Estimación de rendimiento de cultivos agrícolas: integración de datos derivados de monitores de rendimiento y productos del sensoramiento remoto

Por ***Ing. Agr. Diego Nahuel Carcedo***

Presentado ante la Facultad de Matemática, Astronomía, Física y Computación y el Instituto de Altos Estudios Espaciales Mario Gulich como parte de los requerimientos para la obtención del grado de

MAGISTER EN APLICACIONES DE INFORMACIÓN ESPACIAL

UNIVERSIDAD NACIONAL DE CÓRDOBA

Mayo, 2022

©IG-CONAE 2022
©FaMAF-UNC 2022

DIRECTOR
Dr. Marcelo Scavuzzo
Instituto de Altos Estudios Espaciales "Mario Gulich", Córdoba, Argentina

CO-DIRECTOR
Dr. Diego Pons
Instituto Nacional de Tecnología Agropecuaria,Estación Experimental Agropecuaria Manfredi, Córdoba, Argentina

Esta obra está bajo una licencia:
https://creativecommons.org/licenses/by-nc/4.0/

Dedicatoria

A Juan, a quien extraño todos los días.

Juan siempre te hacía avanzar un poco, descubrir algo nuevo en la vida. Desde sus consejos en estadística para escribir un trabajo, hasta sus charlas de música, literatura o política compartiendo un café, siempre te habría la cabeza.

Un profesional, un colega, un amigo super generoso que te podía escuchar con atención por horas repetir la misma historia, y aun así darte una palabra justa al final.

Los amores más grandes generan los duelos más grandes. El duelo de Juan será infinito, como su amor por la investigación y la docencia y toda la sabiduría que él dejó.

A Juan el mejor de los nuestros.

Agradecimientos

Agradezco al equipo de profesores y profesoras, estudiantes y ex-maestrandos del Instituto Gulich (IG).

A los revisores y miembros del tribunal que con dedicación y profundo conocimiento sugirieron mejoras para esta tesis y mi formación profesional.

A mi co-director, Dr. Diego Pons, por las palabras, la guía y la invitación a conocer el Instituto Gulich y la MAIE. A mi director Dr. Marcelo Scavuzzo por su apoyo y estímulo. Al Dr. Giovanni Laneve por haberme recibido en la Scuola di Ingegneria Aerospaziale, en la Università di Roma 'La Sapienza'.

A CONAE por la beca que me permitió iniciar este estudio. A INVAP por haberme facilitado los datos utilizados y el tiempo que me permitió finalizar este estudio. Al equipo de la Cátedra de Estadística y Biometría de la Facultad de Ciencias Agropecuarias de la Universidad Nacional de Córdoba por iniciarme en el hermoso camino académico de satisfacer la infinita curiosidad.

A mis amigos y amigas que me aportaron calma y sugerencias para ordenar el trabajo, que escucharon las ideas todavía sin pulir y me ayudaron a diagramar la tesis y que luego leyeron la tesis y me sugirieron correcciones.

A mi novia, la negrita, por elegir estar a mi lado y ser un apoyo incondicional. A mis hermanos por su ternura, cariño y alegrías. A mi hermana por su sensibilidad y generosidad, por su constante diálogo de saberes y compañerismo. A mi padres por haberme permitido estudiar y por haber, siempre, estimulado mi curiosidad.

A la República Argentina tierra en la que todos los días agradezco haber nacido.

Resumen

En el contexto de un aumento demográfico continuo, la seguridad alimentaria global enfrenta desafíos, acentuados por el cambio climático. La producción de cultivos fundamentales como el maíz se ve afectada por las perturbaciones climáticas, lo que destaca la necesidad de soluciones resilientes. La teledetección emerge como un recurso práctico para estimar los rendimientos de maíz, al proporcionar vasta información de manera remota sobre el crecimiento y las condiciones del cultivo permitiendo ajustes precisos en la gestión agrícola. Esta capacidad no solo optimiza la eficiencia de los recursos para los agricultores, sino que también guía la formulación de políticas adaptativas a nivel local y global, con miras a asegurar la disponibilidad sostenible de alimentos. La recopilación de datos georreferenciados, especialmente a través de sistemas de monitoreo de cosecha, brinda una perspectiva detallada de los rendimientos agrícolas a nivel geográfico. La investigación, basada en información derivada de sensores satelitales, ha generado modelos de predicción de rendimiento de maíz a escala regional y paisajística. En esta investigación se trabajó sobre la hipótesis que la variabilidad de variables derivadas de información de sensado remoto medidas a nivel de lote permiten predecir el rendimiento del cultivo de maíz medido con monitores de cosecha. El objetivo principal de esta tesis fue generar modelos paramétricos y no paramétricos para predecir el rendimiento de maíz en la zona núcleo de la República Argentina a nivel de lote, tomando como datos de entrada información geoespacial de sensores remotos y utilizando observaciones de campo de 33 lotes. Se probó la efectividad de una metodología que permite depurar, estandarizar y re-escalar archivos de datos georreferenciados, que facilitó la interpolación geoestadística de manera cuasiautomática. Se probaron diferentes marcos teóricos para predecir el rendimiento desde información geoespacial de sensores remotos, información climática y de relieve: modelos de regresión lineal múltiple (RLM) y regresión basada en el método de aprendizaje automático “bosque aleatorio” (RFR). Se ponderó la variabilidad intra lote de cada una de las variables predictoras calculando el desvío estándar y el rango. El modelo RLM con interacciones (RMSE: 1.48) explicó un 4,12 % más de la variabilidad del rendimiento que el mejor modelo RFR (RMSE: 1.87). Siendo ambos modelos eficientes para explicar y predecir la variabilidad de los rendimientos. La evidencia aportada por esta tesis sugiere que se pueden construir mejores protocolos de predicción estadística de rendimiento de maíz a nivel regional utilizando el modelo RLM.

Palabras clave: Sensado remoto, Rendimiento, Maíz, Monitores de cosecha, Machine learning, Google Earth Engine.

Abstract

In the context of ongoing demographic growth, global food security faces significant challenges, accentuated by climate change. The production of fundamental crops like maize is affected by climatic disruptions, underscoring the need for resilient solutions. Remote sensing emerges as an essential resource for estimating maize yields, providing extensive information remotely on crop growth and conditions, allowing precise adjustments in agricultural management. This capability not only optimizes resource efficiency for farmers but also guides the formulation of adaptive policies at local and global levels, aiming to ensure sustainable food availability. The collection of georeferenced data, especially through harvest monitoring systems, offers a detailed perspective on agricultural yields at a geographical level. Research, based on information derived from satellite sensors, has generated predictive models for maize yield at regional and landscape scales. In this research, we worked on the hypothesis that the variability of multiple variables derived from remote sensing information measured at the field level allows predicting the yield of the corn crop measured with harvest monitors. The main objective of this thesis was to generate parametric and non-parametric models to predict corn yield in the core zone of the Argentine Republic at the field level, taking as input geospatial data from remote sensors and using data from 33 fields. In first place, the effectiveness of a methodology that allows debugging, standardizing and re-scaling georeferenced data files was tested, which facilitated geostatistical interpolation in a quasi-automatic manner. Secondly, different theoretical frameworks were tested to predict maize yield from climatic, relief and remote sensors geospatial information: multiple linear regression models (RLM) and regression based on the machine learning method random forest"(RFR). The intra-lot variability of each of the predictor variables was weighted by calculating the standard deviation and the range. The RLM model without interactions (RMSE: 1.48) explained 4,12 % more of the performance variability than the best RFR model (RMSE: 1.87). As both models are efficient to explain and predict the variability of yields, the evidence provided by this thesis suggests that better statistical prediction protocols for maize yield can be built at the regional level using the RLM model.

Keywords: Remote sensing. Corn crop yield. Harvest monitors. Machine learning. Google Earth Engine

Índice de Contenidos

Índice de tablas

Índice de figuras

CAPÍTULO 1

Introducción

1.1. Contexto

En un mundo en constante crecimiento poblacional, la necesidad de alimentos se ha convertido en un desafío crítico. A medida que la demanda de comida aumenta, es esencial garantizar un suministro adecuado para alimentar a la población mundial. La producción de cultivos básicos como el maíz (Zea Mays) desempeña un papel fundamental en este contexto, ya que es una fuente crucial de nutrientes y energía para muchas comunidades en todo el mundo.

El cambio climático ha provocado una serie de desafíos para la agricultura global, incluido el cultivo de maíz. Las variaciones en los patrones climáticos, como sequías más intensas y eventos climáticos extremos, han afectado negativamente la producción de cultivos. [3] El maíz, siendo altamente sensible a las condiciones climáticas, ha experimentado cambios en su rendimiento y calidad debido a estas perturbaciones, [4] lo que resalta la importancia de encontrar soluciones resilientes y sostenibles.

En medio de estos desafíos, la teledetección ha emergido como una herramienta invaluable en la estimación de los rendimientos de maíz. Utilizando satélites y tecnologías de observación remota, los científicos pueden monitorear de manera precisa y eficiente las condiciones de cultivo en vastas áreas. Esto incluye la detección temprana de estrés hídrico, radiación en el periodo de floración y otros factores que podrían afectar la producción de maíz. La teledetección ofrece datos en tiempo casi real, lo que permite a los agricultores tomar decisiones informadas para optimizar sus prácticas agrícolas. La capacidad de estimar los rendimientos de maíz de manera precisa a través de la teledetección tiene implicaciones profundas. Los agricultores pueden ajustar sus métodos de riego y manejo de cultivos en función de las condiciones detectadas, lo que resulta en un uso más eficiente de los recursos y una reducción de las pérdidas. Además, los gobiernos y las organizaciones internacionales pueden utilizar esta información para desarrollar políticas agrícolas adaptadas al contexto climático local y global, lo que contribuye a la seguridad alimentaria a largo plazo.

A medida que avanzamos hacia un futuro marcado por la incertidumbre climática [5] y la necesidad de producir más alimentos, la integración continua de la teledetección en la agricultura es esencial. Innovaciones tecnológicas y modelos de análisis de datos más sofisticados prometen mejorar aún más la precisión de las estimaciones de rendimiento de maíz y otros cultivos. Esta colaboración entre la ciencia, la tecnología y la agricultura sienta las bases para enfrentar los desafíos de manera proactiva y trabajar hacia un suministro de alimentos más seguro y sostenible para las generaciones futuras. Un gran cantidad de datos de importancia agronómica son recolectados con sistemas de posición geográfica generando datos geoposicionados, particularmente los monitores de cosecha hoy permiten conocer el rendimiento con gran precisión geográfica. A escala regional, estos datos permiten describir la variación espacial del rendimiento de maíz en distintos lotes agrícolas.

En la República Argentina los productores agropecuarios al reportar al fisco (DGA, ARBA, AFIP) están obligados a informar el dato de rendimiento a nivel de lote o parcela agrícola [6] pero luego estos datos son publicados de manera agregada a nivel regional y/o provincial. Asimismo instituciones que realizan seguimiento estadísticos de rendimiento de maíz (Bolsa de Cereales de Córdoba, Bolsa de Cereales de Buenos Aires, Bolsa de Cereales de Rosario, Dirección de Estimaciones Agrícolas MAGYP, Oficina de Riesgo Agropecuario, entre otras) utilizan datos reportados a escala de lote para realizar sistemas de alarmas regionales de índole climática y fitosanitaria. Finalmente numerosos estudios locales e internacionales han evaluado y generado modelos de predicción y estimación de rendimiento de maíz a escala regional y de paisaje utilizando información derivada de sensores remotos satelitales de alta y media resolución espacial y temporal.

En esta investigación se trabajó sobre la hipótesis que la variabilidad de variables derivadas de información de sensado remoto medidas a nivel de lote permiten predecir el rendimiento del cultivo de maíz medido con monitores de cosecha a nivel de lote.

1.2. Motivación

La República Argentina es un país en el que el sector exportador, en particular el agroindustrial, participa en gran medida del producto interno bruto [7]. Los principales cultivos que se producen en la región pampeana son maíz, trigo y soja. Particularmente el maíz, que participa con un gran porcentaje en la producción nacional anual [8], es el cultivo que presenta el mayor costo inicial de producción y es el más susceptible de reducir su producción ante eventos climáticos extremos [9]. El conocimiento anticipado del rendimiento del cultivo de maíz podría permitir a los agricultores implementar medidas de manejo tanto para mejorar en caso de estimación de rendimientos bajos o para consolidar rendimientos altos esperados, así como para diseñar estrategias futuras de comercialización más convenientes. Además, esta información podría usarse para que las compañías de seguros proporcionen a los productores nuevas pólizas vinculadas a los retornos estimados [10]. Contar con una estimación anticipada de rendimiento de maíz le permitiría tanto al Estado como a las instituciones de referencia del sector realizar y reportar proyecciones sobre el rendimiento total esperado de maíz para un año determinado. Así como diseñar sistemas de monitoreo institucional en tiempo real de las cosechas de cada campaña. Se descuenta que este tipo de información ayudaría a estabilizar los precios con mayor certeza en el mercado de compra y venta de granos [10]. El uso de sensoramiento remoto (SR) para monitorear el crecimiento de los cultivos recientemente ha incrementado de manera

considerable, con el desarrollo de nuevas tecnologías y una mayor disponibilidad de imágenes, permitiendo la explotación de estos datos de una manera cada vez más eficaz [11]. Particularmente, el SR satelital es cada vez más utilizado en el monitoreo de cultivos y variables que describen su crecimiento, ya que permite el monitoreo tanto de grandes zonas a escala regional o local [11], como de áreas más pequeñas a escala de campo [12]. El uso de estos datos ha demostrado ser eficaz en la estimación del rendimiento de los cultivos [13, 14], al combinar el seguimiento de SR de variables biofísicas de cultivos con modelos de crecimiento de cultivos. El incremento en el costo de los insumos (semilla, energía y fitosanitarios) del cultivo de maíz, así como de una disponibilidad creciente de contratistas de servicios de cosecha con máquinas cosechadoras con tecnología de monitoreo de rendimiento, habilitó a que un mayor número de productores utilicen este instrumental aumentando de manera vertiginosa, en los últimos años, la superficie agrícola con observaciones de campo sensibles de ser analizadas. El avance científico en el complejo campo del modelado de variables ambientales, tanto de estrategias estocásticas (orientadas por datos), como de metodologías mecanisticas (orientadas por procesos), hacen del modelado de rendimiento del cultivo de maíz una tarea necesaria para mejorar la agricultura de precisión y la toma de decisiones en la producción agrícola. El uso de datos satelitales, como las imágenes de Sentinel 2 y el cálculo de índices espectrales, brinda una base sólida para llevar a cabo análisis detallados y avanzados que pueden beneficiar tanto a los agricultores como a la comunidad en general.

1.3. Hipótesis

La variabilidad de variables derivadas de información de sensado remoto medidas a nivel de lote permiten estimar el rendimiento del cultivo de maíz en la zona núcleo de la República Argentina.

1.4. Objetivo General

Evaluar modelos paramétricos y no paramétricos para estimar el rendimiento de maíz en la zona núcleo de la República Argentina a nivel de lote, tomando como datos de entrada información geoespacial de sensores remotos.

1.5. Objetivos Específicos

1. Estudiar los antecedentes relativos al modelado de rendimiento de maíz a partir de información ambiental.
2. Implementar y testear herramientas de software que faciliten el preprocesamiento de variables de terreno (rendimiento) y de sensores remotos, que permitan analizar la variabilidad entre lotes agrícolas de maíz.
3. Comparar la performance de distintas estrategias de modelado para caracterizar la relación entre múltiples variables geoespaciales provenientes de sensoramiento remoto y el rendimiento de maíz obtenido en distintas campañas agrícolas.

1.6. Estructura de la tesis

La tesis está estructurada en cinco capítulos. En el primero se introduce la problemática, el contexto general, la motivación, la hipótesis, los objetivos planteados en la investigación. En el segundo capítulo se aborda el marco teórico. En el tercer capítulo se describe la metodología y los datos utilizados. En el cuarto capítulo se presentan los resultados obtenidos y en el capítulo cinco se abordan las conclusiones de la tesis.

CAPÍTULO 2

Marco teórico

2.1. Ecofisiología del rendimiento del maíz.

De manera introductoria podemos mencionar que el rendimiento del cultivo de maíz está estrechamente ligado a la producción de biomasa aérea, la cual depende de la cantidad de radiación fotosintéticamente activa que el cultivo logra interceptar.[15] El maíz, gracias a su sistema fotosintético C4, es más eficiente en la conversión de radiación en biomasa en comparación con los cereales de invierno de sistema C3. La eficiencia en el uso de la radiación (EUR) es fundamental para el rendimiento y si bien en condiciones de laboratorio puede variar con la temperatura [16] y la etapa del ciclo del cultivo [17], tiende a ser constante en las diversas condiciones de cultivo. Por otro lado Satorre [15] documentó que el aumento en la producción de biomasa depende de la duración del ciclo y de la eficiencia con la cual el cultivo captura la luz.

La tasa de producción de biomasa determina a su vez la cantidad y el peso de los granos que la planta genera. Estas dos variables: número de granos por planta (NGP) y el peso individual de los granos (PG) son los componentes numéricos que determinan el rendimiento en maíz. El NGP se establece en un período de tiempo denominada crítico centrado en la etapa fenológica de floración. [15]

La radiación fotosinteticamente activa interceptada (RFAi), es la porción de la radiación solar que el cultivo intercepta, y esta determinada en gran medida por la cobertura que el cultivo consigue realizar sobre el suelo y plantas de otras especies. Esta cobertura, denominada cobertura del dosel esta a su vez determinada por la cantidad de hojas de cada planta. La cantidad de hojas de cada planta, se denomina índice de área foliar (IAF) y es por ende un indicador importante de la RFAi (Figura 2.1), que a su vez es un indicador importante de la producción de biomasa aérea (Figura 2.2) que es finalmente la variable que determina el numero y peso de los granos por ende el rendimiento. [15].

En el ANEXO I se ahonda en los detalles de la ecofisiología de la determinación del rendimiento de maíz.

h]

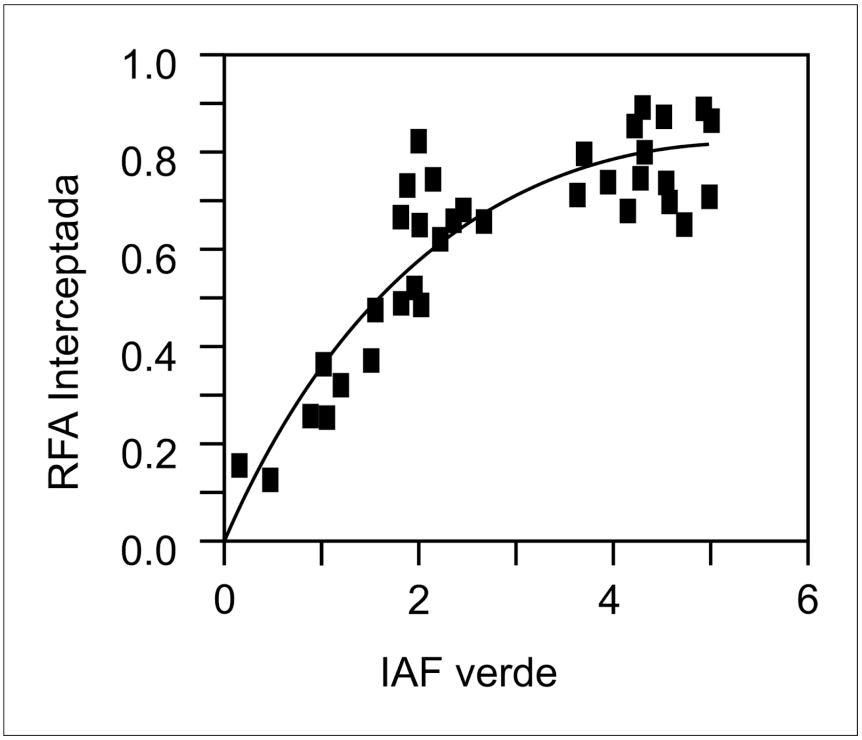

Figura 2.1: Adaptada de [18] Relación entre índice de área foliar y radiación fotosintéticamente activa interceptada (RFAi)

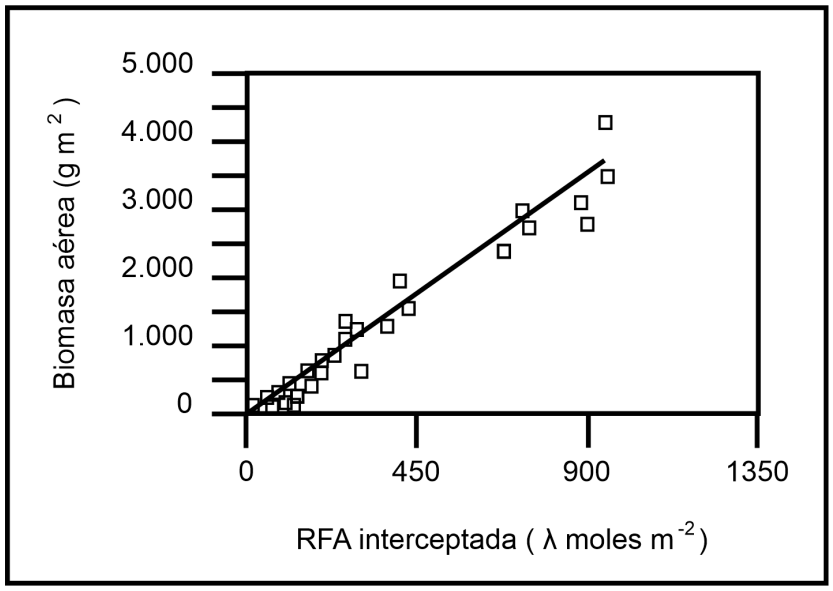

Figura 2.2: Adaptado de [17]. Relación entre la radiación fotosintéticamente activa interceptada (RFAi) y la biomasa aérea.

2.1.1. Relación entre índice de área foliar e índices de vegetación.

La precisión y rapidez para obtener el IAF resulta crucial en la investigación ecológica orientada a procesos, ya que contribuye a comprender los intercambios vegetales (Figura 2.2) en

diversas escalas, desde niveles microscópicos hasta extensos paisajes. A pesar de su importancia, medir directamente el IAF en vastas áreas es complejo y demandante en términos de tiempo y recursos. La aparición de técnicas ópticas y la teledetección activa ha revolucionado estos esfuerzos, mejorando sustancialmente la capacidad de estimar y validar el IAF en entornos regionales y de paisaje de manera rápida y no intrusiva. Un ejemplo de este avance es el trabajo de Peng [1], quien estableció la relación entre el IAF e índices de vegetación calculados con información capturada de manera remota, brindando herramientas valiosas para la evaluación y comprensión de la vegetación en diversos contextos y escalas.

2.2. Relación entre índices de vegetación derivados de sensores remotos y rendimiento de maíz.

Numerosos estudios (Tabla 2.1) han confirmado la utilidad, conveniencia y eficacia de utilizar información de sensores remotos para calcular diversos índices de vegetación, con el propósito de cuantificar o estimar variables biofísicas en cultivos agrícolas [19], [20], [21], [22]. Esta metodología ha sido especialmente valiosa para establecer vínculos entre índices de vegetación y el rendimiento del maíz [23], [24], [25]. A pesar de la amplia aplicación de los índices ópticos de teledetección en la evaluación de características biofísicas de la vegetación en contextos agrícolas, es importante tener en cuenta que estos índices resultan de la combinación de cambios en las propiedades tanto de la vegetación como del entorno. Esto incluye factores como el IAF, el contenido de clorofila en las hojas, las sombras del dosel y la reflectancia del suelo subyacente. Por ello, se sugiere emplear una combinación de diferentes índices [26] para una evaluación del rendimiento del cultivo del maíz. Tal como señala Peng [1], el contenido de clorofila es de particular relevancia para la agricultura de precisión, ya que refleja la actividad fotosintética y, por ende, la generación de biomasa aérea, que como se ha mencionado, guarda una relación directa con el rendimiento del maíz.

Tabla 2.1: Adaptado de [1]. Relación existente entre índices de vegetación y rendimiento de maíz.

Índice de vegetación	Modelo		Modelo validado	
	R2	RMSE	R2	RMSE
NDVI	0.823	0.646	0.784	0.659
OSAVI	0.799	0.669	0.772	0.682
GNDVI	0.792	0.697	0.743	0.718
EVI2	0.724	0.774	0.69	0.791
MSAVI2	0.741	0.728	0.734	0.742
EVI	0.877	0.609	0.795	0.621

2.3. Firma espectral del Maíz

La firma espectral se refiere a la característica única de cómo una sustancia o material interactúa con la luz en las diferentes longitudes de onda a lo largo del espectro electromagnético. Espe-

cíficamente, se trata de la reflectancia o la emisividad de un material en función de la longitud de onda de la luz incidente.

Cada material o sustancia, ya sea un objeto en la Tierra, una planta, una roca o incluso una molécula, tiene una firma espectral única. Esto significa que reflejará o emitirá luz de manera diferente en diferentes partes del espectro electromagnético. La firma espectral de un material se mide en un rango de longitudes de onda que abarca desde la luz visible hasta el infrarrojo cercano o incluso el infrarrojo térmico y revela información valiosa sobre sus propiedades físicas y químicas.

En el marco del proyecto de construcción de una biblioteca de firmas espectrales realizado en conjunto entre CONAE e INVAP [27] se realizaron mediciones de radiancia de un cultivo de maíz en la localidad de Pehuajó, Provincia de Buenos Aires, en febrero de 2015. Utilizando un radiómetro de campo, el ASD FieldSpec FR 1, se realizaron mediciones en 11 puntos a lo largo de una transecta que permitieron luego construir la firma espectral del maíz para el estado fenológico de emergencia de estigma (floración). La firma representa la mediana de las firmas espectrales de los espectros de reflectancia obtenidos en cada punto de la transecta (Figura 2.3)

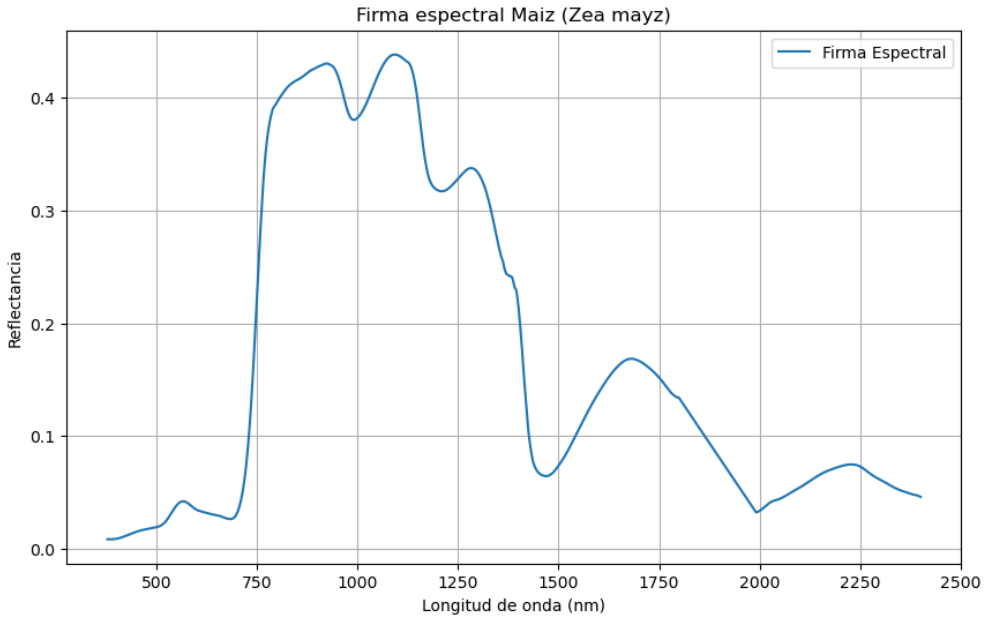

Figura 2.3: Firma espectral del maíz en fase de emergencia de estigma en Pehuajó, Buenos Aires, Febrero 2015.

2.4. Datos.

En este estudio se abordó el desafío metodológico de contar con diferentes escalas entre los datos derivados de plataformas de sensoramiento remoto y datos de monitores de rendimiento integrando los datos en la misma escala a nivel de lote.

2.4.1. Datos de campo: Monitores de rendimiento.

Los monitores de rendimiento son una tecnología ampliamente utilizada en la agricultura de precisión, y los mapas de rendimiento de alta resolución se han convertido en una herramienta

esencial en la agricultura moderna [28] [29]. Con el uso cada vez mayor de monitores de rendimiento y la acumulación de datos de rendimiento a lo largo de varios años, surge la necesidad de desarrollar técnicas robustas para el procesamiento e interpretación de estos datos. Un monitor de rendimiento es un equipo que se integra en una máquina cosechadora (Figura 2.4) y cuenta con los siguientes componentes:

1. Sensor de flujo de grano: Este sensor permite medir el volumen de grano cosechado. Existen diferentes métodos para realizar esta medición, pero generalmente se utiliza un dispositivo que se coloca en el cabezal de la noria, que es el sistema encargado de elevar el grano limpio hacia la tolva de la cosechadora.

2. Sensor de humedad de grano: Este sensor mide la humedad del grano a medida que ingresa a la cosechadora. Esto permite corregir el valor estimado de rendimiento a un valor de humedad estándar.

3. Sensor de velocidad de avance: La velocidad de avance puede medirse directamente o calcularse a partir de la información del sistema GPS. Esta velocidad es esencial para calcular la distancia recorrida y, por lo tanto, el área cosechada, multiplicando este valor por el ancho de trabajo del cabezal de cosecha, que debe ser ingresado manualmente por el operador.

4. Sensor del cabezal: Este sensor define el inicio y el fin del registro de datos según si el cabezal de la cosechadora se encuentra o no en posición de trabajo.

5. Consola del monitor: La consola del monitor, también conocida como "datalogger", es donde se registran los datos de los sensores y se ingresan los parámetros iniciales antes de iniciar la cosecha. Las marcas comerciales más conocidas incluyen .Ag Leader"para cosechadoras CASE y New Holland, "Green Star"para John Deere, y "Field Star"para Massey Ferguson y AGCO Allis. Los datos exportados de los monitores de rendimiento pueden tener diferentes formatos según la marca y el equipo instalado en la cosechadora.

6. Receptor GPS o DGPS: Este sensor determina la posición geográfica exacta de la máquina cosechadora. Todo dato relevado tendrá coordenadas geográficas otorgadas por el GPS.

Los datos del monitor de rendimiento contienen fuentes sistemáticas y aleatorias de variación de rendimiento medido[30], que incluyen:

1. Variabilidad de rendimiento relacionada con las condiciones climáticas y las características del suelo y el paisaje, que suele ser más estable.

2. Variabilidad de rendimiento inducida por prácticas de manejo variables.

3. Errores de medición asociados al proceso de mapeo del rendimiento.

El uso de esta tecnología presenta diversas ventajas, entre las que destacan:

1. Facilita la realización de agricultura por ambiente de similares características, optimización el uso del suelo y los insumos generando una reducción de costos por unidad producida.

2. A través de la recopilación y análisis de datos de rendimiento georreferenciados durante varios años, es posible identificar patrones de rendimiento repetitivos y sus causas naturales. Esto permite separar esta variabilidad de rendimiento consistente de la variación

Figura 2.4: Principales componentes del monitor de rendimiento integrado en una máquina cosechadora.

aleatoria que puede ser inducida por la gestión o las condiciones específicas de cada año.

Por otro lado, es importante mencionar las limitaciones asociadas a este tipo de datos, que incluyen:

1. Se registran errores debido a la variabilidad en las prácticas de manejo, lo que genera eventos aleatorios. Fenómeno típicamente observado en lotes pequeños. Estos eventos pueden incluir saltos en la sembradora, un establecimiento deficiente del cultivo, aplicación desigual de fertilizantes, daños causados por herbicidas, heladas o plagas.

2. Suelen generarse errores de medición debido a fallas en los sensores de flujo de grano y humedad, problemas en la georreferenciación y movimientos de la cosechadora, errores humanos por parte del operador y fallos en el procesamiento de datos [30].

A pesar de que un mapa de rendimiento de un solo año resulta útil para interpretar las posibles causas de la variación en el rendimiento, su utilidad se limita cuando se trata de tomar decisiones estratégicas de manejo específicas para un sitio en períodos de mediano a largo plazo.

2.4.2. Datos satelitales: Plataformas de sensoramiento remoto.

Los datos satelitales son una fuente de datos valiosos para estudiar el comportamiento del cultivo de maíz y estimar su rendimiento, ya que proporciona información precisa, oportuna y a gran escala de manera no intrusiva, es decir sin tener que cortar partes del tejido del cultivo. Utilizar datos satelitales y en particular índices de vegetación para estudiar el comportamiento del cultivo de maíz y predecir su rendimiento tiene varias ventajas significativas:

1. Cobertura global y escalabilidad: Los datos satelitales permiten el monitoreo simultáneo de vastas extensiones de terreno, lo cual es esencial para el estudio de cultivos a gran escala.

2. Frecuencia de observación: Estos datos tienen una periodicidad regular, lo que habilita un seguimiento constante y uniforme durante todo el ciclo fenológico del cultivo, posibilitando la identificación de tendencias y patrones a lo largo del tiempo.

3. Acceso a áreas remotas: Los datos satelitales pueden recopilarse incluso en áreas remotas o de difícil acceso, lo que facilita el monitoreo de cultivos en regiones apartadas o de difícil acceso. dónde puede ser complicado obtener datos de campo.

4. Rápida adquisición de datos: La obtención de datos satelitales es rápida y eficiente, lo que permite una respuesta oportuna ante eventos climáticos extremos o cambios en las condiciones del cultivo.

5. Información multispectral: Los datos satelitales se obtienen a través de muestreos en diferentes longitudes de onda del espectro electromagnético, lo que nos permite tener múltiples datos de píxel o sitio

6. Monitoreo de la variabilidad espacial: Los datos satelitales permiten identificar la variabilidad espacial dentro de un lote agrícola. Esta visión detallada de las diferencias en el terreno y en el crecimiento de las plantas en diversas áreas, facilita la identificación de patrones espaciales (zonas de mayor o menor vigor vegetal) que luego pueden ser utilizadas para optimizar el manejo del cultivo por ambientes de similares características.

7. Costo-efectividad: Utilizar datos satelitales puede ser más económico que realizar un monitoreo aéreo o terrestre en gran escala, lo que lo convierte en una opción atractiva para agricultores, investigadores y planificadores agrícolas.

Las plataformas de sensoramiento remoto también conocidas como satélites son un conjunto diverso de instrumentos y tecnologías empleados con fines de diverso índole, desde facilitar comunicaciones como para observar y monitorear nuestro planeta a distancia. Se dividen principalmente en cinco categorías:

1. Órbita terrestre baja (LEO): Son satélites que operan en altitudes generalmente entre 300 y 1.200 kilómetros y se suelen utilizar para la observación terrestre de alta resolución y la investigación científica.

2. Órbita terrestre media (MEO): Son satélites se utilizan para servicios de comunicación, posicionamiento y navegación , como los del sistema de navegación GPS.

3. Órbita geoestacionaria (GEO): Son satélites que están ubicados a una altitud de aproximadamente 35.786 kilómetros sobre el ecuador, donde parecen "flotar.[en] el mismo lugar en relación con la rotación de la Tierra. Esta posición fija les permite proporcionar un monitoreo continuo de una región específica, lo que los hace ideales para aplicaciones como pronóstico del tiempo y comunicación.

4. Órbita de transferencia geoestacionaria (GTO): Son satélites satélites se utilizan para lanzar otros satélites desde órbitas de transferencia hacia órbitas geoestacionarias. Los satélites en GTO actúan como vehículos lanzadores.

5. Órbita heliosíncronicos (SSO): Estos satélites orbitan a altitudes que van desde aproximadamente 600 kilómetros hasta 800 kilómetros sobre la Tierra y se utilizan para la

observación de la Tierra y la monitorización del medio ambiente, proporcionando imágenes constantes y adecuadas para la detección de cambios, la predicción de ciclones y la monitorización de desastres naturales.

Los satélites SSO son los indicados para utilizar en el monitoreo agropecuario y ambiental debido a su capacidad para proporcionar imágenes constantes y consistentes de la Tierra en condiciones de iluminación similares. Al mantener una órbita que pasa sobre la misma región de la Tierra a la misma hora local todos los días, los SSO capturan imágenes con iluminación consistente, lo que facilita la detección de cambios en la vegetación, el seguimiento de cultivos y la evaluación de patrones climáticos.

Los principales satélites SSO son:

2.4.2.1. Misión MODIS.

La Misión MODIS (Moderate Resolution Imaging Spectroradiometer) es un logro colaborativo entre la NASA y el Instituto Nacional de Estudios Espaciales de Francia (CNES). Su instrumento central, el espectro radiómetro MODIS, está diseñado para capturar datos a escala global con una resolución espacial de 250 metros a 1 kilómetro en sus bandas espectrales visibles e infrarrojas. Esta plataforma se caracteriza por su resolución temporal excepcional, ya que proporciona observaciones diarias y, en algunos casos, incluso varias veces al día, permitiendo el seguimiento en tiempo casi real de cambios en la superficie terrestre. Las diversas bandas espectrales de MODIS, que van desde el ultravioleta hasta el infrarrojo térmico, brindan información espectral detallada sobre la atmósfera, la superficie terrestre y los cuerpos de agua. Una de las principales ventajas de MODIS radica en su capacidad para calcular una amplia gama de índices de vegetación que permiten evaluar la salud y la actividad vegetal a lo largo del tiempo. Estos índices son esenciales para monitorear patrones de sequía, identificar áreas afectadas por incendios forestales y estudiar la expansión urbana, entre otras aplicaciones. MODIS ha demostrado ser muy útil para el seguimiento de fenómenos climáticos y la observación de eventos naturales extremos. Sus datos son de acceso libre, gratuito e irrestricto para volúmenes de datos no comerciales.

2.4.2.2. Misión Landsat.

La misión Landsat, una colaboración entre la NASA y el Servicio Geológico de los Estados Unidos (USGS), es un programa emblemático de observación de la Tierra que ha proporcionado datos cruciales para el monitoreo ambiental y la investigación durante décadas. Su instrumento principal, el Sensor Operativo de Teledetección (ORS), es un avanzado sistema de captura remota que abarca un espectro amplio, capturando información en múltiples bandas espectrales, desde el visible hasta el infrarrojo térmico. Con una resolución espacial de 30 metros ofrece una visión detallada de la superficie terrestre. La resolución temporal, una revisita cada 8 días para ciertas bandas, permite el seguimiento de cambios sutiles en la cobertura terrestre y la detección de eventos a lo largo de semanas, meses e incluso años. La resolución espectral, de 11 bandas distintas, posibilita el análisis detallado de diversos fenómenos, como la salud vegetal. Los datos de esta misión son de acceso libre, gratuito e irrestricto para volúmenes de datos no comerciales.

2.4.2.3. Misión Sentinel-2.

Es una misión satelital, de la agencia espacial europea (ESA), multiespectral de alta resolución, que consiste en dos satélites gemelos situados en la misma órbita pero desfasados 180 grados. Se encuentra diseñado para un revisita temporal alta de 5 días en el Ecuador, y lleva consigo un instrumento óptico que muestra en 13 bandas espectrales distintas: cuatro bandas a 10 metros, seis bandas a 20 metros y tres bandas a 60 m de resolución espacial [2]. El conjunto de datos de nivel 1C, capturados por el instrumento 'MultiSpectral Instrument' (MSI) de la plataforma Sentinel-2 son de acceso libre, gratuito e irrestricto para volúmenes de datos no comerciales. Este conjunto de datos contiene datos continuos desde el año 2015 hasta la actualidad. Las imágenes disponibles cuentan en sus metadatos con información sobre el porcentaje de nubosidad presente en la totalidad de la escena capturada, lo que permite después filtrar y seleccionar aquellas imágenes libres de nubes.

Banda espectral	**Longitud de onda central (nm)**	**Ancho de la banda (nm)**	**Resolución espacial (m)**
B1	443	20	60
B2	490	65	10
B2	490	65	10
B3	560	35	10
B4	665	30	10
B5	705	15	20
B6	740	15	20
B7	783	20	20
B8	842	115	10
B8a	865	20	20
BO	945	20	60
B10	1380	30	60
B11	1610	90	20
B12	2190	180	20

Tabla 2.2: Características de las bandas de Multi Spectral Instrument (MSI) de Sentinel-2. [2]

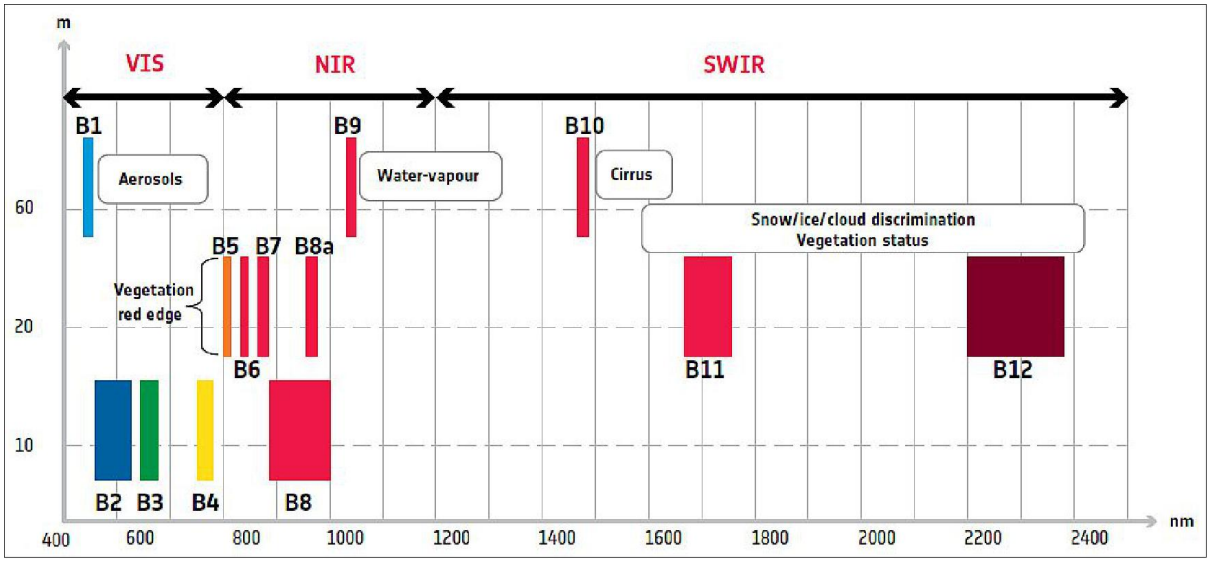

Figura 2.5: Características de las bandas de Multi Spectral Instrument (MSI) de Sentinel-2 [2].

La combinación de resolución espacial y temporal en Sentinel-2 (Figura 2.5) brinda la capacidad de generar índices de vegetación con mayor precisión y frecuencia en comparación con otras plataformas. Esto resulta especialmente valioso en aplicaciones agrícolas, gestión de recursos naturales y seguimiento de la cobertura terrestre, donde los cambios rápidos y sutiles son esenciales para la toma de decisiones informadas

En el contexto de esta tesis se eligió trabajar con datos de Sentinel-2 para calcular índices espectrales focalizados en el monitoreo de vegetación ya que la alta resolución temporal y espacial, mejora la detección y comprensión de los cambios en la vegetación y los ecosistemas terrestres de manera más precisa y oportuna.

2.4.3. Catálogo de activos espacio temporales (STAC) y plataformas de cómputo en la nube.

El Catálogo de Activos Espacio Temporales (STAC) es un marco diseñado para optimizar el acceso y uso de datos en constante evolución, como datos geoespaciales y satelitales. Adoptando estándares abiertos y basados en la web, STAC agiliza la búsqueda y acceso eficiente a diversos activos, como imágenes satelitales, datos climáticos y observaciones terrestres, conservando información esencial de cada recurso. Este enfoque altamente interoperable fomenta la colaboración entre fuentes de datos y aplicaciones, impulsando soluciones innovadoras en áreas como monitoreo ambiental y agricultura de precisión.

La integración de catálogos STAC en plataformas de cómputo en la nube amplía el potencial de análisis e investigaciones geoespaciales. Estas plataformas escalables almacenan, procesan y distribuyen datos globalmente. Al incorporar STAC, los catálogos en la nube simplifican la búsqueda y selección de datos relevantes, mientras que su interoperabilidad facilita la fusión de múltiples fuentes, posibilitando análisis complejos. Potenciadas por la capacidad de cómputo en la nube, se habilita la realización de análisis avanzados a una escala previamente inalcanzable. La combinación de STAC y las plataformas de cómputo en la nube impulsa la innovación en gestión ambiental, agricultura de precisión y respuesta a desastres, permitiendo una exploración y acceso ágil y profundo de los datos geoespaciales.

2.4.3.1. Amazon Web Services (AWS).

Es una plataforma líder en la nube que ofrece una amplia gama de servicios de cómputo, almacenamiento, bases de datos y análisis. Con una presencia global, AWS permite a las usuario implementar, escalar y administrar recursos de tecnología de información de manera flexible y eficiente. Los usuarios pueden acceder a servicios como Amazon EC2 para la creación de máquinas virtuales, Amazon S3 para el almacenamiento escalable, y Amazon RDS para bases de datos administradas. AWS ofrece un ecosistema completo para el desarrollo, implementación y gestión de aplicaciones y servicios en la nube, respaldado por un sólido conjunto de herramientas y APIs para la automatización y el control preciso.

2.4.3.2. Planet Azure.

Es una plataforma innovadora centrada en la observación y monitoreo de la Tierra desde el espacio. Ofreciendo datasets de múltiples plataformas satelitales, Planet Azure disponibiliza imágenes de alta resolución y datos geoespaciales para diversas aplicaciones, desde el seguimiento ambiental hasta la planificación urbana. La plataforma proporciona acceso a un catálogo en constante actualización de imágenes que abarcan amplias áreas geográficas con alta resolución temporal y especial. Mediante herramientas de análisis y visualización integradas, los usuarios pueden explorar y comprender los cambios en la superficie terrestre a lo largo del tiempo. Planet Azure facilita la toma de decisiones informadas al ofrecer información precisa y actualizada sobre nuestro planeta desde una perspectiva espacial.

2.4.3.3. Planetary Computer.

Planetary Computer, impulsado por Microsoft, se destaca como una plataforma innovadora en el ámbito de la ciencia de datos geoespaciales. Está diseñadA específicamente para el análisis de datos geoespaciales a gran escala. Ofrece una amplia gama de servicios para la carga, el almacenamiento, el procesamiento y el análisis de datos geoespaciales:

1. APIs y SDKs para acceder a datos de satélites, sensores remotos y otros fuentes geoespaciales.

2. Herramientas para el procesamiento de datos geoespaciales, como la clasificación de imágenes, la extracción de características y la geocodificación.

3. Herramientas para el análisis de datos geoespaciales, como la visualización de datos, la modelización espacial y la inteligencia artificial.

Planetary Computer, ofrece un ecosistema integral para acceder y analizar conjuntos de datos a escala planetaria. Esta plataforma proporciona un vasto repositorio de datos geoespaciales provenientes de diversas fuentes, permitiendo a los usuarios explorar y utilizar información detallada sobre la Tierra. Ofrece herramientas avanzadas de análisis y visualización que posibilitan el estudio de patrones ambientales, el monitoreo de cambios en el paisaje y la comprensión de fenómenos terrestres a lo largo del tiempo.

2.4.3.4. Google Earth Engine (GEE).

Es una plataforma basada en la nube para el análisis geoespacial a escala planetaria. Es una plataforma integrada que permite realizar descargas masivas y automáticas de imágenes satelitales y distintos conjuntos de datos disponibles de manera libre y accesibles por intermedio de un catálogo. GEE permite realizar procesamiento sobre los datos, así como la construcción de series temporales y los análisis directamente desde su editor de código utilizando el lenguaje de programación JavaScript [31].

En el contexto de esta tesis se eligió trabajar con la plataforma GEE ya que diferentes estudios comprobaron la versatilidad, practicidad y eficiencia de utilizar esta plataforma, para calcular distintos índices de vegetación para cuantificar, estudiar y predecir variables biofísicas de cultivos agrícolas [20].

2.5. Modelos de estimación de rendimiento.

En la estimación de rendimiento anual de los cultivos con sensores remotos se trabaja principalmente con dos estrategias.

1. Modelos mecanísticos alimentados con diferentes fuentes de datos terrestres y satelitales, también conocidos como modelos basados en procesos. La restricción de estos modelos radica en que las variables utilizadas en la operación deben ser parametrizadas con datos que pueden no estar disponibles o ser difíciles de obtener para distintos sitios.

2. Modelos estocásticos conformado por variables aleatorias y analizadas en términos de su probabilidad, son modelos que se alimentan con información que caracteriza el ambiente (índices de vegetación, variables edáficas, relieve, climáticas, entre otras). Existen dos metodologías estocásticas, una que utiliza datos a largo plazo (between-season) y otra que utiliza datos de una campaña (in-season). La metodología de análisis temporal (in-season) estudia los índices espectrales derivados de los datos de imágenes recopilados en diferentes estados fenológicos del ciclo de cultivo y aplica modelos estadísticos para lograr la mejor correlación con el rendimiento final. La metodología (between-season) se basa en el hecho de que la integración de imágenes de varias campañas a lo largo de los años es representativa del desempeño de los cultivos en diferentes condiciones ambientales. La decisión de utilizar los enfoques a largo plazo o a corto plazo se basa en la resolución temporal y espacial de las imágenes disponibles para el estudio.

A continuación, una breve síntesis de los modelos más utilizados en la actualidad.

2.5.1. Modelos mecanísticos.

Estos modelos se destacan por el grado de conocimiento que se necesita para utilizarlos, ya que su premisa es intentar describir y modelar los procesos fisiológicos que subyacen en la estimación de rendimiento. En situaciones prácticas por lo general resultan costosos de utilizar.

2.5.1.1. AquaCrop.

Es un modelo de simulación agrícola desarrollado por la FAO en 2009, que se enfoca en cultivos de cereales y hortalizas, ofreciendo un análisis detallado de cómo la disponibilidad de agua afecta el rendimiento de los cultivos. Utiliza datos climáticos (precipitación, temperatura, etc), de propiedades del suelo (capacidad de retención de agua, tasas de infiltración, etc), de prácticas de manejo (cantidad y momentos de irrigación, etc) y de propiedades de los cultivos (tasas de crecimiento en diferentes fases fenológicas, respuestas al estrés hídrico, etc) para estimar el rendimiento del cultivo en términos de producción de biomasa y cosecha, así como la eficiencia del uso del agua. Aunque su fortaleza radica en su capacidad para abordar interacciones complejas entre suelo, planta y atmósfera, tiene limitaciones al enfocarse en cultivos anuales y depender de la calidad de los datos de entrada.

2.5.1.2. Simulador de sistemas de producción agrícola (APSIM).

Es un modelo de simulación de sistemas de producción agrícola ampliamente utilizado que se desarrolló en Australia. Al igual que AquaCrop, APSIM es una herramienta computacional diseñada para modelar y simular diversos aspectos de los sistemas agrícolas y la producción de cultivos. A diferencia de AquaCrop, APSIM es un modelo más completo y versátil que abarca una amplia gama de cultivos, sistemas de producción y condiciones climáticas.

APSIM se ha utilizado para simular diferentes componentes de sistemas agrícolas, como la interacción suelo-planta-atmósfera, los ciclos de nutrientes, el crecimiento de cultivos, el manejo de plagas y enfermedades, y la gestión del agua. Puede considerar factores como la variabilidad climática, las prácticas agrícolas y la gestión del suelo en sus simulaciones.

El modelo es conocido por su capacidad de adaptación a diversas condiciones y su flexibilidad para simular una amplia gama de cultivos y sistemas de producción, incluyendo pastizales, sistemas de secano y sistemas intensivos de riego. APSIM ha sido utilizado en investigaciones agrícolas, estudios de manejo de cultivos y toma de decisiones agrícolas en todo el mundo.

APSIM también tiene limitaciones, como la necesidad de datos precisos para calibrar y validar las simulaciones, y la complejidad técnica que puede requerir un conocimiento especializado para su implementación y uso efectivo.

2.5.1.3. Enfoque sistémico para la sostenibilidad del uso de la tierra (SALUS) .

Desarrollado por el Dr. Bruno Basso y su equipo de investigación en la Universidad Estatal de Míchigan, SALUS es un modelo de simulación agrícola y de sistemas ecológicos que tiene como objetivo principal evaluar la sostenibilidad de los sistemas de uso de la tierra, considerando aspectos agrícolas, ambientales y económicos.

SALUS toma en cuenta una amplia gama de interacciones en los sistemas agrícolas, incluyendo la dinámica de los cultivos, la gestión del suelo, la gestión de nutrientes y el manejo del agua. También considera aspectos socioeconómicos y ambientales, lo que lo convierte en un enfoque integral para evaluar la sostenibilidad de diferentes sistemas de producción agrícola.

Es un modelo avanzado y holístico que busca evaluar la sostenibilidad de los sistemas agrícolas considerando múltiples factores interconectados. Se utiliza para modelar y predecir los

impactos de diversas prácticas agrícolas y cambios en el uso de la tierra en la producción de cultivos, la calidad del suelo, la retención de nutrientes y la eficiencia en el uso del agua. Su enfoque integral resulta valioso para evaluar los efectos del cambio climático y para optimizar estrategias de manejo que equilibren la producción agrícola con la conservación de recursos y la sostenibilidad a largo plazo.

2.5.1.4. Algoritmo simple para estimaciones de rendimiento (SAFY) .

Es un modelo de rendimiento de cultivos desarrollado por Duchemin en 2008 que simula el crecimiento de cultivos y la biomasa acumulada diariamente. Fue desarrollado originalmente para trigo de invierno bajo riego y luego probado en diferentes condiciones y cultivos. Monteith (1977) desarrolló una teoría simple para vincular la producción de fitomasa seca total y la parte fotosintéticamente activa de la radiación solar (PAR) absorbida por las plantas [32]. El modelo SAFY utiliza esta relación. Opera con un paso temporal diario desde el día de la emergencia de la planta hasta el día de la senescencia de la hoja completa. Durante este período, la producción de fitomasa es impulsada por la radiación PAR entrante absorbida por las hojas, con dos fases fenológicas sucesivas: (1) Extensión de la hoja; (2) Llenado de grano. Los parámetros se determinan a partir de literatura, mediciones in situ y datos de teledetección. Para la determinación efectiva de parámetros, los datos deben ser adquiridos a altas resoluciones espaciales y temporales [32].

2.5.1.5. Estudio de la Alimentación Mundial (WOFOST) .

El modelo de simulación desarrollado por Van Dipen en 1989 es una herramienta para analizar el crecimiento y producción de cultivos bajo una amplia gama de condiciones climáticas y del suelo. Para conducir este análisis es primero necesario conocer en qué medida la producción del cultivo está limitada por los factores de luz, humedad y los macro nutrientes en el sitio y, en segundo lugar, estimar qué prácticas culturales se pueden realizar para mejorar la performance del cultivo. WOFOST funciona de forma jerárquica, evaluando la producción potencial en condiciones de producción limitadas por agua y nutrientes [33].

2.5.1.6. CERES-Maíz .

Es un modelo de simulación computacional diseñado para representar los procesos que tienen lugar en las plantas y el suelo en relación con el crecimiento y desarrollo del cultivo de maíz. En este modelo, se incorporan aspectos teóricos relacionados con la hidratación del suelo, la fenología y la generación de materia seca. Estos aspectos se basan en datos locales que describen el clima, las condiciones del suelo y el propio cultivo.

El modelo incluye una función adicional dedicada al nitrógeno, con el propósito de emular los efectos de las prácticas de fertilización. Esta función contempla las transformaciones que ocurren tanto en el suelo como en la planta en relación con el nitrógeno. Inicialmente, el modelo fue concebido por Jones y J.R. Kiniry en 1986 para aplicaciones en pastizales. Con el tiempo, ha sido adaptado y sometido a pruebas en diversos cultivos a nivel global. Es importante destacar que el desempeño de este modelo depende significativamente de la calidad de los datos del terreno en el que se aplique, lo que representa su principal limitación [34].

2.5.1.7. STICS.

Desarrollado en INRA (Francia) en 1996, este modelo simula el crecimiento de cultivos, el agua del suelo y el balance de nitrógeno basado en datos climáticos diarios. Calcula ambas variables agrícolas (rendimiento, consumo de insumos) y variables ambientales (pérdidas de agua y nitrógeno). Desde el punto de vista conceptual, STICS se basa esencialmente en relaciones conocidas entre variables y sobre simplificaciones de modelos existentes. Uno de los elementos clave de STICS es su adaptabilidad a diversos cultivos. Esto se logra mediante el uso de parámetros genéricos relevantes para la mayoría de los cultivos [35].

2.5.1.8. El sistema de apoyo a la transferencia de agrotecnología (DSSAT).

Este popular modelo ha sido usado intensamente los últimos 15 años por investigadores de todo el mundo. Este sistema combina modelos de 16 cultivos diferentes con un software que facilita la evaluación y aplicación de los modelos de cultivo para diferentes fines. El DSSAT fue desarrollado originalmente por una red internacional de científicos "International Benchmark Sites Network for Agrotechnology Transfer" para facilitar la aplicación de modelos de cultivos en un sistema con enfoque a la investigación agronómica. La base del DSSAT es un modelo del sistema de cultivo (CSM). Este es una estructura modular en la que los componentes se separan a lo largo de líneas de disciplina científica y son estructurados para permitir un fácil reemplazo o adición de módulos. Tiene un módulo de suelo, un cultivo, un módulo de plantillas que puede simular diferentes cultivos definiendo archivos de entrada de especies, una interfaz para agregar modelos de cultivos individuales si tienen el mismo diseño e interfaz, un módulo de clima y, un módulo para tratar la competencia por la luz y el agua entre el suelo, las plantas y la atmósfera. También está diseñado para incorporarse en varios paquetes de aplicaciones, que van desde aquellos que ayudan a los investigadores a adaptar y probar el CSM a aquellos que operan el DSSAT / CSM para simular la producción en el tiempo y el espacio para diferentes propósitos [36].

2.5.2. Modelos estocásticos.

2.5.2.1. Modelo espacio temporal de fusión de imágenes de índice de vegetación (STVIFM).

El STVIFM, desarrollado por Liao en 2017, constituye una herramienta de gran utilidad en la generación precisa de series temporales de índice de vegetación de diferencia normalizada (NDVI) con una resolución espacial de alta calidad. Este enfoque se basa en la combinación de imágenes provenientes de los sensores LANDSAT-8 y MODIS, lo que permite obtener resultados más precisos. A partir de esta serie temporal de NDVI, se deriva la curva fenológica del crecimiento de los cultivos. Esta curva fenológica-NDVI resultante se emplea como entrada en el modelo SAFY, junto con otros datos complementarios como un mapa de clasificación de cultivos, información sobre la fenología, así como registros meteorológicos diarios [37].

2.5.2.2. Modelo de regresión múltiple utilizando datos de imágenes hiperespectrales.

Aguate (2017) llevó a cabo un estudio sobre el uso de datos provenientes de imágenes hiperespectrales en un modelo de regresión múltiple. Los resultados indicaron que la aplicación de ecuaciones de predicción basadas en estos datos hiperespectrales puede generar pronósticos más precisos en cuanto al rendimiento de grano, en comparación con el empleo de índices de vegetación como variables predictoras. Dentro de las ecuaciones de predicción hiperespectrales, se exploraron tres métodos de estimación: mínimos cuadrados ordinarios (OLS), mínimos cuadrados parciales (PLS, una técnica de reducción de dimensionalidad) y un enfoque bayesiano que involucra la contracción y selección de variables (BayesB).

El estudio también analizó los beneficios derivados de la integración de datos de reflectancia recopilados en diferentes momentos durante la misma temporada de cultivo. Los resultados destacaron que esta metodología de combinación temporal mejoró la precisión de las predicciones en comparación con las aproximaciones basadas únicamente en índices de vegetación. En resumen, este estudio resalta el potencial de las imágenes hiperespectrales y sus diferentes métodos de análisis en la mejora de la predicción del rendimiento de grano en comparación con enfoques tradicionales basados en índices de vegetación [38].

2.5.2.3. Regresión lineal entre rendimientos de cultivos medidos a campo y productos derivados de información obtenida con el sensor MODIS.

Johnson [39] realizó una evaluación exploratoria para determinar la fuerza de correlación y sincronización óptima de varios productos de imágenes compuestas de MODIS de uso común versus los rendimientos de los 10 commodities más importantes del mundo. Los cultivos analizados incluyeron cebada, canola, maíz, algodón, papas, arroz, sorgo, soja, remolacha azucarera y trigo. Los datos de MODIS investigados incluyó el NDVI, fracción de radiación fotosintéticamente activa (FPAR), IAF y producción primaria bruta (GPP), además de la temperatura diurna de la superficie terrestre (DLST) y la LST nocturna (NLST). Todas las imágenes utilizadas tenían intervalos de tiempo de 8 días. El NDVI tenía una resolución espacial de 250 m mientras que los otros productos de 1000 m. Todos los cultivos, excepto el arroz, mostraron correlaciones positivas con al menos uno de los índices de vegetación calculados para la temporada de crecimiento del cultivo. El NDVI evidencio una correlación ligeramente mejor que FPAR.

2.5.3. Modelos mecanísticos y estocásticos: Integración utilizando datos de teledetección.

La combinación de modelos mecanísticos y estocásticos con datos de teledetección ha demostrado ser una estrategia de gran utilidad en la simulación del rendimiento agrícola a nivel regional, como ilustran diversos estudios de caso [40]. En este enfoque, los datos proporcionados por la teledetección se emplean para ajustar los parámetros de modelos basados en procesos, lo que conduce a una mejora en la precisión de las simulaciones. Sin embargo, las incertidumbres inherentes en factores como la distribución temporal de la lluvia, las propiedades del suelo y las prácticas agrícolas a nivel regional pueden introducir errores en los resultados de las simulaciones de cultivos.

Un ejemplo de este enfoque se encuentra en el trabajo de De Wit y Van Diepen (2007), quienes utilizaron el Ensemble Kalman Filter (EnKF) para incorporar datos de humedad del suelo obtenidos mediante microondas en el modelo de cultivo WOFOST. Esto permitió corregir las discrepancias en el balance hídrico del modelo [41]. En otro estudio, Li et al. (2014) implementaron el algoritmo EnKF para fusionar datos de IAF con un modelo de crecimiento de cultivos que incorporaba información hidrológica. Esta integración posibilitó obtener una representación espacial y variaciones regionales más precisas en el rendimiento del maíz [42]. Además, Inés et al. (2013) desarrollaron un marco de asimilación de datos llamado EnKF-DSSAT-CSM-Maiz, que combina la humedad del suelo derivada de sensoramiento remoto y datos de IAF provenientes de teledetección en un modelo de cultivo [43].

En resumen, a partir de estos ejemplos se distingue que los modelos estocásticos basados en series temporales de índices de vegetación ofrecen flexibilidad y viabilidad para aplicaciones en escalas regionales y a nivel de parcela. Por otro lado, los modelos basados en procesos, aunque requieren una mayor cantidad de información en su proceso de calibración, tienden a generar resultados más precisos.

2.5.4. Modelos Regresión Lineal Múltiple (RLM)

Los RLM son un enfoque estadístico que se emplea para modelar la relación entre una variable continua y una o más variables independientes. Este método busca ajustar una ecuación lineal que represente esta relación [44]. Específicamente, el modelo de regresión ajustado mediante el método de mínimos cuadrados ordinarios (OLS) es una técnica común utilizada para estimar los coeficientes de las ecuaciones de regresión lineal que describen cómo una o más variables cuantitativas independientes están relacionadas con una variable dependiente. Este enfoque ha sido ampliamente aplicado en diversos contextos, incluyendo la estimación de variables biofísicas a partir de datos relevados por sensores remotos [45].

2.5.4.1. Definición matemática

Faraway [46], referente en el campo de los modelos lineales, propone que para abordar teóricamente este modelo, consideremos un conjunto de datos que involucra un número determinado de variables independientes, denotado por p, y una variable de respuesta, llamada Y. Este conjunto de datos consta de "n"filas u observaciones. Se define, como: X (matriz de variables independientes) = una matriz de tamaño n por p donde X_{i_j} denota los valores de j-ésima característica para la i-ésima observación. Así que,

$$\begin{bmatrix} X_{1_1} & \dots & X_{1_p} \\ X_{2_1} & \dots & X_{2_p} \\ X_{n_1} & \dots & X_{n_p} \end{bmatrix}$$

Y que, Y (vector de respuesta) = un vector de tamaño n donde y_i denota el valor de respuesta para i-ésima observación.

Y =

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

La línea de regresión para las variables independientes (p) se representa como:

$$Y(x_i) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_i p \tag{2.1}$$

donde Y(x_i) es el valor de respuesta pronosticado para la i-ésima observación y β_0, β_1, β_2 son los coeficientes de regresión. Además, podemos escribir:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_i p + \varepsilon_i \tag{2.2}$$

donde ε_i representa el error residual en la i-ésima observación.

Podemos generalizar un poco más nuestro modelo lineal representando la matriz de variables independientes X como:

X=

$$\begin{bmatrix} 1 & X_{1_1} & \dots & X_{1_p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{2_1} & \dots & X_{2_p} \\ 1 & X_{n_1} & \dots & X_{n_p} \end{bmatrix}$$

Ahora, el modelo lineal se puede expresar en términos de matrices como:

Donde,

$$Y = X\beta + \varepsilon \tag{2.3}$$

Aquí, Y_i es el vector de respuesta, X_i es la matriz de variables independientes, β es el vector de coeficientes de regresión, y ε representa el vector de error. La estimación de los coeficientes β se obtiene utilizando el método de los mínimos cuadrados, que busca minimizar el error . Se puede demostrar que la estimación de β se expresa como:

$$\beta = (\hat{x}x)^{-1}\hat{x}y \tag{2.4}$$

donde $\hat{X}$ denota la transposición de la matriz -1 representa la matriz inversa.

Conociendo las estimaciones de mínimos cuadrados, β, el modelo de regresión lineal múltiple ahora se puede estimar como:

$$\hat{y} = x\beta \tag{2.5}$$

donde $\hat{y}$ es el vector de respuesta estimado.

2.5.4.2. Métricas para evaluación de capacidad predictiva

Asumamos que tenemos n observaciones y_i y que tenemos un estimador que estima los valores y^i.

El error cuadrático medio es:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.6}$$

Entonces el error cuadrático medio de la raíz es la raíz cuadrada, por lo tanto:

$$RMSE = \sqrt{MSE} \tag{2.7}$$

El R^2 es igual a:

$$R^2 = 1 - \frac{SSE}{TSS} \tag{2.8}$$

donde SSE es la suma de los errores al cuadrado (SSE por sus siglas en ingles, Sum of Squares Explained) o:

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2) \tag{2.9}$$

y por definición esto es igual a:

$$SSE = n \times MSE \tag{2.10}$$

El TSS suma total de cuadrados (TSS por sus siglas en ingles Total Sum of Squares y es igual a:

$$TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{2.11}$$

Donde

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2.12}$$

Entonces

$$R^2 = 1 - \frac{n \times MSE}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.13}$$

Para una regresión con intercepto, R^2 está entre 0 y 1, y por su definición.

$$R^2 = 1 - \frac{SSE}{TSS} \tag{2.14}$$

Por lo tanto:

$$\frac{SSE}{TSS} \tag{2.15}$$

es la suma de los errores al cuadrado dividida por la suma total de los cuadrados, por lo que es la fracción de la suma total de los cuadrados que está contenida en el término de error. Así que uno menos esto es la fracción de la suma total de cuadrados que no está en el error, o (R^2) es la fracción de la suma total de cuadrados que está 'explicada por' la regresión [44].

En resumen el RMSE es una medida de la desviación promedio de las estimaciones de los valores observados, mientras que el coeficiente de correlación de Pearson (R^2) es una medida de cuán asociadas se encuentran las variables entre sí.

Finalmente, el error porcentual absoluto medio, también conocido como desviación porcentual absoluta media (MAPE), es una métrica de evaluación para problemas de regresión. La idea de esta métrica es ser sensible a los errores relativos. Por ejemplo, no se cambia por una escala global de la variable de destino.

Si $\hat{y}_i$ es el valor pronosticado de la muestra i-ésima y_i es el valor verdadero correspondiente, entonces el error porcentual absoluto medio (MAPE) estimado se define como:

$$\text{MAPE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{|y_i - \hat{y}_i|}{\max(\varepsilon, |y_i|)} \tag{2.16}$$

donde ε es un número arbitrario pequeño pero estrictamente positivo para evitar resultados indefinidos cuando y es cero.

2.5.4.3. Supuestos que debe cumplir un modelo regresión lineal

Los modelos de regresión lineal se basan en supuestos fundamentales que deben cumplirse en el conjunto de datos en el que se aplican:

1. Relación Lineal: La relación entre la variable de respuesta y las variables independientes debe ser lineal, y esta suposición puede ser evaluada a través de análisis de correlación y gráficos de dispersión. Se emplean pruebas como la matriz de correlaciones y el gráfico de dispersión para validar esta relación.

2. Homocedasticidad y Normalidad de Errores: Se asume que los errores siguen una distribución normal y tienen varianza constante (homocedasticidad). Estas suposiciones se pueden evaluar mediante el test de Shapiro-Wilk para la normalidad de los errores y el gráfico de residuos frente a los valores predichos para la homogeneidad de varianzas.

3. No Correlación entre Variables Regresoras: Se supone que no existe correlación significativa entre las variables independientes. Esta suposición se verifica mediante pruebas de correlación, como el test de correlación de Spearman, que evalúan si existe una relación lineal entre las variables independientes.

La combinación de estas suposiciones y pruebas forma la base para validar la idoneidad de un modelo de regresión lineal y garantizar que sus resultados sean confiables y precisos.

2.5.5. Modelos Machine Learning

2.5.5.1. Inteligencia artificial

El ámbito de la Inteligencia Artificial (IA) se encuentra en el cruce de la ciencia informática y otras disciplinas científicas, y su objetivo es desarrollar sistemas capaces de emular la inteligencia o comportamiento humano, una tarea de gran complejidad. Dentro de este campo, destaca el Machine Learning, que ha ganado notoriedad en años recientes. Aunque enmarcado en la informática, se distingue por su enfoque diferenciado respecto a las técnicas tradicionales [47].

En la informática convencional, los algoritmos consisten en un conjunto explícito de instrucciones programadas para resolver problemas específicos. Por contraste, los algoritmos de Machine Learning habilitan a las computadoras para entrenar modelos a partir de datos de entrada y emplear análisis estadísticos para generar salidas. Así, las computadoras pueden crear modelos a partir de datos, lo cual es valioso para automatizar procesos de toma de decisiones basados en información de entrada. En esencia, el aprendizaje automático redefine el paradigma de la programación clásica [47].

En la agricultura, los métodos de aprendizaje automático más relevantes son las redes neuronales artificiales, las redes neuronales profundas y las máquinas de vectores de soporte [48]. Estos se han empleado extensivamente para modelar y predecir el rendimiento de cultivos [49]. En la teledetección, el método Random Forest se ha destacado como un algoritmo de regresión idóneo para fusionar múltiples variables explicativas y estimar parámetros como la biomasa del trigo [50], el IAF [51], el rendimiento de cultivos [52], [53], así como la evapotranspiración del maíz [53]. La principal ventaja del algoritmo Random Forest radica en su capacidad para modelar interacciones complejas entre datos sin hacer suposiciones acerca de la distribución de los mismos.

2.5.5.2. Random Forest Regression (RFR)

Un árbol de decisión individual es un algoritmo de inteligencia artificial que modela decisiones a través de una serie de bifurcaciones o nodos. Cada nodo representa una característica o atributo, y las ramas que emergen indican las posibles opciones basadas en ese atributo. Estos árboles capturan patrones y relaciones en los datos de entrenamiento, dividiendo los casos en subgrupos más homogéneos a medida que avanzan hacia los nodos terminales. Es decir un árbol de decisión organiza datos en un formato de árbol que representa una serie de decisiones basadas en características, lo que permite hacer predicciones o clasificaciones.

El enfoque del modelo RFR [54] se basa en una agrupación (ensemble) de árboles de decisión individuales (Figura 2.6), cada uno de los cuales es entrenado con una muestra aleatoria extraída de los datos de entrenamiento originales mediante el proceso de bootstrapping. Esto implica que cada árbol se forma con datos ligeramente diferentes. En cada árbol individual, las observaciones son distribuidas a lo largo de bifurcaciones (nodos) para construir la estructura del árbol hasta llegar a los nodos terminales. Para predecir una nueva observación, se combinan

las predicciones de todos los árboles individuales que componen el modelo. En aplicaciones de aprendizaje automático orientadas a la predicción, los RFR han demostrado obtener resultados comparables e incluso superiores en algunos casos a los obtenidos con regresiones lineales múltiples[55].

Para comprender el funcionamiento de los modelos RFR, es necesario tener previo conocimiento sobre los conceptos de ensamble y bagging.

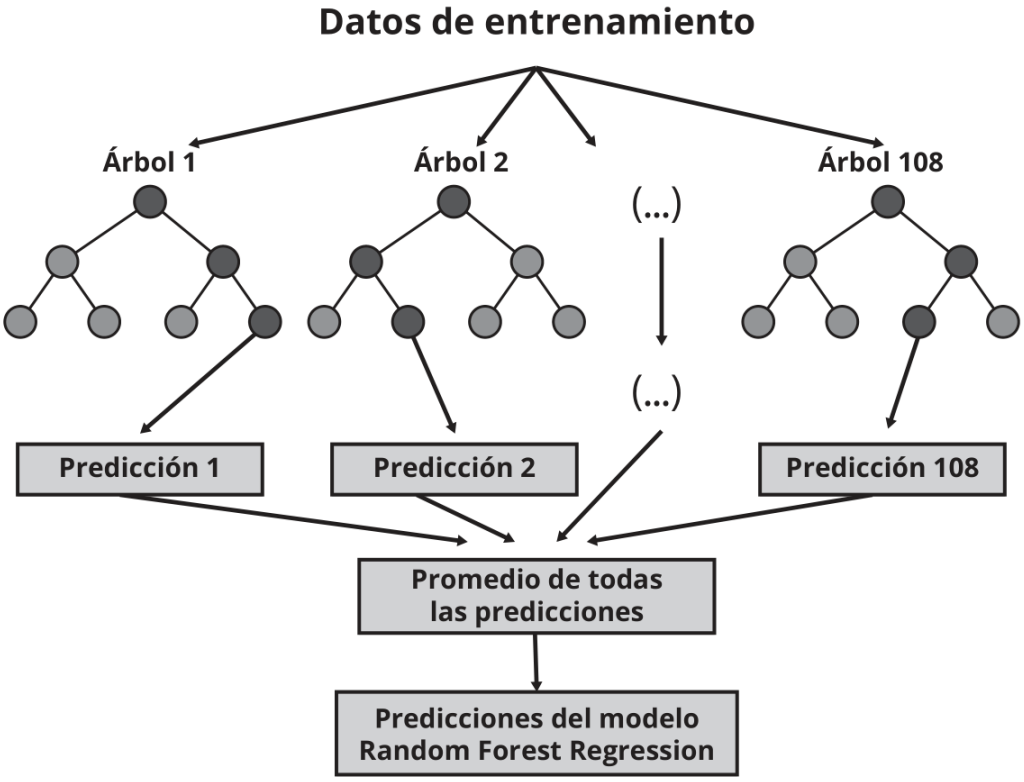

Figura 2.6: Diagrama de la estructura del modelo Random Forest Regression.

2.5.5.2.1. Métodos de ensamble En el ámbito del aprendizaje estadístico y machine learning, todos los modelos enfrentan el reto de balancear el sesgo y la varianza [47]. El término bias (sesgo) se refiere al grado en que las predicciones de un modelo se desvían, en promedio, de los valores reales. Refleja la capacidad del modelo para capturar la verdadera relación entre las variables predictoras y la variable de respuesta. Por ejemplo, si esta relación sigue un patrón no lineal, un modelo de regresión lineal no lograría representarla adecuadamente, resultando en un alto sesgo [47]. El término "varianza", por otro lado, se refiere a la magnitud del cambio en el modelo en función de las variaciones en los datos de entrenamiento [56]. Idealmente, un modelo no debería reaccionar drásticamente ante pequeñas fluctuaciones en los datos de entrenamiento; si lo hace, es posible que esté memorizando datos en lugar de aprender la relación real entre las variables predictoras y la variable de respuesta. Por ejemplo, un árbol de decisión con muchas ramas que cambia considerablemente su estructura ante leves variaciones en los datos, muestra una alta varianza.

A medida que la complejidad de un modelo crece, su flexibilidad para adaptarse a las observaciones aumenta, lo que disminuye el sesgo y mejora su capacidad predictiva. Sin embargo,

cuando la flexibilidad es excesiva, surge el problema del sobreajuste [47], en el cual el modelo se ajusta tanto a los datos de entrenamiento que no logra predecir nuevas observaciones con precisión. El modelo ideal logra un equilibrio óptimo entre sesgo y varianza.

Por lo general, los árboles de decisión pequeños (con pocas divisiones) presentan poca varianza, pero no representan de manera adecuada la relación entre las variables, resultando en un alto sesgo. Por otro lado, los árboles de decisión grandes se ajustan demasiado a los datos de entrenamiento, lo que reduce el sesgo pero aumenta la varianza. Para abordar esta problemática, se emplean los métodos de ensemble [47], que combinan múltiples modelos en uno nuevo con el propósito de alcanzar un equilibrio entre sesgo y varianza, resultando en predicciones más sólidas que las de los modelos individuales originales.

Existen dos categorías de métodos de ensemble: los de promediación y los de impulso.

1. En los métodos de promediación, se construyen varios estimadores de manera independiente y se promedian sus predicciones. En promedio, el estimador combinado suele ser superior a cualquiera de los estimadores individuales, debido a una menor varianza.

2. En los métodos de impulso, los estimadores se construyen secuencialmente, ajustando el sesgo del estimador combinado. La motivación radica en combinar varios modelos débiles para formar un conjunto poderoso [47].

En particular, el modelo de Random Forest emplea un método de promediación llamado Bagging, donde se ajustan múltiples modelos, cada uno con un subconjunto diferente de los datos de entrenamiento. En la etapa de predicción, todos los modelos contribuyen con sus predicciones, y el valor final se obtiene promediando las predicciones de todos los modelos [57].

Mientras que en los métodos de promediación, cada modelo difiere de los demás porque se entrena con una muestra distinta obtenida mediante el proceso de bootstrapping, en los métodos de impulso, los modelos se ajustan secuencialmente y la importancia (peso) de las observaciones cambia en cada iteración, resultando en ajustes diversos [57].

La clave para que los métodos de ensemble superen a los modelos individuales radica en que los modelos que los conforman sean lo más diversos posible, es decir, que sus errores no estén correlacionados.

2.5.5.2.2. Bagging El término *bagging* proviene de *bootstrap aggregation* y se refiere a la utilización de muestreo repetido con reposición (bootstrapping) para disminuir la varianza en modelos de aprendizaje estadístico, especialmente aquellos basados en árboles [57].

Si tenemos n observaciones independientes Z_1 , ... , Z_n , cada una con varianza σ^2 , la varianza de la media de las observaciones Z_x es σ^2/n . En otras palabras, al promediar un conjunto de observaciones, se disminuye la varianza. Basándose en este concepto, una manera de reducir la varianza y mejorar la precisión de un método predictivo es obtener múltiples muestras de la población, ajustar un modelo diferente a cada muestra y luego promediar (o calcular la moda en el caso de variables cualitativas) las predicciones resultantes. Dado que por lo general no se dispone de múltiples muestras, es posible simular este proceso mediante bootstrapping, generando pseudo-muestras con las cuales ajustar diversos modelos y luego combinarlos.

En el contexto de los árboles de decisión, que son propensos a alta varianza y bajo sesgo, el bagging ha demostrado excelentes resultados. El proceso se lleva a cabo de la siguiente manera:

1. Generar B conjuntos de entrenamiento pseudo-aleatorios mediante bootstrapping a partir del conjunto de entrenamiento original.

2. Entrenar un árbol para cada uno de los B conjuntos generados en el paso 1. Cada árbol se construye sin restricciones significativas y no se somete a poda, lo que resulta en alta varianza pero bajo sesgo. Por lo general, la única regla de parada es el número mínimo de observaciones que deben tener los nodos terminales. El valor óptimo de este hiperparámetro se puede determinar comparando el out of bag error o mediante validación cruzada.

3. Finalmente para cada nueva observación, obtener la predicción de todos los B árboles. La predicción final se obtiene promediando las predicciones de los B árboles en el caso de variables cuantitativas, o seleccionando la clase predicha más frecuente (moda) en el caso de variables cualitativas.

El número de árboles en el proceso de bagging no es crítico en términos de riesgo de sobreajuste. A medida que aumenta el número de árboles, la reducción del error de prueba se estabiliza. Sin embargo, cada árbol ocupa espacio en memoria, por lo que es importante no almacenar más árboles de los necesarios [57]

2.5.5.2.3. Entrenamiento de Random Forest El algoritmo de Random Forest es una modificación del proceso de bagging que permite mejoras notables al decorrelacionar aún más los árboles generados. Los beneficios de bagging se basan en la reducción de la varianza mediante el promedio de múltiples modelos, siempre y cuando estos modelos no estén altamente correlacionados. Si la correlación es alta, la reducción de varianza obtenida es limitada. En escenarios donde un solo predictor tiene una gran influencia junto a otros predictores moderadamente influyentes, la mayoría o todos los árboles generados en el proceso de bagging estarán dominados por ese predictor y serán similares entre sí [57]. Esta alta correlación limita la efectividad de bagging para disminuir la varianza y, por consiguiente, no mejora el modelo.

Random Forest supera esta limitación al introducir un paso adicional de selección aleatoria de m predictores antes de cada división en el árbol. Esto garantiza que un promedio de (p - m)/p divisiones no involucre el predictor influyente, permitiendo que otros predictores tengan la oportunidad de ser elegidos. Esta adición logra una mayor decorrelación entre los árboles, lo que conduce a una mayor reducción de la varianza al momento de combinarlos.

Los métodos de Random Forest y Bagging siguen esencialmente el mismo algoritmo, con la diferencia clave de que en Random Forest, se realiza una selección aleatoria de m predictores antes de cada división. El impacto en los resultados depende del valor específico de m elegido. La manera óptima de determinar m es evaluando el out-of-bag error o utilizando la validación cruzada. Similar a Bagging, Random Forest no sufre problemas de sobreajuste al aumentar el número de árboles en el proceso. Una vez que se alcanza un cierto número, la reducción del error en las pruebas se estabiliza en el proceso.

2.5.5.2.4. Predicción de Random Forest La predicción en un modelo de Random Forest es el resultado del promedio de las predicciones de todos los árboles que componen el conjunto [57].

2.5.5.2.5. Out of Bag Error Debido a la naturaleza del proceso de bagging, es posible estimar el error de prueba sin recurrir a métodos de validación cruzada. Dado que los árboles se ajustan usando muestras generadas mediante bootstrapping, en promedio, cada ajuste utiliza aproximadamente dos tercios de las observaciones originales. El tercio restante se denomina Out-of-bag (OOB) [57].

Registrando las observaciones usadas en cada árbol ajustado durante el proceso de bagging, se puede predecir la respuesta de la observación i utilizando los árboles en los que esa observación fue excluida, y luego promediando las predicciones. De esta manera, se pueden obtener predicciones para las "n.ºbservaciones y calcular el error cuadrático medio OOB (para regresión) o el error de clasificación OOB (para árboles de clasificación). Dado que cada observación solo se predice con los árboles en los que no participó en el ajuste, el error OOB sirve como estimación del error de prueba. Con un número suficiente de árboles, el error OOB es prácticamente equivalente al error de validación cruzada "leave-one-out"[57]. Esta característica es una ventaja adicional de los métodos de bagging, incluido Random Forest, ya que evita el costo computacional de la validación cruzada para la optimización de hiperparámetros. En un muestreo mediante bootstrapping, si el tamaño de los datos de entrenamiento es "n", cada observación tiene una probabilidad de ser seleccionada de 1/n. Por lo tanto, la probabilidad de no ser seleccionada en todo el proceso es de $(1-1/n)^n$, que tiende a converger en $1/\varepsilon$, aproximadamente un tercio. [57]

2.5.5.2.6. Importancia de los predictores Aunque es cierto que el proceso de bagging mejora la capacidad predictiva en comparación con modelos basados en un solo árbol, esto tiene un costo: la interpretabilidad del modelo disminuye. Al ser una combinación de múltiples árboles, no es posible representar gráficamente el modelo de manera sencilla ni identificar visualmente qué predictores son más importantes. Sin embargo, existen estrategias para cuantificar la importancia de los predictores, convirtiendo a los modelos de bagging en herramientas poderosas no solo para la predicción, sino también para el análisis exploratorio[57]. Las estrategias más utilizadas son la importancia por permutación y la impureza de los nodos.

2.5.5.2.7. Importancia por permutación Evalúa la influencia de cada predictor en una métrica específica de evaluación del modelo (calculada mediante el error OOB o la validación cruzada). El valor asignado a cada predictor se obtiene de la siguiente manera:

1. Se crea el conjunto de árboles del modelo.
2. Se calcula la métrica de error (RMSE). Este valor es la referencia (error 0).
3. Para cada predictor j:

I - Se permutan los valores del predictor j en todos los árboles del modelo, manteniendo el resto constante.
II - Se recalcula la métrica de error tras la permutación, llamémosla error j.
III - Se calcula el incremento en la métrica debido a la permutación del predictor j como porcentaje:

$$Incremento_j = \frac{error - error_j}{error_0} \times 100$$

Si el predictor permutado estaba contribuyendo al modelo, es probable que el modelo aumente su error, ya que se pierde la información proporcionada por esa variable. El porcentaje de au-

mento en el error debido a la permutación del predictor j puede interpretarse como su influencia en el modelo. Es importante mencionar que este incremento puede ser negativo. Si la variable no contribuye al modelo, es posible que, al reorganizarla aleatoriamente, por pura casualidad, se logre mejorar ligeramente el modelo, lo que resultaría en un valor negativo para (error - error j). Por lo general, se considera que las variables con incrementos cercanos a cero tienen una importancia limitada [57].

2.5.5.2.8. Incremento de la pureza de nodos Cuantifica el aumento total en la pureza de los nodos debido a las divisiones en las que participa el predictor (promedio de todos los árboles). Para calcularlo, se registra el descenso en la métrica utilizada como criterio de división (índice Gini, MSE, entropía, etc.) en cada división de los árboles. Para cada predictor, se calcula el descenso promedio en el conjunto de árboles del conjunto. Cuanto mayor sea este promedio, mayor será la contribución del predictor al modelo [57].

CAPÍTULO 3

Materiales y Métodos

3.1. Área de estudio

El área de estudio (Figura 3.1) se encuentra ubicada en la "Pampa Arenosa" o "Mar de Arena Pampeano, región que cubre unos 200.000 km2 en las provincias de Buenos Aires (O y centro), La Pampa (NE), Córdoba (S), San Luis (centro y S) y Santa Fe (SO). La altura sobre el nivel del mar varía entre 400 m al O y 30 m al E.

El clima de esta área es una variante del subtropical húmedo, también denominado templado, caracterizado por tener la estación más cálida coincidiendo con la más lluviosa. Los veranos son cálidos y los inviernos frescos y cambiantes, con una temperatura promedio anual de alrededor de 17°C. Se encuentra entre las isohietas de 750 mm (que marca la transición entre la pampa subhúmeda y la pampa húmeda) y 1200 mm (que separa la pampa húmeda de la pampa perihúmeda). Las precipitaciones presentan un comportamiento cíclico que se manifiesta en periodos con aumento o disminución de lluvias. [58]

Su suelo está compuesto principalmente por Molisoles y Hapludoles Énticos, de textura Franca Limosa y Gruesa, Mixta, térmica. Está compuesto esencialmente por materiales originarios de los suelos de origen eólico, con predominio de texturas franco arenosas y en menor proporción arenoso francas finas. El relieve se caracteriza por ser una extensa planicie arreica con una pendiente regional de 0,05 %, que comprende lomas onduladas y suavemente onduladas con gradiente regional suave hacia el sudeste. Como unidades menores se observan lomas medanosas, médanos algunos parcialmente fijados y hoyas medanosas originadas por acciones eólicas del pasado reciente. Estas formas menores normalmente están orientadas de noreste a sudoeste. La pendiente puede variar de 0 a 3 %.

Si bien históricamente la región fue dominada por pastizales nativos, se transformó poco a poco a una matriz mixta de pastizales nativos, cultivos anuales y pasturas a partir de finales del siglo XIX [59]. Sin embargo, desde la década de 1980, y especialmente en la última década, el remanente de pastizales y de praderas fue casi completamente convertido a la agricultura anual,

siendo la soja el cultivo dominante [60]. Actualmente tiene como principal actividad productiva la agrícola-ganadera.

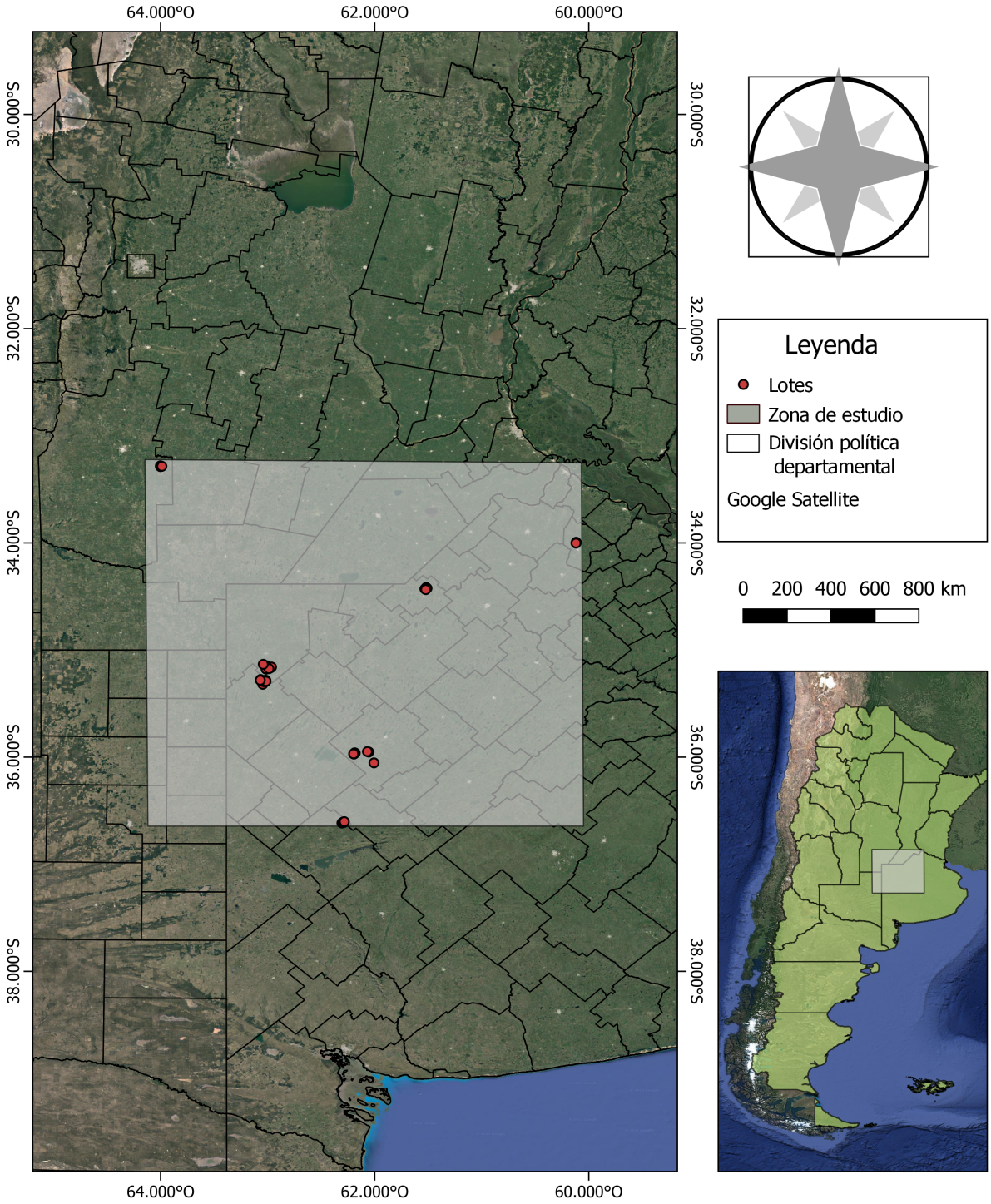

Figura 3.1: Área de estudio y lotes muestreados,en zona núcleo de la pampa argentina. Elaborado en Qgis 3.22.4 en base a imágenes Google Earth 2021 CNES / Airbus

La extensión y los límites del área de estudio se determinaron en función de la ubicación geográfica de los datos disponibles de 33 monitores de rendimiento, correspondientes a 27 lotes agrícolas pertenecientes a 8 establecimientos diferentes. (Figura 3.2 De estos, 7 están ubicados

en la Provincia de Buenos Aires y 1 en la Provincia de Córdoba. Para comprender la variabilidad de los suelos en los sitios estudiados, se identificaron las 7 series de suelos correspondientes a los lotes: Lincoln, La Albina, María Teresa, Piedritas, Bolívar, Cañada Seca y Olaeta.

Tabla 3.1: Resumen de la ubicación (localidad y provincia) de los 27 lotes, su superficie sembrada y la serie de suelo a la que pertenece cada uno.

Lote	Sup. Sembrada	Localidad	Provincia	Serie
1_A	52	Gral Villegas	BA	Lincoln
1_B	52	Gral Villegas	BA	Lincoln
1_C	207	Gral Villegas	BA	Lincoln
1_D	14	Gral Villegas	BA	Lincoln
1_E	68	Gral Villegas	BA	Lincoln
2_A	100	Pehuajó	BA	Piedritas
2_B	84	Pehuajó	BA	Bolívar
3_A	51	Daireaux	BA	Piedritas
3_B	70	Pehuajó	BA	Bolívar
3_C	115	Alem	BA	Bolívar
4_A	84	Gral Villegas	BA	Lincoln
4_B	98	Gral Villegas	BA	Lincoln
4_C	103	Gral Villegas	BA	Lincoln
4_D	77	Gral Villegas	BA	Cañada Seca
4_E	89	Pehuajó	BA	Lincoln
4_F	96	Gral Villegas	BA	Lincoln
5_A	247	Pehuajó	BA	Piedritas
6_A	117	Alem	BA	María Teresa
6_B	80	Alem	BA	La Albina
6_C	119	Daireaux	BA	La Albina
6_D	52	Alem	BA	María Teresa
6_E	94	Alem	BA	María Teresa
6_F	88	Daireaux	BA	María Teresa
7_A	63	Las Acequias	CBA	Olaeta
7_B	90	Las Acequias	CBA	Olaeta
7_C	76	Las Acequias	CBA	Olaeta
8_A	256	Pehuajó	BA	Piedritas

Estas series varían en su clasificación taxonómica, índice de productividad, limitaciones de uso, capacidad de uso, tipo de drenaje, tipo de paisaje, textura y pendiente (Tabla 3.2). Se observa que en general los suelos de las series son suelos pardos, profundos y de aptitud agrícola. En la seria La Albina se describe la posible presencia de un horizonte petrocálcico con escasas concreciones de Carbonato de Calcio (CAC03) entre los 80 y 100 cm de profundidad, no se identifica la presencia de Tosca en ninguna serie de los suelos estudiados. [61]

La superficie promedio de los lotes es de 96 hectáreas (Figura 3.2), con un lote con una superficie mínima de 14 hectáreas y un lote con una superficie máxima de 256 hectáreas.

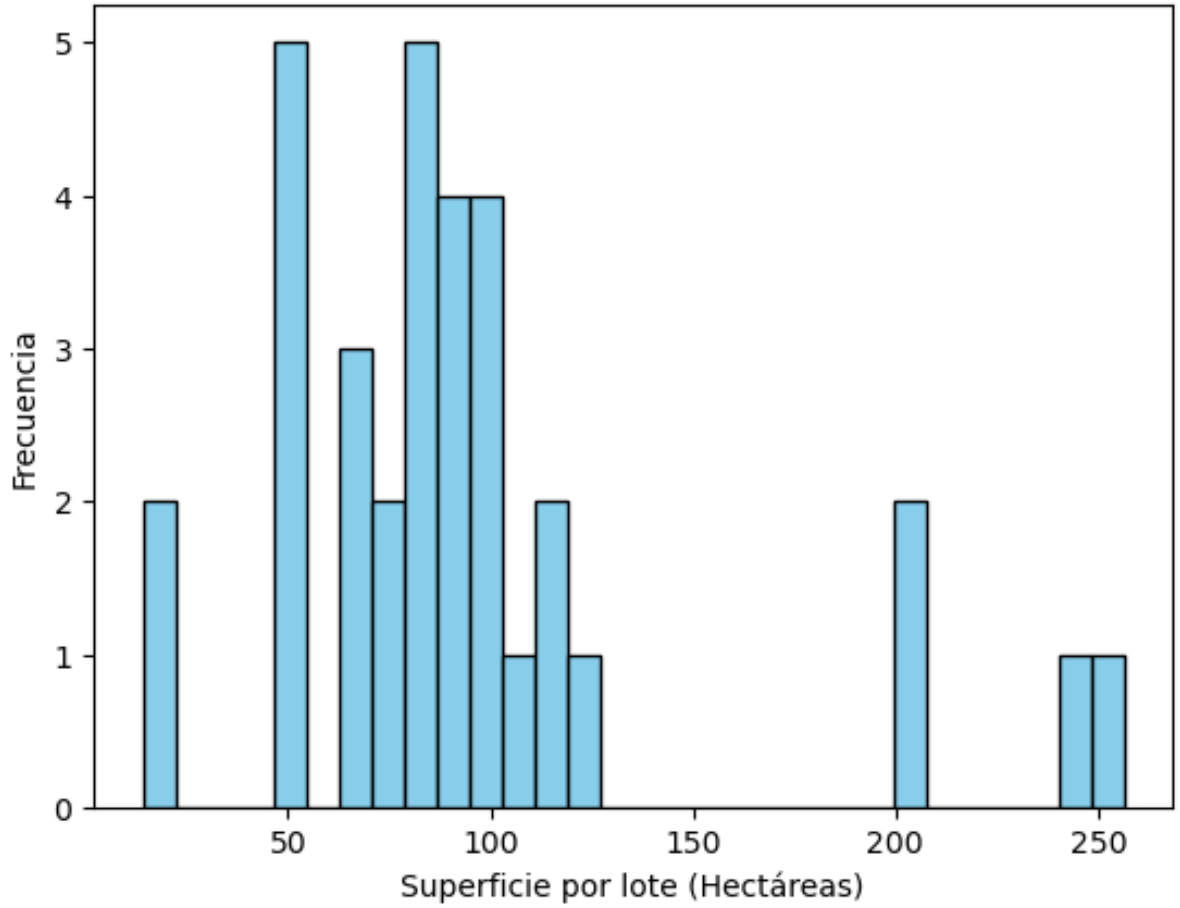

Figura 3.2: Histograma de distribución de lotes en función de su superficie

Tabla 3.2: Principales características de las series de suelo de los lotes estudiados. RH: Retención de humedad, EO: Erosión eólica, CIC: Capacidad de intercambio catiónico.

[61]

Serie	Clasificación Taxonómica	IP	Limitaciones de Uso	CU	Drenaje	Paisaje	Textura	Pendiente
Lincoln (Ln)	Hapludol típico	64.8	RH	IIs	Bien a algo excesivamente drenado	Paisaje suavemente ondulado	Franco a franco arcillo arenoso	0-1 %
La Albina (LAb)	Hapludol thapto nátrico	41.9	Drenaje y alcalinidad sódica	IVws	Algo pobremente drenado	Planicies suavemente onduladas	Franco arcillo arenoso	0-0.5 %
María Teresa (MT)	Hapludol típico, limosa	51	Drenaje, susceptibilidad a EO, baja CIC	IIIs	Algo excesivamente drenado	Medias lomas	Franco arenosa	0-1 %
Piedritas (Pas)	Hapludol éntico	58.4	Baja RH, leve susceptibilidad a la EO, baja CIC	IIIs	Algo excesivamente drenado	Planicie arenosa amplia	Franco arenoso grueso	0-1 %
Bolívar (Bv)	Hapludol éntico	47.2	Baja RH, leve susceptibilidad a la EO, baja CIC	IIIs	Algo excesivamente drenado a bien drenado	Lomas medanosas suavemente onduladas	Franco arenoso grueso	0-1 %
Cañada Seca (CSe)	Hapludol thapto árgico	52.6	Baja RH, baja CIC	IVs	Algo excesivamente drenado	Paisaje suavemente ondulado	Arenas finas de origen eólico	0-1 %
Olaeta (Ol)	Hapludol éntico	34	Baja RH, moderada susceptibilidad a EO, baja estabilidad estructural, baja CIC	IVes	Algo excesivamente drenado a bien drenado	Lomas medanosas onduladas	Franco arenosa	0-2 %

3.2. Flujo de trabajo

En la primera parte de la Figura 3.3 se muestra la sección donde se abordó la adquisición y el pre-procesamiento de las datos de campo y satelitales. La segunda parte muestra todos los pasos del procesamiento de datos para obtener los compuestos estacionales de los índices espectrales, variables ambientales y el dato de rendimiento por lote. La tercera parte indica todos los componentes del modelado, mientras que el cuarto se refiere a la evaluación de los modelos para identificar la precisión para obtener el resultado final (el último paso). Estimación de rendimiento y evaluación de resultados.

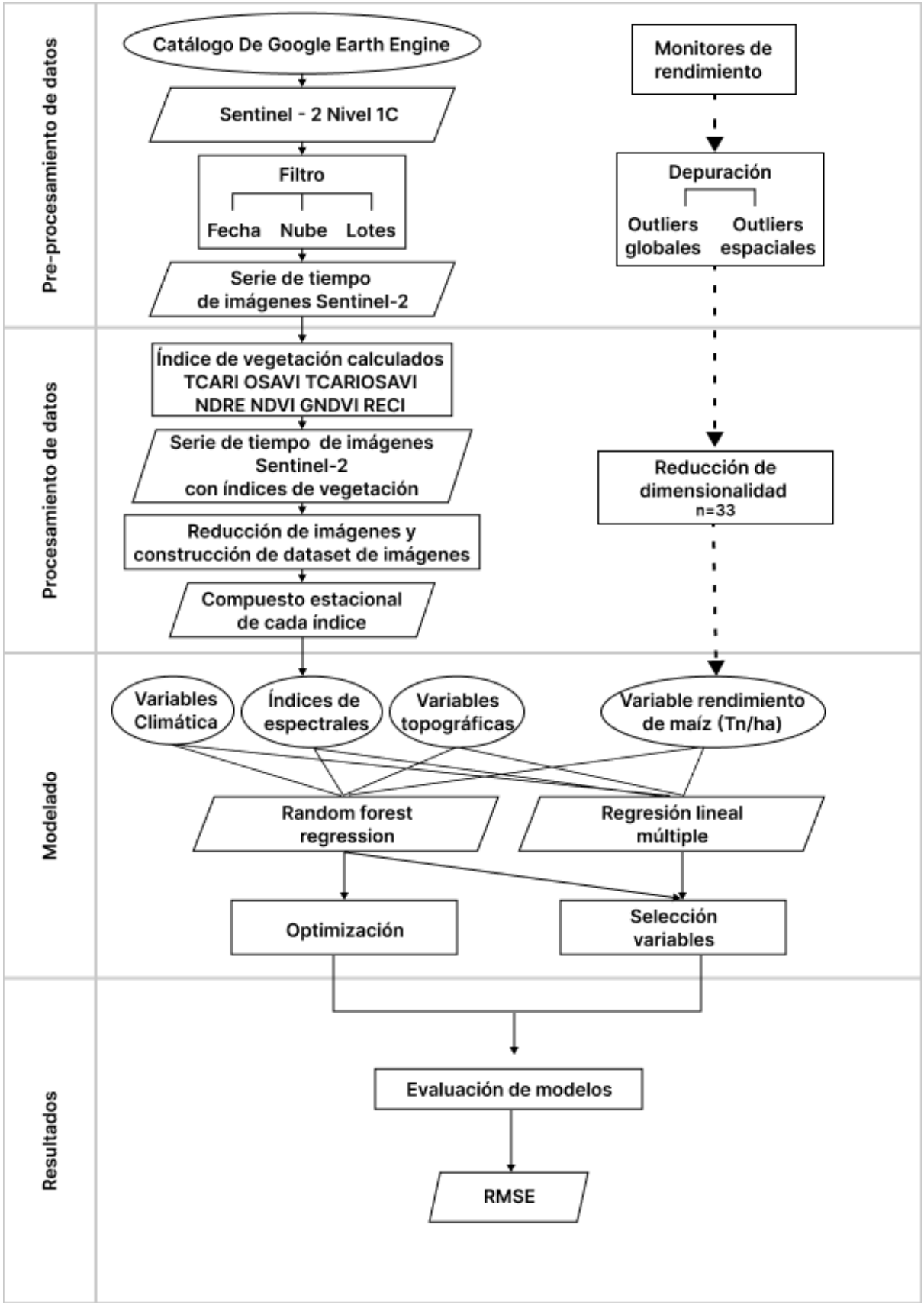

Figura 3.3: Diagrama de flujo de trabajo con la organización de las tareas y procedimientos llevados a cabo para la construcción de la base de datos y los ajustes y validación de los modelos. .

3.3. Observaciones de campo: Monitores de rendimiento

Los datos de rendimiento de Maíz fueron proporcionados por INVAP S.E en formato vectorial. Se pusieron a disposición 33 monitores de rendimiento, correspondientes a 3180 Hectáreas (Tabla 3.4), correspondientes a distintas campañas agrícolas, desde 2015 hasta 2021 (Figura 3.4) lo que permitió absorber la variabilidad climática de 5 años del área de estudio.

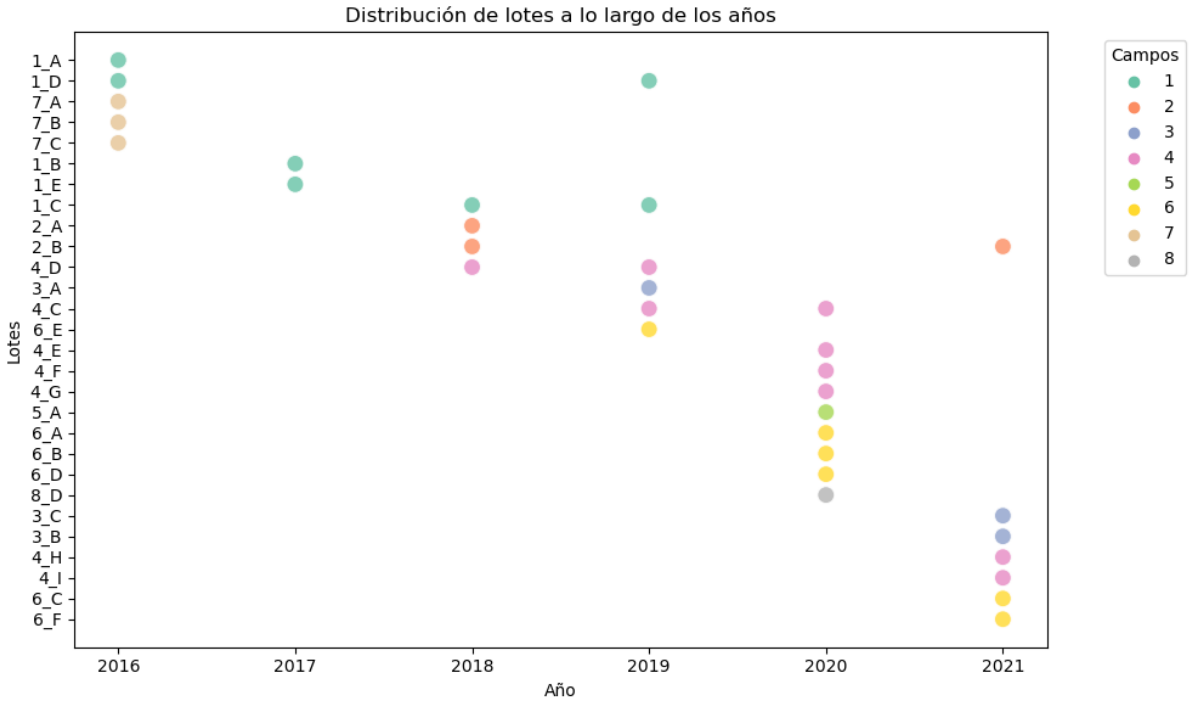

Figura 3.4: Distribución de los lotes a lo largo de los años y agrupados en colores por campo.

La distribución de los monitores no es consistente en el rango temporal (Figura 3.4) Hay solo 5 lotes que tienen datos de más de una campaña (1_D, 1_C, 2_B, 4_D, 4_C) mientras que la mayoría de los lotes solo cuenta con un dato por campaña. Los datos corresponden a 8 establecimientos distintos (Figura 3.5). El campo denominado 4 proporcionó la mayor cantidad de monitores de rendimiento: 8, mientras que los campos 5 y 8 sólo proporcionaron 1 monitor.

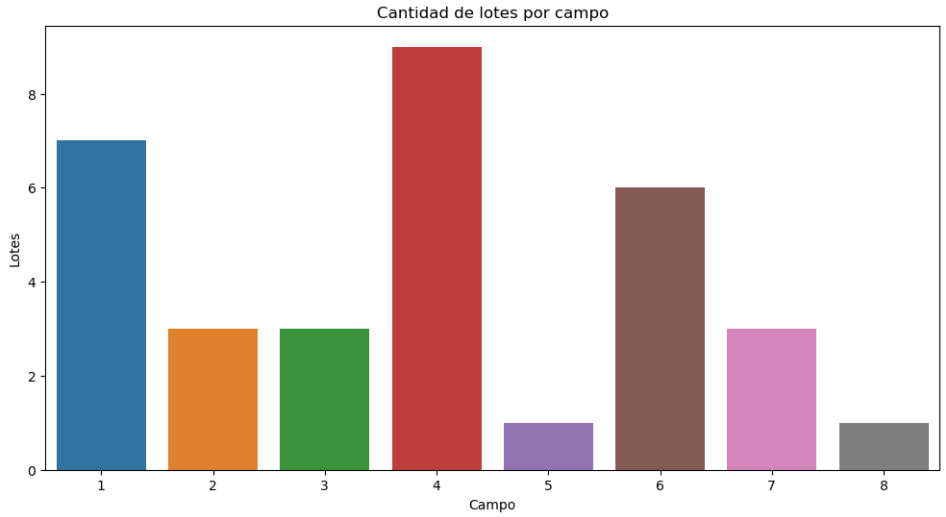

Figura 3.5: Los monitores de rendimiento corresponden a 8 establecimientos.

La disposición de los lotes en cada establecimiento se caracteriza por su proximidad cercana o contigüidad, lo que significa que están ubicados adyacentes unos a otros o en una proximidad inmediata, sin grandes distancias o separaciones entre ellos.

Los lotes se encuentran en las proximidades de 5 localidades (Figura 3.6), cuatro pertenecientes a la Provincia de Buenos Aires (General Villegas, Pehuajó, Daireaux, Alem) y un lote a la Provincia de Córdoba (Las Acequias).

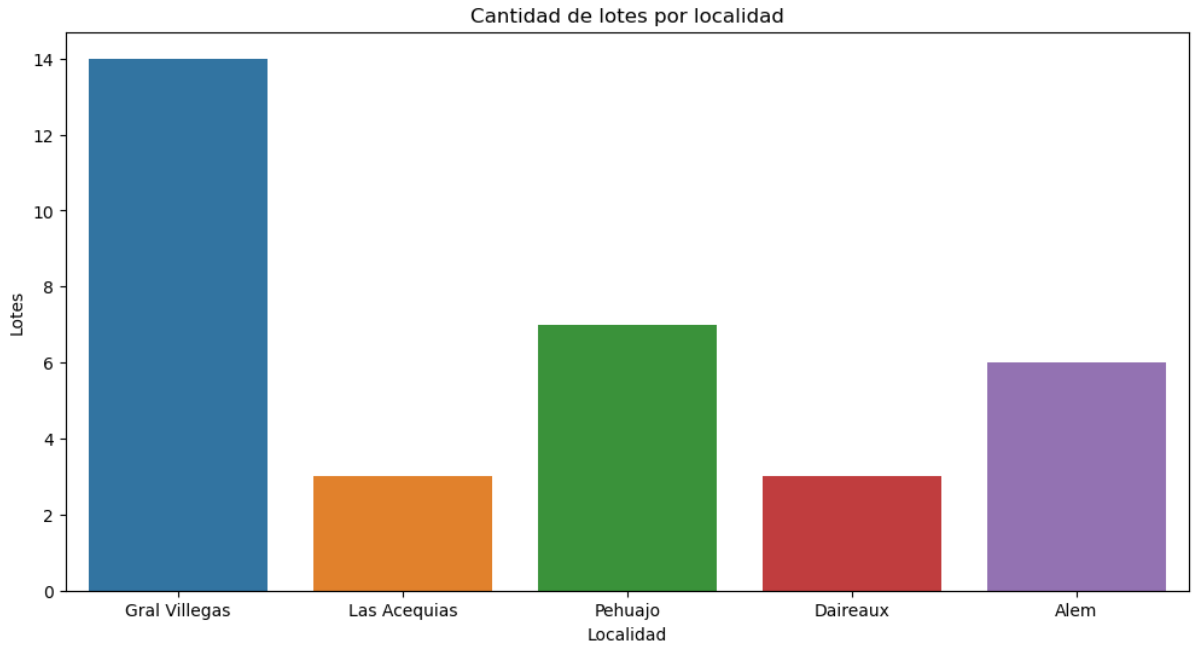

Figura 3.6: Cantidad de lotes agrupados a partir de la localidad más cercana.

La Figura 3.7 muestra una representación gráfica detallada de los datos de rendimiento en toneladas por hectárea (tn/ha) obtenidos de los monitores de los lotes adyacentes A, B y C pertenecientes al establecimiento 7. Esta representación proporciona una visualización clara y comparativa de los rendimientos entre los diferentes lotes colindantes dentro de dicho establecimiento.

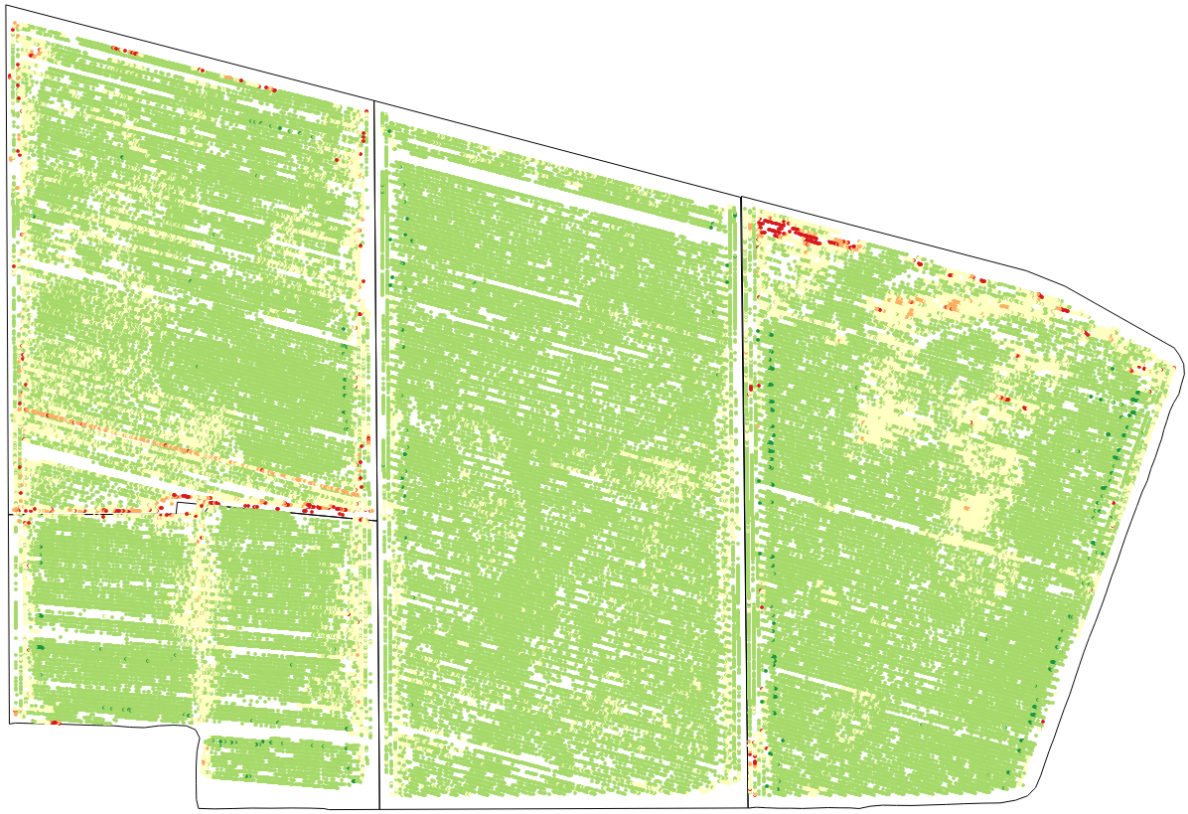

Figura 3.7: Representación de datos (tn/ha) de monitores de rendimiento de los lotes colindantes A, B y C pertenecientes al establecimiento 7.

3.3.1. Descripción de los datos

La generación de datos mediante el uso de monitores de rendimiento y otras maquinarias precisas usualmente trae aparejado la generación de numerosos datos espurios asociados a la técnica y procedimiento de medición. Consecuentemente, previo al análisis de los datos georreferenciados, es necesario implementar varios algoritmos de preprocesamiento como son aquellos orientados a la depuración o eliminación de valores aberrantes, la transformación e interpolación espacial de cada una de las capas de datos disponibles en sistema y escala común y la unión de la/las capas de información disponibles. Para el caso de datos provenientes de monitores de rendimiento en agricultura de precisión (AP), se han desarrollado protocolos tanto para identificarlos como para limpiarlos o depurar las bases de datos antes del análisis geoestadístico [62], [63].

Los principales causas de registros erróneos en el momento de la cosecha de cultivos de grano son reconocidas:

1. Dinámica de la cosechadora (tiempo de demora entre el corte y el registro del sensor de rendimientos, errores por diferencias entre el ancho realmente cosechado respecto del indicado en la cosechadora).
2. Variabilidad en las mediciones relacionadas a la heterogeneidad en la humedad del grano.
3. Falta de precisión del sistema de posicionamiento satelital.
4. Fallas de operación de la maquinaria (velocidad, giros en la cabecera con el cabezal bajo.)

[64].

Debido a lo cual, los datos erróneos deben eliminarse antes del estudio de variabilidad espacial. Los protocolos de depuración sustentados en criterios estadísticos para la identificación de outliers, protegen el sesgo y la precisión de los resultados del análisis estadístico y, por tanto, generan información más confiable para la toma de decisiones agronómicas. La interpolación espacial de los datos georreferenciados es atravesada por el concepto de correlación espacial. La autocorrelación espacial es la correlación de los valores de una misma variable entre sitios del dominio espacial y se expresa como función de la distancia que existe entre los sitios [65]. Es necesario recurrir a la modelación de este proceso estocástico espacial para poder generar las estimaciones en sitios donde el atributo de interés no ha sido medido [66].

3.3.2. Procesamiento de los datos de monitores de rendimiento

3.3.2.1. Protocolo para la eliminación automática de datos

Siguiendo el protocolo elaborado por [63] se llevó a cabo una evaluación secuencial de los conjuntos de datos de cada monitor de rendimiento con el objetivo de detectar y eliminar de manera automática diversos tipos de valores incorrectos. Este proceso se dividió en dos etapas:

En la primera etapa, se eliminaron los errores técnicos comunes que suelen encontrarse en los datos de rendimiento [67]. Esto incluyó la eliminación de puntos de datos con valores de rendimiento igual a cero, así como la corrección de efectos de borde y la identificación de valores atípicos globales.

En la segunda etapa, se procedió a eliminar pequeñas áreas o franjas con rendimientos extremadamente altos o bajos que no guardaban una relación cercana con sus vecinos inmediatos (valores atípicos espaciales).

Las estadísticas utilizadas en cada una de estas etapas se detallan a continuación.

1. Se aplicó un umbral sensible para restringir los datos, eliminando todos los valores de rendimiento iguales a cero.
2. Se procedió a la eliminación de todos los puntos de datos de los monitores de rendimiento ubicados a una distancia de 5 metros desde los bordes del campo. Esto tenía como objetivo mitigar los efectos de borde y los errores inherentes al monitoreo de rendimiento en la periferia del campo.
3. Se calculó la media y el desvío estándar (DE) de los datos, y se identificaron automáticamente los valores de rendimiento que se encontraban fuera del rango de la media más/menos tres veces el desvío estándar. Este filtro se aplicó al final de esta fase de limpieza de datos con el fin de evitar la inflación de la varianza causada por una estimación errónea de valores de rendimiento muy bajos.
4. Se empleó el Índice de Autocorrelación Espacial Local de Moran (IM) [68] como una herramienta para identificar los valores inusuales en los datos de rendimiento. El IM es un indicador que evalúa la agrupación espacial de valores similares o diferentes en un conjunto de datos. Un valor positivo del IM indica una tendencia a la agrupación de valores similares, mientras que uno negativo sugiere una agrupación de valores diferentes, como cuando un sitio con un bajo rendimiento está rodeado de vecinos con un alto rendimiento.

Para determinar la relevancia estadística de los valores de IM asociados con cada punto de datos de rendimiento, se utilizó un valor de p. Estos valores de probabilidad se ajustaron de acuerdo con el criterio de Bonferroni debido a la evaluación simultánea de múltiples índices en el campo. Este ajuste redujo el nivel de significancia a 0.05 para mantener la tasa de error del estudio en ese mismo valor [69].

El proceso se llevó a cabo de la siguiente manera:

1. En primer lugar, se identificaron los valores internos a eliminar, que correspondían a aquellos con valores negativos de IM que además eran estadísticamente significativos. Estos valores se consideraron atípicos y se marcaron para su exclusión.
2. Luego, se creó un diagrama de Moran, que es un gráfico de dispersión que muestra la relación entre los datos de rendimiento espacial y sus valores retrasados espacialmente, proporcionando un resumen de la medida de influencia para la regresión lineal.
3. Se aplicaron diversas medidas de influencia, como la distancia de Cook, el apalancamiento, DF-FITS, DFBETAS y COVRATIO [70], a la regresión lineal entre los datos de rendimiento y el rendimiento rezagado. Estas medidas ayudaron a identificar los datos influyentes en el diagrama de Moran.
4. Los datos influyentes detectados en el diagrama de Moran estaban relacionados con valores atípicos espaciales que no habían sido identificados por el IM, ya que estaban agrupados en lugar de estar aislados. Estos puntos con valores relativamente altos o bajos podían encontrarse no solo en los bordes del campo, sino también en su interior.

El diagrama de Moran se incorporó al protocolo para identificar de manera más exhaustiva los valores atípicos espaciales y garantizar la calidad de los datos de rendimiento.

$$LM_i = \frac{Z_i - \hat{Z}}{\sigma^2} \sum_{i=1, j\neq i}^{n} W_{ij}(Z_j - \hat{Z}) \tag{3.1}$$

donde $\hat{Z}$ es el valor medio de los rendimientos; Z_i es el valor de rendimiento en la ubicación i; Z_j es el valor de rendimiento en otros lugares ($j \neq i$); σ^2 es la varianza del rendimiento, n es el tamaño de la muestra; y W_{ij} es una distancia ponderada entre Z_i y Z_j. Para calcular W_{ij}, los puntos vecinos se definieron como contiguos puntos; por defecto se utilizaron puntos ubicados a una distancia ≤ 40 m. Este es un umbral común utilizado para procesar mapas de rendimiento de granos en la pampa argentina [71].

donde $\hat{Z}$ es el valor medio de los rendimientos; Z_i es el valor de rendimiento en la ubicación i; Z_j es el valor de rendimiento en otros lugares ($j \neq i$); σ^2 es la varianza del rendimiento, n es el tamaño de la muestra; y W_{ij} es una distancia ponderada entre Z_i y Z_j. Para calcular W_{ij}, los puntos vecinos se definieron como contiguos puntos; por defecto se utilizaron puntos ubicados a una distancia ≤ 40 m. Este es un umbral común utilizado para procesar mapas de rendimiento de granos en la pampa argentina [71].

Los resultados de este índice se usaron para etiquetar las observaciones periféricas como se describe a continuación: Un valor positivo del índice IM indica una agrupación espacial de valores similares, mientras que uno negativo sugiere una agrupación de valores diferentes, por ejemplo, un sitio con un valor de rendimiento bajo está rodeado de vecinos con valores de rendimiento altos.

La importancia estadística del valor de IM asociado con cada punto de datos de rendimiento se evalúa mediante un valor de p. Estos valores de probabilidad son ajustados por el criterio de Bonferroni debido a la multiplicidad de índices evaluados simultáneamente dentro del campo. Este ajuste desinfla el $\alpha = 0{,}05$ aplicado a cada prueba IM, por lo que la tasa de error de todo el estudio permanece en 0,05 [69].

3.3.2.2. Limpieza de datos

Los mapas de rendimiento son componentes clave de la agricultura de precisión debido a su utilidad tanto en el desarrollo como en la evaluación de estrategias de manejo de precisión. Sin embargo, los mapas de rendimiento brutos contienen una variedad de errores inherentes [72]. Los investigadores han informado que del 10 % al 50 % de las observaciones revelan errores de medición [73].

En la Tabla 3.3 se observa la cantidad de datos de rendimiento originalmente registrados en cada año y lote, así como la cantidad depurada después de aplicar el protocolo de limpieza de datos. Se calculó el porcentaje de datos descartados en el proceso de depuración, lo que refleja la importancia de este procedimiento en la obtención de datos confiables y precisos para su uso en la agricultura de precisión.

Resumen de las observaciones de campo a lo largo de varios años y en diferentes lotes agrícolas. En promedio se obtuvieron 57102 observaciones por lote. En promedio se depuraron 32 % de las observaciones de cada campo, obteniéndose un promedio de 39623 datos por campo. Este porcentaje de eliminación de errores en las observaciones está dentro del rango informado en la literatura [63], [73], [62] [74].

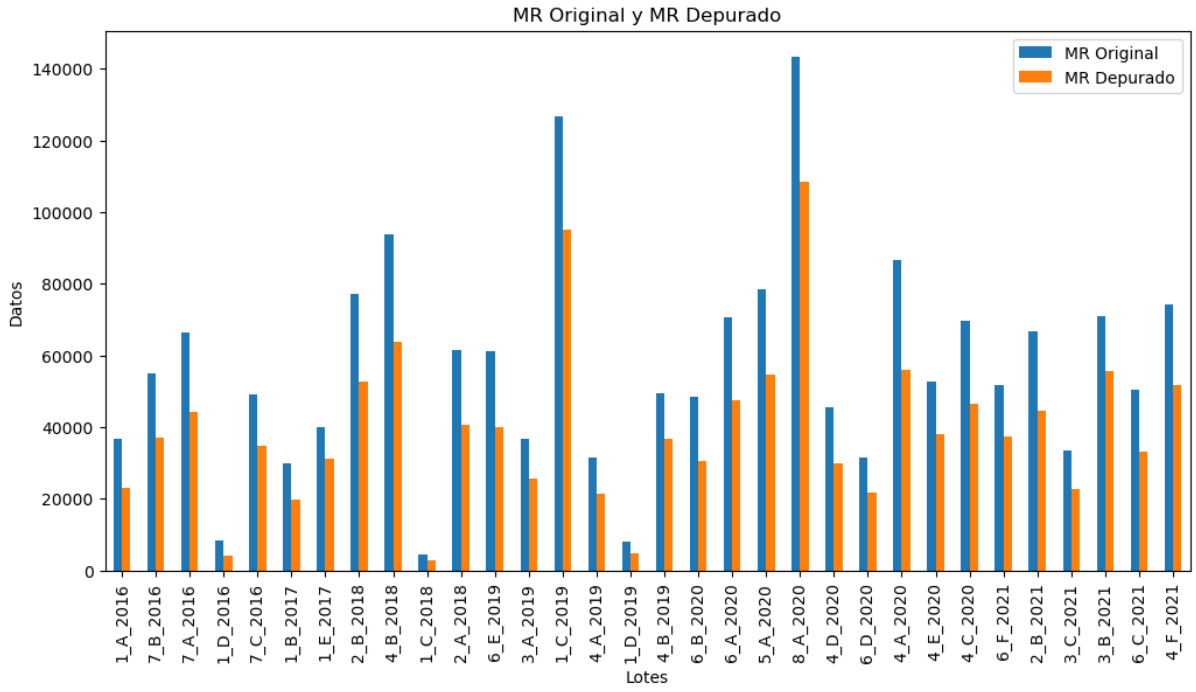

Figura 3.8: Comparación de la cantidad de datos de los monitores de rendimiento originales (barras azules) y los monitores depurados (barras anaranjadas) para los diferentes lotes a lo largo de los años.

Tabla 3.3: Observaciones de campo a lo largo de varios años y en diferentes lotes agrícolas.

Año	Lote	Observaciones de campo	Observaciones utilizadas	Observaciones descartadas (%)
2016	1_A	36772	23086	37
2016	1_D	8345	4226	49
2016	7_A	66527	44137	34
2016	7_B	55008	37034	33
2016	7_C	49068	34768	29
2017	1_B	29939	19838	34
2017	1_E	40073	31162	22
2018	1_C	4510	2704	40
2018	2_A	61410	40779	34
2018	2_B	77278	52671	32
2018	4_B	93780	63831	32
2019	1_C	126643	94937	25
2019	1_D	7986	4901	39
2019	3_A	36742	25630	30
2019	4_A	31440	21283	32
2019	4_B	49456	36860	25
2019	6_E	61193	40067	35
2020	4_A	86715	55916	36
2020	4_C	69505	46358	33
2020	4_D	45485	29985	34
2020	4_E	52767	37930	28
2020	5_A	78354	54650	30
2020	6_A	70575	47585	33
2020	6_B	48312	30598	37
2020	6_D	31396	21700	31
2020	8_A	143469	108290	25
2021	2_B	66828	44511	33
2021	3_B	70913	55571	22
2021	3_C	33483	22682	32
2021	4_F	92638	63691	31
2021	4_F	55568	39769	28
2021	6_C	50514	33137	34
2021	6_F	51663	37279	28
Promedio		57102	39623	32

En el primer paso del proceso de limpieza de datos, se eliminan los valores cero, ya que pueden interferir en la detección de valores atípicos globales. Según Leroux et al. [74], no se esperaría encontrar observaciones con un valor de rendimiento igual a cero, ya que esto podría haber ocurrido cuando se ajustó el umbral de corte después de la cosecha del cultivo. Los valores atípicos globales también pueden ser eliminados mediante algoritmos no paramétricos [75]. Es importante señalar que la presencia de rendimiento reducido en los bordes de los campos es un fenómeno agronómico legítimo y no debe considerarse un error de rendimiento. Los errores de mapeo de rendimiento generalmente ocurren en la cabecera del lote debido a que la cosechadora para poder efectuar el giro disminuye la velocidad y retrasa el mecanismo de llenado.

En la figura 3.9 se observa la eliminación de datos de los bordes y los errores debidos al inicio y al final de los errores del monitor de rendimiento de campo fue sustancial.

El efecto de borde es inherente a la forma del lote y puede establecerse en un valor mayor a los 5 m utilizados en este estudio. Sin embargo, para automatizar el protocolo de limpieza, se utilizó un valor bajo para ajustarse a la mayoría de los mapas de rendimiento.

El protocolo utilizado eliminó un gran número de observaciones, pero los tamaños de los mapas de rendimiento limpios aún eran suficientes para realizar las predicciones de rendimiento.

Es importante destacar que el filtro de borde también eliminó los puntos de datos recogidos durante el proceso de vaciado y llenado de la cosechadora al comienzo y al final de la cosecha. Sudduth y Drummond [73] argumentaron que la eliminación de estos puntos es esencial en el proceso de limpieza de los mapas de rendimiento.

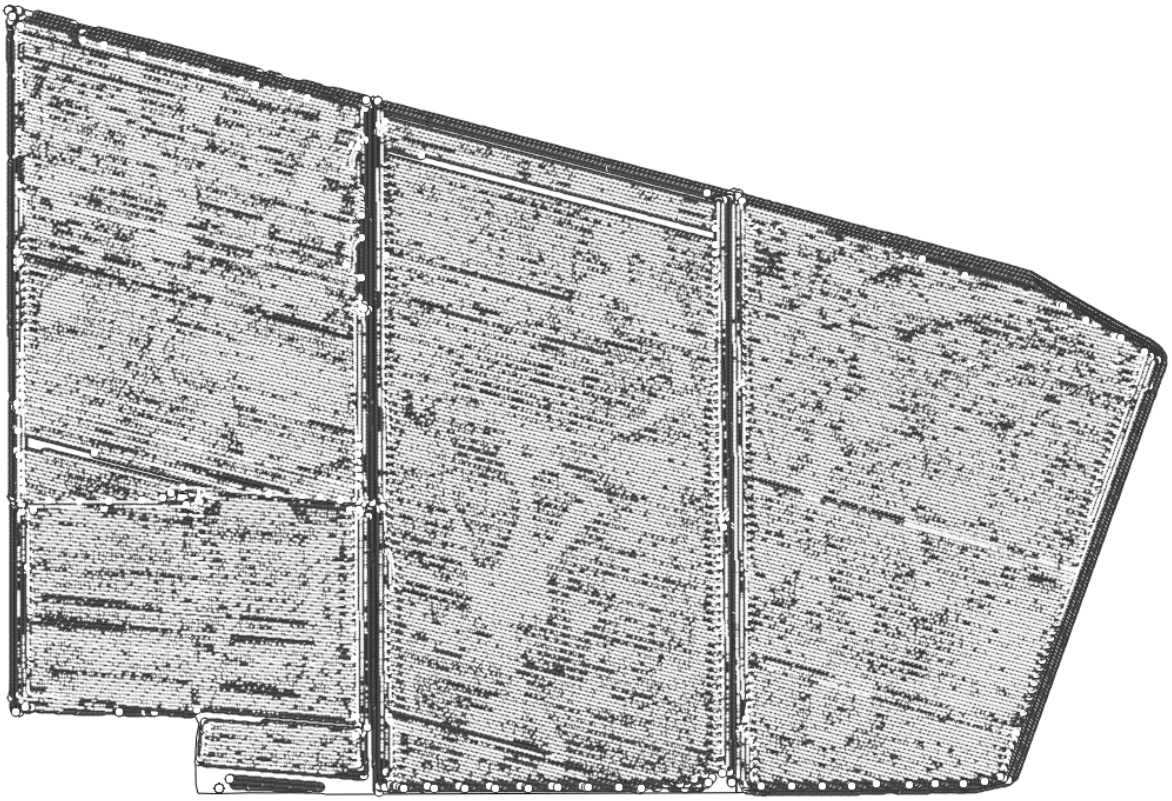

Figura 3.9: Los puntos negros son los datos originales, los puntos blancos son los datos que quedan luego del proceso de limpieza y filtrado.

Después de completar el proceso de limpieza de los datos obtenidos de los monitores de rendimiento, se calculó la variable dependiente, rendimiento de maíz a escala de lote, mediante la obtención de la media de los datos para cada lote. Finalmente, con el propósito de evaluar la representatividad de esta media, se calculó el desvío estándar y el rango correspondientes en cada lote.

El rendimiento medio en la muestra es de 9.28 toneladas por hectárea (Tn/ha), con un desvío estándar (DS) de 2.24 Tn/ha, un coeficiente de variación (58.5 %), un rango de 5.84 Tn/ha y una superficie sembrada total (SST) de 3180 hectáreas (Ha).

Tabla 3.4: Resumen de las observaciones de campo y sus principales estadísticos.

Cultivo	Obs.	Media	Mediana	Min	Max	DS	CV(%)	Rango	SST
Maíz	33	9.28	9.30	3.59	14.41	2.24	58.5	5.84	3180

3.4. Datos satelitales.

3.4.1. Rango temporal de datos

El periodo crítico del maíz, rango temporal donde se define el rendimiento, se ubica desde 20 días antes a 10 días después de la floración. La fecha de ocurrencia de la floración del maíz varía según las características climáticas, principalmente temperatura, de cada sitio y según las prácticas de manejo de siembra que se realice (siembra temprana o siembra tardía).

Para establecer el rango temporal se tomaron en cuenta las fechas de las tablas 3.5, 3.6 elaboradas por la oficina de riesgo agropecuario (ORA) correspondientes al periodo crítico de maíz para las zonas de estudio (Sur provincia de Córdoba, Oeste Provincia de Buenos Aires, Norte Provincia de Buenos Aires). Estas tablas están elaborada teniendo en cuenta el coeficiente de cultivo (KC) de maíz que es un valor que se utiliza en agricultura para estimar las necesidades de agua de un cultivo en diferentes etapas de crecimiento, contiene a su vez los días que pasan desde la siembra (DDS), la fecha en formato juliano y la fecha en formato "día-mes".

En la figura 3.10 se visualiza como la fecha de ocurrencia del período crítico del maíz varía según varían las características climáticas de la región donde se siembra y la modalidad de manejo de fecha de siembra que se emplea.

Tabla 3.5: confeccionada por ORA de datos de desarrollo de maíz en diferentes etapas de crecimiento y fechas de siembra en la región del en las regiones del oeste y norte de la Provincia de Buenos Aires.

Siembra Temp. - 1°Fecha		**O DE BS AS**			**N DE BS AS**		
ETAPA	**KC**	**DDS**	**JULIANO**	**FECHA**	**DDS**	**JULIANO**	**FECHA**
Siembra	0.40	0	275	1-oct	0	254	10-sept
Emergencia	0.50	12	287	13-oct	13	267	23-sept
Fin fase juvenil	0.80	45	320	15-nov	42	296	22-oct
Inicio floración	0.90	55	330	25-nov	47	301	27-oct
Floración (antesis)	1.20	88	363	28-dic	91	345	10-dic
Inicio llenado de granos	0.90	123	32	1-feb	103	357	22-dic
Madurez fisiológica	0.60	167	77	17-mar	144	32	1-feb
Cosecha	0.40	177	87	27-mar	152	40	9-feb
Siembra Temp. - 2°Fecha		**O DE BS AS**			**N DE BS AS**		
ETAPA	**KC**	**DDS**	**JULIANO**	**FECHA**	**DDS**	**JULIANO**	**FECHA**
Siembra	0.40	0	315	10-nov	0	284	10-oct
Emergencia	0.50	8	323	18-nov	9	293	19-oct
Fin fase juvenil	0.80	35	350	15-dic	32	316	11-nov
Inicio floración	0.90	55	330	25-dic	35	319	14-nov
Floración (antesis)	1.20	67	16	16-ene	66	350	15-dic
Inicio llenado de granos	0.90	100	49	18-feb	87	5	5-ene
Madurez fisiológica	0.60	157	107	16-abr	127	45	4-feb
Cosecha	0.40	167	117	26-abr	137	55	24-feb
Siembra Tardía		**O DE BS AS**			**N DE BS AS**		
ETAPA	**KC**	**DDS**	**JULIANO**	**FECHA**	**DDS**	**JULIANO**	**FECHA**
Siembra	0.40	0	345	10-dic	0	345	10-dic
Emergencia	0.50	7	352	17-dic	13	352	17-dic
Fin fase juvenil	0.80	39	18	18-ene	42	18	18-ene
Inicio floración	0.90	50	29	29-ene	47	29	29-ene
Floración (antesis)	1.20	72	51	20-feb	91	51	20-feb
Inicio llenado de granos	0.90	95	75	15-mar	103	75	15-mar
Madurez fisiológica	0.60	131	111	20-abr	144	111	20-abr
Cosecha	0.40	141	121	30-abr	152	121	30-abr

Tabla 3.6: confeccionada por ORA de datos de desarrollo de maíz en diferentes etapas de crecimiento y fechas de siembra en la región del centro de Córdoba.

Siembra Temprana		**CTRO DE CBA**		
ETAPA	**KC**	**DDS**	**JULIANO**	**FECHA**
Siembra	0.40	0	275	1-oct
Emergencia	0.50	8	287	13-oct
Fin fase juvenil	0.80	35	317	12-nov
Inicio floración	0.90	55	322	17-nov
Floración (antesis)	1.20	67	345	10-dic
Inicio llenado de granos	0.90	100	366	31-dic
Madurez fisiológica	0.60	157	59	28-feb
Cosecha	0.40	167	70	10-mar
Siembra Tardía		**CTRO DE CBA**		
ETAPA	**KC**	**DDS**	**JULIANO**	**FECHA**
Siembra	0.40	0	336	1-dic
Emergencia	0.50	7	342	7-dic
Fin fase juvenil	0.80	39	10	10-ene
Inicio floración	0.90	50	15	15-ene
Floración (antesis)	1.20	72	45	14-feb
Inicio llenado de granos	0.90	95	61	1-mar
Madurez fisiológica	0.60	131	81	21-mar
Cosecha	0.40	141	91	31-mar

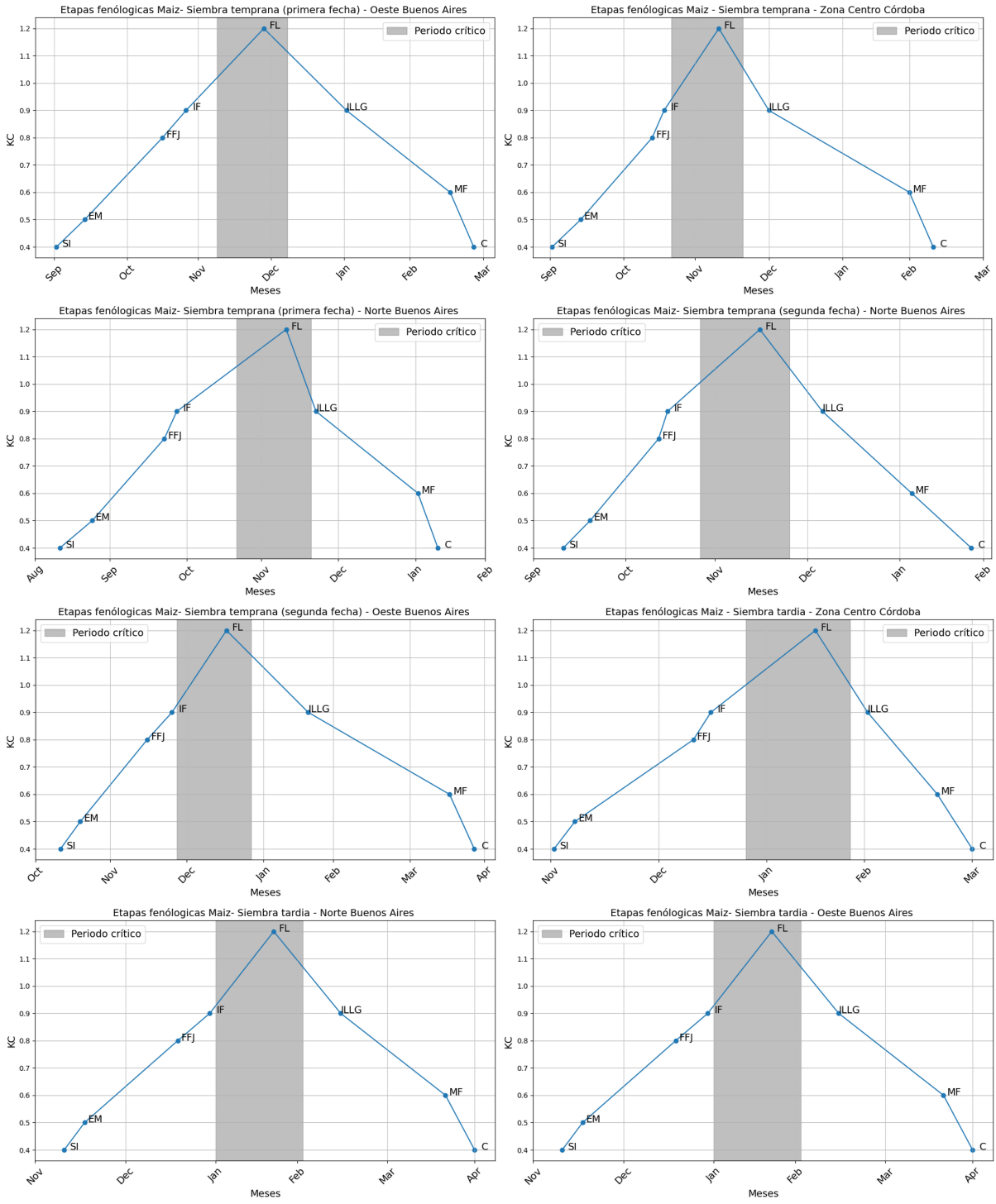

Figura 3.10: Elaboración propia a partir de los datos de la Oficina de riesgo agropecuario (ORA). Evolución del coeficiente de cultivo (KC) a lo largo de las etapas fenológicas de desarrollo de maiz: Siembra(SI), Emergencia(EM), Fin fase juvenil(FFJ), Inicio floración (IF), Floración (FL), Inicio llenado de granos (ILLG), Madurez fisiológica (MF), Cosecha (C) para las distintas zonas y modalidades de manejo.

Debido a la necesidad de contar con un único rango de tiempo que permita construir un dataset homogéneo de datos satelitales se estableció un rango único de fechas para todo el área de estudio.

Se estblecio una ventana temporal comenzando 20 días antes de la primer fecha de floración (10/12, modalidad siembra temprana en el centro de Córdoba y norte de Buenos Aires) finalizando 10 días después de la última floración (20/02, modalidad siembra tardía en el norte y el oeste de Buenos Aires), aplicable a las campañas agrícolas 2015/2016, 2016/207, 2017/2018, 2019/2020 y 2020/2021.

3.4.2. Selección y filtrado de imágenes

Se seleccionó la plataforma de sensoramiento remoto Sentinel-2 debido a sus mejores prestaciones en cuanto a la frecuencia de provisión de datos y su resolución espacial.

La plataforma de cálculo en la nube GEE permite acceder, procesar y descargar datos de sensoramiento remoto de manera práctica y eficiente. Se evitaron las descargas innecesarias de grandes volúmenes de datos utilizando su potencia de cómputo para calcular los índices espectrales que luego se descargaron. Se realizo el siguiente procedimiento:

1. Se accedió al dataset de Sentinel-2, del cual se seleccionaron las imágenes que presentaban un porcentaje de nubes por debajo de un umbral específico. El 'Cloud Pixel Percentage (CPP)' es un dato que proporciona el proveedor en el metadato de cada escena (tile). Se estableció un filtro de tolerancia con un máximo del 15 % de nubes en las imágenes utilizadas.

2. Se filtraron las imágenes según su ubicación geográfica utilizando el perímetro de cada lote para seleccionar y recortar las imágenes.

3. A partir de las imágenes seleccionadas y recortadas (entre 10 y 15 para cada sitio y año), se calculó el valor mediano de cada píxel en cada una de las bandas de interés. Esto se hizo con el objetivo de eliminar los valores extremos y reducir el ruido no deseado, como la neblina. De esta manera, se obtuvieron compuestos estacionales que se acercaban al "mejor"valor representativo de cada píxel durante el periodo crítico establecido [76].

4. A partir de los 33 compuestos estacionales, se calcularon los índices espectrales que se detallan en la siguiente sección.

5. Se descargaron los índices espectrales.

Para concluir el procedimiento, se empleó el plugin 'Estadísticas de Zona' en la versión estable 3.22.7 Białowieza de QGIS [77] para calcular las estadísticas, media, desvío estándar y el rango, para cada lote. Estas estadísticas se calcularon para cada uno de los índices espectrales previamente generados, con el objetivo de caracterizar la variabilidad de los índices espectrales y la pendiente dentro de cada lote.

La media proporciona un valor representativo, el desvío estándar indica la dispersión de los datos, y el rango señala la amplitud de las variaciones. Estas métricas permiten una evaluación más precisa y detallada de los datos, proporcionando información sobre la variabilidad de los índices.

3.4.3. Índices espectrales focalizados en tejidos vegetales

En esta sección se describen los índices espectrales seleccionados por su capacidad para identificar el tejido vegetal, caracterizar el ambiente y estimar el rendimiento del maíz.

3.4.3.1. Índice de Absorción de Clorofila en Reflectancia Transformado - TCARI

El MCARI (Índice de Absorción Modificado de Clorofila en Reflectancia) [78] es un índice que proporciona una medida de la profundidad de la absorción de clorofila por los tejidos vegetales y es altamente sensible a las variaciones en las concentraciones de clorofila, así como a las variaciones en el IAF. Los valores de MCARI (Ecuación 3.2) se ven afectados por las condiciones de iluminación, la reflectancia del fondo del suelo y otros materiales no fotosintéticos observados. Su principal limitación es su alta sensibilidad a los efectos de elementos no fotosintéticos, especialmente a bajas concentraciones de clorofila.

$$MCARI = (R_{700} - R_{670}) - 0{,}2(R_{700} - R_{550})\frac{R_{700}}{R_{670}} \tag{3.2}$$

Motivado por el potencial uso operativo del MCARI en el contexto de la agricultura de precisión, Haboudane [79] desarrolló una versión modificada mejorando su sensibilidad a valores bajos de clorofila, y lo denominó Índice de Absorción de Clorofila Transformada en Reflectancia (TCARI) (Ecuación 3.3). Además, Kim [80] demostró que el cambio en la reflectancia de fondo afecta la pendiente de reflectancia entre 550 y 700 nm. Las diferencias en la relación (R700/R550) están estrechamente relacionadas con las variaciones en las características de reflectancia de los materiales de fondo, como el suelo y los componentes no fotosintéticos. Para compensar estos efectos, se utiliza la relación (R700/R670) para contrarrestar la influencia del fondo solo en la diferencia (R700 - R550), de modo que el TCARI se define de la siguiente manera:

$$TCARI = 3(R_{700} - R_{670}) - 0{,}2(R_{700} - R_{550})\frac{R_{700}}{R_{670}} \tag{3.3}$$

A pesar de las mejoras observadas con respecto a los efectos de la biomasa no verde, este índice intrínseco aún es sensible a las propiedades subyacentes de la reflectancia del suelo, particularmente para los IAF bajos [81]. Para superar este problema, Daughtry [78] propusieron que MCARI se combinará con un índice de vegetación de línea de suelo como el Índice de Vegetación Ajustado al Suelo Optimizado (OSAVI; [81]. Tal integración reduce las contribuciones de reflectancia de fondo y mejorará la sensibilidad a la variabilidad del contenido de clorofila de la hoja.

3.4.3.2. Índice de Vegetación Ajustado al Suelo Optimizado - OSAVI

El OSAVI pertenece a la familia de Índices de Vegetación Ajustados al Suelo Index (SAVI) [82] y se define mediante la siguiente ecuación:

Este índice (Ecuación 3.4) se caracteriza por su fácil uso en el contexto de las observaciones operativas en los paisajes agrícolas. De hecho, su determinación no necesita información sobre

las propiedades ópticas del suelo y, además, ofrece los mejores resultados para la mayoría de los cultivos agrícolas (Rondeaux et al., 1996). Por otro lado, Haboudane [79] identificó que OSAVI tiene un comportamiento y tendencias similares al Índice de Vegetación Ajustado al Suelo Transformado (TSAVI); [83], cuya determinación requiere el conocimiento de los parámetros de la línea del suelo. Este último debe calcularse a partir de las áreas de la imagen correspondientes a suelos desnudos. Esto supone la presencia de tales áreas en la escena observada y depende de diferentes condiciones del suelo, como variaciones en los niveles de humedad, para establecer la línea del suelo mediante el trazado de la reflectancia del suelo en el espacio rojo versus infrarrojo. Estos cálculos adicionales (limitaciones) para TSAVI no son obligatorios para OSAVI, lo que hace que este último sea más adecuado para observar y monitorear los cambios de cultivos en el contexto de la agricultura de precisión.

La forma de reflectancia del dosel resulta de una interacción compleja entre las concentraciones de pigmentos, el desarrollo estructural del dosel y, en algunos aspectos, la contribución del suelo subyacente por lo cual una evaluación precisa del estado de la clorofila del cultivo a partir de datos de detección remota requiere índices espectrales que respondan a la concentración de clorofila, al IAF e insensibles a los efectos de fondo. Para este propósito, se ha demostrado que un uso combinado de TCARI y OSAVI tuvo éxito en la producción de isolíneas de las concentraciones de clorofila en las hojas [78].

$$OSAVI = \frac{(1+0{,}16)(R_{800}-R_{670})}{R_{800}+R_{670}+0{,}16} \tag{3.4}$$

3.4.3.3. TCARI/OSAVI

Haboudane [79] introdujeron el uso de la relación TCARI/OSAVI (Ecuación 3.5) para realizar predicciones precisas del contenido de clorofila y la actividad fotosintética de los cultivos a partir de imágenes hiperespectrales de teledetección. Demostraron que esta relación es relativamente insensible a las variaciones de la cobertura del dosel, incluso para valores IAF muy bajos. Este índice combina las capacidades de los índices que responden a las variaciones de clorofila (TCARI) y los que minimizan los efectos de fondo del suelo en la reflectancia del canopeo (OSAVI).

$$TCARIOSAVI = \frac{TCARI}{OSAVI} \tag{3.5}$$

3.4.3.4. Índice de borde rojo de diferencia normalizada - NDRE

El NDRE (Ecuación 3.6) es un índice de vegetación utilizado en sensores remotos para medir el contenido de clorofila en las plantas. Está representado por un determinado valor calculado mediante una combinación de una banda de infrarrojo cercano (NIR) y el rango RedEdge entre el rojo visible y el NIR. La banda del borde rojo es muy sensible a niveles medios a altos de contenido de clorofila. Por lo tanto, el borde rojo es un buen indicador de la salud del cultivo en cultivos de etapa media a tardía donde la concentración de clorofila es relativamente más alta.

$$NDRE = \frac{R_{665}-R_{705}}{R_{665}+R_{705}} \tag{3.6}$$

La banda de borde rojo es capaz de penetrar la hoja mejor que la banda roja que es absorbida por la clorofila en las primeras capas.

3.4.3.5. Índice de vegetación de diferencia normalizada - NDVI

NDVI (Ecuación 3.7) es uno de los índices más utilizados e implementados de los calculados a partir de información multiespectral [84]. Se calcula a partir de la relación normalizada entre las bandas roja e infrarroja cercana. Un uso directo del NDVI es para caracterizar el crecimiento o vigor del dosel; por lo tanto, es muchos estudios lo han comparado con el IAF. [26]

$$NDVI = \frac{R_{842} - R_{665}}{R_{842} + R_{665}} \tag{3.7}$$

3.4.3.6. Índice de vegetación de diferencia normalizada verde - GNDVI

El GNDVI (Ecuación 3.8) [85] a diferencia del NDVI usa el canal visible verde en lugar del rojo. GNDVI se desempeña mejor que el NDVI convencional para predecir IAF [85] por lo cual es ampliamente utilizado para estimar varias características agronómicas como el IAF en poroto seco [86], así como la biomasa de los brotes y el contenido de clorofila y nitrógeno en el trigo [87] y el maíz [88].

$$GNDVI = \frac{R_{842} - R_{560}}{R_{842} + R_{560}} \tag{3.8}$$

3.4.3.7. Índice de vegetación de clorofila de borde rojo - RECl

El índice de vegetación ReCI (Ecuación 3.9), también conocido como CIred-Edge, fue específicamente diseñado para responder al contenido de clorofila en las hojas [89]. El índice de clorofila de borde rojo se calcula usando una estrecha banda espectral entre el rojo y el infrarrojo cercano (NIR) del rango de reflectancia de la planta.

$$RECI = \frac{R_{842} - R_{705}}{-1} \tag{3.9}$$

Se considera un buen indicador de la actividad fotosintética de la cubierta del dosel ya que la banda de borde rojo es muy sensible a la luz reflejada por la estructura celular de una planta [90]. Una reflectancia más brillante corresponde a un mayor número de células vegetales y, por extrapolación, al área más verde.

Tabla 3.7: Ecuaciones de índices espectrales y bandas de Sentinel-2 utilizadas para cada longitud de onda

Variable	Ecuación
GNDVI	$B8 - B3/B8 + B3$
NDRE	$B4 - B5/B4 + B5$
NDVI	$B8 - B4/B8 + B4$
OSAVI	$\frac{(1+0,16)(B8-B4)}{B8+B4+0,16}$
RECI	$B8/B5 - 1$
TCARIOSAVI	$TCARI/OSAVI$
TCARI	$3(B5 - B4) - 0,2(B5 - B3)\left(\frac{B5}{B4}\right)$

3.4.4. Variables climáticas

3.4.4.1. Precipitación

Se utilizó la plataforma Google Earth Engine (GEE) [31] donde por medio de un código en Java-Script se puede acceder al producto estimación de precipitación multisatélite calibrado con manómetros (precipitationCal) puesto a disposición a una resolución espacial de 11,2 hectáreas (11132 metros cuadrados) por Global Precipitation Measurement (GPM) [91] cada 3 horas.

GPM es una misión satelital internacional que proporciona observaciones de lluvia y nieve en todo el mundo con alta frecuencia. "Integrated Multi-satellite Retrievals for GPM" (IMERG) es el algoritmo unificado que proporciona las estimaciones de lluvia. Este algoritmo está diseñado para intercalibrar, fusionar e interpolar todas las estimaciones de precipitación de microondas satelitales (GMS, MTSat, Himawari series, JMA, GOES series, NESDIS,Meteosat y EUMETSAT) junto con estimaciones satelitales de infrarrojos (NOAA, GOES, Meteosat, SNPP y Aqua AIRS), calibradas por microondas y análisis de pluviómetros sobre todo el globo. El sistema se ejecuta varias veces para cada tiempo de observación, dando primero una estimación rápida y, sucesivamente, proporcionando mejores estimaciones a medida que llegan más datos. El último paso utiliza datos de indicadores mensuales para crear productos de nivel de investigación. El sesgo en el acierto de IMERG está fuertemente relacionado con la intensidad de lluvia en las mediciones de pluviómetros, por lo que se recomienda su uso principalmente en zonas templadas [65].

En el presente estudio se calculó la suma de todas las mediciones disponibles entre las fechas correspondientes al periodo crítico de maíz desde la campaña 2015 a la campaña 2021 para cada lote bajo estudio.

3.4.4.2. Evapotranspiración y Temperatura

Desde la plataforma Google Earth Engine (GEE) [31] dataset ERA5-Land [92] perteneciente al Climate Data Store (CDS) [93]. Este es un conjunto de datos de reanálisis que proporciona una visión coherente de la evolución de las variables terrestres durante varias décadas con una resolución espacial de 11,1 hectáreas (11132 metros). ERA5-Land está producido reproduciendo el componente terrestre del reanálisis climático ECMWF-ERA5. El reanálisis combina datos de modelos con observaciones de todo el mundo en un conjunto de datos globalmente completo y consistente utilizando las leyes de la física. El reanálisis produce datos que se remontan varias décadas atrás en el tiempo, proporcionando una descripción precisa del clima del pasado. Entre numerosas variables, este dataset cuenta con información de temperatura y Evapotranspiración.

3.4.4.2.1. Evapotranspiración El producto "total_evaporation" [94] es el promedio mensual de metros de agua equivalente acumulada que se ha evaporado de la superficie de la Tierra, incluida una representación simplificada de la transpiración (de la vegetación), en forma de vapor en el aire de arriba. La convención del Sistema Integrado de Pronóstico del ECMWF es que los flujos descendentes son positivos. Por lo tanto, los valores negativos indican evaporación y los valores positivos indican condensación.

En el presente estudio se calculó la media de todas las mediciones mensuales disponibles entre las fechas correspondientes al periodo crítico de maíz desde la campaña 2015 a la campaña 2021 para cada lote bajo estudio.

3.4.4.2.2. Temperatura La teoría de similitud de Monin-Obukhov (M-O) [95] describe el flujo medio no dimensional y la temperatura media en la capa superficial en condiciones no neutrales como una función del parámetro de altura adimensional, llamado así por los científicos rusos A. S. Monin y A. M. Obukhov . Este flujo medio o difusión turbulenta es el intercambio vertical de calor, cantidad de movimiento y humedad a través de la atmósfera. En el modelo ECMWF-IFS (Integrated Forecast System) [96] la temperatura superficial se calcula interpolando entre el nivel más bajo del modelo (MO) y la superficie de la Tierra, teniendo en cuenta las condiciones atmosféricas. El producto "temperature_2m" es la temperatura del aire a 2 m sobre la superficie de la tierra. La temperatura medida en Kelvin fue convertida a grados Celsius (°C) restando 273,15.

Para el presente estudio se calculó la media de todas las mediciones disponibles entre las fechas correspondientes al periodo crítico de maíz desde la campaña 2015 a la campaña 2021 para cada lote bajo estudio.

3.4.4.3. Variables topográficas.

La Agencia de Exploración Aeroespacial de Japón (JAXA) lleva adelante desde 2014 el proyecto ALOS World 3D con el objetivo de cubrir las áreas terrestres globales mediante el uso de 3 millones de archivos de escenas adquiridos por el sensor de mapeo estéreo pancromático PRISM montado en la plataforma Advanced Land Satélite de observación "DAICHI"(ALOS) operado desde el año 2006 hasta el año 2011. Los mapas 3D digitales desarrollados consisten en el modelo de elevación digital (DEM) o el modelo de superficie digital (DSM) [97] que pueden representar terrenos terrestres con 5 metros de resolución espacial y 5 metros de precisión

de altura (desviación estándar) e imágenes de aspecto nadir PRISM ortorrectificado. Los mapas 3D digitales se han utilizado en una amplia variedad de aplicaciones, como el desarrollo de mapas, la predicción de daños por desastres naturales y la investigación de recursos hídricos.

3.4.4.3.1. Altitud El Delta Surface Fill (DSF) es un método de relleno de datos faltantes que reemplaza los valores en el DEM original con los valores ajustados calculados a partir de los píxeles válidos circundantes en otro DSM de referencia. Mediante el uso de la diferencia de valores de altura válidos en el DEM original y de referencia, este proceso garantiza la continuidad uniforme de la topografía en los límites de los rellenos vacíos.

Utilizando el método Delta Surface Fill (DSF) la JAXA puso a disposición con acceso irrestricto el conjunto de datos del modelo de superficie digital global (DSM) [97] con una resolución horizontal de malla de aproximadamente 30 metros (1 arcosegundo de latitud y longitud).

Desde la plataforma GEE se accedió al producto DSM y se calculo la media de la altitud para cada lote bajo estudio.

3.4.4.3.2. Pendiente A diferencia de la mayoría de los DEM de acceso gratuito, en la colección DSM antes mencionada, debido a que los datos no permiten realizar un único mosaico de toda la superficie global se precisa realizar una reproyección para realizar los cálculos de pendiente. Luego de reproyectar el dataset a el sistema de coordenadas de latitud/longitud basado en el centro de masa de la Tierra denominado EPSG:4326 - WGS 84, se utilizó la librería ee.Terrain disponible en el editor de código de GEE para calcular la pendiente media para cada lote bajo estudio.

Tabla 3.8: Variables utilizadas para ajustar los modelos. La variable dependiente es rendimiento de maíz, mientras que las variables predictoras son, por un lado, la media (μ), el desvío estándar (σ) y el rango (r) de los índices espectrales y la pendiente de cada lote y, por otro, el mapa de superficie digital y las variables climáticas. Todas las variables están medidas a escala de lote.

Tipo de variables	Variables	Símbolo	Código
Datos de campo	Rendimiento de maíz	RENDμ REND_σ REND_r	REND_mean REND_stde REND_range
Índices espectrales	Índice de vegetación de diferencia normalizada verde	GNDVIμ GNDVI_σ GNDVI_r	GNDVI_mean GNDVI_stde GNDVI_range
	Índice de borde rojo de diferencia normalizada	NDREμ NDRE_σ NDRE_r	NDRE_mean NDRE_stde NDRE_range
	Índice de vegetación de diferencia normalizada	NDVIμ NDVI_σ NDVI_r	NDVI_mean NDVI_stde NDVI_range
	Índice de Vegetación Ajustado al Suelo Optimizado	OSAVIμ OSAVI_σ OSAVI_r	OSAVI_mean OSAVI_stde OSAVI_range
	Índice de vegetación de clorofila de borde rojo	RECIμ RECI_σ RECI_r	RECI_mean RECI_stde RECI_range
	Índice de Absorción de Clorofila en Reflectancia Transformado y de Vegetación Ajustado al Suelo Optimizado	TCARIOSAVIμ TCARIOSAVI_σ TCARIOSAVI_r	TCARIOSAVI_mean TCARIOSAVI_stde TCARIOSAVI_range
	Índice de Absorción de Clorofila en Reflectancia Transformado	TCARI_μ TCARI_σ TCARI_r	TCARI_mean TCARI_stde TCARI_range
Topográficas	Pendiente	SLOPE_μ SLOPE_σ SLOPE_r	SLOPE_mean SLOPE_stde SLOPE_range
	Mapa de superficie digital	DSM	DSM
Climáticas	Precipitaciones acumuladas Temperatura Evapotranspiración total	PP TEMP EVAPO	PP TEMP EVAPO

3.5. Modelos de estimación de rendimiento

Como se mencionó en el marco teórico, los modelos mecanísticos requieren una gran cantidad de datos de campo específicos. Para este estudio, la única variable de campo con la que se contó en los lotes estudiados, fue la variable de interés a estimar: Rendimiento de maíz en toneladas por hectárea. Al no contar con otras variables mecanisticas el trabajo de investigación se centró exclusivamente en modelos estocásticos.

Se desarrollaron dos enfoques de modelado estocástico para estimar el rendimiento del maíz por lote:

1. En primer lugar, se ajustó un modelo de regresión lineal múltiple. Este modelo estadístico busca establecer una relación lineal entre una variable dependiente y las demás variables independientes. En este contexto, la variable dependiente, es decir la variable que se busca estimar, es el rendimiento del maíz por lote. Las variables independientes son los índices espectrales, las variables climáticas y las topográficas.
2. En segundo lugar, se ajustó un modelo de random forest regression. En términos generales, esta técnica de aprendizaje automático se basa en la construcción de múltiples árboles de decisión y su combinación para obtener predicciones más robustas y precisas. En este caso, se utilizó el RFR para abordar la complejidad inherente a la agricultura, donde múltiples factores interactúan de manera no lineal para influir en el rendimiento del maíz.

En las siguientes secciones, se profundizará en la descripción del desarrollo de estos dos enfoques de modelado.

3.5.1. Modelo Regresión Lineal Múltiple - RLM

3.5.1.1. Construcción

El modelo de regresión lineal múltiple se construyó en Python utilizando un entorno de Jupyter Notebook [98] proporcionado por Anaconda [99]. Se utilizó la librería Statsmodels [100], que facilita la implementación de modelos estadísticos. Para implementar la regresión lineal múltiple (RLM), se dividió el conjunto de datos de la variable dependiente, es decir el rendimiento de maíz por lote, en dos subconjuntos: datos de entrenamiento (70 %) y datos de prueba (30 %).

Para realizar el acondicionamiento de los datos se utilizaron las siguientes librerías:

1. numpy: Significa Python numérico y es una librería de Python diseñada para el cálculo y el procesamiento de matrices multidimensionales y unidimensionales [101].
2. pandas: Es una librería que proporciona herramientas de manipulación de datos de alto rendimiento [102].
3. Matplotlib: Es una librería utilizada para la visualización de datos y se enfoca principalmente en gráficos básicos. Permite la creación de gráficos de barras, gráficos de torta, gráficos de líneas y diagramas de dispersión [103].
4. Seaborn: Es una librería utilizada para crear gráficos estadísticos a partir de conjuntos de datos. Ofrece una variedad de patrones de visualización y utiliza una sintaxis más simple

que Matplotlib. Se utiliza para resumir datos y mostrar la distribución de los mismos [104].

El ajuste del modelo se realizo siguiendo el siguiente procedimiento:

1. En primer lugar se realizo un estudio de las relaciones individuales que existe entre las distintas variables. Esta información es crítica a la hora de identificar cuáles pueden ser los mejores predictores para el modelo.
2. En segundo lugar, mediante histogramas, se estudió la distribución de cada variable predictora y de la variable a predecir.
3. En tercer lugar se realizo el primer ajuste del RLM mediante una regresión iterativa de mínimos cuadrados ordinarios (OLS) sin tener en cuenta las correlaciones entre las variables.
4. Por último se realizo el nuevo ajuste del RLM teniendo en cuenta las interacciones entre las variables identificadas como las más influyentes.

Para poder ajustar el modelo lineal múltiple primero se realizo un estudio de la relación que existe entre las variables. Esta información es crítica a la hora de identificar cuáles pueden ser los mejores predictores para el modelo. Asimismo se estudió la distribución de cada variable mediante histogramas.

Se realizó una regresión iterativa de mínimos cuadrados ordinarios (OLS) para desarrollar dos modelos, en primer lugar un modelo sin tener en cuenta las interacciones entre las variables y un segundo modelos teniendo en cuenta las interacciones entre las variables más influyentes.

3.5.1.2. Evaluación

En la evaluación del modelo, se examinó en primer lugar la significancia de los resultados utilizando las métricas de bondad de ajuste [46].

Luego se realizo un diagnóstico de los residuos del modelo por intermedio de una inspección visual con el objetivo de corroborar que se comportan de manera aleatoria e independiente, lo cual es un indicio de que no hay una estructura sistemática en los datos que el modelo no haya capturado.

El procedimiento consistió en corroborar los cuatro supuestos:

1. Linealidad: Para verificar la linealidad de los residuos se utiliza el gráfico: Valor Predicho vs Valor Real. Si los puntos se distribuyen alrededor de la línea diagonal (línea punteada de referencia), esto sugiere que la relación entre las variables independientes y la variable dependiente es aproximadamente lineal. Si se observan patrones curvos o no lineales en este gráfico, podría indicar violaciones de la suposición de linealidad.
2. Homocedasticidad: La homocedasticidad se refiere a la igualdad de varianza de los residuos en función de las variables independientes. Si los puntos en el gráfico residuos del modelo vs predicción están dispersos de manera uniforme alrededor de la línea horizontal punteada en y=0, es una señal de homocedasticidad. Si los puntos forman un patrón de abanico o cono, podría indicar heterocedasticidad, lo que significa que la varianza de los errores cambia con las predicciones.

3. Normalidad: La normalidad de los residuos se puede verificar utilizando el gráfico distribución de residuos del modelo y el gráfico Q-Q residuos del modelo. [51] En el primero, si la distribución de los residuos se asemeja a una campana de Gauss, es un indicio de normalidad. En el segundo, los puntos deben estar cerca de la línea diagonal, lo que sugiere que los residuos se ajustan a una distribución normal. Desviaciones significativas en cualquiera de estos gráficos podrían indicar que los residuos no siguen una distribución normal.
4. Independencia de los Datos: El gráfico residuos del modelo permite verificar que los errores (residuos) no estén correlacionados entre sí. Si no se observa ningún patrón en la dispersión de los residuos a lo largo del eje x (por ejemplo, no hay autocorrelación temporal o espacial), es una indicación de independencia de los datos.

Finalmente se comprobó si los residuos siguen una distribución normal empleando el test estadístico Shapiro-Wilk. En este test, la hipótesis nula considera que los datos siguen una distribución normal, por lo tanto, si el p-value no es inferior al nivel de referencia ($p<0.05$) no hay evidencias para descartar que los datos se distribuyen de forma normal.

3.5.2. Modelo Random Forest Regression - RFR

El algoritmo Random Forest Regression (RFR) es un enfoque de aprendizaje conjunto (en ensambles) que se utiliza para crear modelos de regresión no paramétricos. Este método se basa en la combinación de múltiples árboles de decisión. RFR destaca por su capacidad para manejar grandes volúmenes de datos en tiempos de cómputo relativamente cortos. [54].

3.5.2.1. Construcción

La librería de Python, Sklearn, [105] cuenta con la clase *sklearn.ensemble.RandomForestRegressor*, que permite crear un modelo de regresión utilizando el algoritmo Random Forest. [105]

Se utilizó este algoritmo para entrenar el modelo random forest. Los parámetros e hiperparámetros se optimizaron utilizando las funcionalidades proporcionadas por la librería.

Se realizó un procedimiento de ajuste iterativo de modelos de regresión de random forest con el objeto de mejorar la performance del modelo en cada iteración. En este procedimiento se comenzó con un modelo inicial con sus hiperparametros sin optimizar, al que luego en sucesivas etapas se le fueron ajustando los hiperparametros.

Para realizar el ajuste de los modelos se dividió el dataset en datos de entrenamiento (24 lotes) y datos de testeo (9 lotes). Se evaluó la performance de los modelos mediante el cálculo de MAPE y el RMSE.

3.5.2.2. Optimización de hiperparámetros

3.5.2.2.1. Número de Árboles El número de árboles (n_estimators) es un hiperparámetro crítico en cuanto a que, añadir árboles puede hacer que mejore el resultado por sobreajuste y por otro lado añadir árboles una vez que la mejora se estabiliza es una pérdida de recursos

computacionales. Se realizó un ajuste por intermedio de un bucle iterativo donde se entrenó un modelo Random Forest con diferentes valores de n_estimators y se evaluó su rendimiento con dos métodos Out-of-Bag (OOB) y validación cruzada (cv).

1. El método Out-of-Bag (OOB) [57]: Es una forma de validación cruzada integrada en el proceso de entrenamiento. Cada vez que se entrena un árbol en el bosque (con un valor específico de n_estimators), algunos datos no se utilizan en ese proceso de entrenamiento (datos *fuera de la bolsa* o out-of-bag). Después de entrenar todos los árboles, se calcula una puntuación OOB promedio (oob_scores) para cada valor de n_estimators. Esta puntuación refleja cuán bien generaliza el modelo a datos que no se utilizaron en su entrenamiento. Se calcula utilizando la métrica R^2 (coeficiente de Pearson), que mide la proporción de la varianza en la variable objetivo (en este caso, y_train) que es explicada por el modelo. Un valor de R^2 cercano a 1 indica un buen ajuste a los datos de entrenamiento. Asimismo se evaluó el rendimiento del modelo en los datos de entrenamiento (train-scores). Cada puntuación en train_scores representa cuán bien se ajusta el modelo a los datos de entrenamiento con un valor específico de n_estimators. Finalmente el valor max score es el valor más alto de la puntuación OOB que se alcanza durante el proceso de entrenamiento e indica el número óptimo de árboles (n_estimators).

2. El método validación cruzada (cross-validation) [106] Este método divide los datos de entrenamiento en varios subconjuntos ("folds") y evalúa el modelo en múltiples combinaciones de entrenamiento y validación, lo que proporciona una medida más precisa de la capacidad de generalización del modelo. Se utiliza la métrica neg_root_mean_squared_error que es la versión negativa de RMSE para calcular el error en cada fold y luego informa el promedio de estos errores como la puntuación de validación cruzada (cv_scores). Esto se hace porque la función cross_val_score espera que una métrica de puntuación sea mayor cuando el modelo es mejor, pero el RMSE es mejor cuando es menor. Entonces, al usar la versión negativa, se invierte la métrica para que mayores valores negativos indiquen un mejor rendimiento. El objetivo es encontrar el valor de n_estimators que minimice el error de validación cruzada (min-score), que indica que el modelo que generaliza bien a datos no vistos.

La elección de los parámetros iniciales del modelo, en particular de árboles, generalmente no se basa en una regla matemática rígida, sino en una combinación de consideraciones prácticas, el tamaño del conjunto de datos, el equilibrio entre sesgo y varianza. Para un primer ajuste se propuso trabajar con 24 árboles, como valor inicial y se seleccionaron todas las variables predictoras disponibles.

3.5.2.2.2. Cantidad máxima de variables El valor del número máximo de variables dependientes (max_features) es uno de los hiperparámetros más importantes en Random Forest, ya que controla la correlación entre los árboles. Para determinar el valor óptimo de max_features, se calcularon las mismas dos métricas que para identificar el número de árboles: el error Out-of-Bag y la validación cruzada .

3.5.2.3. Importancia de las variables predictoras

Se calculó el grado de contribución individual de cada variable en la predicción global del modelo, calculando en primer lugar la importancia por Pureza de Nodos y luego por permutación.

3.5.2.3.1. Importancia por Pureza de Nodos Se calculó la medida de cuánto contribuye cada predictor (variable) en un modelo de árbol de decisión a la reducción de impureza en los nodos del árbol. La impureza se refiere a cuán mezcladas están las clases en un nodo. Los nodos con alta impureza contienen una mezcla de clases, mientras que los nodos con baja impureza tienen una predominancia de una sola clase. La importancia por pureza de nodos mide cuánto se reduce la impureza promedio en el árbol al dividir los datos en función de un predictor en particular.

El propósito de calcular la importancia por pureza de nodos es identificar qué predictores son más útiles para tomar decisiones en el árbol de decisión. Se utilizan para seleccionar las variables más relevantes y simplificar el árbol, lo que puede llevar a modelos más interpretables y eficientes.

3.5.2.3.2. Importancia por Permutación Se calculó cuánto disminuye el rendimiento del modelo cuando se permutan los valores de un predictor específico. Esta metodología se basa en la idea de que si una variable es importante en un modelo, al permutar sus valores (alterar su orden) debería tener un impacto negativo en el rendimiento del modelo. Es decir, si se permuta una variable importante, el modelo debería empeorar. Por lo tanto, la importancia por permutación mide cuánto empeora el rendimiento del modelo cuando se permutan los valores de un predictor específico.

El propósito de calcular la importancia por permutación es proporcionar una evaluación más robusta de la importancia de las variables en un modelo. Puede detectar relaciones no lineales y efectos no obvios que la importancia por pureza de nodos podría pasar por alto.

3.5.2.4. Evaluación con gráficos de dependencia parcial (PDP)

Se construyeron los gráficos de dependencia parcial (PDP, por sus siglas en inglés "Partial dependance plot") que indican el efecto marginal que tienen una o dos características en el resultado previsto de un modelo de aprendizaje automático [107]. Un gráfico de dependencia parcial puede mostrar si la relación entre el objetivo y una característica es lineal, monótona o más compleja.[108] Por ejemplo, cuando se aplica a un modelo de regresión lineal, las gráficas de dependencia parcial siempre muestran una relación lineal.

Las X_S son las características para las que se debe trazar la función de dependencia parcial y X_C son las otras características utilizadas en el modelo de aprendizaje automático f

$$pd_{X_S}(x_S) \overset{def}{=} \mathbb{E}_{X_C}[f(x_S, X_C)] = \int f(x_S, x_C) p(x_C) dx_C \tag{3.10}$$

Por lo general, solo hay una o dos características en el conjunto S. Las características en S son aquellas para las que queremos saber el efecto en la predicción. Los vectores de características X_S y X_C combinados forman el espacio total de características x.

La dependencia parcial funciona al marginar la salida del modelo de aprendizaje automático sobre la distribución de las funciones en el conjunto C, de modo que la función muestre la relación entre las funciones en el conjunto S que interesan y el resultado previsto.

Al marginar las otras funciones, obtenemos una función que depende solo de las funciones en S, incluidas las interacciones con otras funciones. La función parcial dice, para valores dados

de las características S, cuál es el efecto marginal promedio en la predicción. En esta fórmula, $x_(i)_C$ son valores de características reales del conjunto de datos para las características que no interesan, y n es el número de instancias en el conjunto de datos. Una suposición del PDP es que las características en C no están correlacionadas con las características en S. Si se viola esta suposición, los promedios calculados para la gráfica de dependencia parcial incluirán puntos de datos que son muy poco probables o incluso imposibles [65].

Ventajas El cálculo de gráficos de dependencia parcial es intuitivo: la función de dependencia parcial en un valor de la variable predictora particular representa la predicción promedio si obligamos a todos los datos a asumir ese valor. Si la variable para la que se calculó el PDP no está correlacionada con las otras características, entonces los PDP representan perfectamente cómo la variable influye en la predicción en promedio.

En el caso no correlacionado, la interpretación es clara: la gráfica de dependencia parcial muestra cómo la predicción promedio en su conjunto de datos cambia cuando se cambia la función j-ésima. El cálculo de las dependencias parciales tiene una interpretación causal. Intervenimos en una variable y medimos los cambios en las predicciones. Al hacerlo, analizamos la relación causal entre la variable y la predicción.[109]

La relación es causal para el modelo, porque modela explícitamente el resultado como una función de las variables, pero no necesariamente para el mundo real.

Desventajas El número máximo realista de variables predictoras en una función de dependencia parcial es dos. Esto no es culpa de las PDP, sino de la representación bidimensional (papel o pantalla) y también de nuestra incapacidad para imaginar más de 3 dimensiones.

La suposición de independencia es el mayor problema con las gráficas de PDP. Se supone que las variables para las que se calcula la dependencia parcial no están correlacionadas con otras variables. En el caso de que estén correlacionadas la curva PDP se presenta como una línea horizontal, ya que los efectos de ambas mitades del conjunto de datos se cancelan entre sí. Lo que permite concluir que la variable no tiene ningún efecto en la predicción [108].

Los códigos desarrollados para realizar el análisis de los datos y distintos modelos se encuentran en los Anexos V, VI y VII. También se encuentran publicados en https://github.com/diegocarcedo/Corn-Yield-Estimation.

CAPÍTULO 4

Resultados

En este capítulo se presenta, en primer lugar, el resultado de un análisis exploratorio de la base de datos y las relaciones existentes entre las variables estudiadas. En segundo lugar se exponen los resultados obtenidos con los distintos modelos evaluados. Como fue mencionado en el apartado de materiales y métodos, se realizaron múltiples pruebas, con el fin de ajustar los parámetros asociados a cada modelo.

Utilizando los valores de R2, RMSE y MAPE como métricas, se evaluaron con los datos de prueba la capacidad predictiva de todos los modelos y se compararon para identificar el mejor modelo asimismo se realizaron diagramas de dispersión entre los resultados reales y los predichos. Finalmente en el mejor modelo obtenido, se estudiaron las dependencias parciales de las variables modeladas.

4.1. Análisis exploratorio de la base de datos.

4.1.1. Variable dependiente

La variable dependiente, conocida también como variable de respuesta, está conformada por las mediciones de rendimiento de maíz tomadas en el campo y promediadas a nivel de cada lote. Con el objeto de conocer si la distribución de la variable es normal, se realizo en primer lugar un test de Shapiro-Wilk y luego un histograma que grafique el comportamiento de la variable. Finalmente se exploró el comportamiento de la variable mediante gráficos boxplot.

4.1.1.1. Test de Shapiro-Wilk

El test de Shapiro-Wilk es una prueba estadística utilizada para evaluar si una muestra de datos sigue una distribución normal. Se utiliza para verificar si los datos siguen una distribución que

se asemeja a una campana de Gauss, que es comúnmente asociada con la distribución normal.

El ρ-valor que se obtiene al realizar el test de Shapiro-Wilk es una medida de la evidencia en contra de la hipótesis nula. En este contexto, la hipótesis nula (H0) es que los datos de la muestra siguen una distribución normal. Si se obtiene como resultado del test ρ-valor alto, significa que no tienes evidencia suficiente para rechazar la hipótesis nula es decir, no hay suficiente evidencia para afirmar que los datos no siguen una distribución normal. El resultado obtenido para el test fue de 0.62 lo sugiere que no hay razones estadísticas para afirmar que los datos no siguen una distribución normal.

4.1.1.2. Histograma

Se graficó la distribución de la variable dependiente y (Figura 4.1) se corroboró que tiene una distribución normal con una cola levemente negativa debido a que, unos pocos lotes, tienen un rendimiento muy inferior a la media de la variable.

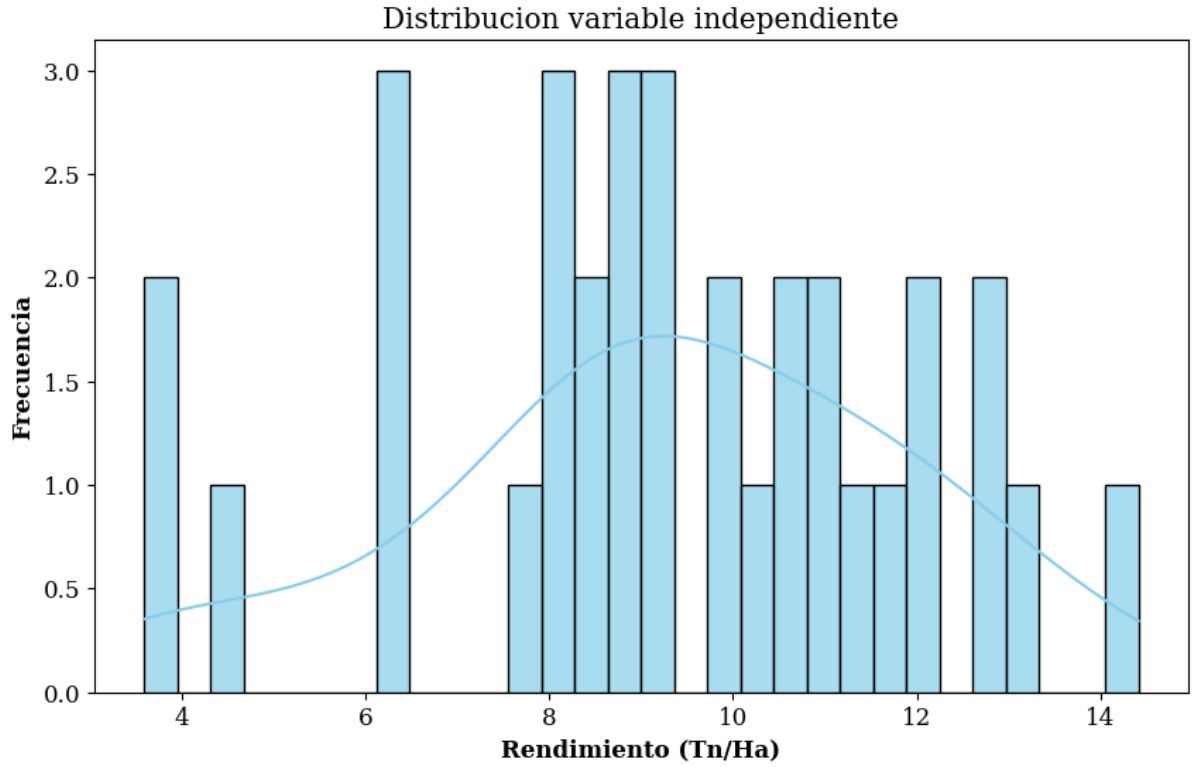

Figura 4.1: Distribución de la variable dependiente , rendimiento de toneladas por hectárea por lote.

4.1.1.3. Boxplot

Con el objeto de caracterizar la variabilidad a escala de lote, tanto de la variable medida a campo como de las variables derivadas de sensores, se calculó el desvío estándar y el rango de la distribución de los datos. Con la información provista por estos estadísticos se estudió la distribución de la variable por medio de gráficos boxplot.

Un boxplot (o diagrama de caja) es una representación gráfica que muestra la distribución de un conjunto de datos de manera resumida. Los componentes principales de un boxplot son:

1. La caja (box) representa el rango intercuartílico (IQR), que es la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1) de los datos. El interior de la caja contiene el 50 % central de los datos. La longitud de la caja es proporcional al IQR. El borde inferior de la caja está en Q1, y el borde superior de la caja está en Q3.
2. Dentro de la caja, se traza una línea que representa la mediana (Q2) de los datos. La mediana es el valor que divide el conjunto de datos en dos mitades iguales.
3. Las barras son las líneas que se extienden desde la parte superior e inferior de la caja hasta los valores extremos dentro de un rango aceptable. Los valores extremos se calculan como 1.5 veces el IQR. Las barras representan la dispersión de los datos dentro de un rango razonable.
4. Los puntos individuales fuera de las barras se consideran valores atípicos o extremos. Estos son valores que están significativamente alejados de la tendencia central.

   En líneas generales se interpreta que cuanto más grande es la caja, mayor es la dispersión de los datos, es decir el rendimiento medido a campo de ese año contó con una mayor variabilidad en los datos. Asimismo si la linea central de la caja es maś alta, significa que la media es mayor en ese año en particular, lo que sugiere un mejor rendimiento.

   El boxplot de la variable dependiente, denominada rendimiento medido a campo, se construyó por años para poder evaluar el comportamiento de la variable en el rango de tiempo estudiado. El gráfico permitió observar que la mediana de la variable rendimiento presenta valores consistentes a lo largo del tiempo y no presenta valores anómalos o outliers. Asimismo se observaron valores muy bajos en el año 2018, lo que condice con la sequía reportada para ese año [110] y una gran variabilidad en los datos recolectados en el año 2019.

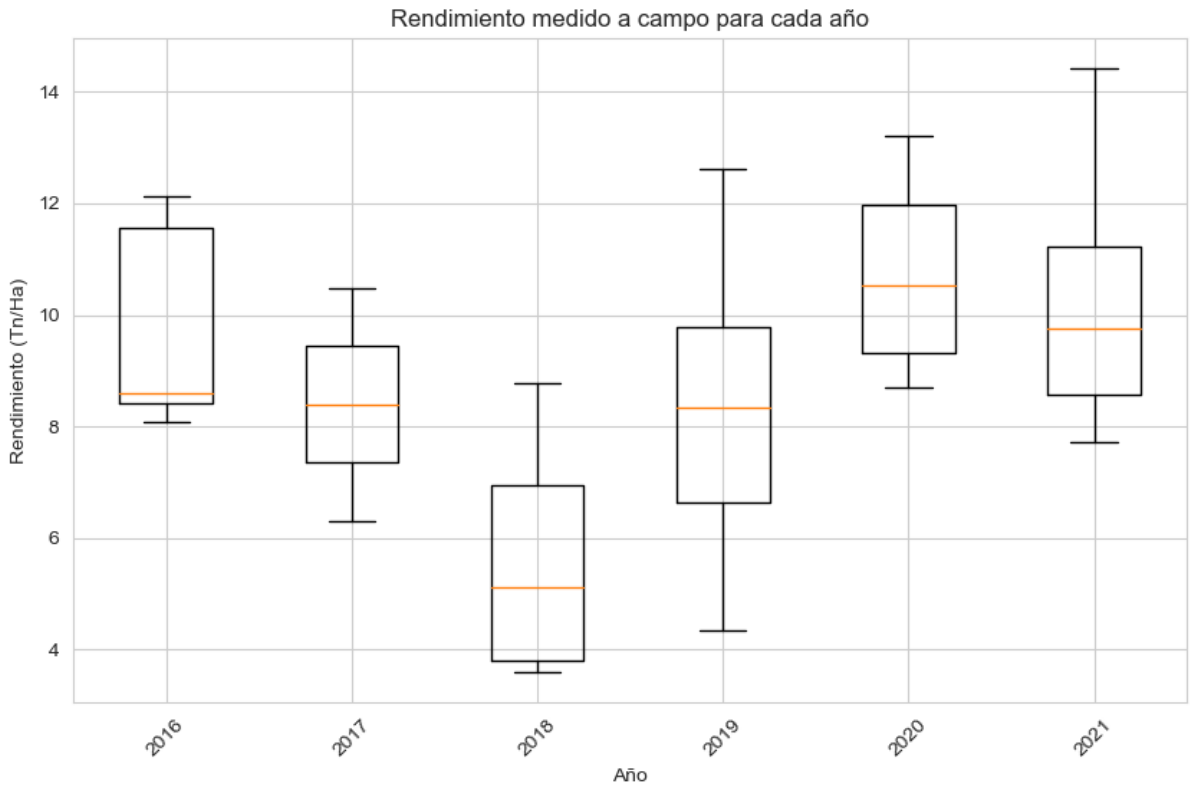

Figura 4.2: El boxplot representa la distribución de los datos de rendimiento para cada año de una manera visual y proporciona información sobre la tendencia central (mediana), la distribución (tamaño de la caja) y la dispersión (barras).

4.1.2. Variables predictoras

Las variables seleccionadas que conformaron la base de datos para la evaluación de algoritmos de predicción fueron variables derivadas del sensoramiento remoto satelital. Las variables predictoras, también llamadas independientes, se agruparon según su foco de medición, en variables focalizadas en tejidos vegetales, variables climáticas, variables topográficas (Tabla 3.8).

Con el objeto de conocer en qué medida la media espacial calculada a escala de lote es representativa de cada lote, se calculó el desvío estándar y el rango.

Este procedimiento se realizo en QGis. En primer lugar se extrajo la media, luego el desvío estándar y finalmente el rango para todas las variables para cada lote con excepción de las variables modelo digital de superficie (DSM), precipitación (PP), temperatura (TEMP) y evapotranspiración (EVAPO) ya que por su resolución espacial mayor al tamaño del perímetro de lote solo se contó con un dato por lote.

Con la información provista por estos estadísticos se estudió la distribución de los datos de las variables independientes mediante la utilización de gráficos boxplot.

4.1.2.1. Box-plot variables predictoras

Como se observa en la Figura 4.1 cada boxplot muestra la distribución de una variable específica, GNDVI, NDRE, NDVI, OSAVI, RECI, TCARIOSAVI y SLOPE, para los diferentes años,

que van desde 2016 hasta 2021. Las diferencias en la forma y la dispersión de los boxplots reflejan cómo estas variables se comportan tanto intralote como en los diferentes sitios en los diferentes años.

En las variables NDVI, NDRE, NDVI, OSAVI y RECI se observan similaridades en el comportamiento de los datos:

1. La distribución de los datos es similar.
2. Varían de la misma manera entre los distintos años.
3. Las medianas no están cerca de los extremos evidenciando una distribución no simétrica.
4. Se identifican outliers en los mismos años, en el año 2021 para todas las variables y en el año 2020 para todas excepto GNDVI.

Por otro lado la variable TCARI si bien tiene un comportamiento a través de los años muy similar a las variables mencionadas en el párrafo anterior la distribución de sus datos presentan menor variabilidad ya que las barras que muestran los valores extremos son cortas. Asimismo se observa que presenta valores outliers en el año 2019 y 2021. La variable TCARIOSAVI presenta outliers solo en el año 2021 y una distribución menos variables a través de los años. Por último la variable SLOPE presenta datos similares en cuanto a mediana (pendientes entre 1 y 1,5 %) y distribución a lo largo de los años, con la excepción del año 2017 donde todos los datos provienen del mismo establecimiento, en particular de dos lotes con gran presencia de lomas con pendientes mayor a 2 %.

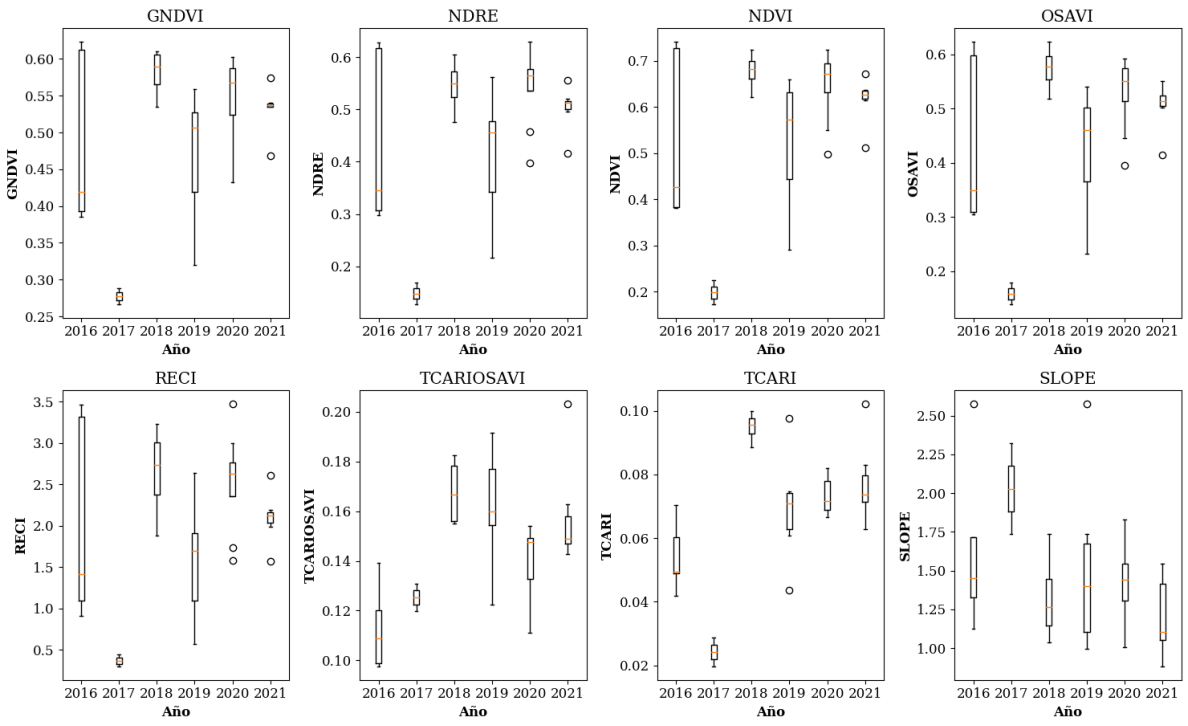

Figura 4.3: El boxplot representa la distribución de los datos de las variables GNDVI, NDRE, NDVI, OSAVI, RECI, TCARIOSAVI y SLOPE para cada año de una manera visual y proporciona información sobre la tendencia central (mediana), la distribución (tamaño de la caja) y la dispersión (barras).

4.1.2.2. Histograma

Con el objeto de conocer la distribución para las variable topográfica, DSM, y las climáticas, PP, TEMP y EVAPO cuya resolución espacial es mayor a la superficie de lotes, por lo cual no contamos con información sobre su variabilidad espacial intralote, se estudió su distribución con un histograma que grafique el comportamiento entre los distintos lotes y años.

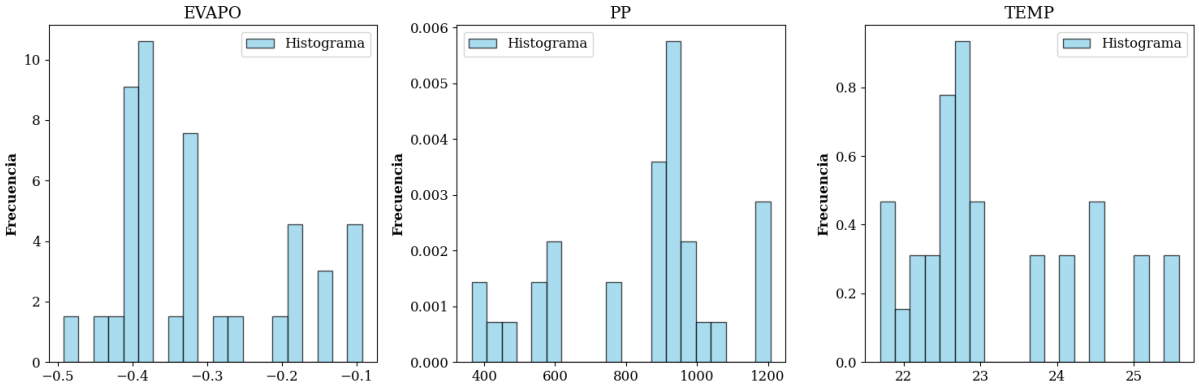

Figura 4.4: Distribución de las variables climáticas - Evapotranspiración, Precipitación y Temperatura.

La distribución de las variables climáticas (Figura 4.4) evidenció que los ambientes estudiados son distintos entres si. En particular en lotes del sur de Córdoba se acumuló a lo largo del periodo de tiempo estudiado un promedio de 500 mm, mientras que en lotes del norte y oeste de la provincia de buenos aires se acumularon entre 800 y 1200 mm de agua. Por otro lado la temperatura y la evapotranspiración, se distribuyeron en la misma medida a lo largo de los lotes, evidenciando la alta correlación entre estas variables. Los valores negativos de evapotranspiración, se explican a partir de que en la convención del modelo ECMWF- IFS (Integrated Forecast System) es que los flujos descendentes son positivos y los ascendentes (transpiración y evaporación) son negativos.

Las variables (Figura 4.5) focalizadas en el relieve dan cuenta de la no homogeneidad de los lotes estudiados en cuanto a sus características topográficas. Si bien la mayoría de los lotes se encuentran en zonas de alrededor de 100 metros de altitud (oeste y norte buenos aires), se estudiaron lotes a 300 metros de altura correspondientes al sur de la provincia de Córdoba.

Se constató que las variables NDVI_range, RECI_range, TCARIOSAVI_mean, RECI_mean, TCARI_range, OSAVI_range, TCARI_mean son las únicas que presentan una distribución normal mientras que las restantes presentan una distribución no normal.

En el Anexo 2 se detallan el resultado de los test de Shapiro-Willk y los histogramas de las distribuciones de las variables predictoras.

Finalmente se calculó la correlación lineal mediante el cálculo del coeficiente de determinación (R^2) entre cada variable independiente y la variable dependiente. En líneas generales se constató que la correlación individual de cada variable independientes con la variable dependiente es baja (R^2 <0.25).

En el Anexo 3 se presentan los gráficos de las correlaciones lineales y los coeficientes de determinación obtenidos.

En resumen la caracterización y el análisis exploratorio de la variable dependiente permitió por un lado corroborar que cuenta con una distribución normal, aspecto imprescindible para poder

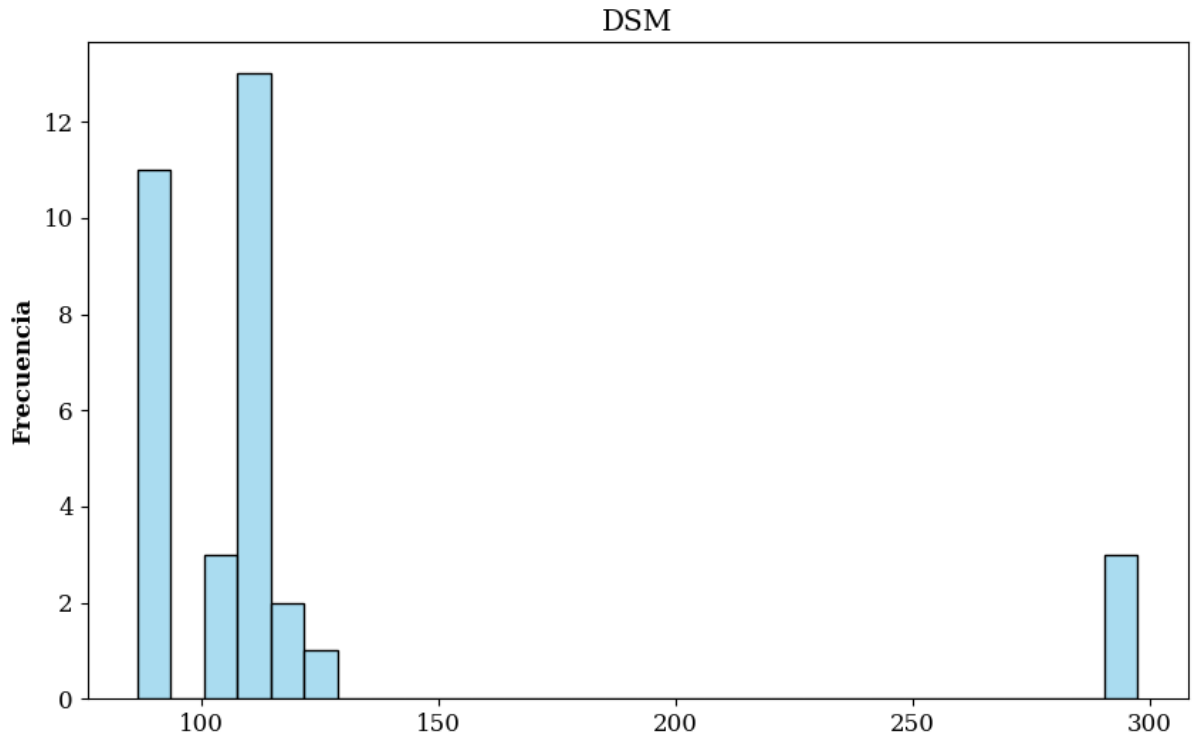

Figura 4.5: Distribución de las variables Altitud (DSM).

construir un modelo de regresión lineal, y por otro conocer que las variables desvío estándar y rango son una herramienta útil para conocer la variabilidad de la variable tanto intralote como entre distintos sitios y años.

Asimismo el análisis exploratorio de las variables seleccionadas como predictoras permitió caracterizar la distribución de cada variable a través de los años, identificar el comportamiento similar entre variables y la presencia de valores anómalos, y constatar que las variables desvío estándar y rango calculadas en cada lote para cada variable son una herramienta útil para caracterizar la variabilidad espacial.

4.1.2.3. Estructura de correlación entre variables

Se estudió la estructura de correlación entre las variables de la base de datos mediante una matriz de correlación, en las que se muestra el coeficiente de correlación de pearson para cada par de variables [111].

El coeficiente de correlación de Pearson ρ es una medida estadística que se utiliza para evaluar la relación lineal entre dos conjuntos de datos numéricos. Este coeficiente se calcula de la siguiente manera:

1. Se obtiene la covarianza entre las dos variables (en este caso, un índice espectral y el rendimiento del maíz).
2. Luego, se calculan las desviaciones estándar de ambas variables.
3. Finalmente, el coeficiente de correlación de Pearson se obtiene dividiendo la covarianza entre las dos variables por el producto de sus desviaciones estándar.

El coeficiente de correlación de Pearson puede tomar valores entre -1 y 1:

1. Si el valor es 1, indica una correlación positiva perfecta, lo que significa que a medida que una variable aumenta, la otra también lo hace de manera lineal.
2. Si el valor es -1, indica una correlación negativa perfecta, lo que significa que a medida que una variable aumenta, la otra disminuye de manera lineal.
3. Si el valor es 0, indica que no hay correlación lineal entre las dos variables.

En primer lugar (Figura 4.6) se calculó una matriz de correlación de todas las variables predictoras, es decir las variables GNDVI, NDRE, NDVI, OSAVI, RECI, TCARIOSAVI, SLOPE, DSM, PP, TEMP EVAPO, AÑO y sus variables asociadas (rango y desvío estándar). Se identificó que existe alta correlación entre las variables derivadas de sensoramiento remoto focalizadas en tejidos, en particular entre las variables medias y su variables asociadas (desvío estándar y rango).

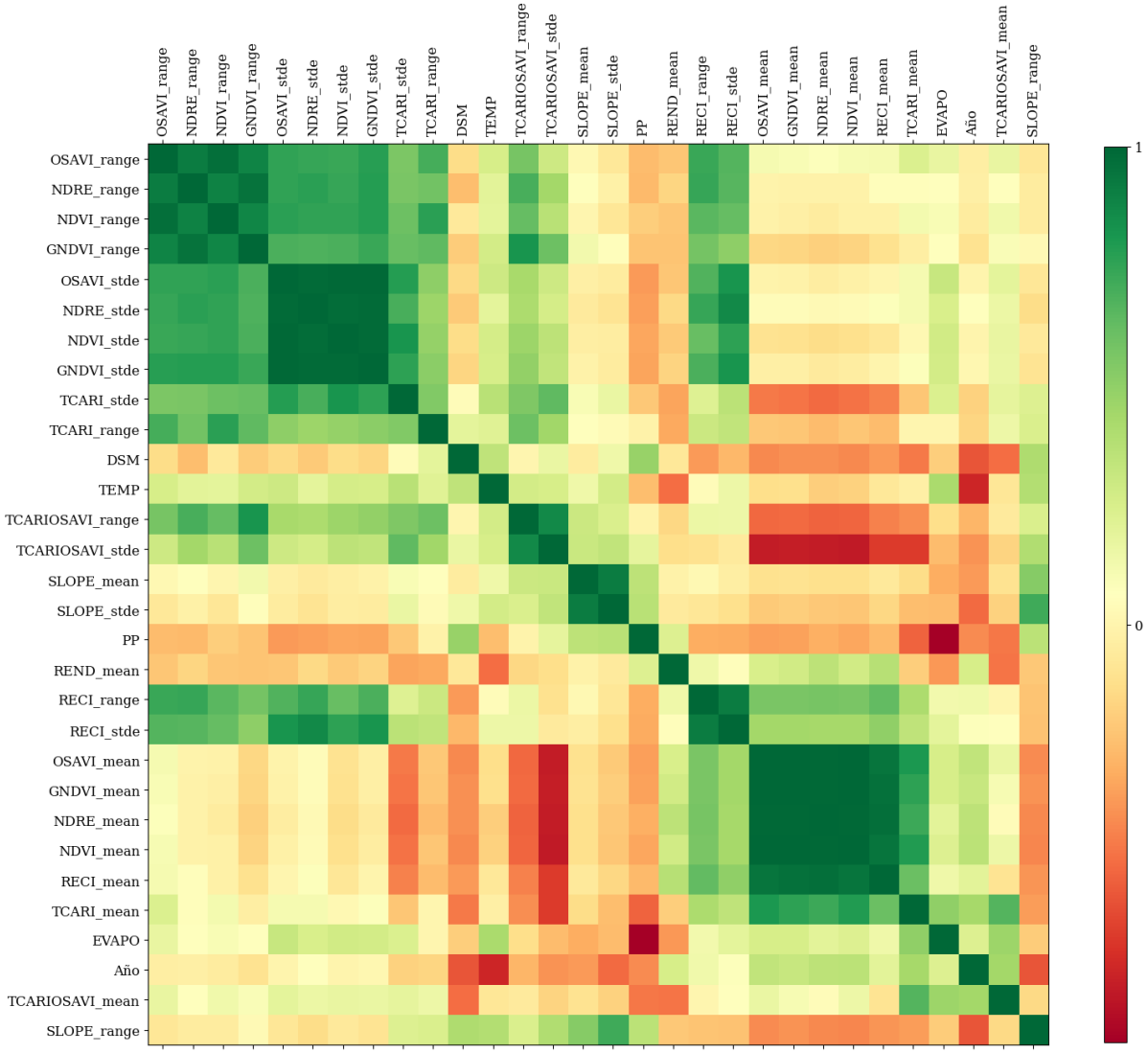

Figura 4.6: Correlograma de las variables de la base de datos. Los colores indican la correlación entre las variables que varían desde un color verde (correlación positiva) a color rojo (correlación negativa) Colores claros indican poca correlación entre el par de variables.

Asimismo se identificó una alta correlación ($\rho > 0{,}85$) entre la variables medias y el desvío

estándar de los índices espectrales GNDVI, OSAVI, NDVI, NDRE, RECI, NDRE.

Con el objeto de visualizar particularmente, en aquellas variables que cuentan con variables que miden su variabilidad, las correlaciones entre las variables medias y sus respectivas medidas de variabilidad (desvío estándar y rango), se calculó una matriz (Figura 4.7) solo con estas variables.

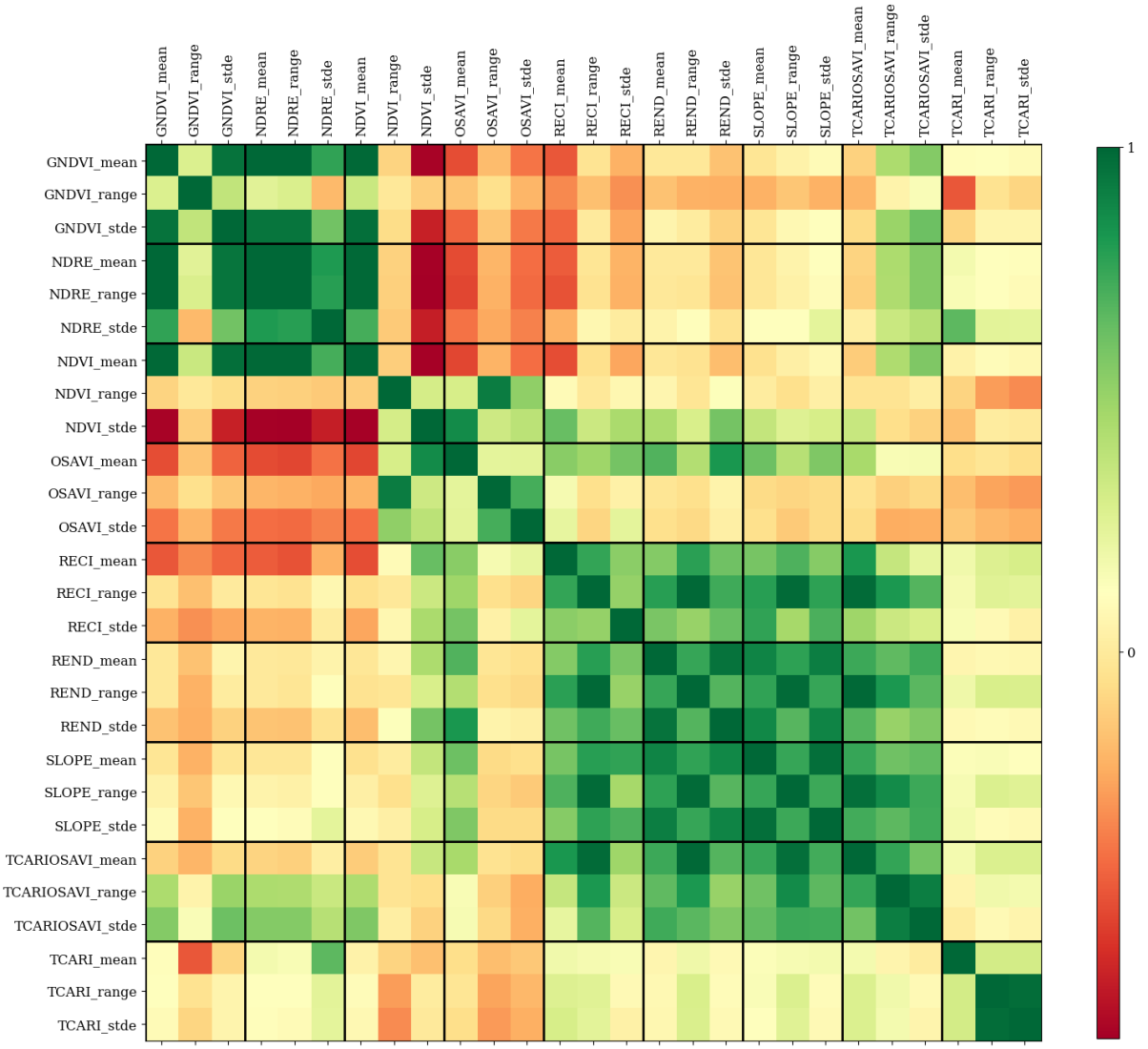

Figura 4.7: Matriz de correlación de las variables de la base de datos ordenadas de a conjuntos de tres variable con el objeto de realzar la correlación entre la variable y las variables que la caracterizan (desvío estándar y rango). Los colores indican la correlaciónentre las variables que varían desde un color verde (correlación positiva) a color rojo(correlación negativa) Colores claros indican poca correlación entre el par de variables.

Se identificó que existe una alta correlación ($\rho > 0{,}85$) entre las variables GNDVI, NDRE, RECI, REND, SLOPE, TCARIOSAVI y TCARI, con sus respectivas variables asociadas (desvío estándar y rango) que son una medida de su variabilidad, y en menor medida (($\rho > 0{,}75$)entre las variables NDVI, OSAVI y sus respectivas variables asociadas.

La correlación de Spearman, también conocida como coeficiente de correlación de Spearman ψ, es una medida estadística que evalúa la relación entre dos variables. Se utiliza para determinar si existe una asociación monotónica (no necesariamente lineal) entre las variables, es decir,

si el aumento o disminución en una variable se relaciona con un aumento o disminución en la otra variable, aunque no necesariamente en una relación lineal.

Las características de este tipo de correlación son:

1. No paramétrica: A diferencia de la correlación de Pearson, que asume que las variables están distribuidas de manera normal y tienen una relación lineal, la correlación de Spearman no hace estas suposiciones y es adecuada para datos que no siguen una distribución normal o que muestran relaciones no lineales.
2. Basada en Rangos: Para calcular la correlación , primero se convierten los valores de las variables en rangos. Los rangos son números enteros que indican la posición relativa de un valor en un conjunto de datos ordenado. Luego, se calcula la correlación entre los rangos en lugar de los valores originales.
3. Valores de Correlación: El coeficiente de correlación ψ varía en el rango de -1 a 1.
    a) ψ = 1: Indica una correlación perfectamente positiva, lo que significa que a medida que una variable aumenta, la otra también lo hace en un patrón estrictamente monotónico.
    b) ψ = -1: Indica una correlación perfectamente negativa, lo que significa que a medida que una variable aumenta, la otra disminuye en un patrón estrictamente monotónico.
    c) ψ = 0: Indica que no hay una relación monotónica entre las variables, pero no necesariamente que no haya ninguna relación en absoluto.

Esta correlación es útil cuando se trabaja con datos numéricos que no se ajustan a los supuestos de la correlación de Pearson.

Con el objeto de visualizar las correlaciones específicas entre las variables medias de los índices espectrales se calculó una matriz de correlación de Spearman (Figura 4.8). Esta matriz permite visualizar las correlaciones entre múltiples variables, así como identificar patrones de relación entre ellas.

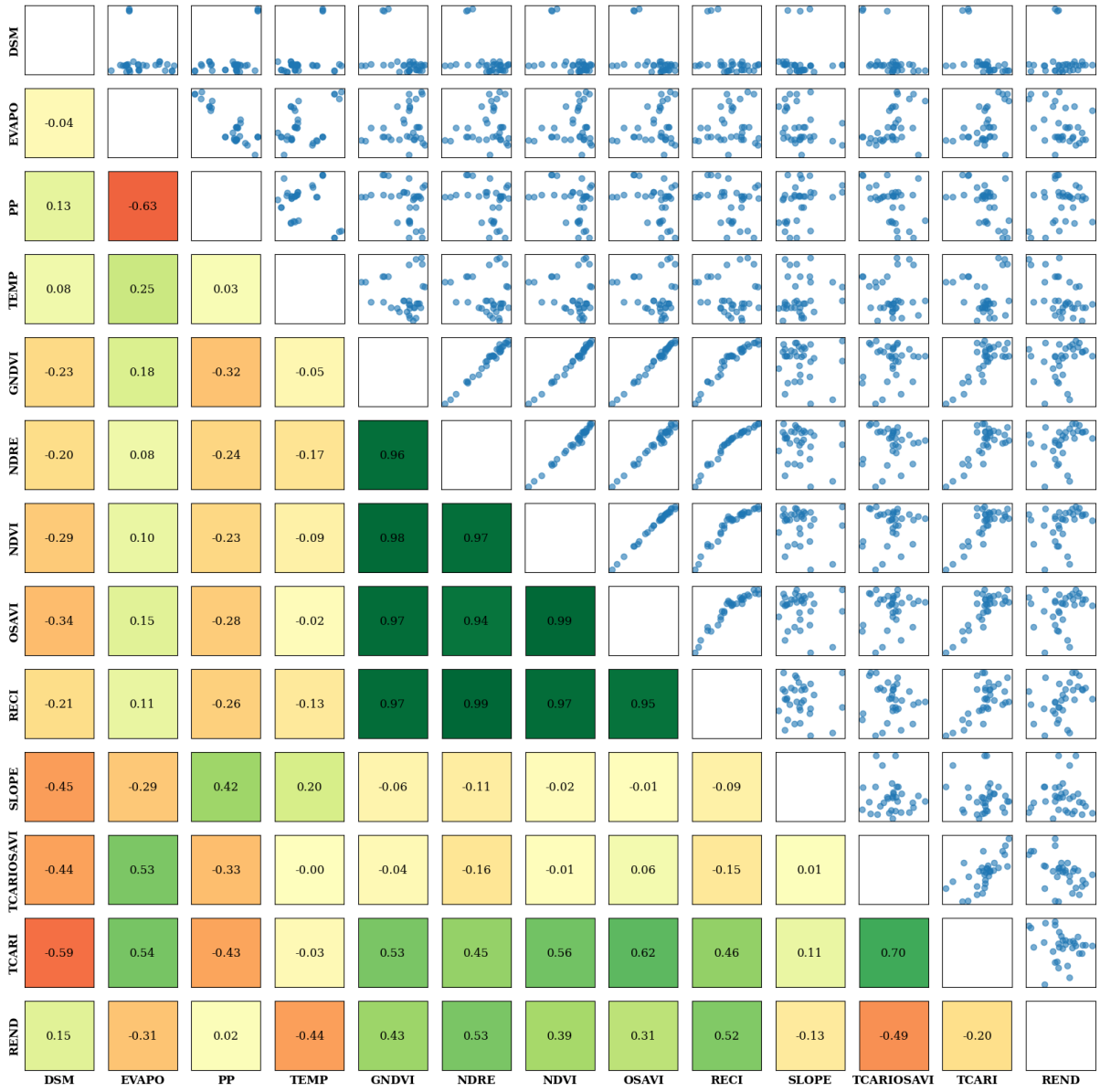

Figura 4.8: Matriz de correlación con el valor del coeficiente de spearman y el diagrama de dispersión de cada par de relaciones.

La matriz de correlación permitió visualizar que existen altas correlaciones ($\rho > 0{,}75$) entre las variables derivadas de sensoramiento remoto focalizadas en tejidos vegetales, en particular entre los índices RECI, OSAVI, NDVI y NDRE, y en menor medida TCARI ($\rho > 0{,}45$) esto posiblemente se debe a que todos los índices espectrales comparten alguna banda Tabla 3.7 en su cálculo por lo cual cuentan con datos similares.

Por otro lado se observa una correlación negativa ($\rho < -0{,}63$) no esperable entre PP y EVAPO, ya que se conoce que a mayor precipitación mayor evapotranspiración. Asimismo se observa que la variable rendimiento se correlaciona de manera positiva ($\rho > 0{,}31$) con los índices espectrales GNDVI, NDVI, OSAVI, RECI y de manera negativa ($\rho > 0{,}49$) con el índice TCARIOSAVI.

Finalmente la matriz de correlaciones permitió identificar que las variables climáticas y las topográficas no guardan una correlación alta con la variable rendimiento (REND).

El análisis de la estructura de correlación entre las variables permitió conocer y caracterizar cómo se correlacionan las variables predictoras de manera individual entre sí y con sus variables que miden su variabilidad. Este análisis nos sugiere que existe multicolinealidad en la base de datos, información muy valiosa para utilizar en la construcción de los modelos de estimación de rendimiento .

4.2. Modelo Regresión Lineal Múltiple - RLM.

Como se observó, en el análisis exploratorio de la sección anterior, existe una alta correlación lineal entre las variables y sus respectivas variables asociadas que caracterizan su variabilidad (desvío estándar y rango). La multicolinealidad [112] en el contexto de modelos de regresión predictivos, usualmente se trabaja con la eliminación de variables redundantes ya que se asume que la disminución de la dimensionalidad por redundancia no implica pérdida significativa de la capacidad predictiva.

La etapa de selección de las variables para ajustar un modelo de regresión es una etapa crucial en el ajuste de modelos predictivos. Las variables disponibles con potencialidad explicativa suelen estar altamente correlacionadas (colinealidad) y complicar el ajuste de modelos lineales [113].

Con el objeto de evitar utilizar datos redundantes y variables colineales se construyeron modelos solo con las variables medias (mean) de los índices espectrales y de la variable pendiente (slope), mas las variables climáticas y la altitud (DSM) y se realizó un procedimiento de ajuste iterativo de modelos de regresión lineal múltiple. En este procedimiento en cada iteración se modificó el conjunto y tipo de variables de input con el objeto de incrementar la capacidad explicativa del modelo.

Para realizar el ajuste de los modelos se dividió el dataset en datos de entrenamiento (24 lotes) y datos de testeo (9 lotes).

Los resultados obtenidos para cada modelo de regresión lineal se compilaron en la Tabla 4.1).

Se realizo un diagnóstico de los residuos del modelo por realizando en primer lugar una inspección visual con el objetivo de corroborar que se comportan de manera aleatoria e independiente. Y en segundo lugar se calcularon estadísticos que corroboran objetivamente lo identificado visualmente.

El procedimiento consiste en corroborar los cuatro supuestos:

1. Linealidad: Para verificar la linealidad de los residuos se utiliza el gráfico: Valor Predicho vs Valor Real. Si los puntos se distribuyen alrededor de la línea diagonal (línea punteada de referencia), esto sugiere que la relación entre las variables independientes y la variable dependiente es aproximadamente lineal. Si se observan patrones curvos o no lineales en este gráfico, podría indicar violaciones de la suposición de linealidad.

2. Homocedasticidad: La homocedasticidad se refiere a la igualdad de varianza de los residuos en función de las variables independientes. Si los puntos en el gráfico residuos del modelo vs predicción están dispersos de manera uniforme alrededor de la línea horizontal punteada en y=0, es una señal de homocedasticidad. Si los puntos forman un patrón

de abanico o cono, podría indicar heterocedasticidad, lo que significa que la varianza de los errores cambia con las predicciones.

3. Normalidad: La normalidad de los residuos se puede verificar utilizando el gráfico distribución de residuos del modelo y el gráfico Q-Q residuos del modelo. En el primero, si la distribución de los residuos se asemeja a una campana de Gauss, es un indicio de normalidad. En el segundo, los puntos deben estar cerca de la línea diagonal, lo que sugiere que los residuos se ajustan a una distribución normal. Desviaciones significativas en cualquiera de estos gráficos podrían indicar que los residuos no siguen una distribución normal.
4. Independencia de los Datos: El gráfico residuos del modelo permite verificar que los errores (residuos) no estén correlacionados entre sí. Si no se observa ningún patrón en la dispersión de los residuos a lo largo del eje x (por ejemplo, no hay autocorrelación temporal o espacial), es una indicación de independencia de los datos.

Finalmente se comprobó si los residuos siguen una distribución normal empleando el test estadístico Shapiro-Wilk. En este test, la hipótesis nula considera que los datos siguen una distribución normal, por lo tanto, si el ρ-valor no es inferior al nivel de referencia (ρ<0.05) no hay evidencias para descartar que los datos se distribuyen de forma normal.

4.2.1. Modelo inicial

En primer modelo, RLM -1, se ajustó con las variables correspondientes a los índices espectrales, las variables climáticas y las topográficas. Se obtuvo un R^2 = 0.68 y un RMSE de 1.83. Se corroboró la normalidad de los residuos con el test de Shapiro-Wilk (0.97) con un ρ-valor de 0.71.

Se observa en los distintos gráficos que se cumplen los supuestos de linealidad, independencia y normalidad. Al respecto de la homocedasticidad de los residuos, se observa entre los valores 5 y 10 de predicción en el gráfico de residuos del modelo versus predicción una posible estructura en la distribución de los datos.

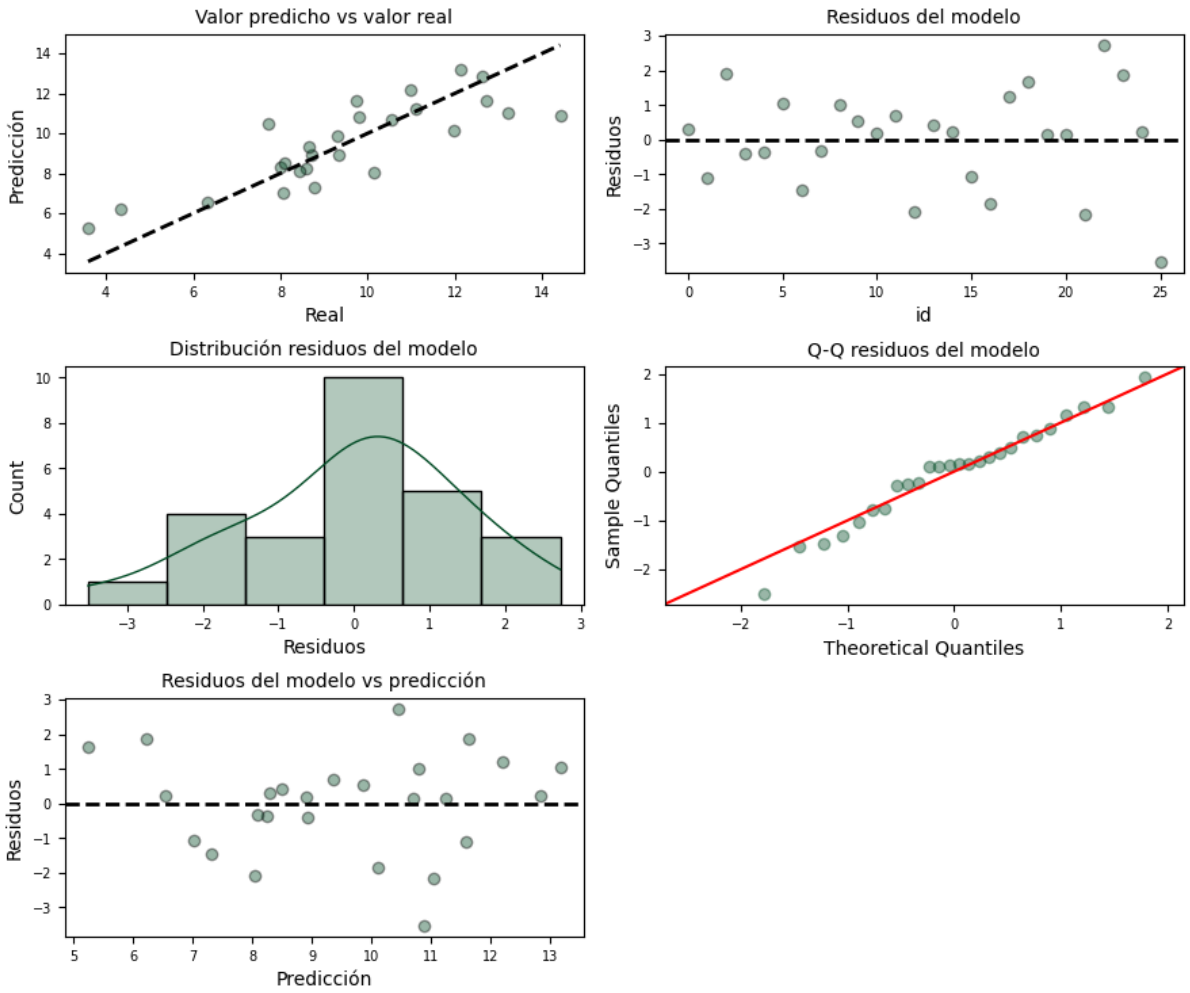

Figura 4.9: Diagnóstico de los residuos del modelo de regresión lineal múltiple con las variables índices espectrales, climáticas y topográficas.

4.2.2. Modelo con análisis de componentes principales

Con el objeto de reducir la multicolinealidad entre las variables predictoras se utilizó una técnica estadística y de aprendizaje automático que reduce la dimensionalidad de un conjunto de datos denominada análisis de componentes principales (ACP).

Este método consiste en transformar un conjunto de variables correlacionadas en un conjunto de variables no correlacionadas llamadas componentes principales. Estas componentes principales se ordenan en función de su varianza, de modo que las primeras componentes principales explican la mayor parte de la variabilidad en los datos originales.

En primer lugar se procedió a calcular las componentes principales. Luego se seleccionaron las primeras 3 componentes con las cuales se logra explicar un 80 % de la varianza. Se procedió finalmente a ajustar un nuevo modelo, RLM -2, con estas nuevas variables sintéticas. Se obtuvo un $R^2 = 0.64$ y un RMSE de 1.80.

Se corroboró la normalidad de los residuos con el test de Shapiro-Wilk (0.94) con un ρ-valor de 0.11.

Se observan en los distintos gráficos que se cumplen los supuestos de linealidad y normalidad

. Al respecto de la linealidad en el gráfico residuos del modelo los datos no se distribuyen de manera aleatoria si no en una serie sinusoidal. Situación similar observamos en el supuesto de homocedasticidad de los residuos, se observa entre en el gráfico de residuos del modelo vs predicción una posible estructura en la distribución de los datos con forma de serie sinusoidal.

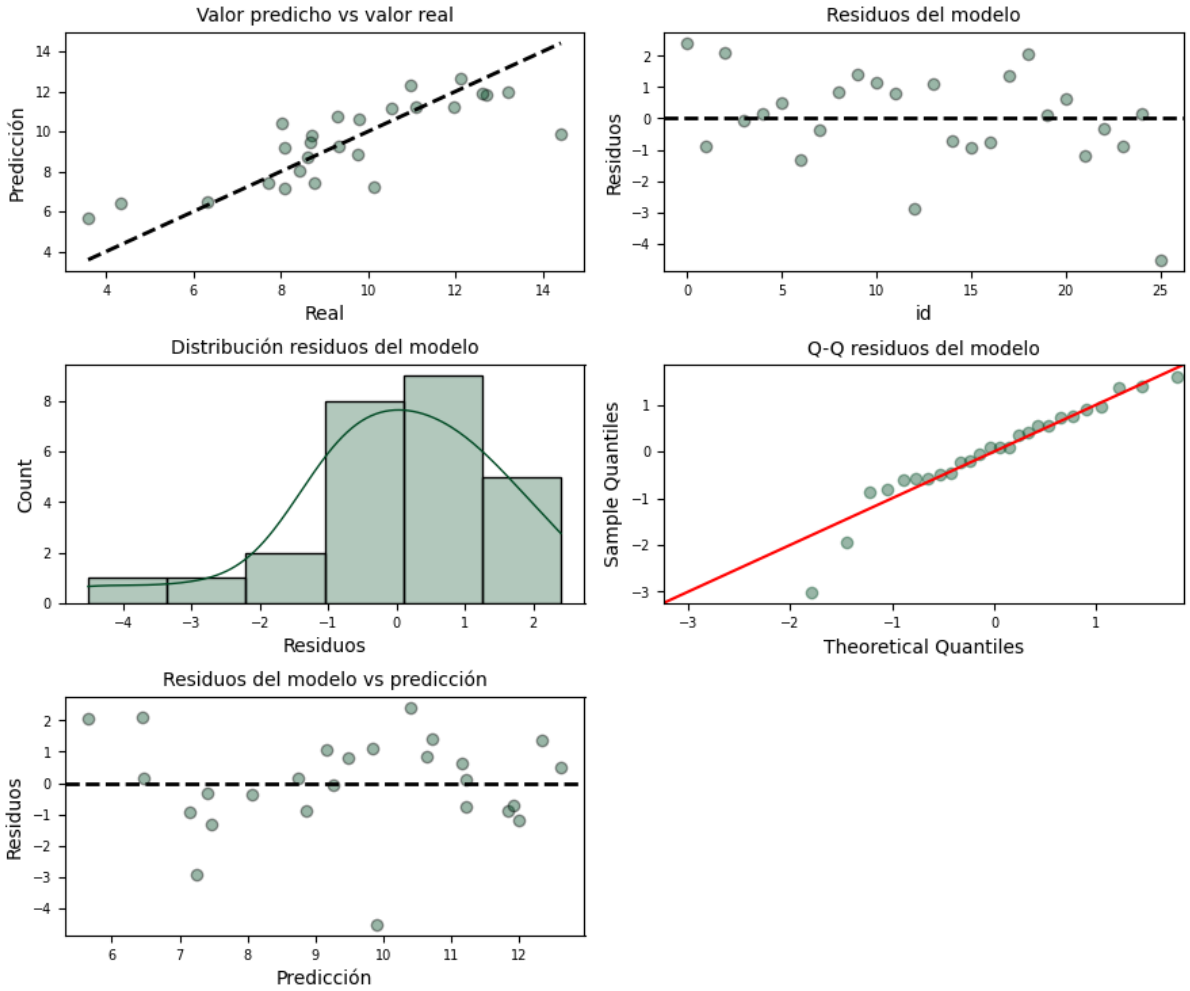

Figura 4.10: Diagnóstico de los residuos del modelo de regresión lineal múltiple de componentes principales.

4.2.3. Modelo con selección de variables

Existen distintos métodos para seleccionar las variables más relevantes y evitar problemas de multicolinealidad [112][113].

1. Métodos de Penalización (L1 - Lasso): El método Lasso es una técnica de regularización que agrega un término de penalización L1 a la función de pérdida en un modelo de regresión. Este término de penalización fuerza algunos coeficientes de regresión a ser exactamente cero. La principal característica del Lasso es que puede llevar a la selección automática de variables, ya que los coeficientes de algunas variables se vuelven cero, lo que significa que esas variables se eliminan del modelo. Es útil cuando se sospecha que muchas de las variables son irrelevantes o colineales.

2. Utilización de Coeficientes de Regresión: Este enfoque implica ajustar un modelo de regresión lineal utilizando todas las variables independientes y examinar los coeficientes

de regresión resultantes. Se evalúa la magnitud y el signo de los coeficientes para determinar la importancia relativa de cada variable. Se eliminan las variables con coeficientes cercanos a cero, ya que se asume que tienen un impacto limitado en la predicción.

3. Utilización de una métrica de validación (validación cruzada): En este método, se utiliza una métrica de evaluación, como el error cuadrático medio (MSE) o el coeficiente de determinación (R^2), para evaluar el rendimiento del modelo en diferentes conjuntos de datos de validación. Se prueban diferentes conjuntos de variables predictoras, y se selecciona el conjunto que produce el mejor rendimiento en términos de la métrica de evaluación. La validación cruzada es útil para evaluar la capacidad de generalización del modelo.

4. Método paso a paso (stepwise): El método stepwise es un enfoque de selección de variables que evalúa diferentes conjuntos de variables predictoras de manera iterativa. Se puede realizar en:

   I - Dirección forward: El modelo inicial no contiene ningún predictor, solo el parámetro β_0. A partir de este se generan todos los posibles modelos introduciendo una sola variable de entre las disponibles. Aquella variable que mejore en mayor medida el modelo se selecciona. A continuación, se intenta incrementar el modelo probando a introducir una a una las variables restantes. Si introduciendo alguna de ellas mejora, también se selecciona. En el caso de que varias lo hagan, se selecciona la que incremente en mayor medida la capacidad del modelo. Este proceso se repite hasta llegar al punto en el que ninguna de las variables que quedan por incorporar mejore el modelo.

   II - Dirección backward: El modelo se inicia con todas las variables disponibles incluidas como predictores. Se prueba a eliminar una a una cada variable, si se mejora el modelo, queda excluida. Este método permite evaluar cada variable en presencia de las otras. El método paso a paso requiere de algún criterio matemático para determinar si el modelo mejora o empeora con cada incorporación o extracción. Se utilizó el parámetro Akaike (AIC) que tiende a ser más restrictivo e introducir menos predictores que el R^2 ajustado.

Con el objeto de reducir la colinealidad de las variables predictoras, se utilizó el método stepwise que emplea criterios matemáticos para decidir qué predictores contribuyen significativamente al modelo y en qué orden se deben introducir.

4.2.4. Selección Stepwise - Dirección backward

Se aplicó el método de selección de variables predictoras en dirección backward, con el cual se identificó que el mejor modelo es el formado por los predictores NDVI, TCARI y TEMP.

Se procedió a ajustar un modelo de regresión lineal múltiple (RLM - 3) con los predictores identificados y se obtuvo un $R^2 = 0.64$ y un RMSE de 1.79.

Se corroboró la normalidad de los residuos con el test de Shapiro-Wilk (0.95) con un ρ-valor de 0.26

Se observa en los distintos gráficos que se cumplen los supuestos de linealidad, independencia y normalidad y homocedasticidad.

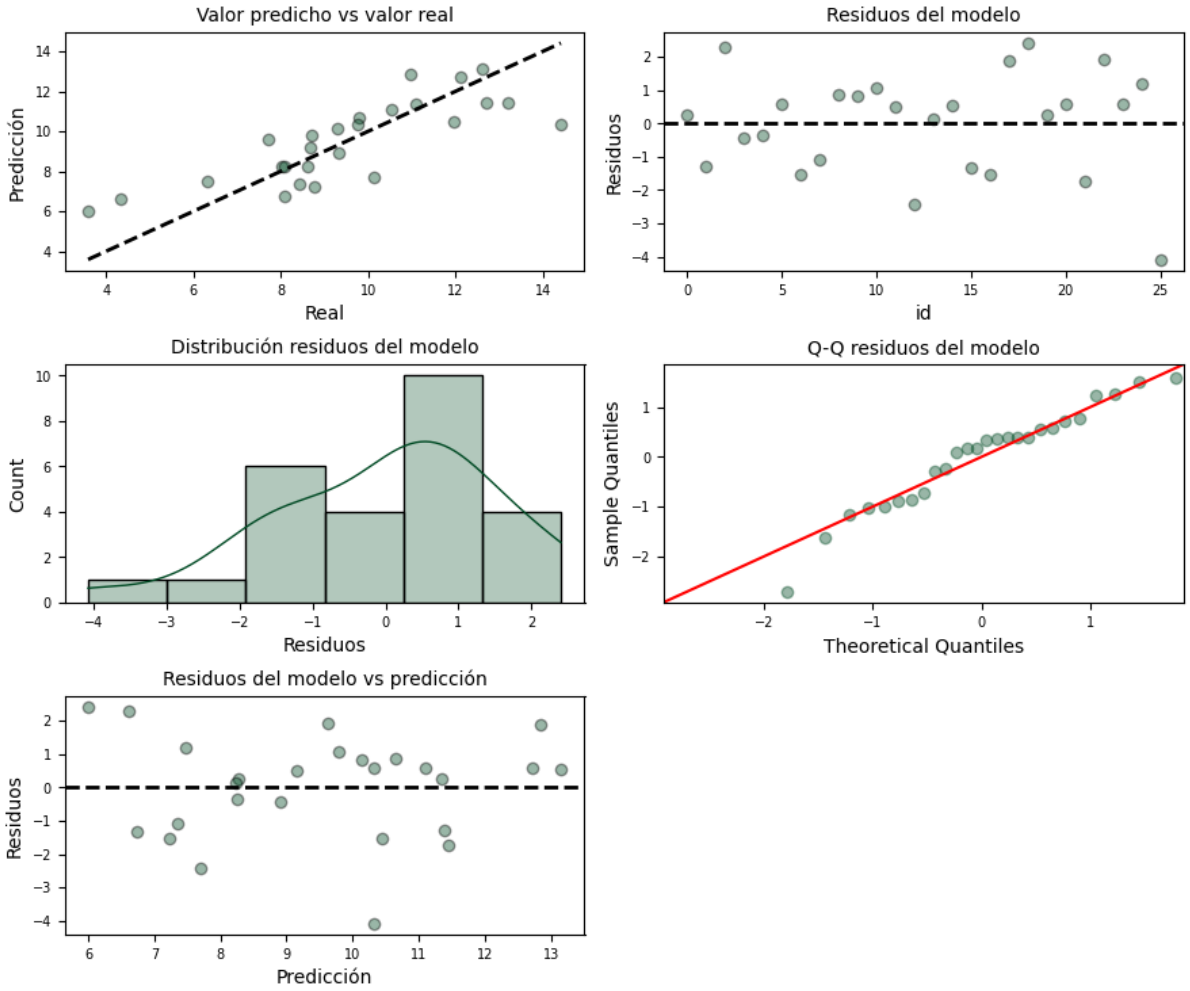

Figura 4.11: Diagnóstico de los residuos del modelo de regresión lineal múltiple con selección de variables por el método stepwise en dirección backward.

4.2.5. Selección Stepwise - Dirección forward

Se aplicó el método de selección de variables predictoras en dirección forward, con el cual se identificó que el mejor modelo es el formado por los predictores EVAPO, GNDVI y TCARI. Se procedió a ajustar un modelo de regresión lineal múltiple (RLM - 4) con los predictores identificados y se obtuvo un $R^2 = 0.97$ y un RMSE de 2.24.

Se corroboró la normalidad de los residuos con el test de Shapiro-Wilk (0.94) con un ρ-valor de 0.18

Se observa en los distintos gráficos que se cumplen los supuestos de linealidad, independencia y normalidad y homocedasticidad.

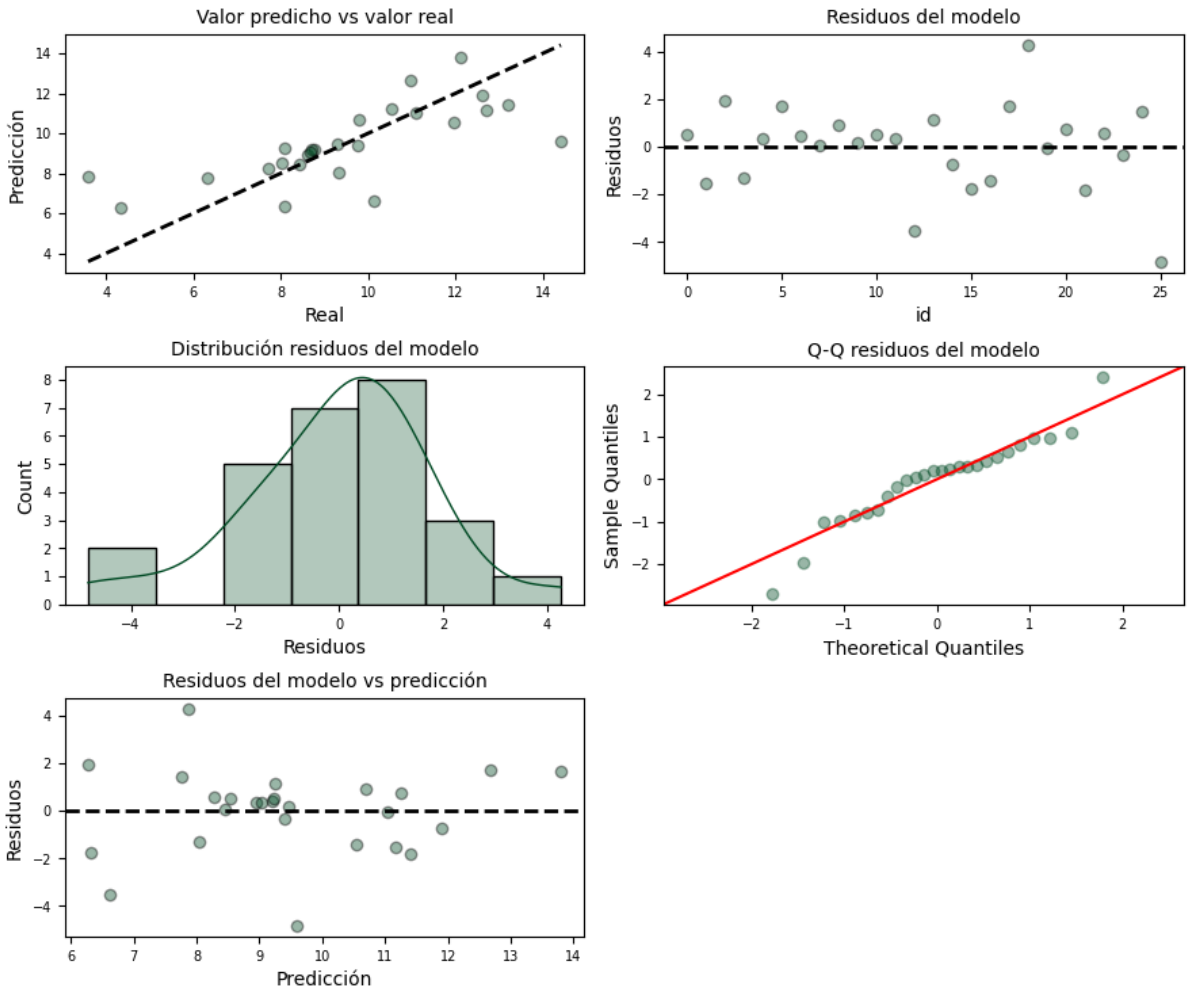

Figura 4.12: Diagnóstico de los residuos del modelo de regresión lineal múltiple con selección de variables por el método stepwise en dirección forward.

4.2.6. Modelo con interacciones

La interacción entre variables se refiere a la situación en la que el efecto de una variable independiente sobre la variable dependiente depende del nivel de otra variable independiente.

El aspecto teórico matemático de la interacción entre variables en un modelo de regresión lineal se puede explicar de la siguiente manera:

$$Y = a + b_1X_1 + b_2X_2 + b_3X_1 \cdot X_2 + e \tag{4.1}$$

donde: Y es la variable dependiente, X_1 y X_2 son variables independientes, b_1 y b_2 son los coeficientes de regresión de X_1 y X_2, b_3 es el coeficiente de regresión de la interacción entre X_1 y X_2, y e es el error.

La ecuación anterior indica que el efecto de X_1 sobre Y depende del nivel de X_2. El coeficiente b_3 mide el cambio en el efecto de X_1 sobre Y por cada unidad de cambio en X_2.

Se ajustó un modelo de regresión lineal múltiple (RLM - 5) agregando variables considerando las interacciones entre las variables que se identificaron previamente que contribuyen en ma-

yor medida a mejorar la performance del modelo mediante el método de stepwise (forward), EVAPO,GNDVI, TCARI. Se obtuvo un $R^2 = 0.73$ y un RMSE de 1.48.

Se corroboró la normalidad de los residuos con el test de Shapiro-Wilk (0.95) con un ρ-valor de 0.33

Se observa en los distintos gráficos que se cumplen los supuestos de linealidad, independencia y normalidad y homocedasticidad.

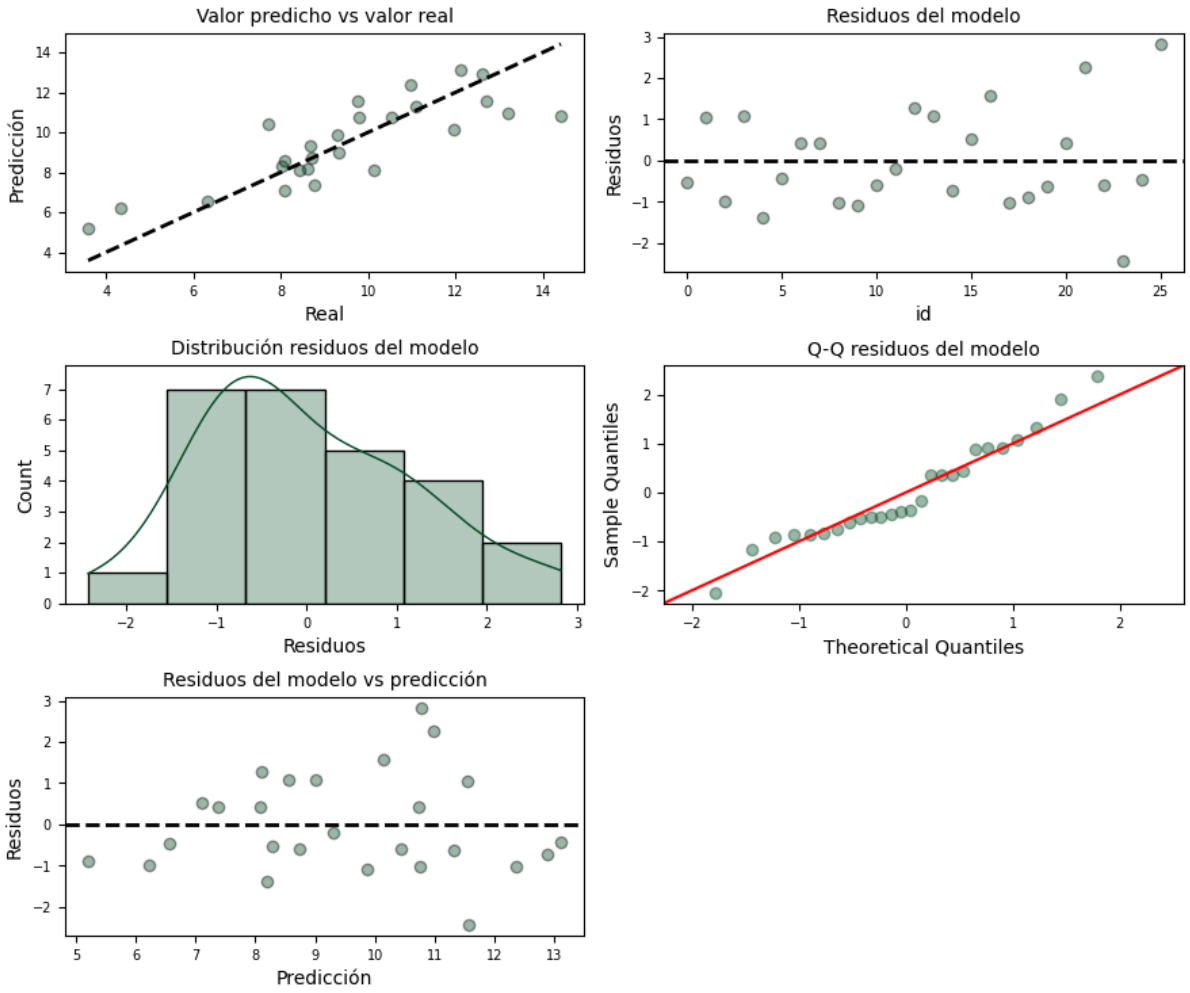

Figura 4.13: Diagnóstico de los residuos del modelo de regresión lineal múltiple con selección de variables por el método stepwise en dirección forward.

Tabla 4.1: Resumen de Resultados

Modelo	Tipo de Variables	R^2	RMSE	SW (est)	SW (ρ-valor)	Variables
RLM-1	Índices espectrales Climáticas Topográficas	0.68	1.83	0.97	0.71	gndvi, osavi, ndvi, reci, ndre, tcari, tcariosavi, pp, evapo, temp, dsm, slope
RLM - 2	PCA	0.64	1.80	0.94	0.11	pc1, pc2, pc3
RLM - 3	Selec. por stepwise backward	0.64	1.79	0.95	0.26	ndvi, tcari, temp
RLM - 4	Selec. por stepwise forward	0.97	2.24	0.94	0.18	evapo, gndvi, tcari
RLM - 5	Índices espectrales Climáticas Topográficas Interacción de las variables seleccionadas por stepwise forward	0.73	1.48	0.95	0.33	gndvi, osavi, ndvi, reci, ndre, tcari, tcariosavi, pp, evapo, temp, dsm, slope, evapo*gndvi, evapo*tcari, tcari*gndvi

El modelo de regresión lineal ajustado sobre el conjunto de datos correspondiente a las variables derivadas de sensores remotos focalizadas en tejidos vegetales (índices espectrales), las variables climáticas, las variables topográficas y que tiene en cuenta el efecto de las interacciones entre las variables EVAPO, GNDVI y TCARI resultó ser el modelo con mejor capacidad explicativa. El resultado (RMSE 1,48) es equivalente a decir que hay 1,48 unidades de distancia a la media, es decir 1,48 Tn/ha de error de predicción de la variable rendimiento de maíz por lote. Resultados similares a los encontrados en investigaciones realizadas utilizando modelos mecanísticos (DDSAT, CENTURY modelo de suelo) [114] [115].

Los modelos de RML demostraron su validez al satisfacer los supuestos fundamentales requeridos para su aplicación.

4.3. Modelo Random Forest Regression - RFR

Como ya se menciono en el capitulo 3 el procedimiento de ajuste de los modelos RFR se realizo de manera iterativa. En primer lugar se construyeron modelos con un conjunto de variables con valores iniciales para los hiperparámetros, que luego se optimizaron para mejorar el comportamiento del algoritmo. Luego se fue optimizando el conjunto de variables seleccionadas así como los hiperparametros para obtener el resultado más robusto y preciso posible.

En la Tabla 4.4 se compilaron los parámetros ajustados y los resultados obtenidos con cada uno de los modelos de random forest construido.

4.3.1. Modelo inicial

En el modelo inicial el valor de los hiperparámetros fue el siguiente:

1. n_estimators: Se especificaron 24 árboles.
2. criterion: El criterio utilizado para medir la calidad de una división. En este caso, se utilizó 'squared_error', que es equivalente a 'mse' (mean squared error).
3. max_depth: La profundidad máxima de los árboles. Aquí, se dejó como None, lo que significa que los árboles se expandirán hasta que contengan menos de min_samples_split muestras por hoja.
4. max_features: El número de características a considerar al buscar la mejor división. Se consideraron las 28 características (variables predictoras).
5. oob_score: Se estableció en False, lo que significa que no se calculó el error out-of-bag.
6. n_jobs: El número de trabajadores utilizados para entrenar el modelo -1 significa que se utilizaron todos los núcleos disponibles en la CPU.
7. random_state: Al igual que en la división de datos, se utilizó para garantizar la reproducibilidad del modelo.

Una vez establecidos los parámetros se entrenó el modelo utilizando los datos de entrenamiento X_train e y_train. El modelo Random Forest aprendió a hacer predicciones basadas en las características proporcionadas.

Este modelo entrenado, denominado RFR - 1, se utilizó para hacer predicciones en el conjunto de datos de prueba (X_test). Se calculó el error de las predicciones: 2.21 (RMSE) y 0.30 (MAPE) y se evaluó el gráfico de resultados predichos versus observaciones reales (Figura 4.14)

El promedio de la variable respuesta en el conjunto de la base de datos es 9.27 Tn/Ha. Lo que nos indica que el modelo RFR - 1 (RMSE:2,21; Tabla 4.4) tiene un 23,8 % de error, es decir el modelo cuenta con una capacidad explicativa del 76.2 % de la variabilidad de la variable rendimiento.

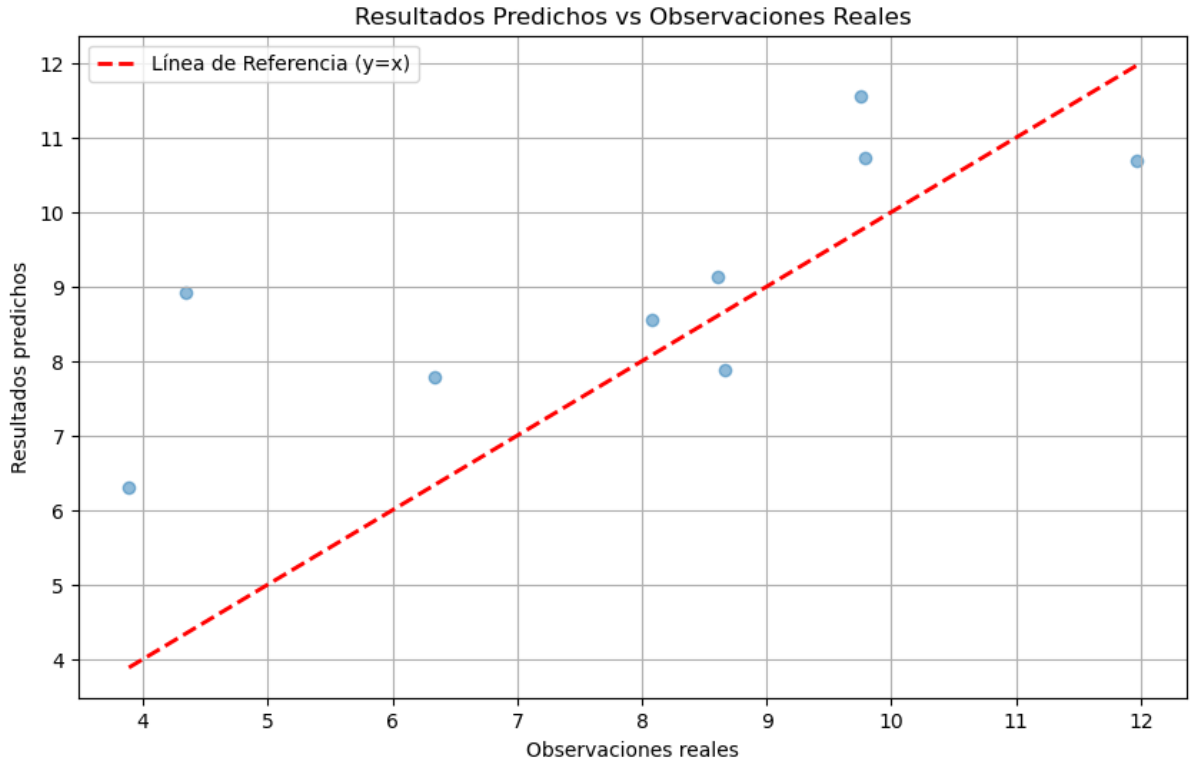

Figura 4.14: Resultados obtenidos con el modelo inicial random forest evaluados en término de las observaciones de la variable dependiente utilizadas en el testeo.

En la Figura 4.14 se observa que los resultados predichos se encuentran próximos a la curva de referencia (y=x) para valores de rendimiento medio (6 a 10) mientras que para valores extremos (4 a 5) se observa una sobreestimación y para valores cercanos a 12 se observa una subestimación. Con el objeto de mejorar la performance del modelo y evitar la subestimación de la variable respuesta, se procedió a optimizar los hiperparámetros del modelo.

4.3.2. Modelo optimizado

El objetivo en la optimización de hiperparámetros es determinar el valor óptimo para cada hiperparámetro. Se trabajo optimizando uno por vez.

4.3.2.1. Optimización del número de árboles

Se realizo en primer lugar una optimización del número de árboles (n_estimators) por el método Out-of-Bag (OOB). En la Figura 4.15 se observó que si bien el resultado se optimiza

rápidamente con un número bajo de árboles (<20) el método continuó buscando y se detuvo en el máximo valor de la métrica OOB, equivalente a una cantidad de 112 árboles. En cuanto al rendimiento del modelo en los datos de entrenamiento (train-scores) se observa que se mantiene relativamente constante a lo largo de toda la iteración.

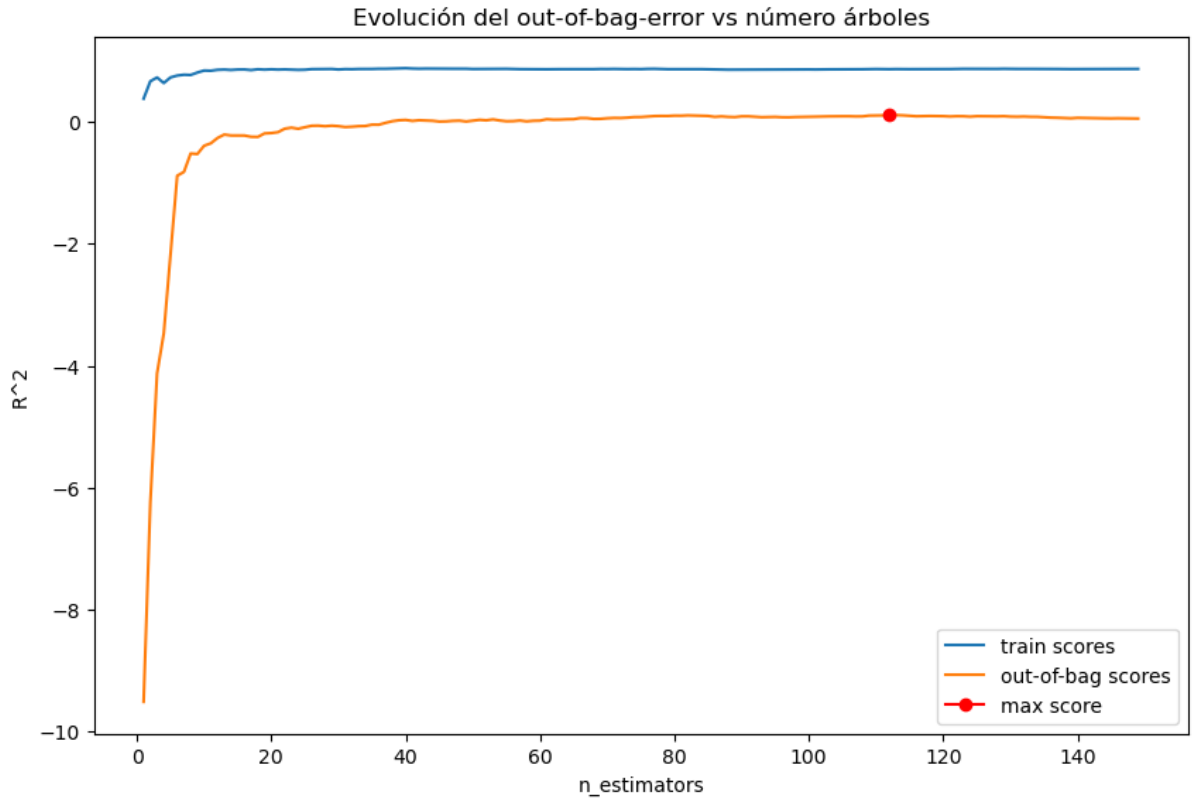

Figura 4.15: Gráfico de evolución de la métrica OOB, a medida que se incrementa el número de árboles del modelo RFR. Se identifica que se optimiza el resultado (max score) con una cantidad de 112 árboles (n_estimators).

En segundo lugar se realizó una optimización del número de árboles por el método de validación cruzada. En la Figura 4.16 se observa que 5 es la cantidad de árboles que minimiza el error de validación cruzada (min-score).

El método OOB se basa en una estimación interna durante el entrenamiento mientras que la validación cruzada es una técnica externa que se utiliza para evaluar el rendimiento del modelo de manera más general y robusta.

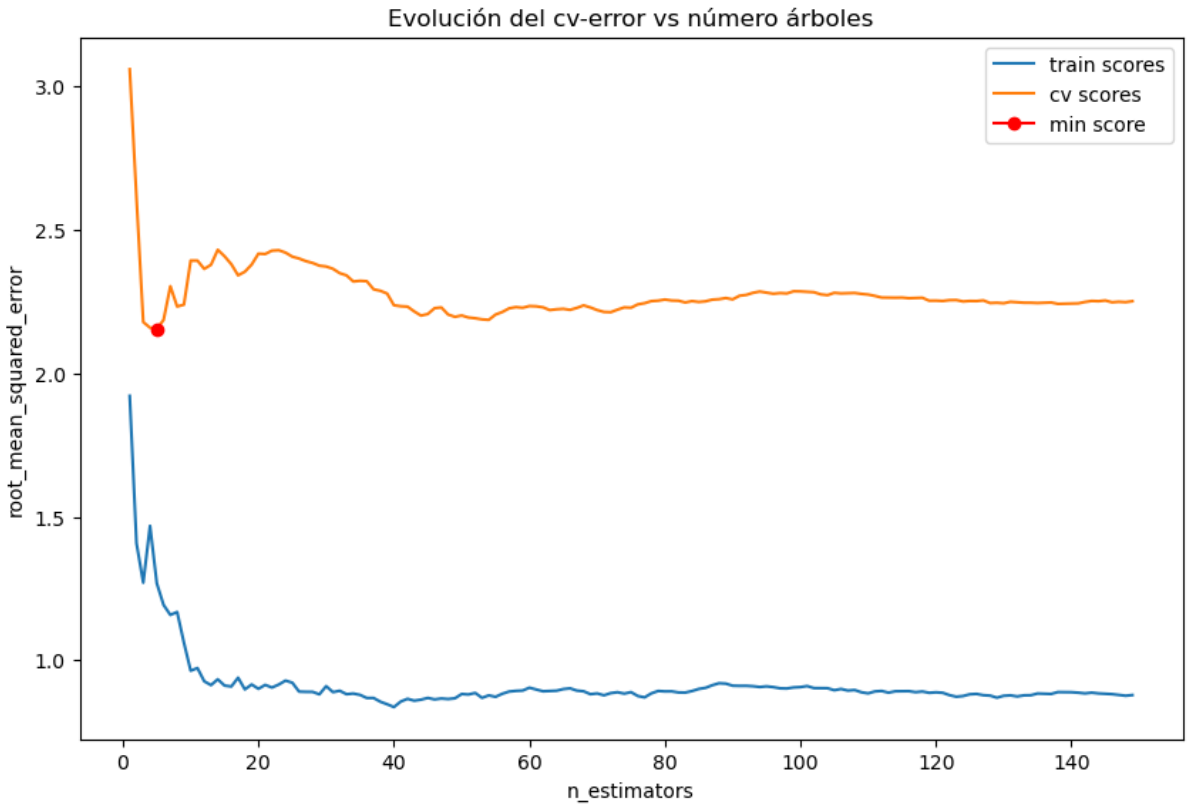

Figura 4.16: Gráfico de evolución de la métrica CV, a medida que se incrementa el número de árboles del modelo RFR. Se identifica que se optimiza el resultado (min score) con una cantidad de 5 árboles (n_estimators).

4.3.2.2. Optimización del la cantidad máxima de características

Max_features, es el hiperparámetro que controla la cantidad de características que se consideran al dividir los nodos de cada árbol. En su optimización se emplearon en primer lugar el método OOB y luego el método CV. En la Figura 4.17 se observó que el máximo valor de la métrica OOB se alcanza con 4 características, es decir con 4 variables. En cuanto al rendimiento del modelo en los datos de entrenamiento (train-scores) se observa que se mantiene relativamente constante a lo largo de toda la iteración.

Mientras que con la estimación OOB se obtuvo un valor de 4 características (variables) con el método validación cruzada se obtuvo un valor de 25 variables. Estas optimizaciones sugieren que con entre 4 y 5 variables se obtienen los mejores resultados con el modelo.

4.3.2.3. Optimización simultánea de número de árboles y cantidad máxima de características

Se realizó una búsqueda exhaustiva en un conjunto de posibles de valores de hiperparámetros para encontrar la combinación de hiperparámetros que obtenga el mejor rendimiento del modelo. En primer lugar se realizo utilizando la métrica OOB y en segundo lugar se realizo utilizando la métrica CV. En ambos métodos los valores de optimización de los parámetros obtenidos fueron: 'max_features': 7, 'n_estimators': 150. Se realizo un ajuste de RFR con cada combinación de parámetros resultantes de la optimización y se compilaron los resultado (RMSE) en la tabla 4.2). Se observó que el mejor resultado se obtuvo con el método de optimización OOB, que

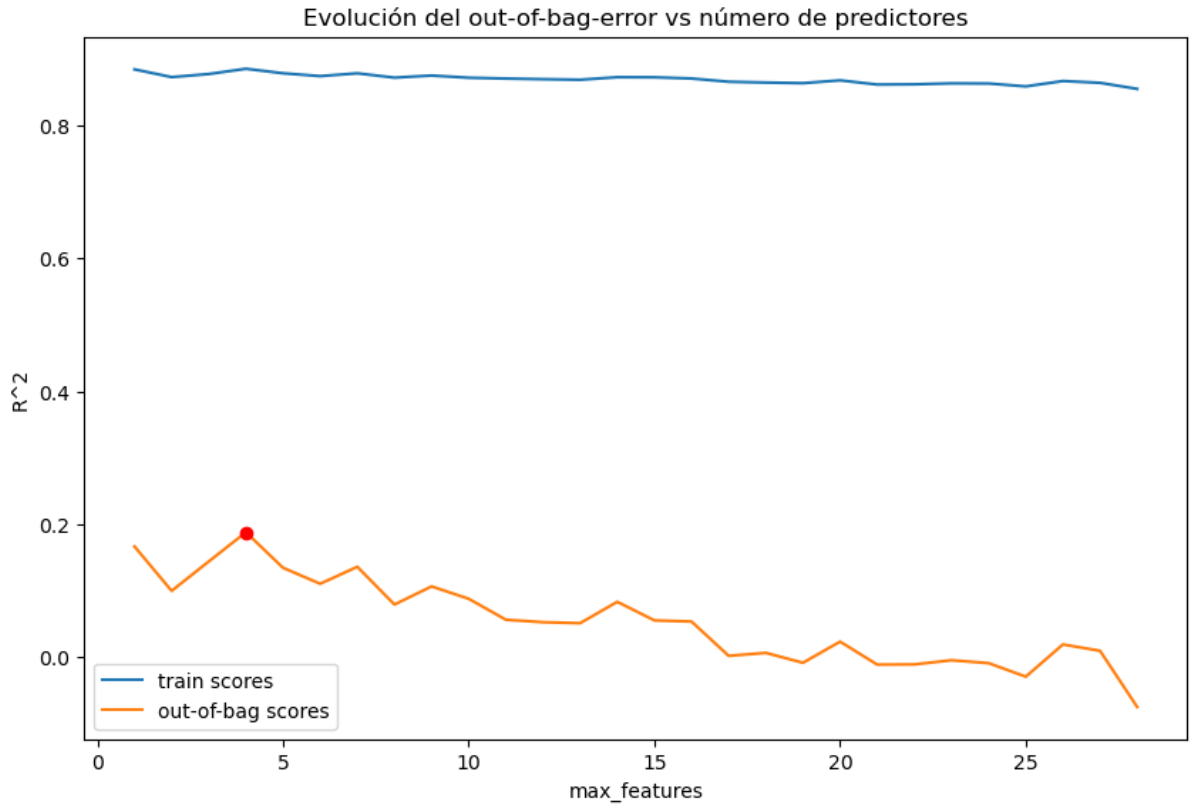

Figura 4.17: Gráfico de evolución de la métrica OOB, a medida que se incrementa la cantidad de características del modelo RFR. Se identifica que se optimiza el resultado (max_features) con una cantidad de 4 características.

indicó utilizar 112 árboles y 7 características.

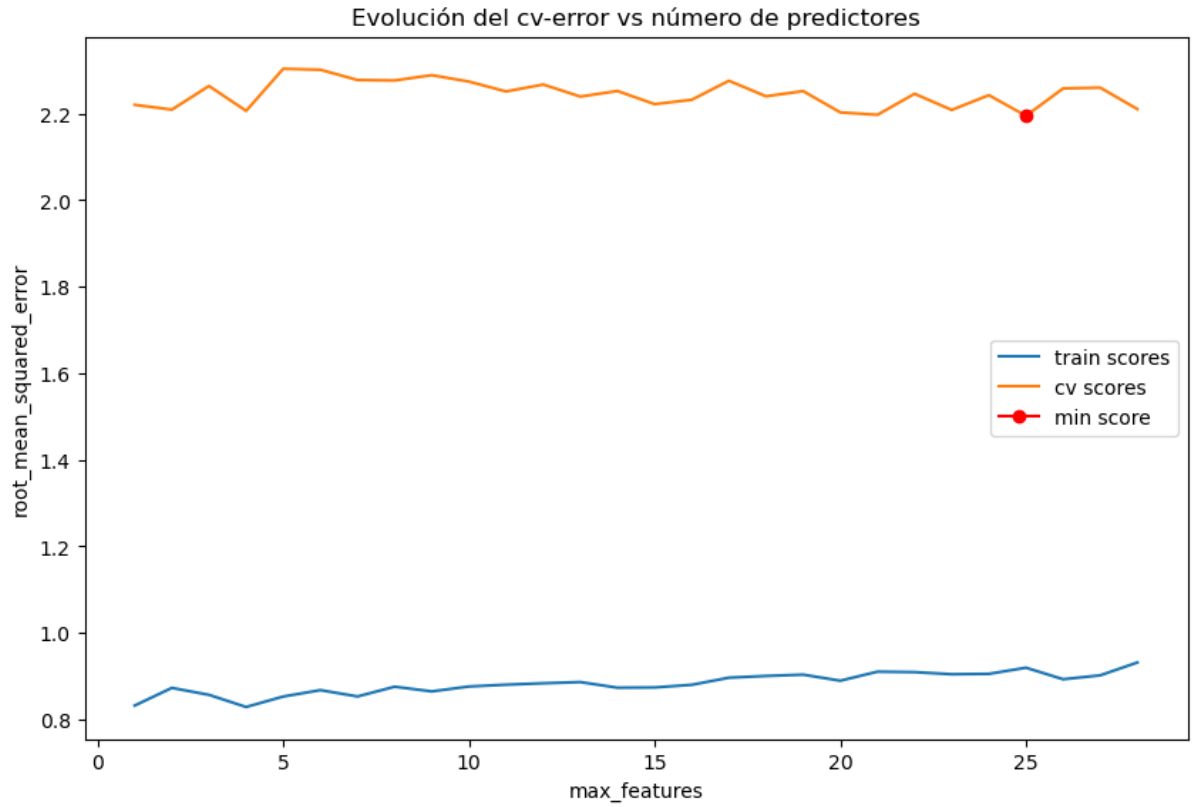

Figura 4.18: Gráfico de evolución de la métrica CV, a medida que se incrementa la cantidad de características del modelo RFR. Se identifica que se optimiza el resultado (min score) con una cantidad de 25 características.

Tabla 4.2: Tabla comparativa del resultado (RMSE) obtenido con cada combinación de parámetros optimizados.

	OOB	VC	Grid Search (OOB)	Grid Search (VC)
Arboles (n_estimators)	112	4	150	150
Variables (max_features)	7	25	7	7
RMSE	2.52	3.11	2.55	2.55

El modelo optimizado (RFR-2) mediante el método OOB obtuvo el mejor resultado con un RMSE de 2.52. (Tabla 4.4) En la Figura de 4.19 se observa que la distancia de los valores predichos a la curva de referencia es mayor que en el modelo sin optimizar. Esto podría indicar que los valores iniciales de los hiperparámetros utilizados en el modelo sin optimización eran adecuados para el conjunto de datos. A su vez, a medida que se optimizan los hiperparámetros, es común encontrar que los incrementos en el rendimiento se vuelven cada vez más marginales. Esto puede hacer que los cambios no tengan un impacto significativo en la métrica de evaluación.

Este resultado sugiere por un lado que el modelo inicial, sin ningún tipo de optimización, ya estaba funcionando de manera eficiente y por otro que en un conjunto de datos compuesto de n=33, puede que no sea necesario ni conveniente llevar a cabo una optimización exhaustiva de los parámetros. Se procedió a seleccionar las variables a utilizar en un siguiente ajuste del modelo utilizando como criterio de selección la importancia individual (el peso) de las variables predictoras en el modelo inicial sin optimizar con el cual se obtuvo una mejor performance.

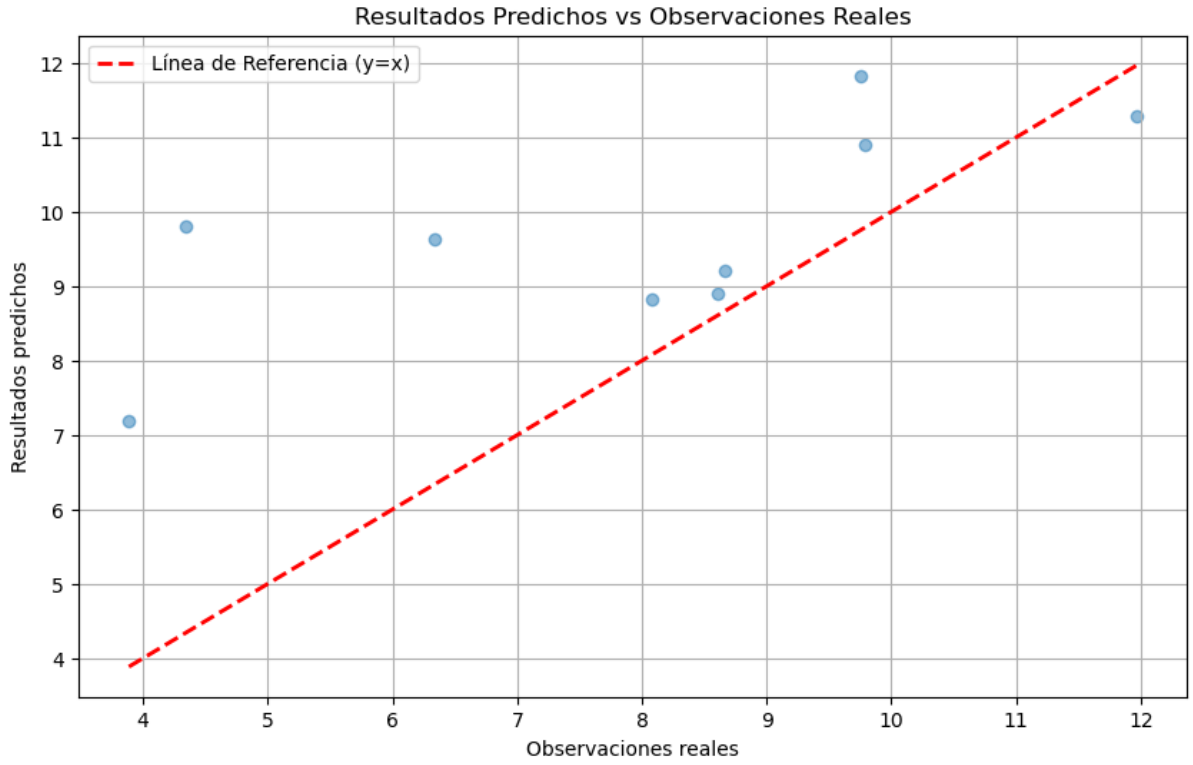

Figura 4.19: Resultados obtenidos con el modelo optimizado random forest evaluados en término de las observaciones de la variable dependiente utilizadas en el testeo.

4.3.3. Modelo con variables seleccionadas (importancia)

4.3.3.0.1. Importancia de las variables predictoras por pureza de nodos. Las importancias de las variables predictoras se encuentran calculadas dentro del modelo ajustado, específicamente en el atributo"feature_importances_" y se calculan como la media y la desviación estándar de la acumulación de la disminución de impurezas dentro de cada árbol. Del modelo RFR-1 Se extrajeron los valores de feature_importance asociado a cada variable (característica) y se realizo el gráfico de importancia (Figura 4.20) . Se observó que las variables TCARI_stde, NDRE_mean y PP se destacan por su importancia, en un segundo lugar se ubican GNDVI_mean, TARIOSAVI_mean, finalmente TEMP, EVAPO, TCARI_mean y TCARI_range completan el grupo de variables que se encuentran por encima de 0.05 de importancia, (equivalente al 30 % de la importancia total)

4.3.3.0.2. Importancia de las variables por permutación. Para calcular la importancia por permutación se utiliza la función "permutation_importance". Tienen una serie de parámetros que se deben ajustar, principalmente:.

1. n_repeats: Es el número de permutaciones aleatorias que se realizan para cada característica.Se utilizaron 5 permutaciones.
2. scoring: Es la métrica utilizada para evaluar el rendimiento del modelo. Se utilizó neg_root_mean_squared_error, lo que significa que se está calculando el error cuadrático medio negativo de las predicciones.

Los datos obtenidos se almacenaron en una base de datos que luego se utilizó para construir

el gráfico (Figura 4.21) de importancia por permutación para evaluar la influencia de cada característica.

Se observó que las variables TCARI_stde y NDRE_mean se destacan por su importancia, en un segundo lugar se ubican las variables TARIOSAVI_mean, PP y GNDVI_mean, finalmente TEMP, EVAPO, TCARI_mean, TCARI_range, DSM y TCARIOSAVI_stde completan el grupo de variables que se encuentran por encima de 0.05 de importancia, (equivalente al 10 % de la importancia total)

Del análisis de ambas métricas se obtuvo que las características más importantes son:

Tabla 4.3: Tipos y Nombres de Variables

Tipo	Nombre
Índice espectral	TCARI_stde
Índice espectral	NDRE_mean
Climática	PP
Índice espectral	TCARIOSAVI_mean
Índice espectral	GNDVI_mean
Climática	TEMP
Climática	EVAPO
Índice espectral	TCARI_mean
Índice espectral	TCARI_range
Topográfica	DSM
Índice espectral	TCARIOSAVI_stde

El modelo de las variables seleccionadas a partir de su grado de contribución individual (RFR - 3) se ajustó con 24 árboles y 12 características y se obtuvo un error de 2.27 (RMSE) y de 0.33 (MAPE). Se realizo la optimización de sus parámetros (RFR - 4) y se ajustó nuevamente con 150 árboles y 7 características, obteniéndose 2.31 (RMSE) y 0.32 (MAPE). Tabla 4.4.

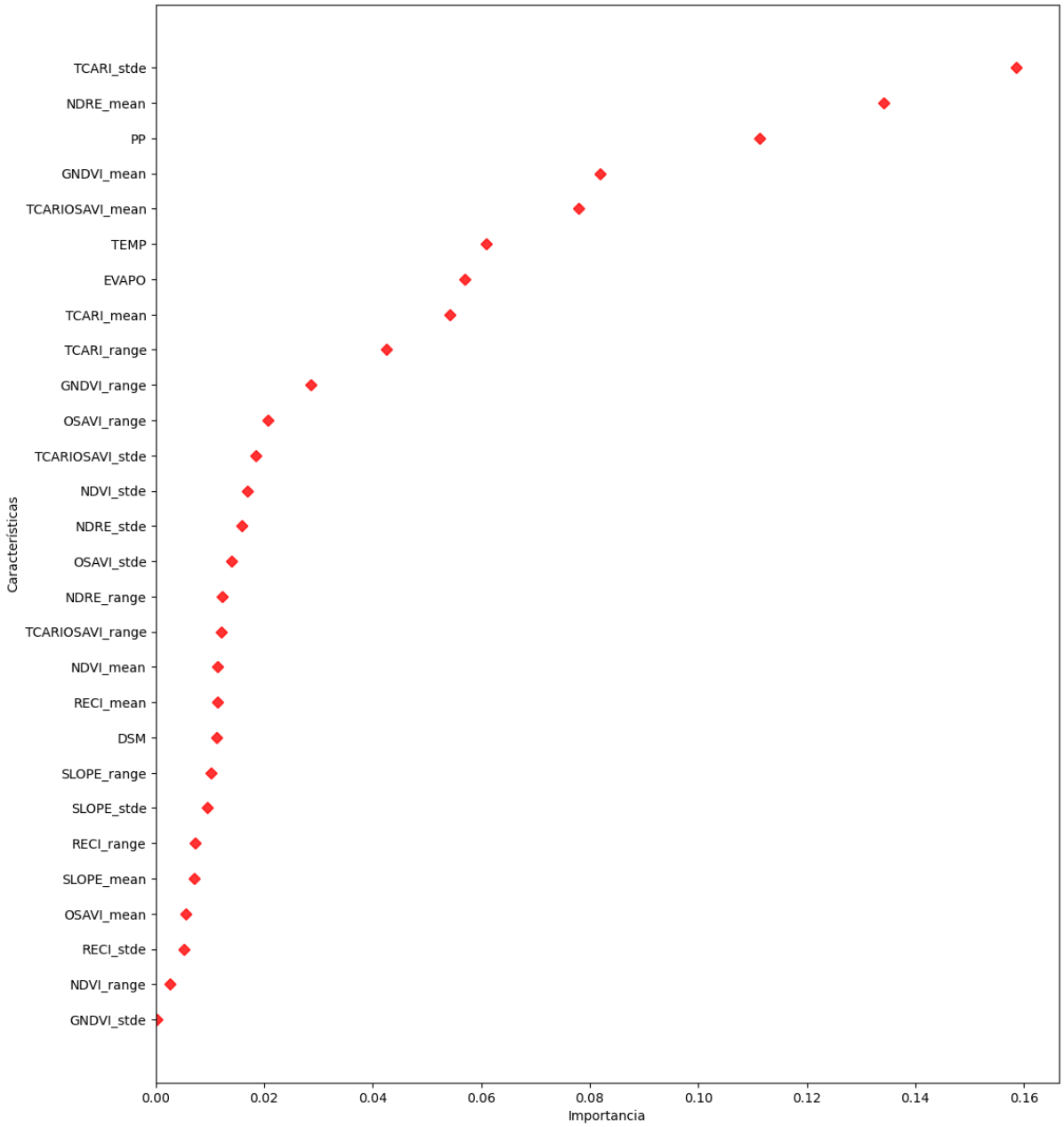

Figura 4.20: Importancia de las variables predictoras según la pureza del nodo final.

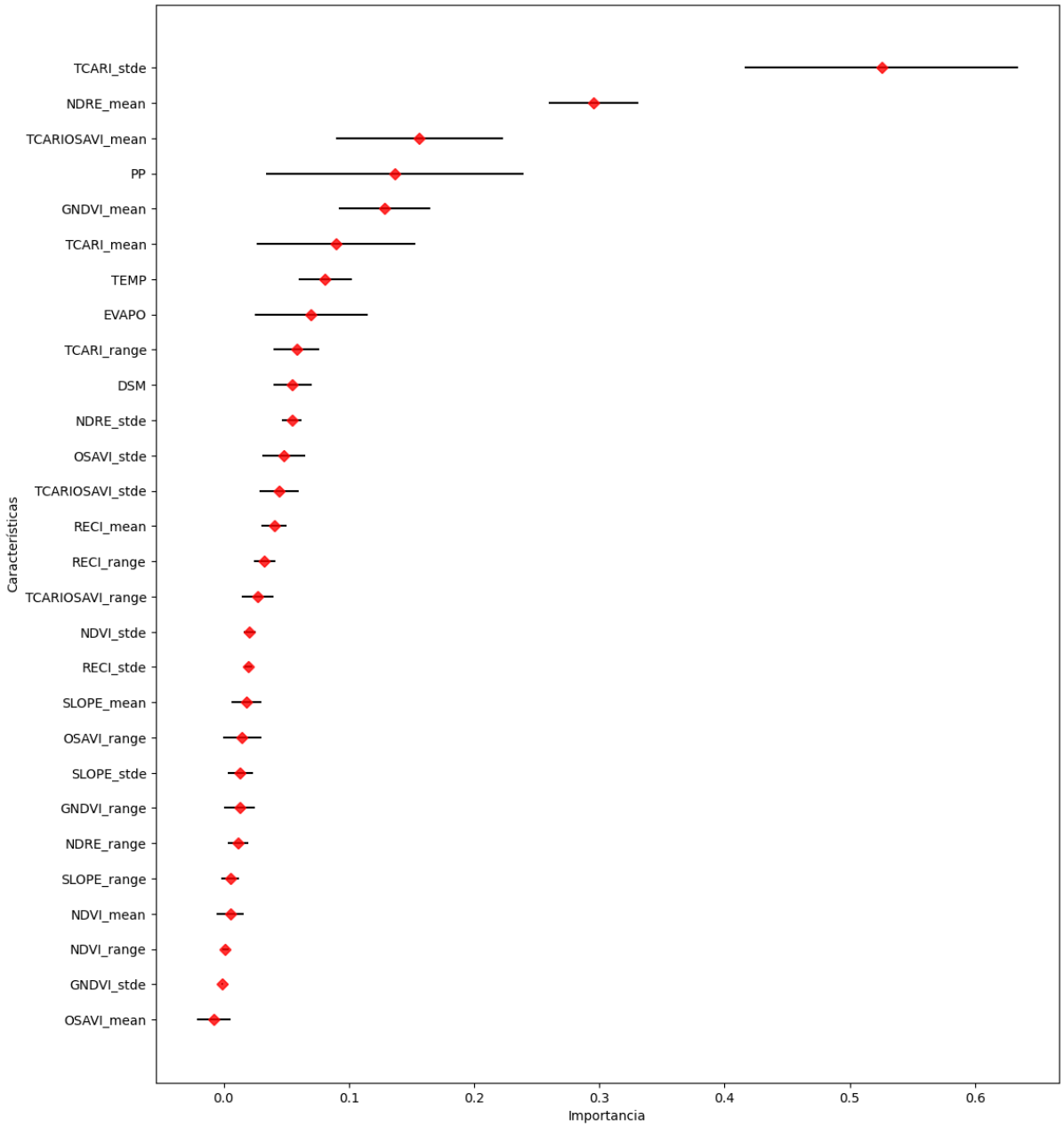

Figura 4.21: Importancia de los predictores según cuánto empeora el rendimiento del modelo cuando se permutan los valores de cada predictor.

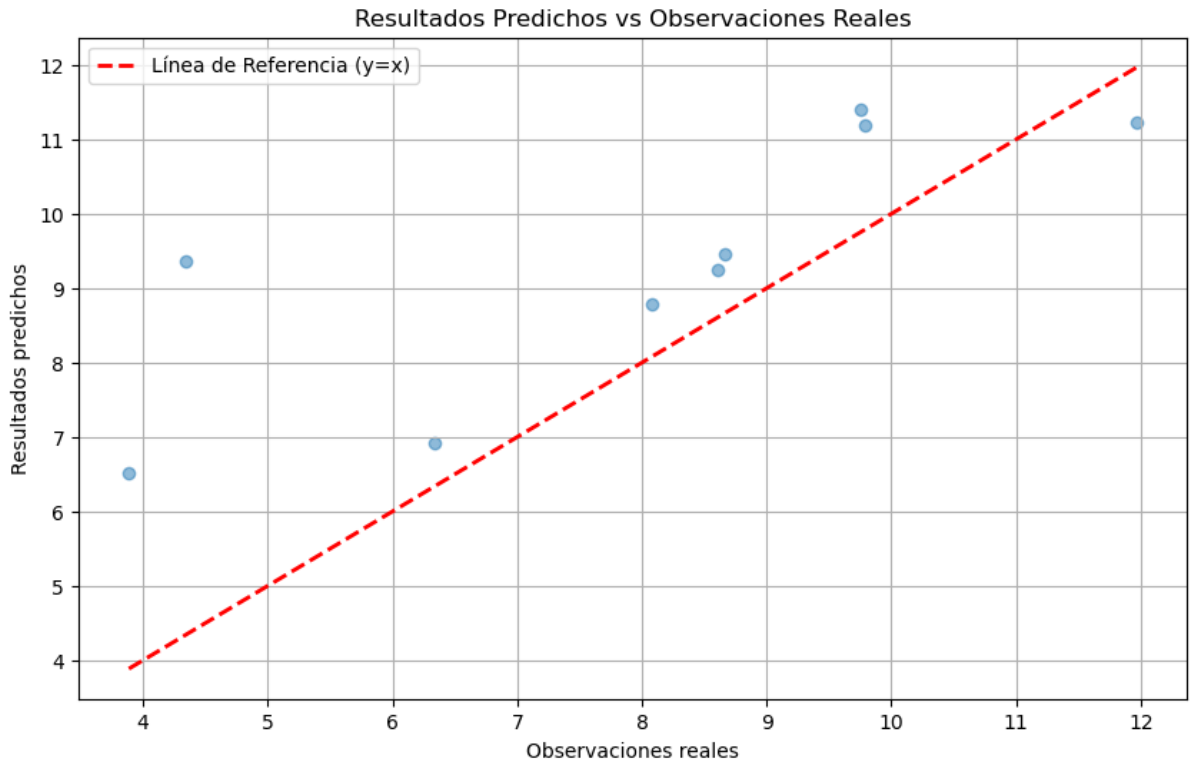

Figura 4.22: Resultados obtenidos con el modelo de las variables seleccionadas por importancia.

Se observó que el método de selección de variables no permitió disminuir el error, ni acortar la distancia de los valores predicho en la curva de referencia en el gráfico de resultado predicho versus observaciones reales, lo que sugiere que para la cantidad y tipo de datos los métodos usados no funcionaron correctamente.

Se observó nuevamente que la optimización de los parámetros en este conjunto de datos (n=33) no mejora el resultado del modelo

Se procedió a seleccionar las variables a utilizar en un siguiente ajuste del modelo empleando como criterio de selección la variables con menor colinealidad identificada en el análisis exploratorio 4.7)

4.3.4. Modelo con variables seleccionadas (análisis exploratorio)

Los modelos de RFR son robustos frente a la multicolinealidad. Esto significa que pueden manejar cierto grado de correlación entre las variables predictoras sin sufrir una degradación significativa en su rendimiento.

Aún así ante la existencia de multicolinealidad, los modelos de RFR tienden a asignar una importancia compartida entre las variables altamente correlacionadas. Esto significa que si dos variables están fuertemente relacionadas, ambas pueden recibir importancia en el modelo, pero la suma de sus importancias no será necesariamente mayor que la de una sola variable no correlacionada. El modelo considera la información redundante y distribuye la importancia en consecuencia. Esto dificulta determinar cuál variable es más relevante para obtener conclusiones agronómicas. Por tanto, puede ser útil reducir la dimensionalidad de los datos si hay muchas variables altamente correlacionadas. Esto simplifica el modelo y acelera su tiempo de entrenamiento. [112] [113].

Con el objeto de identificar si la existencia de colinealidad entre las variables predictoras identificada en el análisis exploratorio está dificultando el ajuste del modelo, se procedió a ajustar un modelo solo con las variables medias (mean) de los índices espectrales y de la variable pendiente (slope), más las variables climáticas y de altitud (DSM).

Se realizó un modelo (RFR -5) solo con las siguientes 12 variables: GNDVI, OSAVI, NDVI, RECI, NDRE, TCARI, TCARIOSAVI, PP, EVAPO, TEMP, DSM, SLOPE. Con los parámetros del modelo establecidos en 12 árboles y 10 características, se obtuvo un error de 1.87 (RMSE) y 0.24 (MAPE. Se procedió a optimizar los parámetros (RFR - 6) y se ajustó nuevamente con 150 árboles y 7 características obteniéndose un error de 1.99 (RMSE) y 0.28 (MAPE). Tabla 4.4.

El modelo ajustado con las variables seleccionadas obtuvo un mejor resultado, indicando que si bien el modelo RFR por lo general no es sensible a la multicolinealidad en conjuntos de datos con n chico la reducción de dimensionalidad permitió una mejora en su performance. Nuevamente se corroboró que la optimización de hiperparametros no funciona en situaciones de poca cantidad de datos.

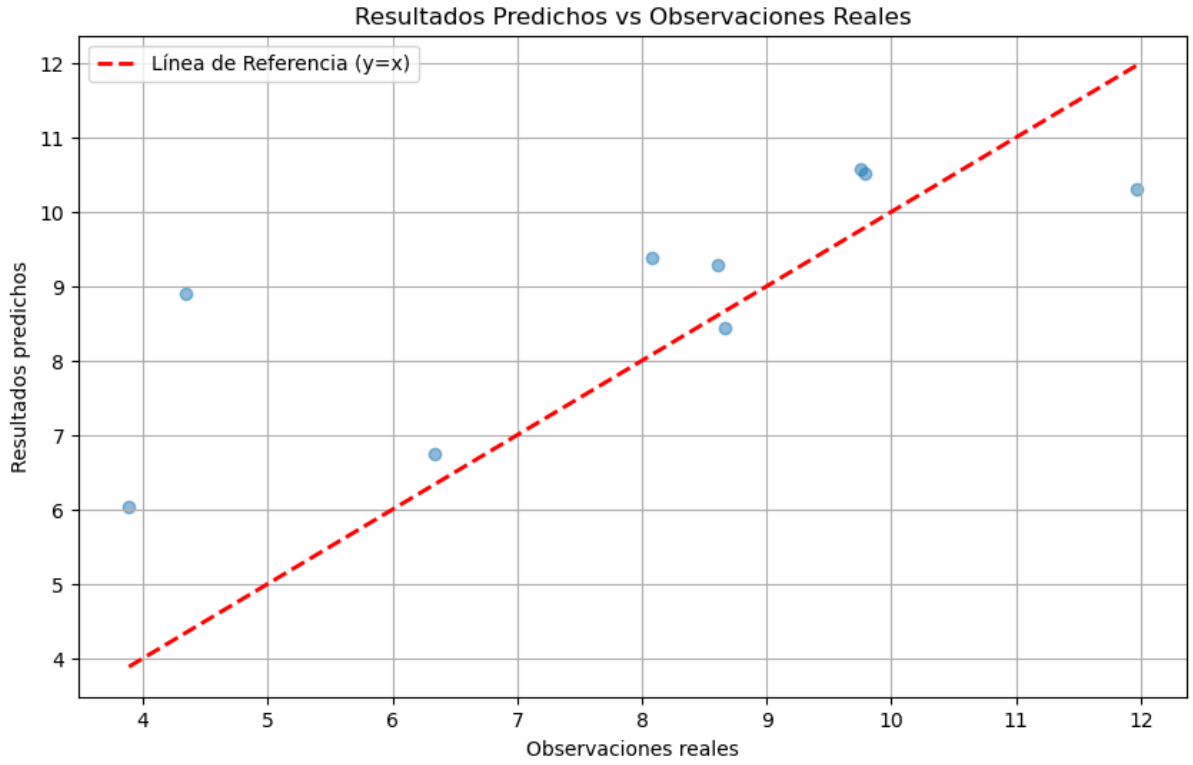

Figura 4.23: Resultados obtenidos con el modelo de variables medias.

Tabla 4.4: Resultados obtenidos en los modelos Random Forest Regression

Modelo	Variables	N° árboles	Características	RMSE	MAPE
RFR - 1	Todas	24	28	2.21	0.30
RFR - 2	Todas optimizado	150	7	2.52	0.35
RFR - 3	Seleccionadas	24	12	2.27	0.33
RFR - 4	Seleccionadas optimizado	150	7	2.31	0.32
RFR - 5	Medias	24	10	1.87	0.24
RFR - 6	Medias optimizado	150	7	1.99	0.28

Con el objeto de comprender cómo las diferentes variables afectan la predicción del rendimiento del maíz, se calcularon las relaciones parciales.

4.3.5. Relaciones parciales entre las variables

Se realizaron los gráficos de dependencias parciales para cada una de las 12 variables del modelo random forest con el cual se obtuvo el mejor resultado (RFR-5).

1. En el eje X de cada gráfico, se representó el rango de valores de la variable predictora específica que se analizó.
2. En el eje Y, se representó el efecto de esa variable en las predicciones del modelo. La escala de valores numéricos asociada al rango de valores de la variables respuesta, rendi-

miento de maíz en toneladas por hectárea.

Finalmente se realizo una interpretación visual de la información presentada observando el comportamiento de la curva del gráfico que representa cómo cambian las predicciones del modelo a medida que varía el valor de la variable predictora en el eje X. Si la curva es plana o casi plana, indica que esa variable tiene poco impacto en las predicciones. Si la curva es pronunciada, muestra que la variable tiene un impacto significativo. [120]

4.3.5.0.1. Dependencias parciales: GNDVI, NDRE, NDVI, OSAVI. En la Figura 4.24) se observó que el rendimiento responde positivamente (mayor GNDVI mayor rendimiento) en gran medida a la variabilidad de la variable GNDVI, específicamente entre los valores 0.5 y 0.6. Luego a mayores valores ya no parecía responder. Asimismo se observa que a medida que la variable NDRE incrementa de valor por encima de 0.45 el rendimiento se incrementa hasta 0.6 donde se ameseta el crecimiento.

La variable NDVI no evidencia tener una dependencia con rendimiento de maíz, mientras que para el caso de la variables OSAVI a medida que esta aumenta por encima de 0.5 el rendimiento disminuye.

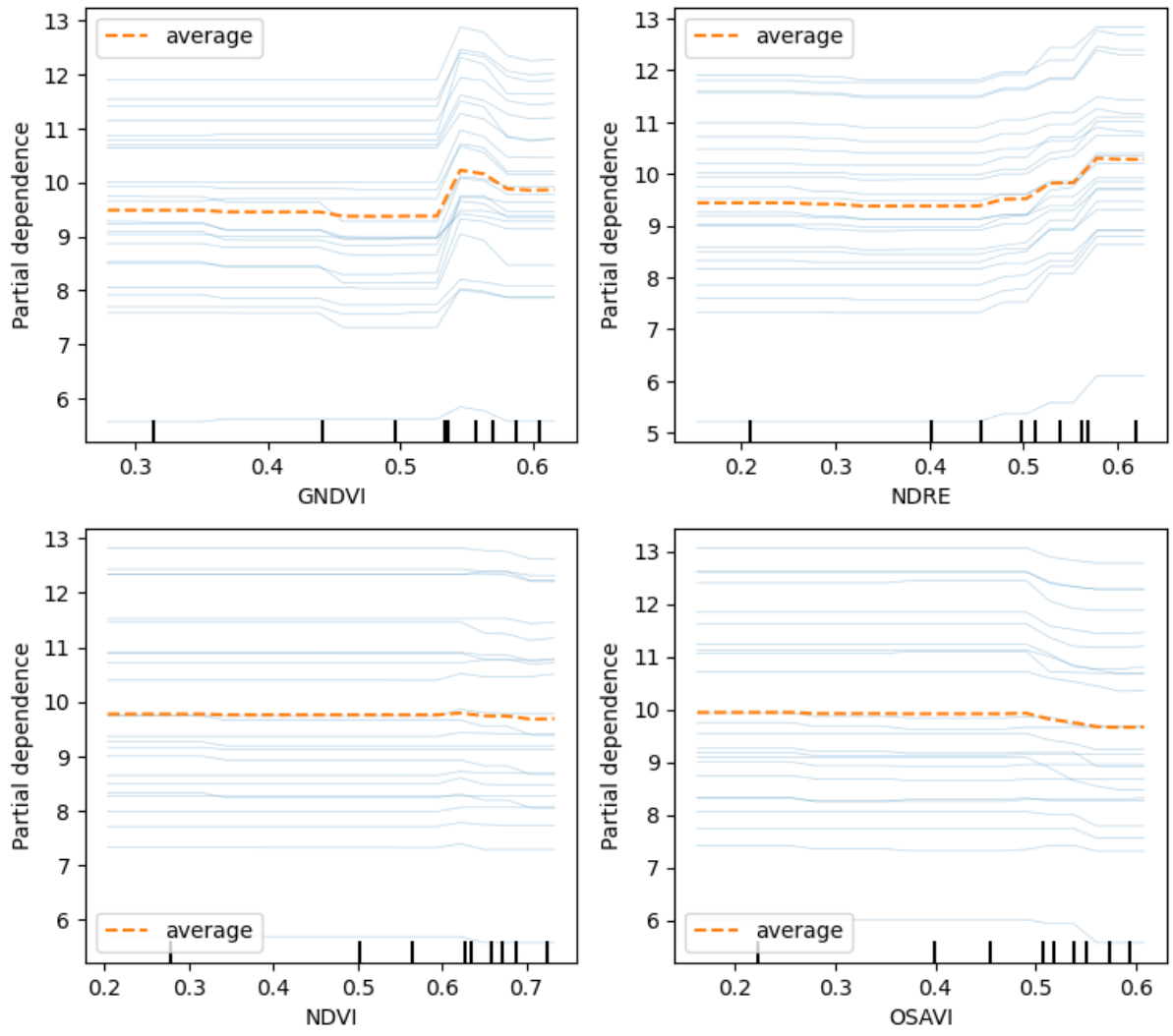

Figura 4.24: Gráficos de dependencias parciales para los índices espectrales GNDVI, NDRE, NDVI, OSAVI.

4.3.5.0.2. Dependencias parciales: SLOPE, RECI , TCARI, TCARIOSAVI . En la Figura 4.25) se observa que la variable RECI influye positivamente en el rendimiento particularmente por encima de 2, hasta 3.3 donde el efecto se torna negativo. Por otro lado la variable que caracteriza la pendiente (SLOPE) evidentemente que se obtiene mejores rendimiento en pendientes de 1 % y a medida que esta aumenta va disminuyendo el rendimiento.

El índice TCARIOSAVI es la variable que evidencia mayor influencia sobre el rendimiento, identificándose que para valores de 0.14 el maíz rinde 10,5 Tn/Ha y en valores de 0.16 el rendimiento ya es de 9 Tn/Ha. Es decir, a medida que aumenta el valor de este índice el rendimiento disminuye, siendo consistente esto con la literatura [79].

Por otro lado el índice TCARI si bien se mantiene estable su influencia gran parte del rango de sus valores, a partir de 0.08 se comienza a evidenciar que el rendimiento disminuye en gran medida, lo cual también es consistente con la bibliografía [79].

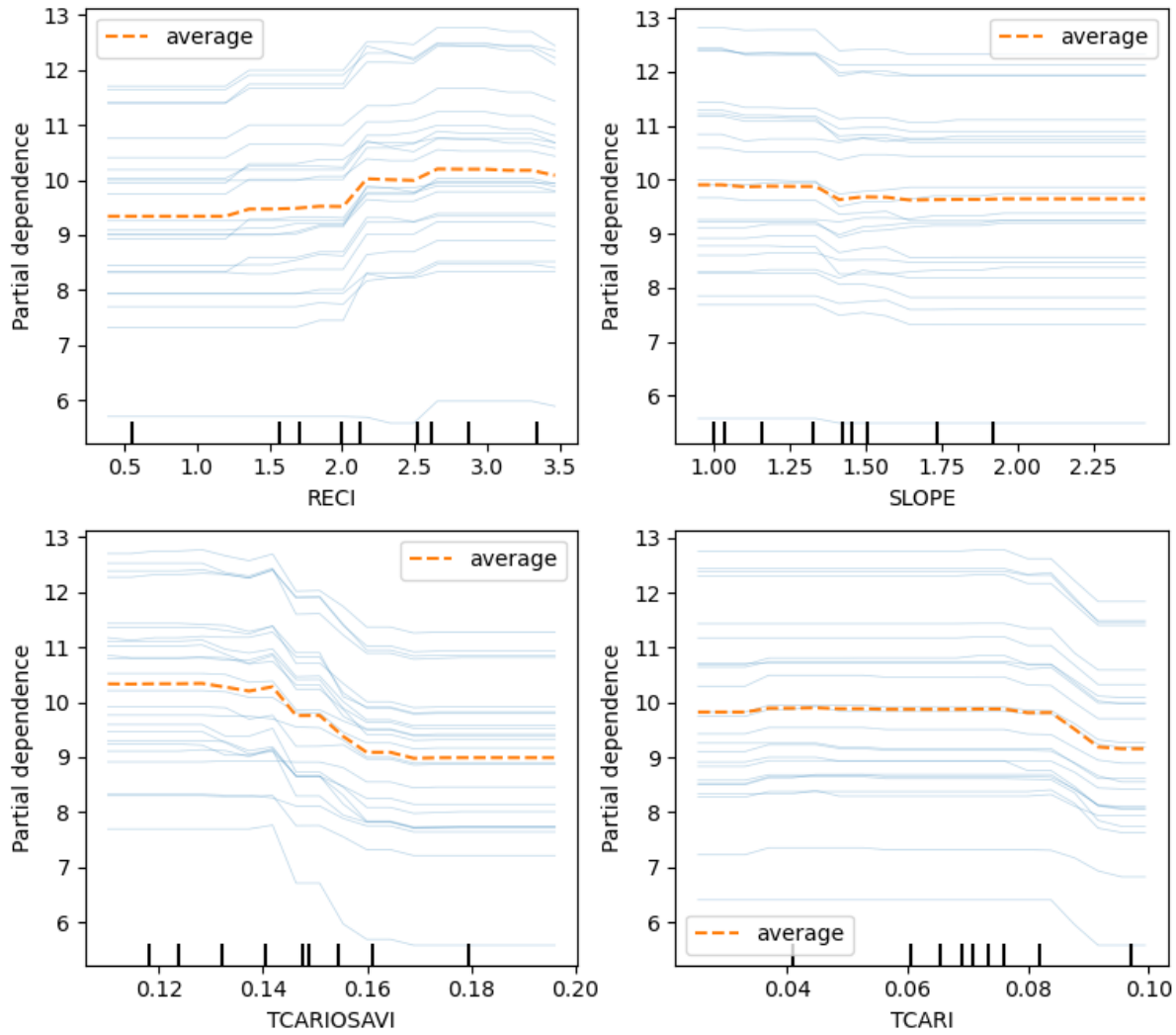

Figura 4.25: Gráficos de dependencias parciales para los índices espectrales RECI, SLOPE, TCARIOSAVI, TCARI.

4.3.5.0.3. Dependencias parciales: DSM, EVAPO, PP, TEMP. En la Figura 4.26 se observa que la altitud (DSM) es una variable influyente sobre el rendimiento, de todos modo el

promedio de los valores de dependencia parcial se mantiene estable a lo largo de todo el rango de valores.

La variables climáticas EVAPO evidenció una leve influencia sobre el rendimiento, observándose que para valores mayores a -0.3, el rendimiento disminuye levemente.

Por otro lado las precipitaciones (PP) no evidencian tener una dependencia parcial de magnitud significativa con rendimiento y finalmente la variable temperatura (TEMP) indica que desde los 23 grados hasta los 25 el rendimiento disminuye.

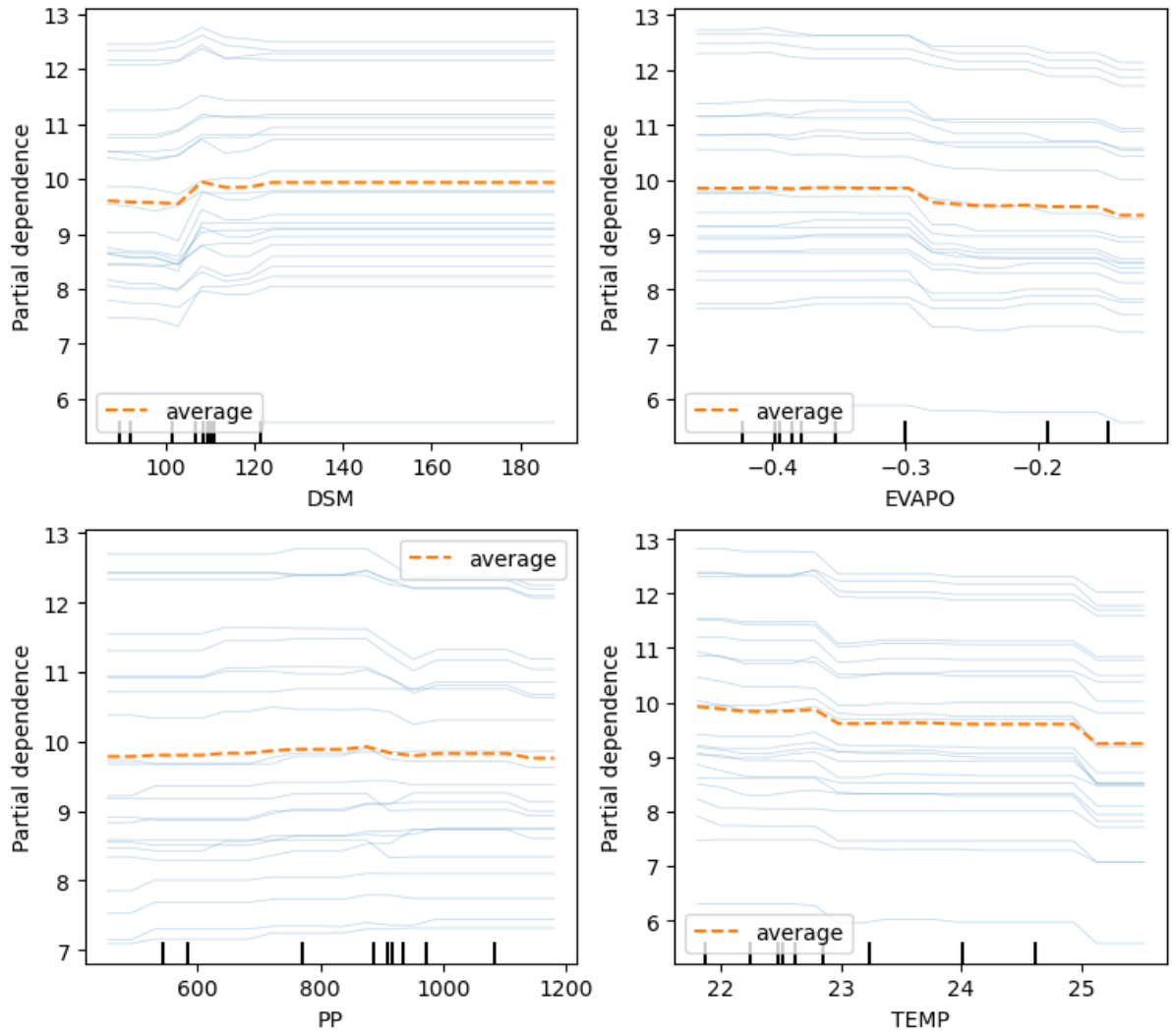

Figura 4.26: Gráficos de dependencias parciales para las variables ambientales espectrales DSM, EVAPO, PP, TEMP.

4.3.5.1. Gráfico de árbol de decisión

Con el propósito de visualizar gráficamente la estructura de uno de los árboles en el modelo que permita la comprensión de cómo se toman las decisiones en el modelo, en la Figura 4.27 se muestra un árbol de decisión con sus nodos y ramas.

Cada nodo representa una decisión, es decir una sumatoria de características del conjunto de datos utilizada para dividir los datos en subconjuntos más pequeños. Las ramas representan las

diferentes opciones o caminos que se pueden seguir en función de las condiciones en los nodos.

Esta visualización resultó útil para entender la lógica detrás de las predicciones del modelo (6 divisiones hasta la máxima profundidad) y para identificar los valores y las características más importantes en la toma de decisiones.

Se observa que a mayor profundidad disminuye el número de muestras con las cuales se toma la decisión, mientras el número de características se mantiene variable a lo largo de toda la estructura.

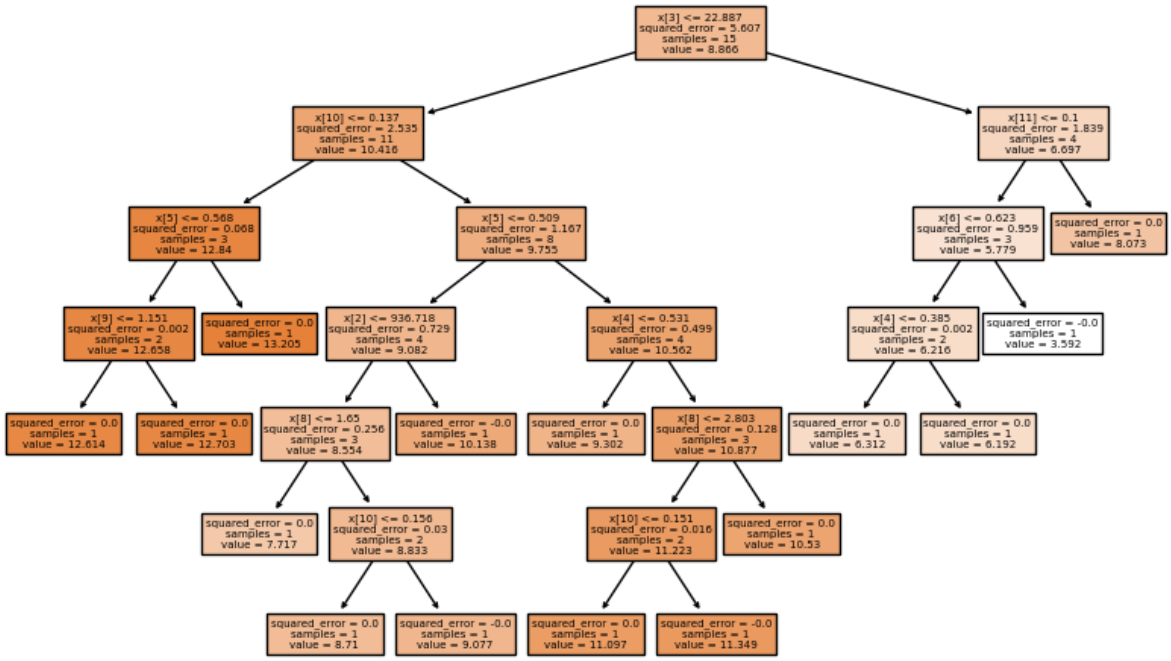

Figura 4.27: Representación gráfica del primer árbol construido en el proceso de entrenamiento del modelo RFR-5 (RMSE = 1.47).

Finalmente se seleccionó el mejor modelo resultante de cada estrategia en la tabla 4.5 y se calculó qué porcentaje de error significan los RMSE obtenidos en relación a la media de rendimiento de la muestras 9.27 (Tn/ha). Se identificó que la metodología de modelado con regresión lineal múltiple superó en un 4,21 % la performance de la metodología de modelado con random forest regresión. Esto nos indica que la relación entre el conjunto de las variables predictoras y la variable respuesta es aproximadamente lineal.

Probablemente la diferencia de rendimiento de los modelos se deba a que en general, se recomienda que en random forest regression el número de muestras sea al menos 10 veces el número de variables predictoras. Esto se debe a que cada árbol de decisión en el bosque aleatorio se entrena con una muestra aleatoria de los datos de entrenamiento. Si el número de muestras es muy pequeño, los árboles de decisión pueden ser demasiado simples y no podrán capturar la complejidad de los datos. [112] [116].

Tabla 4.5: Resultados finales obtenidos con las distintas metodologías de modelado.

	RLM	RFR
Observaciones	33	33
Variables	12	12
RMSE	1.48	1.87
% ERROR	15.97	20.17

CAPÍTULO 5

Conclusiones

Esta tesis explora las herramientas de sensado remoto y de modelado estadístico disponibles y de libre acceso para estimar el rendimiento del cultivo de maíz en la región agrícola del sur de la Provincia de Córdoba y Oeste de la provincia de Buenos Aires. La actividad agrícola se considera de sumo valor en tanto motor de desarrollo de economías regionales y fuente crucial de nutrientes y energía para muchas comunidades en todo el mundo, el conocimiento anticipado de la cantidad de toneladas que se espera cosechar en una determinada campaña permite realizar planificaciones adecuadas tanto a las instituciones públicas y privadas como a los productores en el manejo de su establecimiento agropecuario.

En primera instancia se expuso el resultado de la revisión bibliográfica de las metodologías relativas al modelado de producción de maíz a partir de información ambiental. Se identificaron como grandes ordenadores los modelos estocásticos y los modelos mecanísticos. Estos últimos son de difícil utilización ya que precisan de gran cantidad de información específica sobre el comportamiento del cultivo y el ambiente que buscan modelar. Este trabajo aportó en fortalecer el argumento de la factibilidad de implementar modelos estocásticos sin contar con información del manejo del lote agrícola, ni de la genética de maíz utilizada a campo.

El trabajo de ingeniería de datos, es decir la adquisición, gestión, depuración y acondicionamiento tanto de las observaciones de campo como de los índices espectrales y las variables derivadas de sensores remotos resultó en un tarea determinante que permitió construir una base de datos completa, relevante, precisa y confiable.

La caracterización de las observaciones de campo, el conocimiento de sus particularidades ambientales y su distribución en el territorio permitieron la comprensión y cuantificación de la variabilidad ambiental del rendimiento de maíz en el área estudiada con rendimiento promedio de 9.27 Tn/ha (CV 58.5 %).

Se comprobó que las fuentes de datos utilizadas para el cálculo de índices espectrales así como las fuentes utilizadas para caracterizar la topografía y el clima de los sitios estudiados resultaron de utilidad para el objeto de esta tesis. El futuro en este sentido es prometedor en cuanto la creciente construcción, lanzamiento y disponibilidad de plataformas satelitales con sensores

ópticos y de radar de mayor resolución espacial y temporal, el creciente uso de sensores montados en vehículos aéreos no tripulados (VANT), el crecimiento exponencial de la potencia del procesamiento computacional, la posibilidad de fusión de datos y la política de datos libres, que facilita el acceso a grandes bases de datos digitales (por ejemplo series meteorológicas y datos de campo), implicarán avances rápidos y consistentes en el campo de modelado de variables ambientales.

La técnica de selección secuencial de variables predictoras (stepwise forward) permitió identificar las variables más influyentes y reducir la fuerte colinealidad de la base de datos. La incorporación de las interacciones entre estas variables, resultó determinante en la mejora de los resultados obtenidos.

El modelo de aprendizaje automático Random forest regression tiene pocos hiperparámetros, lo que hizo sencillo su ajuste y optimización. El tamaño relativamente pequeño de la muestra (n<50) limitó su performance (Error=20,17 %). Por otro lado con el modelo de regresión lineal múltiple, con una necesidad computacional menor y sin demandar una optimización de sus parámetros se obtuvo un rendimiento en la estimación superador (Error=15.97 %).

Este error resalta el gran potencial de la técnica desarrollada en la tesis en la aplicación práctica para estimar el rendimiento del cultivo de maíz en la zona núcleo agrícola del país. Se logró estimar el rendimiento del maíz con 33 observaciones de campo y 12 variables derivadas de sensores remotos, con un error similar a un estudio que usó 908 observaciones de campo y 160 variables para estimar el rendimiento en la Provincia de Córdoba (IDECOR, 2022) [115]

En un contexto creciente de disponibilidad de datos de monitores de rendimiento el presente estudio aportó herramientas metodológicas para el análisis de grandes volúmenes de datos y ajustes de modelos usando herramientas estadísticas tradicionales y algoritmos de aprendizaje automático para la estimación de rendimiento. Esta metodología probó ser computacionalmente de bajo costo y obtener resultados positivos a nivel de lotes con potencial uso regional. Asimismo, presenta potencial para la incorporación de nuevas variables predictoras de diferente naturaleza y resoluciones, que podrían mejorar la performance de los modelos.

Esta metodología podría ser utilizada para la construcción de un sistema de monitoreo de rendimiento de maíz regional institucionalizado (Dirección de Estimaciones Agrícolas perteneciente al Ministerio de Agricultura, Ganadería y Pesca de la República Argentina, INTA y/o Mercados granarios). Un sistema que ante escenarios de emergencia provea de información sensible a los decisores públicos para diseñar la mejor respuesta ante el evento adverso. Por otro lado, la metodología ajustada en este estudio se puede utilizar como método de validación de lo reportado por productores antes instituciones fiscales (ARBA, AFIP). Se considera que el conocimiento generado en este estudio puede contribuir al diseño y desarrollo de políticas públicas particularmente orientadas al sector agroindustrial.

Sobre los resultados y experiencia adquirida en las metodologías exploradas en el camino hacia la estimación robusta regional de rendimiento de cultivos anuales, junto con modelos de fusión de datos de distintas fuentes, la clasificación de coberturas agrícolas y detección automática de lotes, serán las bases para la construcción de mapas de rendimiento agrícola y de un servicio de monitoreo de rendimiento regional.

Más allá de los resultados obtenidos en términos cuantitativos, el desarrollo de la tesis ha significado un avance significativo del autor en términos de ingeniería y análisis de datos, construcción e implementación de algoritmos de regresiones lineales y de aprendizaje automático.

Así como del estudio, a partir de información espacial, de variables biofísicas ambientales con potencialidad para el desarrollo de aplicaciones. El desarrollo entero del trabajo se realizó exclusivamente con herramientas de procesamiento de imágenes y datos de acceso libre y gratuito, entre ellas, R, Python, GEE, y QGIS.

En trabajos futuros, en la aproximación a optimizar la metodología, se incorporará mayor frecuencia temporal, cantidad y diversidad de índices espectrales (i.e EVI, NDWI) e información ambiental (i.e variables asociadas al volumen de agua en distintas profundidades del suelo, variables asociadas a la textura del suelo) derivada de sensores remotos. Se conducirá el análisis a una escala espacial de píxel y se explorará la optimización de la estrategia de modelado trabajando con una mayor cantidad de datos para cada lote, con un número mayor de lotes y en una mayor diversidad de ambientes y años.

Referencias bibliográficas

[1] X. Peng, W. Han, J. Ao, and Y. Wang, "Assimilation of lai derived from uav multispectral data into the safy model to estimate maize yield," *Remote Sensing*, vol. 13, no. 6, p. 1094, 2021.

[2] S. U. Handbook and E. Tools, "Sentinel-2 user handbook," *ESA Standard Document Date*, vol. 1, pp. 1–64, 2015.

[3] A. Zaytseva, *Spatio-temporal patterns of extreme weather events and their impacts on corn (Zea mays) and soybeans (Glycine max) in eastern Ontario*. PhD thesis, Carleton University, 2016.

[4] V. Kaminskiy, N. Asanishvili, V. Bulgakov, V. Kaminska, I. Dukulis, and S. Ivanovs, "Impact of global and regional climate changes upon the crop yields.," *Journal of Ecological Engineering*, vol. 24, no. 4, 2023.

[5] M. Yasin, A. Ahmad, T. Khaliq, M. Habib-ur Rahman, S. Niaz, T. Gaiser, I. Ghafoor, H. S. u. Hassan, M. Qasim, and G. Hoogenboom, "Climate change impact uncertainty assessment and adaptations for sustainable maize production using multi-crop and climate models," *Environmental Science and Pollution Research*, pp. 1–22, 2022.

[6] A. F. d. I. Publicos, "Biblioteca electrónica."

[7] G. D. Andrea, "La Contribución del Agro Mitos y Realidades," *Ciclo Alumni Agronegocios*, 2019.

[8] INDEC, "Informe de Avance del Nivel de Actividad - Abril 2019," 2009.

[9] B. d. C. d. C. Departamento de Economia, "Informe de Mercados Agricolas 31," 2019.

[10] D. M. Johnson, "An assessment of pre- and within-season remotely sensed variables for forecasting corn and soybean yields in the United States," *Remote Sensing of Environment*, vol. 141, pp. 116–128, 2014.

[11] D. J. Mulla, "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps," *Biosystems Engineering*, vol. 114, no. 4, pp. 358–371, 2013.

[12] R. Casa, M. Rossi, G. Sappa, and A. Trotta, "Assessing crop water demand by remote sensing and GIS for the Pontina Plain, Central Italy," *Water Resources Management*, vol. 23, no. 9, pp. 1685–1712, 2009.

[13] R. Casa, H. Varella, S. Buis, M. Guérif, B. De Solan, and F. Baret, "Forcing a wheat crop model with LAI data to access agronomic variables: Evaluation of the impact of model and LAI uncertainties and comparison with an empirical approach," *European Journal of Agronomy*, vol. 37, no. 1, pp. 1–10, 2012.

[14] C. Atzberger, "Advances in remote sensing of agriculture: Context description, existing operational monitoring systems and major information needs," *Remote Sensing*, vol. 5, no. 2, pp. 949–981, 2013.

[15] E. H. Satorre, R. Benech, G. A. Slafer, E. B. De la Fuente, D. J. Miralles, M. E. Otegui, and R. Savin, "Producción de granos: bases funcionales para su manejo," *Universidad de Buenos Aires, Facultad de Agronomía, Buenos Aires, Argentina*, 2003.

[16] F. H. Andrade, S. A. Uhart, and A. Cirilo, "Temperature affects radiation use efficiency in maize," *Field Crops Research*, vol. 32, no. 1-2, pp. 17–25, 1993.

[17] M. E. Otegui, M. G. Nicolini, R. A. Ruiz, and P. A. Dodds, "Sowing date effects on grain yield components for different maize genotypes," *Agronomy Journal*, vol. 87, no. 1, pp. 29–33, 1995.

[18] G. Maddonni and M. Otegui, "Leaf area, light interception, and crop development in maize," *Field Crops Research*, vol. 48, no. 1, pp. 81–87, 1996.

[19] J. L. Hatfield, J. H. Prueger, T. J. Sauer, C. Dold, P. O'Brien, and K. Wacha, "Applications of vegetative indices from remote sensing to agriculture: Past and future," *Inventions*, vol. 4, no. 4, p. 71, 2019.

[20] L. Valderrama-Landeros, F. Flores-Verdugo, R. Rodríguez-Sobreyra, J. M. Kovacs, and F. Flores-de Santiago, "Extrapolating canopy phenology information using sentinel-2 data and the google earth engine platform to identify the optimal dates for remotely sensed image acquisition of semiarid mangroves," *Journal of Environmental Management*, vol. 279, p. 111617, 2021.

[21] L. Liu, X. Xiao, Y. Qin, J. Wang, X. Xu, Y. Hu, and Z. Qiao, "Mapping cropping intensity in china using time series landsat and sentinel-2 images and google earth engine," *Remote Sensing of Environment*, vol. 239, p. 111624, 2020.

[22] R. Ni, J. Tian, X. Li, D. Yin, J. Li, H. Gong, J. Zhang, L. Zhu, and D. Wu, "An enhanced pixel-based phenological feature for accurate paddy rice mapping with sentinel-2 imagery in google earth engine," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 178, pp. 282–296, 2021.

[23] X. Liu, H. Zhai, Y. Shen, B. Lou, C. Jiang, T. Li, S. B. Hussain, and G. Shen, "Large-scale crop mapping from multisource remote sensing images in google earth engine," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 414–427, 2020.

[24] Z. Jin, G. Azzari, C. You, S. Di Tommaso, S. Aston, M. Burke, and D. B. Lobell, "Smallholder maize area and yield mapping at national scales with google earth engine," *Remote Sensing of Environment*, vol. 228, pp. 115–128, 2019.

[25] F. Gao and M. Anderson, "Evaluating yield variability of corn and soybean using landsat-8, sentinel-2 and modis in google earth engine," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 7286–7289, IEEE, 2019.

[26] J. Xue and B. Su, "Significant remote sensing vegetation indices: A review of developments and applications," *Journal of sensors*, vol. 2017, 2017.

[27] M. Horlent, G. Ibañez, N. Horlent, J. Otero, L. P. Thomas, J. J. Fernandez, and S. E. Torrusio, "Proyecto biblioteca de firmas espectrales: Base de datos para el almacenamiento, gestión y distribución de datos espectrales obtenidos a campo," in *VIII Congreso Argentino de AgroInformática (CAI-2016)-JAIIO 45 (Tres de Febrero, 2016).*, 2016.

[28] "9 - precision crop production," in *Lockhart and Wiseman' s Crop Husbandry Including Grassland (Tenth Edition)* (A. Samuel and L. Dines, eds.), Woodhead Publishing Series in Food Science, Technology and Nutrition, pp. 273–284, Woodhead Publishing, tenth edition ed., 2023.

[29] C. Mariano and B. Mónica, "A random forest-based algorithm for data-intensive spatial interpolation in crop yield mapping," *Computers and Electronics in Agriculture*, vol. 184, p. 106094, 2021.

[30] S. Arslan and T. S. Colvin, "Grain yield mapping: Yield sensing, yield reconstruction, and errors," *Precision Agriculture*, vol. 3, no. 2, pp. 135–154, 2002.

[31] N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, "Google earth engine: Planetary-scale geospatial analysis for everyone," *Remote sensing of Environment*, vol. 202, pp. 18–27, 2017.

[32] B. Duchemin, P. Maisongrande, G. Boulet, and I. Benhadj, "A simple algorithm for yield estimates: Evaluation for semi-arid irrigated winter wheat monitored with green leaf area index," *Environmental Modelling and Software*, vol. 23, no. 7, pp. 876–892, 2008.

[33] C. A. van Diepen, J. Wolf, H. van Keulen, and C. Rappoldt, "WOFOST: a simulation model of crop production," *Soil Use and Management*, vol. 5, no. 1, pp. 16–24, 1989.

[34] MacKINNON, "Book Review A SIMULATION MODEL OF MAIZE GROWTH AND DEVELOPMENT OF JONES AND J.R. KINIRY," *Computers and Electronics in Agriculture*, vol. 84, no. 24, pp. 9150–9154, 1987.

[35] N. Brisson, C. Gary, E. Justes, R. Roche, B. Mary, D. Ripoche, D. Zimmer, J. Sierra, P. Bertuzzi, P. Burger, *et al.*, "An overview of the crop model stics," *European Journal of agronomy*, vol. 18, no. 3-4, pp. 309–332, 2003.

[36] J. W. Jones, G. Hoogenboom, C. H. Porter, K. J. Boote, W. D. Batchelor, L. A. Hunt, P. W. Wilkens, U. Singh, A. J. Gijsman, and J. T. Ritchie, *The DSSAT cropping system model*, vol. 18. 2003.

[37] C. Liao and J. Wang, "Evaluation of spatio-temporal data fusion methods for generating NDVI time series in cropland areas," *International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2016-Novem, pp. 2570–2573, 2016.

[38] F. M. Aguate, S. Trachsel, L. González Pérez, J. Burgueño, J. Crossa, M. Balzarini, D. Gouache, M. Bogard, and G. de los Campos, "Use of hyperspectral image data outperforms vegetation indices in prediction of maize yield," *Crop Science*, vol. 57, no. 5, pp. 2517–2524, 2017.

[39] D. M. Johnson, "A comprehensive assessment of the correlations between field crop yields and commonly used modis products," *International Journal of Applied Earth Observation and Geoinformation*, vol. 52, pp. 65–81, 2016.

[40] Z. Pan, J. Huang, C. Wei, and H. Zhang, "Remote sensing of agricultural disasters monitoring: Recent advances," *economy and society*, vol. 10, no. 13, p. 16.

[41] A. d. De Wit and C. Van Diepen, "Crop model data assimilation with the ensemble kalman filter for improving regional crop yield forecasts," *Agricultural and Forest Meteorology*, vol. 146, no. 1-2, pp. 38–56, 2007.

[42] Y. Li, Q. Zhou, J. Zhou, G. Zhang, C. Chen, and J. Wang, "Assimilating remote sensing information into a coupled hydrology-crop growth model to estimate regional maize yield in arid regions," *Ecological modelling*, vol. 291, pp. 15–27, 2014.

[43] A. V. Ines, N. N. Das, J. W. Hansen, and E. G. Njoku, "Assimilation of remotely sensed soil moisture and vegetation with a crop simulation model for maize yield prediction," *Remote Sensing of Environment*, vol. 138, pp. 149–164, 2013.

[44] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[45] E. Davis, C. Wang, and K. Dow, "Comparing sentinel-2 msi and landsat 8 oli in soil salinity detection: A case study of agricultural lands in coastal north carolina," *International Journal of Remote Sensing*, vol. 40, no. 16, pp. 6134–6153, 2019.

[46] J. J. Faraway, *Linear models with R*. Chapman and Hall/CRC, 2004.

[47] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. .ºReilly Media, Inc.", 2016.

[48] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, p. 2674, 2018.

[49] T. Van Klompenburg, A. Kassahun, and C. Catal, "Crop yield prediction using machine learning: A systematic literature review," *Computers and Electronics in Agriculture*, vol. 177, p. 105709, 2020.

[50] X. Zhou, X. Zhu, Z. Dong, W. Guo, *et al.*, "Estimation of biomass in wheat using random forest regression algorithm and remote sensing data," *The Crop Journal*, vol. 4, no. 3, pp. 212–219, 2016.

[51] R. Srinet, S. Nandy, and N. Patel, "Estimating leaf area index and light extinction coefficient using random forest regression algorithm in a tropical moist deciduous forest, india," *Ecological Informatics*, vol. 52, pp. 94–102, 2019.

[52] L. Leroux, M. Castets, C. Baron, M.-J. Escorihuela, A. Bégué, and D. L. Seen, "Maize yield estimation in west africa from crop process-induced combinations of multi-domain remote sensing indices," *European Journal of Agronomy*, vol. 108, pp. 11–26, 2019.

[53] T. Sakamoto, "Incorporating environmental variables into a modis-based crop yield estimation method for united states corn and soybeans through the use of a random forest regression algorithm," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 160, pp. 208–228, 2020.

[54] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[55] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS journal of photogrammetry and remote sensing*, vol. 114, pp. 24–31, 2016.

[56] J. A. Di Rienzo, F. Casanoves, L. A. González, E. M. Tablada, and M. d. P. Díaz, *Estadística para las ciencias agropecuarias*. No. 630.21 E79e, Córdoba, AR: Edit. Brujas, 2008.

[57] T. Hastie, R. Tibshirani, and J. Friedman, "Random forests," in *The elements of statistical learning*, pp. 587–604, Springer, 2009.

[58] O. E. Scarpati and A. D. Capriolo, "Sequías e inundaciones en la provincia de buenos aires (argentina) y su distribución espacio-temporal," *Investigaciones geográficas*, no. 82, pp. 38–51, 2013.

[59] A. J. Hall, C. M. Rebella, C. M. Ghersa, and J. P. Culot, "Field-crop systems of the pampas," *Ecosystems of the World (Netherlands)*, 1992.

[60] S. I. Ballesteros, "Inundaciones y su relación con el clima y la hidrología subterránea en el noroeste de buenos aires (1980–2010): Aplicación de percepción remota," *Universidad de Buenos Aires*, 2014.

[61] I. I. de Suelos CIRN, "Cartas de Suelos República Argentina - Provincia de Buenos Aires," May 2022.

[62] W. Sun, B. Whelan, A. B. McBratney, and B. Minasny, "An integrated framework for software to provide yield data cleaning and estimation of an opportunity index for site-specific crop management," *Precision Agriculture*, vol. 14, no. 4, pp. 376–391, 2013.

[63] A. Vega, M. Córdoba, M. Castro-Franco, and M. Balzarini, "Protocol for automating error removal from yield maps," *Precision Agriculture*, vol. 20, no. 5, pp. 1030–1044, 2019.

[64] G. Lyle, B. A. Bryan, and B. Ostendorf, "Post-processing methods to eliminate erroneous grain yield measurements: review and directions for future development," *Precision agriculture*, vol. 15, no. 4, pp. 377–402, 2014.

[65] B. Matérn, "Lecture notes in statistics. spatial variation, vol. 36," 1986.

[66] R. Webster and M. A. Oliver, *Geostatistics for environmental scientists*. John Wiley & Sons, 2007.

[67] S. Drummond and K. Sudduth, "Analysis of errors affecting yield map accuracy," in *Proceedings of the 7th international conference on precision agriculture*, pp. 1478–1490, 2005.

[68] L. Anselin, "The moran scatterplot as an esda tool to assess local instability in spatial," *Spatial Analytical*, vol. 4, no. 111, pp. 9780203739051–8, 1996.

[69] S. P. Wright, "Adjusted p-values for simultaneous inference," *Biometrics*, pp. 1005–1013, 1992.

[70] N. R. Draper and H. Smith, *Applied regression analysis*, vol. 326. John Wiley & Sons, 1998.

[71] N. R. Peralta, J. L. Costa, M. Balzarini, and H. Angelini, "Delineation of management zones with measurements of soil apparent electrical conductivity in the southeastern pampas," *Canadian Journal of Soil Science*, vol. 93, no. 2, pp. 205–218, 2013.

[72] S. Blackmore, "Remedial correction of yield map data," *Precision agriculture*, vol. 1, no. 1, pp. 53–66, 1999.

[73] K. A. Sudduth and S. T. Drummond, "Yield editor: Software for removing errors from crop yield maps," *Agronomy Journal*, vol. 99, no. 6, pp. 1471–1482, 2007.

[74] C. Leroux, H. Jones, A. Clenet, B. Dreux, M. Becu, and B. Tisseyre, "A general method to filter out defective spatial observations from yield mapping datasets," *Precision Agriculture*, vol. 19, no. 5, pp. 789–808, 2018.

[75] M. Hubert and S. Van der Veeken, "Outlier detection for skewed data," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 22, no. 3-4, pp. 235–246, 2008.

[76] J. N. Hird, J. Kariyeva, and G. J. McDermid, "Satellite time series and google earth engine democratize the process of forest-recovery monitoring over large areas," *Remote Sensing*, vol. 13, no. 23, p. 4745, 2021.

[77] QGIS Development Team, *QGIS Geographic Information System*. QGIS Association, 2022.

[78] C. S. Daughtry, C. Walthall, M. Kim, E. B. De Colstoun, and J. McMurtrey Iii, "Estimating corn leaf chlorophyll concentration from leaf and canopy reflectance," *Remote sensing of Environment*, vol. 74, no. 2, pp. 229–239, 2000.

[79] D. Haboudane, J. R. Miller, N. Tremblay, P. J. Zarco-Tejada, and L. Dextraze, "Integrated narrow-band vegetation indices for prediction of crop chlorophyll content for application to precision agriculture," *Remote sensing of environment*, vol. 81, no. 2-3, pp. 416–426, 2002.

[80] M. S. Kim, *The Use of Narrow Spectral Bands for Improving Remote Sensing Estimations of Fractionally Absorbed Photosynthetically Active Radiation*. PhD thesis, 1994.

[81] G. Rondeaux, M. Steven, and F. Baret, "Optimization of soil-adjusted vegetation indices," *Remote sensing of environment*, vol. 55, no. 2, pp. 95–107, 1996.

[82] A. R. Huete, "A soil-adjusted vegetation index (savi)," *Remote sensing of environment*, vol. 25, no. 3, pp. 295–309, 1988.

[83] F. Baret, G. Guyot, and D. Major, "Crop biomass evaluation using radiometric measurements," *Photogrammetria*, vol. 43, no. 5, pp. 241–256, 1989.

[84] F. N. Kogan, "Remote sensing of weather impacts on vegetation in non-homogeneous areas," *International Journal of remote sensing*, vol. 11, no. 8, pp. 1405–1419, 1990.

[85] F.-M. Wang, J.-F. Huang, Y.-L. Tang, and X.-Z. Wang, "New vegetation index and its application in estimating leaf area index of rice," *Rice Science*, vol. 14, no. 3, pp. 195–203, 2007.

[86] M. Tackenberg, C. Volkmar, and K.-H. Dammer, "Sensor-based variable-rate fungicide application in winter wheat," *Pest Management Science*, vol. 72, no. 10, pp. 1888–1896, 2016.

[87] A. L. Vian, C. Bredemeier, M. A. Turra, C. P. d. S. Giordano, E. Fochesatto, J. A. d. Silva, and M. A. Drum, "Nitrogen management in wheat based on the normalized difference vegetation index (ndvi)," *Ciência Rural*, vol. 48, 2018.

[88] J. Bragagnolo, T. J. C. Amado, and R. P. Bortolotto, "Use efficiency of variable rate of nitrogen prescribed by optical sensor in corn1," *Revista Ceres*, vol. 63, pp. 103–111, 2016.

[89] A. A. Gitelson, A. Viña, T. J. Arkebauer, D. C. Rundquist, G. Keydan, and B. Leavitt, "Remote estimation of leaf area index and green leaf biomass in maize canopies," *Geophysical research letters*, vol. 30, no. 5, 2003.

[90] Y.-P. Wang, Y.-C. Chang, and Y. Shen, "Estimation of nitrogen status of paddy rice at vegetative phase using unmanned aerial vehicle based multispectral imagery," *Precision Agriculture*, vol. 23, no. 1, pp. 1–17, 2022.

[91] G. J. Huffman, D. T. Bolvin, D. Braithwaite, K. Hsu, R. Joyce, P. Xie, and S.-H. Yoo, "Nasa global precipitation measurement (gpm) integrated multi-satellite retrievals for gpm (imerg)," *Algorithm Theoretical Basis Document (ATBD) Version*, vol. 4, p. 26, 2015.

[92] J. Muñoz-Sabater, E. Dutra, A. Agustí-Panareda, C. Albergel, G. Arduini, G. Balsamo, S. Boussetta, M. Choulga, S. Harrigan, H. Hersbach, *et al.*, "Era5-land: A state-of-the-art global reanalysis dataset for land applications," *Earth System Science Data*, vol. 13, no. 9, pp. 4349–4383, 2021.

[93] J.-N. Thepaut and D. Dee, "The copernicus climate change service (c3s): Open access to a climate data store," in *EGU General Assembly Conference Abstracts*, pp. EPSC2016–9826, 2016.

[94] J. Muñoz Sabater *et al.*, "Era5-land hourly data from 1981 to present," *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*, vol. 10, 2019.

[95] A. S. Monin and A. M. Obukhov, "Basic laws of turbulent mixing in the surface layer of the atmosphere," *Contrib. Geophys. Inst. Acad. Sci. USSR*, vol. 151, no. 163, p. e187, 1954.

[96] *IFS Documentation CY47R3 - Part I: Observations*. No. 1 in IFS Documentation, ECMWF, 10 2021.

[97] T. Tadono, H. Nagai, H. Ishida, F. Oda, S. Naito, K. Minakawa, and H. Iwamoto, "Generation of the 30 m-mesh global digital surface model by alos prism.," *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 41, 2016.

[98] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (F. Loizides and B. Schmidt, eds.), pp. 87 – 90, IOS Press, 2016.

[99] "Anaconda software distribution," 2020.

[100] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *Proceedings of the 9th Python in Science Conference*, vol. 57, p. 61, Austin, TX, 2010.

[101] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, p. 357–362, 2020.

[102] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Austin, TX, 2010.

[103] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[104] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee, and A. Qalieh, "mwaskom/seaborn: v0.8.1 (september 2017)," Sept. 2017.

[105] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[106] B. Efron, "Estimating the error rate of a prediction rule: improvement on cross-validation," *Journal of the American statistical association*, vol. 78, no. 382, pp. 316–331, 1983.

[107] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[108] C. Molnar, *Interpretable Machine Learning*. 2 ed., 2022.

[109] Q. Zhao and T. Hastie, "Causal interpretations of black-box models," *Journal of Business & Economic Statistics*, vol. 39, no. 1, pp. 272–281, 2021.

[110] A. G. Cabello and A. Ciancio, "El impacto de la sequía en la economía argentina. el caso del cultivo de soja," 2018.

[111] T. Wei and V. Simko, "R package "corrplot": Visualization of a correlation matrix (version 0.84)," 2017.

[112] M. Kuhn, K. Johnson, *et al.*, *Applied predictive modeling*, vol. 26. Springer, 2013.

[113] C. F. Dormann, J. Elith, S. Bacher, C. Buchmann, G. Carl, G. Carré, J. R. G. Marquéz, B. Gruber, B. Lafourcade, P. J. Leitão, *et al.*, "Collinearity: a review of methods to deal with it and a simulation study evaluating their performance," *Ecography*, vol. 36, no. 1, pp. 27–46, 2013.

[114] R. Beulah, "A survey on different data mining techniques for crop yield prediction," *Int. J. Comput. Sci. Eng*, vol. 7, no. 1, pp. 738–744, 2019.

[115] G. d. l. P. d. C. Ministerio de Agricultura y Ganadería e IDECOR, "Mapa de Área sembrada, rindes y producción de soja y maíz campaña 2021/2022, provincia de córdoba," septiembre 2022.

[116] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[117] F. Andrade, "Ecofisiología del cultivo de maíz," 1996.

[118] D. Wilson, R. Muchow, and C. Murgatroyd, "Model analysis of temperature and solar radiation limitations to maize potential productivity in a cool climate," *Field crops research*, vol. 43, no. 1, pp. 1–18, 1995.

[119] M. E. Otegui and R. Bonhomme, "Grain yield components in maize: I. ear growth and kernel set," *Field Crops Research*, vol. 56, no. 3, pp. 247–256, 1998.

[120] R. Jones, J. Roessler, and S. Ouattar, "Thermal environment during endosperm cell division in maize: Effects on number of endosperm cells and starch granules 1," *Crop Science*, vol. 25, no. 5, pp. 830–834, 1985.

[121] L. M. Thompson, "Climatic change, weather variability, and corn production 1," *Agronomy Journal*, vol. 78, no. 4, pp. 649–653, 1986.

[122] J. Schoper, R. Johnson, and R. Lambert, "Maize yield response to increased assimilate supply 1," *Crop Science*, vol. 22, no. 6, pp. 1184–1189, 1982.

[123] M. Tollenaar and T. Daynard, "Effect of defoliation on kernel development in maize," *Canadian Journal of Plant Science*, vol. 58, no. 1, pp. 207–212, 1978.

[124] R. Grant, B. Jackson, J. Kiniry, and G. Arkin, "Water deficit timing effects on yield components in maize," *Agronomy Journal*, vol. 81, no. 1, pp. 61–65, 1989.

[125] A. Hall, J. Lemcoff, and N. Trapani, "Water stress before and during flowering in maize and its effects on yield, its components, and their determinants," *Maydica (Italy)*, 1971.

[126] M. E. Westgate and J. S. Boyer, "Reproduction at low and pollen water potentials in maize 1," *Crop science*, vol. 26, no. 5, pp. 951–956, 1986.

[127] J. Kiniry and J. Ritchie, "Shade-sensitive interval of kernel number of maize 1," *Agronomy Journal*, vol. 77, no. 5, pp. 711–715, 1985.

[128] F. H. Andrade, C. Vega, S. Uhart, A. Cirilo, M. Cantarero, and O. Valentinuz, "Kernel number determination in maize," *Crop Science*, vol. 39, no. 2, p. crops-ci1999.0011183X0039000200026x.

[129] F. H. Andrade, L. Echarte, R. Rizzalli, A. Della Maggiora, and M. Casanovas, "Kernel number prediction in maize under nitrogen or water stress," *Crop Science*, vol. 42, no. 4, pp. 1173–1179, 2002.

[130] M. E. Otegui and F. H. Andrade, "New relationships between light interception, ear growth, and kernel set in maize," *Physiology and modeling kernel set in maize*, vol. 29, pp. 89–102, 2000.

[131] F. Andrade, C. Vega, S. A. Uhart, A. Cirilo, M. Cantarero, and O. Valentinuz, "Kernel number determination in maize," *Crop Science - CROP SCI*, vol. 39, 03 1999.

[132] J. L. Monteith, "Climate and the efficiency of crop production in britain," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 281, no. 980, pp. 277–294, 1977.

[133] V. Sadras and D. Connor, "Physiological basis of the response of harvest index to the fraction of water transpired after anthesis: a simple model to estimate harvest index for determinate species," *Field Crops Research*, vol. 26, no. 3-4, pp. 227–239, 1991.

[134] J. Mercau, V. Sadras, E. Satorre, C. Messina, C. Balbi, M. Uribelarrea, and A. Hall, "On-farm assessment of regional and seasonal variation in sunflower yield in argentina," *Agricultural Systems*, vol. 67, no. 2, pp. 83–103, 2001.

[135] G. Zheng and L. M. Moskal, "Retrieving leaf area index (lai) using remote sensing: theories, methods and sensors," *Sensors*, vol. 9, no. 4, pp. 2719–2745, 2009.

Anexos

1. Anexo I

1.1. Rendimiento de maíz: Factores determinantes.

1.1.1. Fisiología de la generación del rendimiento

Al igual que en la mayoría de los cultivos, en Maíz (Zea Mayx) (Figura 5.1) existe una estrecha relación entre el rendimiento y la producción de biomasa aérea [15], la cual depende de la cantidad de radiación fotosintéticamente activa interceptada por el canopeo. Por su sistema fotosintético (C4) el maíz es más eficiente que los cereales de invierno (sistema fotosintético C3) para convertir radiación en biomasa.

Si bien en condiciones de laboratorio la eficiencia en el uso de la radiación (EUR), puede variar con la temperatura [16] y la etapa del ciclo considerada [17]. En condiciones de cultivo, se ha observado que la disminución de la EUR por temperaturas frescas tienen trascendencia menor en zonas donde las bajas temperaturas ocurren por períodos breves al comienzo del ciclo del cultivo (e.g. siembras tempranas en el norte de Buenos Aires), al igual que en zonas templadas de mayor latitud (e.g. SE de Buenos Aires) donde se manifiestan durante un tiempo más prolongado, tanto al comienzo como al final de la estación de crecimiento.

En condiciones de laboratorio se determino que el valor de la EUR en post-floración es sensible a la relación fuente destino establecida durante el llenado de los granos [15], disminuyendo cuando la relación aumenta y viceversa.

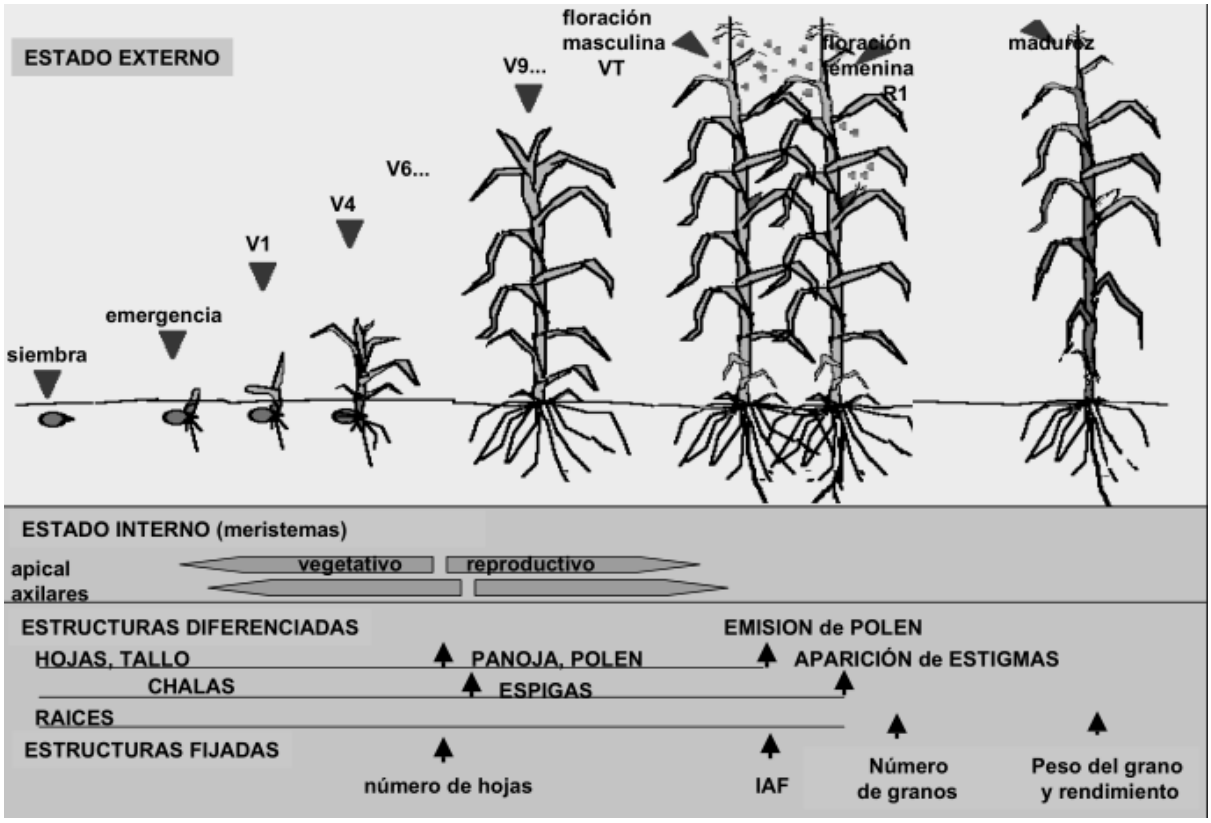

Figura 5.1: Esquema del Desarrollo y Crecimiento del cultivo de maíz. Fuente: Adaptado de Satorre (2003) [15].

En condición de cultivo se considera que el valor de EUR es relativamente constante para la mayoría de las condiciones ambientales [117], lo que indica que el incremento en la producción de biomasa por parte del cultivo dependerá fundamentalmente de la duración de su ciclo y de la eficiencia con que capture el recurso luz (Figura 5.2). La constancia observada en el índice de cosecha del maíz (IC= proporción de la biomasa total aérea que se encuentra en los granos a madurez fisiológica) cuando no existen déficits hídricos ni temperaturas de crecimiento muy bajas [118], determinaría un mayor rendimiento a mayores valores de biomasa.

La variable principal para modificar la cantidad de radiación interceptada por el cultivo es la eficiencia de intercepción (e_i), que está fuertemente asociada a la generación y mantenimiento del área foliar. En los estadios tempranos del ciclo, cuando la cobertura del suelo es baja, se produce una gran ineficiencia en la captación de radiación. Esta restricción impuesta por el desarrollo del área follar es modulada por el ambiente: con temperaturas más frescas (e.g. siembras-tempranas) el área producida por una hoja dada puede ser menor que con temperaturas más cálidas (50°C), determinando finalmente un menor IAF, que puede traer aparejado una menor intercepción de radiación. La respuesta diferencial del crecimiento a la temperatura según genotipos es un parámetro adicional a considerar. Estas modificaciones, impuestas por la variación en la temperatura de crecimiento debida a la fecha de siembra suelen ser compensadas a través del ajuste correcto en la densidad de plantas y la uniformidad de siembra.

En las condiciones de producción de la Región Pampeana, pese al menor crecimiento foliar observado en siembras tempranas, la mayor duración del ciclo determina normalmente una mayor cantidad de radiación acumulada hasta la madurez. La consecuencia directa de una mayor cantidad de radiación interceptada por el cultivo es una mayor biomasa a madurez, la cual, debido a la constancia del Índice de cosecha (IC), se traduce en un mayor rendimiento.

1.2. Componentes numéricos

Al igual que en otros cultivos de grano, el rendimiento en maíz puede ser estudiado a través de sus componentes numéricos; el número de granos (NG) por unidad de superficie y su peso individual. A su vez, el NG es producto del número de plantas por unidad de superficie, del número de espigas granadas por planta (prolificidad) y del número de granos por espiga. El peso del grano, por su parte, es función de la duración del período de llenado y de la tasa de llenado (g día ¹).

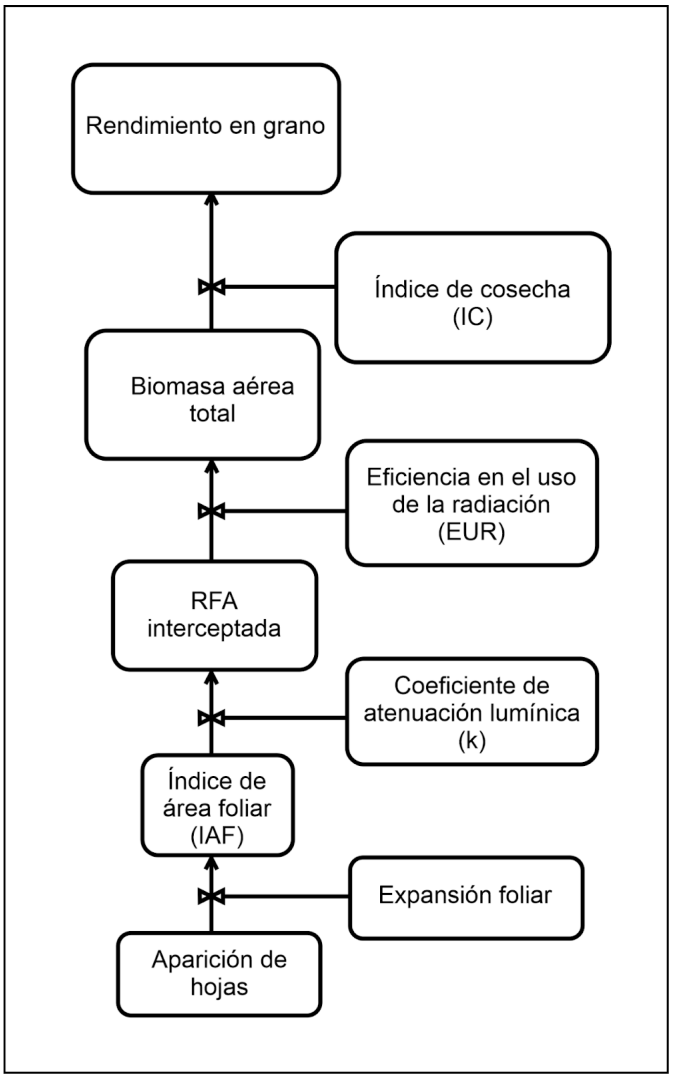

Figura 5.2: Esquema de determinación del rendimiento de maíz.Adaptado de Satorre (2003) [15].

1.2.1. Número de granos

En maíz, el rendimiento está más asociado al número final de granos logrados (Figura 5.3) que al peso de los mismos. La generación de estructuras capaces de dar origen y sostén a un grano son un factor determinante del número final de granos que alcanzan la madurez. Satorre [15] ha señalado que para lograr aumentos de rendimiento es más importante aumentar la supervivencia de dichas estructuras que el número potencial de granos. Por lo cual se torna muy importante

el análisis de los factores determinantes del aborto de estructuras potencialmente aptas para generar un grano cosechable.

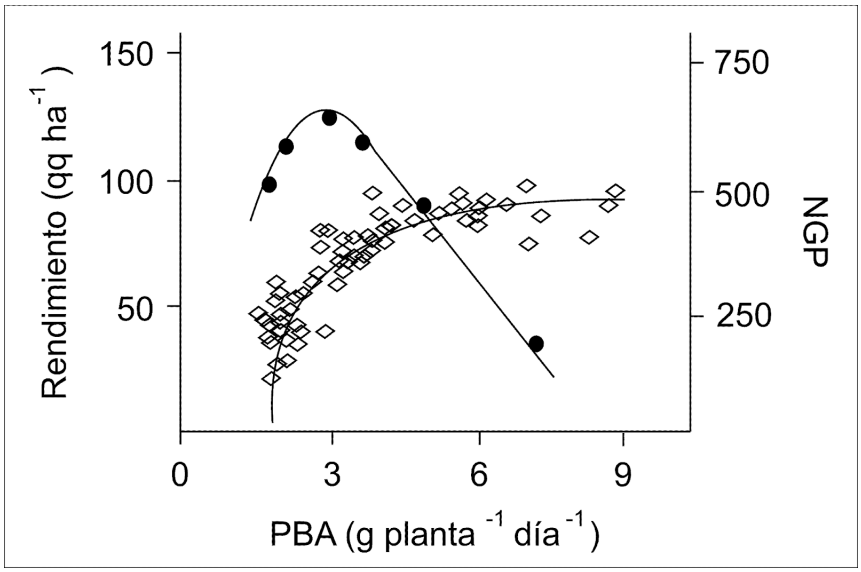

Figura 5.3: Adaptado de [117]. Respuesta del número de granos por planta (NGP) y del rendimiento en grano a la tasa de producción de biomasa aérea (PBA) durante el periodo crítico alrededor de la floración.

1.2.2. Peso del grano

El peso del grano se genera durante una sucesión de etapas a partir de la fecundación del ovario. La primer fase de llenado del grano, comúnmente denominada fase "*lag*" corresponde a un período formativo, de lenta acumulación de biomasa en el grano, durante el cual se establece el número de células endospermáticas y comienzan a formarse los posibles lugares para la deposición del almidón, los amiloplastos. Esta fase, durante la cual se comienza a determinar el peso potencial del grano de maíz, se superpone con el período post-floración de elongación de la espiga y determinación del número de granos [119].

En maíz el peso del grano se encuentra altamente relacionado tanto con el número de células endospermáticas como con la cantidad de gránulos de almidón formados. Estos dos componentes definen conjuntamente la capacidad como destino del grano, debido a que conforman los lugares potenciales de deposición de reservas. Temperaturas muy elevadas durante la fase "*lag*" pueden crear reducciones en el peso potencial del grano, medido como un menor número de células endospermáticas por grano, y determinan un peso final menor [120].

Luego de la fase "*lag*" continúa una segunda fase denominada de llenado efectivo, donde ocurre más del 80 % del incremento de peso y se depositan los principales componentes del grano. Diferencias en el peso final del grano pueden ser explicadas mediante (i) variaciones en la duración del llenado efectivo, o (ii) modificaciones en la tasa de acumulación de materia seca durante esta etapa. Los genotipos de alto peso de grano están relacionados con altas tasas de llenado durante la fase de llenado efectivo y con altos valores de gránulos de almidón formados.

Esta relación entre alto peso de grano, altas tasas de llenado, y altos valores de gránulos de almidón por grano se deben a que la tasa de deposición de almidón es constante por unidad de superficie de gránulo de almidón a lo largo de todo el endosperma, excepto en los gránulos muy chicos. Variaciones en las condiciones ambientales durante la fase de llenado efectivo modulan

el peso final de los granos. Aumentos de temperatura por encima de 23 °C durante esta fase promueven un incremento en la tasa de llenado que no compensa la disminución del tiempo de llenado [121], creando reducciones en el peso final del grano.

Manipulaciones de la cantidad de fuente disponible por grano durante la etapa de llenado efectivo modifican el peso final de los mismos, y si bien el período de formación del grano es muy importante para la determinación del peso potencial, las condiciones durante el llenado efectivo pueden modificar el peso dependiendo de la cantidad de fuente disponible por grano.

Aumentos en la cantidad de fuente por grano mediante reducciones en la densidad del cultivo tres semanas post-floración (i.e. más allá de la fase "*lag*"; [122]) o reducciones en la cantidad de fuente por grano mediante defoliaciones durante el período de llenado efectivo [123] promueven variaciones significativas en el peso de los granos sin modificar el número de los mismos. En climas templados, el llenado de los granos tiene lugar bajo una oferta de radiación elevada, excepto en siembras muy tardías. Esta situación de elevada radiación resulta en un crecimiento post-floración que generalmente excede el crecimiento de los granos, pudiendo dar lugar durante esta etapa principalmente a una limitación del rendimiento por el número de destinos reproductivos fijados (i.e. número de granos por planta).

En términos generales el peso del grano es considerado el componente del rendimiento más estable, ya que pese a la reducción en el peso individual del grano, el rendimiento en maíz aumenta al incrementarse el NGP [123].

1.2.3. Periodo crítico

A través de diferentes aproximaciones experimentales se determinó que el NG queda establecido en un período de aproximadamente 30 días centrado en la floración, motivo por el cual se definió a esta etapa como período crítico. La incidencia de un estrés hídrico [124], [125], [126]o lumínico [127] provoca mayores mermas en el NG cuando tiene lugar en este período, que coincide con el crecimiento activo de la espiga, la emergencia de estigmas y el inicio del llenado del grano. La fuente de variación más importante en el número de granos por planta del maíz está constituida por el aborto de flores fecundadas en este período.

Durante el periodo crítico, la tasa de producción de biomasa aérea (PBA) es indicativa de la condición fisiológica de las mismas y, por lo tanto, de su capacidad para fijar granos [128]. La PBA depende de numerosos factores, como la densidad de siembra, la temperatura, los niveles de radiación, la disponibilidad de agua y nutrientes y el genotipo [129], [130].

Sin deficiencias hídricas y/o nutricionales, el NG fijado por planta se relaciona con la PBA en el período que rodea a la floración [117]. En la figura se observa que la respuesta curvilínea observada se caracteriza por un umbral de PBA por debajo del cual no hay fijación de granos, un valor de PBA por encima del cual incrementos de la PBA sólo generan pequeños aumentos del NG, esto determina un umbral de PBA para lograr prolificidad, es decir, fijación de granos en la segunda espiga [131].

1.2.4. Cobertura de canopeo

Monteith [132], Sadras y Connor [133] y Mercau [134], establecieron que:

1. El rendimiento es proporcional a la intercepción de la luz durante la temporada de crecimiento y, por lo tanto, proporcional a la cubierta vegetal del dosel
2. El rendimiento es proporcional a la fracción de lluvia estacional que ocurre después de la antesis.

Esto se basa en la relación entre el índice de cosecha y la fracción del uso de agua estacional que ocurre después de la antesis. (Figura 5.4)

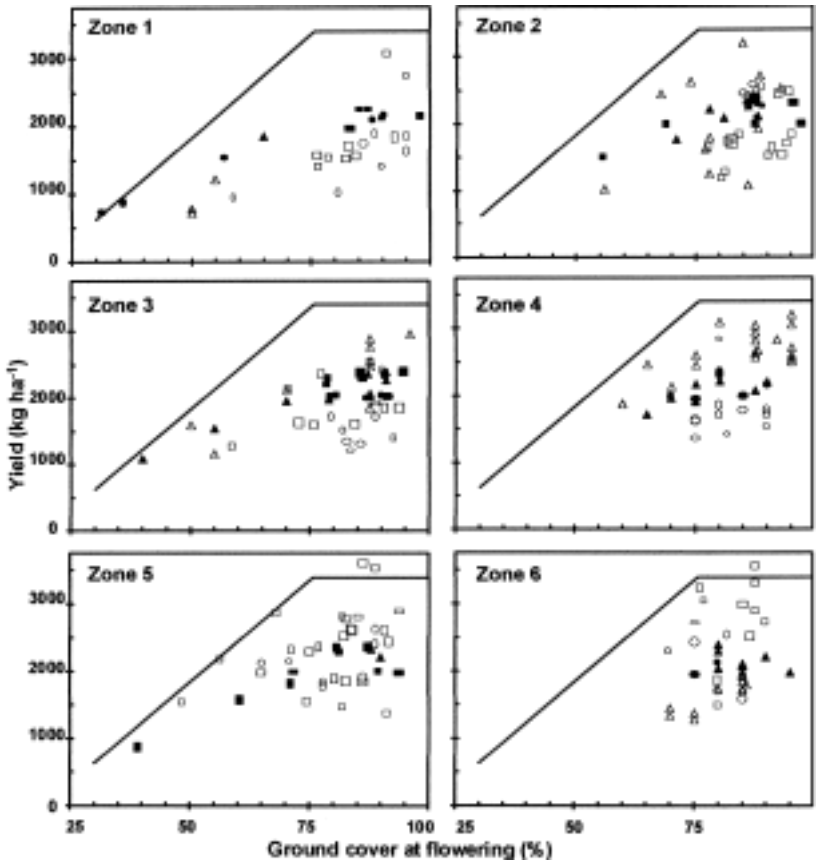

Figura 5.4: Mercau [134] documentó la relación entre el rendimiento y la cubierta vegetal del dosel en seis zonas agroecológicas.

1.2.5. Relación entre canopeo y el índice de área foliar

El índice de área foliar (LAI), un importante parámetro biofísico de la vegetación, es un variable adimensional calculada a partir de establecer una relación entre el área foliar y la unidad de superficie del suelo [135].

Maddonni [18], utilizando este índice, y estudios realizados previamente por[17] midió las relaciones encadenadas desde una escala de hoja (Figura 5.5) a nivel de laboratorio y la radiación interceptada por el canopeo de un cultivo agrícola (Figura 2.1), estableciendo una relación entre la biomasa aérea y la radiación interceptada por el canopeo (Figura 2.2) .

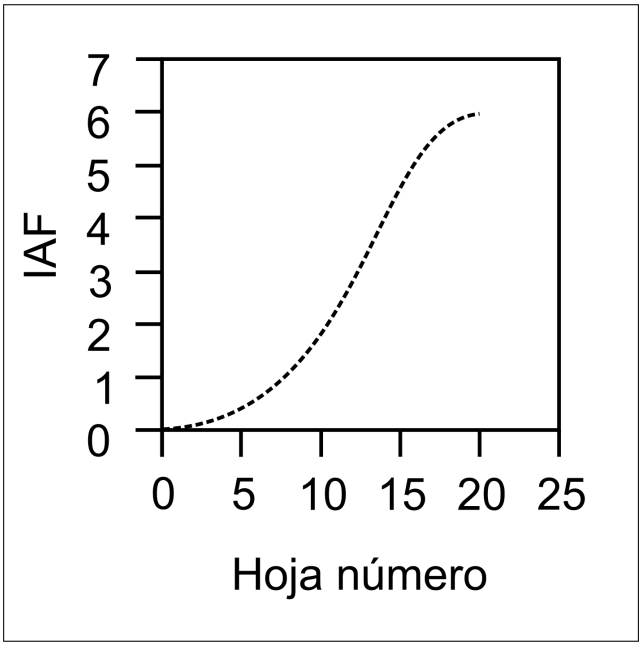

Figura 5.5: Adaptada de [18] Relación entre número de hojas y índice de área foliar.

2. Anexo II

2.1. Histogramas de las distribuciones de las variables predictora.

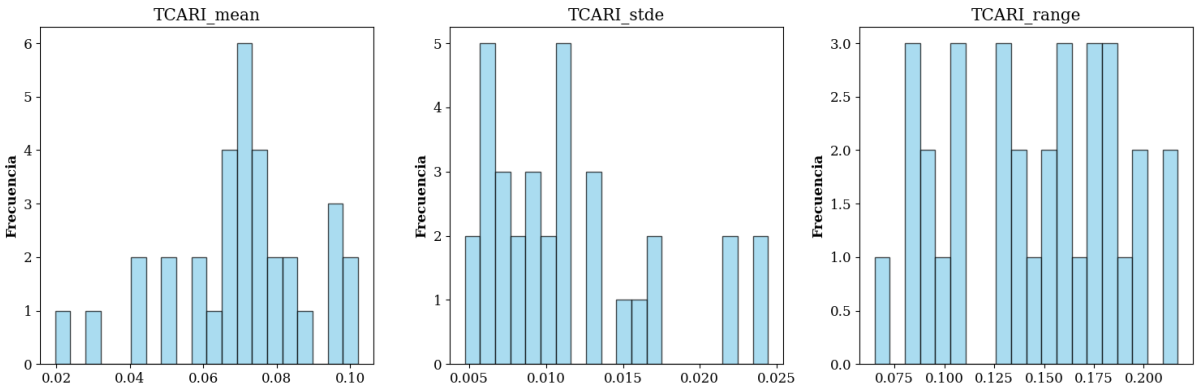

Figura 5.6: Distribución de la variable - Índice de Absorción de Clorofila en Reflectancia Transformado - TCARI. (i) media, (ii) desvío estándar y (iii) rango.

Como se observa en la 5.6) , la variable media TCARI presenta una distribución normal. Por otro lado, la variable desvío estándar de TCARI da cuenta de que esta variable presenta gran variación entre los lotes estudiados.

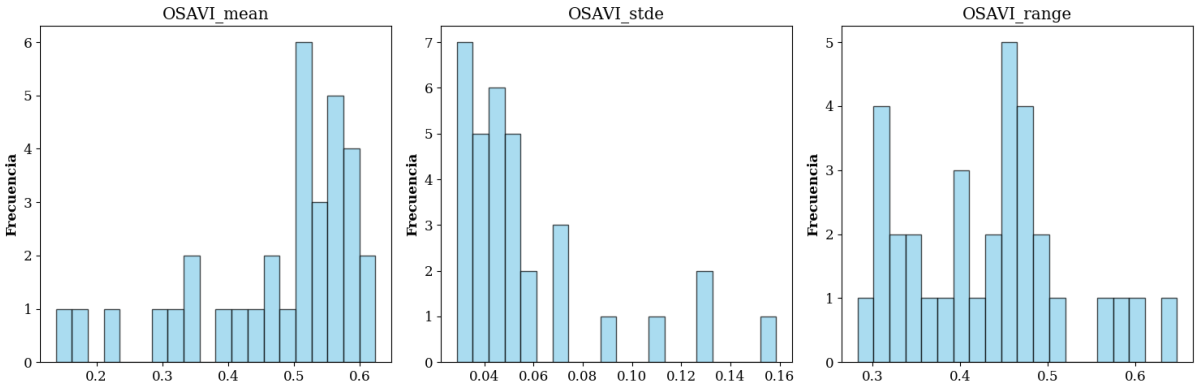

Figura 5.7: Distribución de la variable - Índice de Vegetación Ajustado al Suelo Optimizado - OSAVI (i) media, (ii) desvío estándar y (iii) rango.

La variable media de OSAVI 5.7) presenta valores altos para la mayoría de los lotes evidenciando una distribución normal con una gran cola negativa. Por otro lado, la variable desvío estándar y rango dan cuenta de que la variable no varía mucho entre los lotes estudiados.

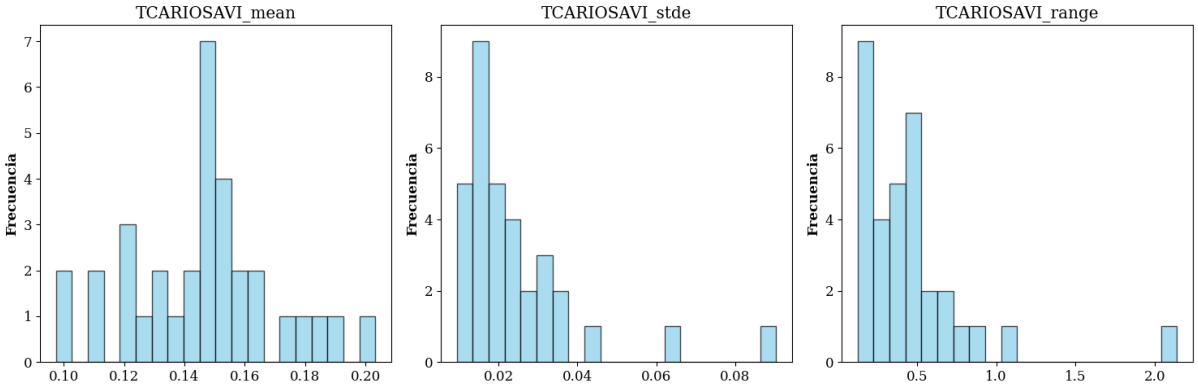

Figura 5.8: Distribución de la variable - Índice TCARI/OSAVI (i) media, (ii) desvío estándar y (iii) rango.

En la Figura 5.8) se evidencia que la variable media de TCARI/OSAVI presenta una distribución normal de los datos y la variable desvío estándar y rango dan cuenta de la poca variabilidad y robustez de la variable a lo largo de los lotes analizados.

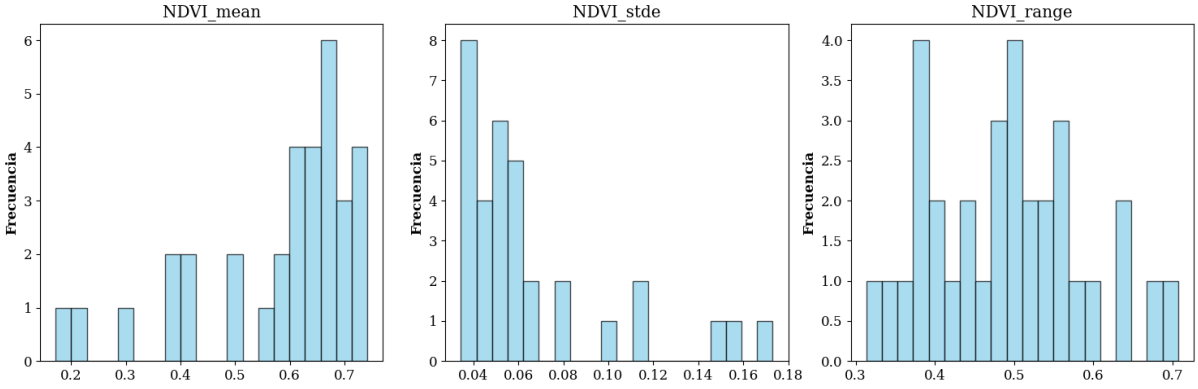

Figura 5.9: Distribución de la variable - Índice de vegetación de diferencia normalizada - NDVI (i) media, (ii) desvío estándar y (iii) rango.

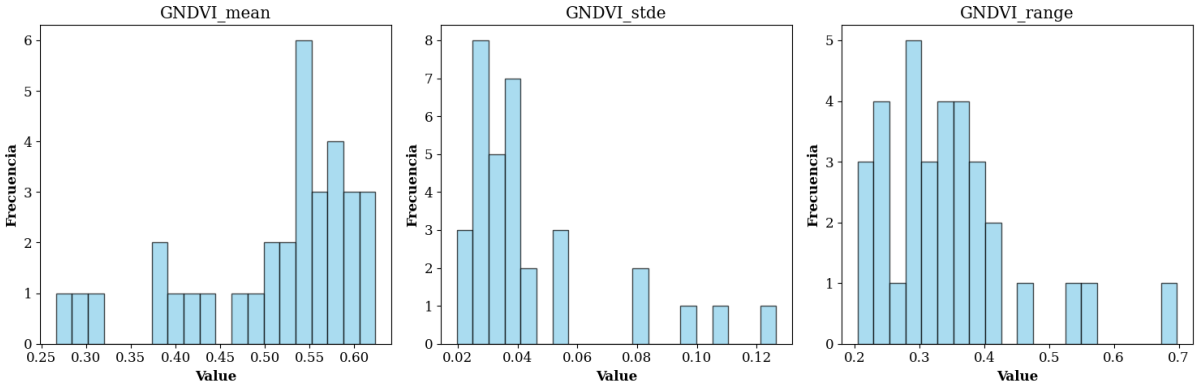

Figura 5.10: Distribución de la variable - Índice de vegetación de diferencia normalizada verde - GNDVI (i) media, (ii) desvío estándar y (iii) rango.

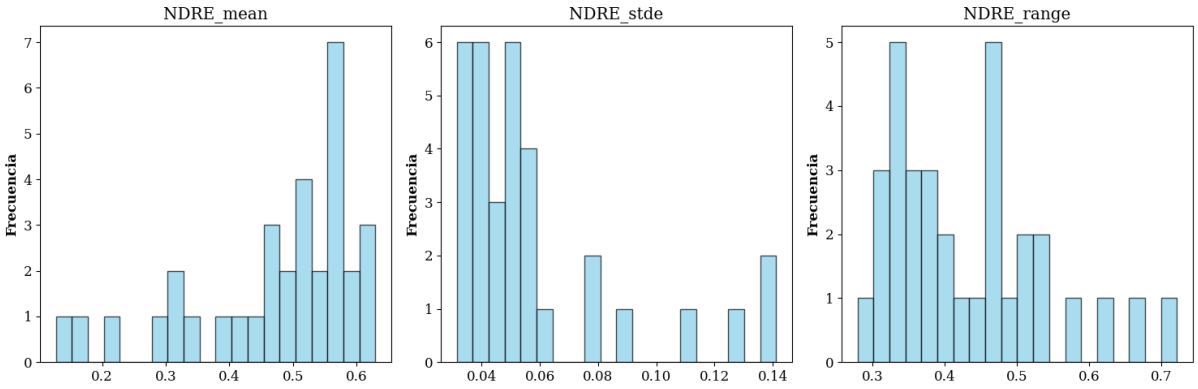

Figura 5.11: Distribución de la variable - Índice de borde rojo de diferencia normalizada - NDRE (i) media, (ii) desvío estándar y (iii) rango.

Como se observa en las figuras5.11, 5.9, 5.10 las variables medias de NDVI, NDRE y GNDVI presentan distribuciones normal con valores relativamente altos, mientras que las variables desvío estándar dan cuenta de la relativamente poca variabilidad de los datos.

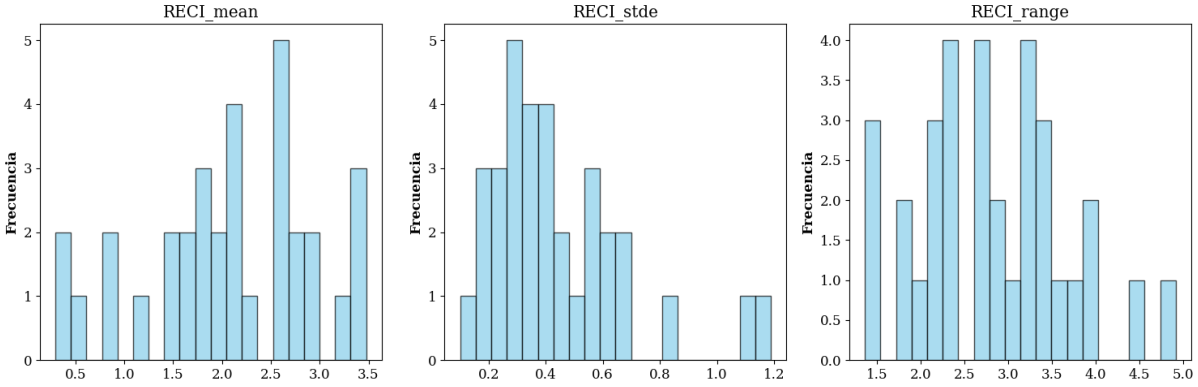

Figura 5.12: Distribución de la variable - índice de vegetación de clorofila de borde rojo- RECI (i) media, (ii) desvío estándar y (iii) rango.

En la Figura 5.12 se observa que la variable media de RECI tiene una distribución normal, al igual que su desvío estándar y rango.

Las variables 5.13 focalizadas en el relieve dan cuenta de la no homogeneidad de los lotes estudiados en cuanto a sus características topográficas. Si bien la mayoría de los lotes se encuentran en zonas de alrededor de 100 metros de altitud (oeste y norte buenos aires), se estudiaron lotes a 300 metros de altura correspondientes al sur de la provincia de Córdoba. Por otro lado, las pendientes observadas fueron mayoritariamente bajas (menores a 1,6 %) y de poca variabilidad.

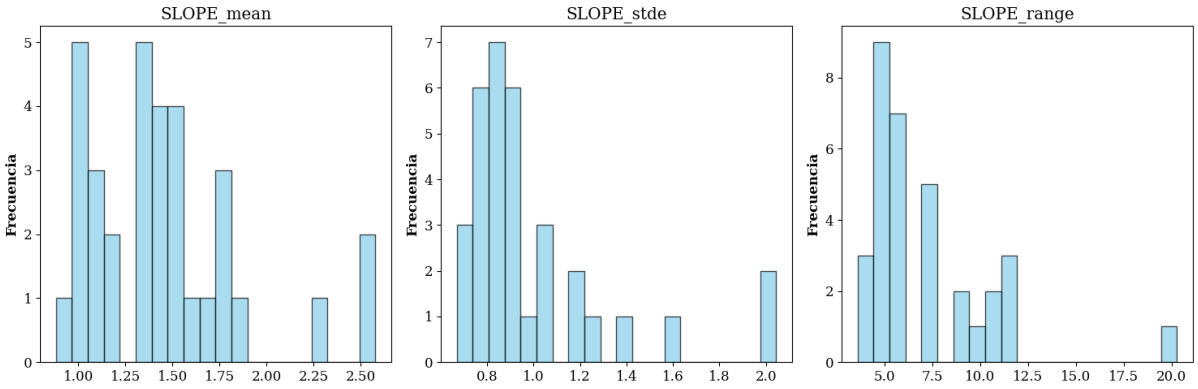

Figura 5.13: Distribución de la variables topográficas. (i) modelo de superficie digital (DSM), (ii) desvío estándar de la pendiente (SLOPE) (iii) media de la pendiente (iv) rango de la pendiente.

2.2. Test Shapiro Wilk de las distribuciones de las variables predictora.

Tabla 5.1: Resultados del test de Shapiro-Wilk para las variables

Variable	p-valor del Shapiro-Wilk Test	Distribución
NDVI_range	0.840073	Normal
RECI_range	0.798197	Normal
TCARIOSAVI_mean	0.592774	Normal
RECI_mean	0.434631	Normal
TCARI_range	0.363971	Normal
OSAVI_range	0.265543	Normal
TCARI_mean	0.100235	Normal
NDRE_range	0.022036	No es normal
GNDVI_range	0.003628	No es normal
TCARI_stde	0.002359	No es normal
NDRE_mean	0.002062	No es normal
SLOPE_mean	0.001602	No es normal
GNDVI_mean	0.001380	No es normal
OSAVI_mean	0.000985	No es normal
RECI_stde	0.000858	No es normal
NDVI_mean	0.000259	No es normal
SLOPE_range	0.000011	No es normal
OSAVI_stde	0.000007	No es normal
NDRE_stde	0.000004	No es normal
NDVI_stde	0.000004	No es normal
GNDVI_stde	0.000004	No es normal
SLOPE_stde	0.000003	No es normal
TCARIOSAVI_stde	0.000002	No es normal
TCARIOSAVI_range	0.000001	No es normal

3. Anexo III

3.1. Correlación lineal entre las variables independientes y la variable dependiente.

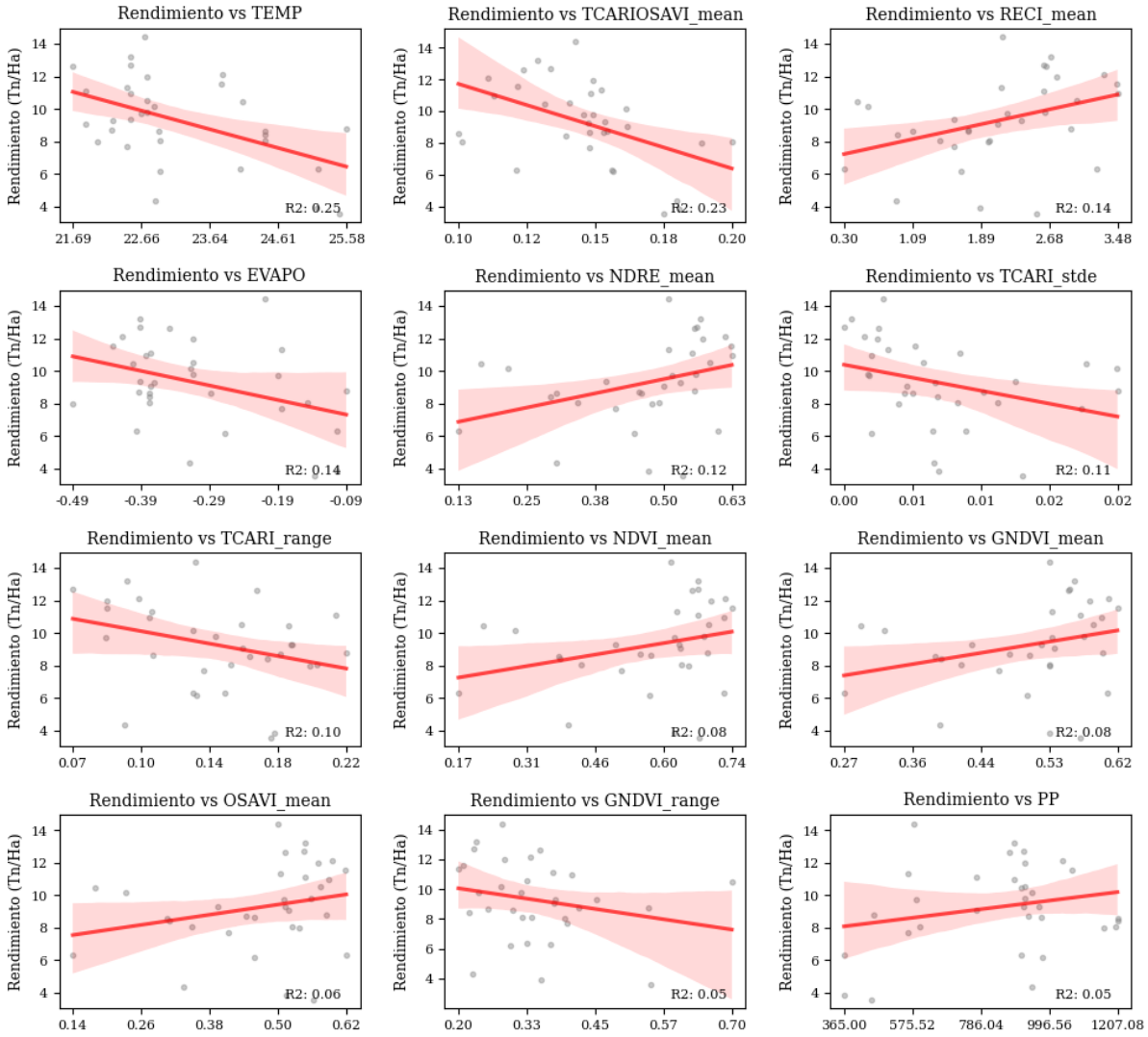

Figura 5.14: Correlación lineal entre las variables independientes y la variable dependiente

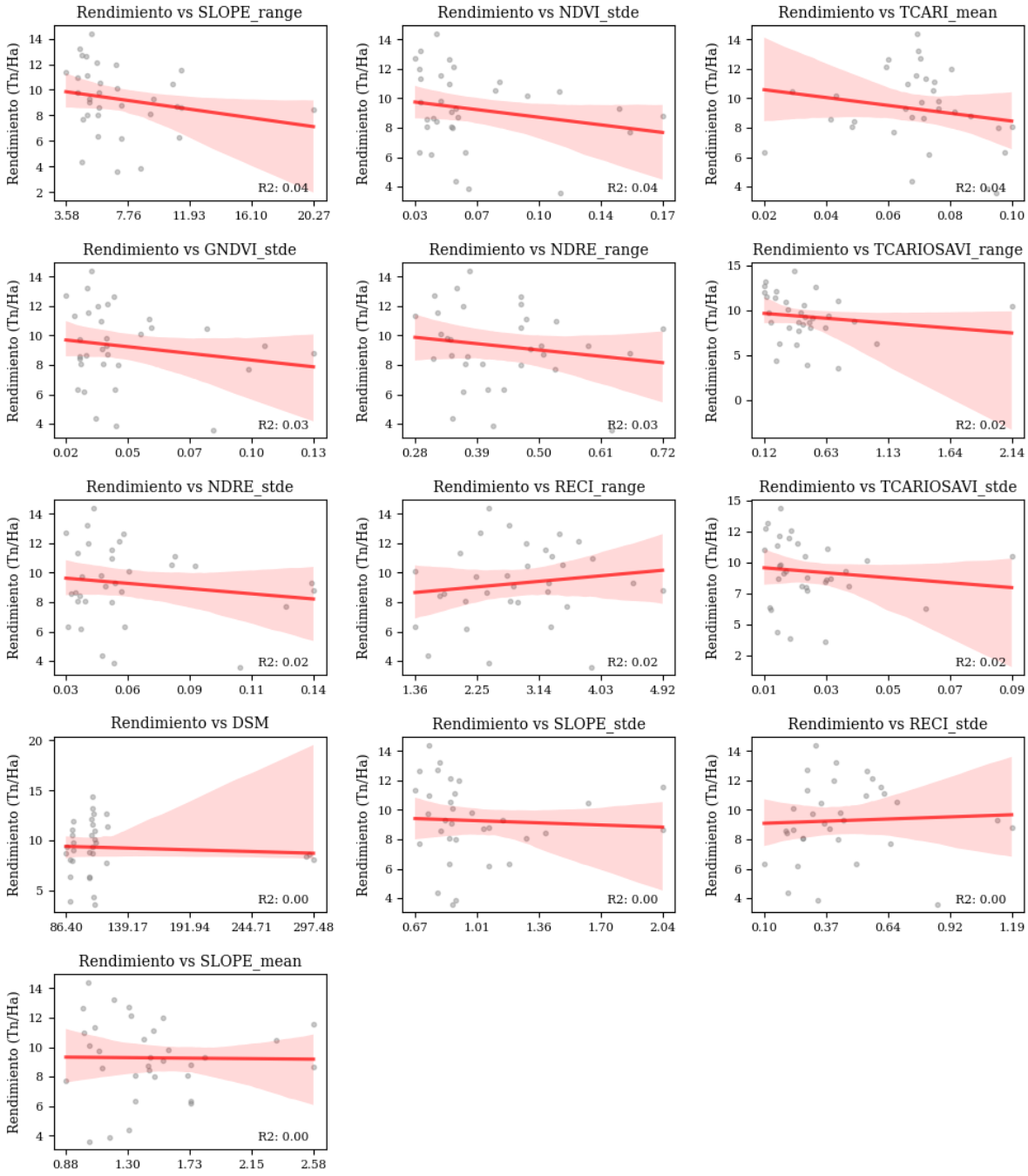

Figura 5.15: Correlación lineal entre las variables independientes y la variable dependiente

4. Anexo IV

4.1. Importancia de variables por pureza de nodos y por permutación.

Tabla 5.2: Importancia de variables por pureza de nodos y por permutación

Variable	Pesos (Media)	Pesos (Desvío Estándar)
TCARI_stde	0.158606	0.525354
NDRE_mean	0.134242	0.295387
PP	0.11128	0.109272
TCARIOSAVI_mean	0.156412	0.066755
GNDVI_mean	0.081966	0.102862
TEMP	0.060988	0.063268
EVAPO	0.056898	0.021508
TCARI_mean	0.054269	0.054594
TCARI_range	0.042476	0.058098
GNDVI_range	0.028563	0.054788
OSAVI_range	0.020686	0.00782
NDRE_stde	0.054594	0.01853
OSAVI_stde	0.047838	0.017274
NDVI_stde	0.016874	0.015632
RECI_mean	0.040504	0.010034
RECI_range	0.032666	0.008645
NDRE_range	0.012332	0.012605
TCARIOSAVI_range	0.027185	0.012168
NDVI_stde	0.020785	0.004831
NDVI_mean	0.011444	0.003168
RECI_mean	0.011402	0.012037
DSM	0.018043	0.011184
OSAVI_range	0.014683	0.015328
SLOPE_range	0.010202	0.013282
SLOPE_stde	0.013282	0.009619
GNDVI_range	0.012863	0.012302
RECI_range	0.007356	0.011549
SLOPE_mean	0.007211	0.008433
SLOPE_range	0.005346	0.007142
OSAVI_mean	0.00564	0.010709
NDVI_mean	0.005305	0.005225
NDVI_range	0.001239	0.00245
NDVI_range	0.002661	-0.001329
GNDVI_stde	0.000708	0.000245
OSAVI_mean	-0.008146	0.013483

5. Anexo V

5.1. Código Análisis exploratorio.

5.1.1. Importar librerías y configuraciones

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import numpy as np
import matplotlib.ticker as ticker
from scipy.stats import linregress
from scipy.stats import spearmanr
import scipy
import scipy.cluster.hierarchy as sch

# Set the default language for Matplotlib to Spanish
plt.rcParams["font.family"] = "serif"
plt.rcParams["font.size"] = 12
plt.rcParams['axes.labelweight'] = 'bold'
```

5.1.2. Importar base de datos

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df.info()
```

```
<class 'pandas.core.frame.df'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 33 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Lote              33 non-null     object
 1   Año               33 non-null     int64
 2   DSM               33 non-null     float64
 3   EVAPO             33 non-null     float64
 4   PP                33 non-null     float64
 5   TEMP              33 non-null     float64
 6   GNDVI_mean        33 non-null     float64
 7   GNDVI_stde        33 non-null     float64
 8   GNDVI_range       33 non-null     float64
 9   NDRE_mean         33 non-null     float64
 10  NDRE_stde         33 non-null     float64
```

```
 11  NDRE_range          33 non-null     float64
 12  NDVI_mean           33 non-null     float64
 13  NDVI_stde           33 non-null     float64
 14  NDVI_range          33 non-null     float64
 15  OSAVI_mean          33 non-null     float64
 16  OSAVI_stde          33 non-null     float64
 17  OSAVI_range         33 non-null     float64
 18  RECI_mean           33 non-null     float64
 19  RECI_stde           33 non-null     float64
 20  RECI_range          33 non-null     float64
 21  SLOPE_mean          33 non-null     float64
 22  SLOPE_stde          33 non-null     float64
 23  SLOPE_range         33 non-null     float64
 24  TCARIOSAVI_mean     33 non-null     float64
 25  TCARIOSAVI_stde     33 non-null     float64
 26  TCARIOSAVI_range    33 non-null     float64
 27  TCARI_mean          33 non-null     float64
 28  TCARI_stde          33 non-null     float64
 29  TCARI_range         33 non-null     float64
 30  REND_mean           33 non-null     float64
 31  REND_stde           33 non-null     float64
 32  REND_range          33 non-null     float64
dtypes: float64(31), int64(1), object(1)
memory usage: 8.6+ KB
```

5.1.3. Análisis de la variable independiente

5.1.4. Histograma

```
# Select the independent variable (Año) for analysis
independent_variable = 'REND_mean'

# Filter out rows with missing values in the selected␣
 ↪independent variable
data = df.dropna(subset=[independent_variable])

# Convert the independent variable column to the appropriate␣
 ↪data type (float)
data[independent_variable] = data[independent_variable].
 ↪astype(float)

# Descriptive statistics
statistics = data[independent_variable].describe()

# Distribution plot
plt.figure(figsize=(10, 6))
```

```
#sns.histplot(data=data, x=independent_variable, bins=10,␣
 ↪kde=True)
sns.histplot(data=data, x=independent_variable, bins=30,␣
 ↪color='skyblue', edgecolor='black', alpha=0.7 , kde=True␣
 ↪)#, kde_kws={'color': 'black'})
plt.title(f'Distribucion variable independiente')
plt.xlabel('Rendimiento (Tn/Ha)')
plt.ylabel('Frecuencia')
plt.show()
```

5.1.5. Test de normalidad Shapiro-Wilk

```
from scipy.stats import shapiro
stat, p = shapiro(data[independent_variable])
print(f"Shapiro-Wilk test p-value: {p}")
if p > 0.05:
    print(f"{independent_variable} appears to be normally␣
 ↪distributed.")
else:
    print(f"{independent_variable} does not appear to be␣
 ↪normally distributed.")

# Example: T-test to compare means
from scipy.stats import ttest_ind
group1 = data[data[independent_variable] ==␣
 ↪2016][independent_variable]
group2 = data[data[independent_variable] ==␣
 ↪2017][independent_variable]
t_stat, p_value = ttest_ind(group1, group2)
print(f"T-test p-value: {p_value}")
if p_value < 0.05:
    print(f"There is a significant difference in means␣
 ↪between 2016 and 2017.")
else:
    print(f"No significant difference in means between 2016␣
 ↪and 2017.")
```

```
Shapiro-Wilk test p-value: 0.6197720766067505
REND_mean appears to be normally distributed.
T-test p-value: nan
No significant difference in means between 2016 and 2017.
```

5.1.6. Boxplot

```
# Group the data by 'Año'
grouped = df.groupby('Año')

# Create an empty list to store the data for each year
data_to_plot = []

# Iterate over each group (each year) and extract the␣
 ↪required columns
for year, group_data in grouped:
    rend_media = group_data['REND_mean']
    rend_stde = group_data['REND_stde']
    rend_range = group_data['REND_range']

    # Combine the data into a single df for the box plot
    combined_data = pd.df({'Year': year, 'REND_mean':␣
 ↪rend_media, 'REND_stde': rend_stde, 'REND_range':␣
 ↪rend_range})

    # Append the combined data to the list
    data_to_plot.append(combined_data)

# Create a box plot for each year using the 'Rend_media',␣
 ↪'Rend_stde', and 'Rend_var' variables
fig, ax = plt.subplots(figsize=(10, 6))
plt.title('Rendimiento medido a campo para cada año')
plt.xlabel('Año')
plt.ylabel('Rendimiento (Tn/Ha)')
#print(data_to_plot)
# Use the boxplot function to create the box plot
ax.boxplot([data['REND_mean'] for data in data_to_plot],␣
 ↪labels=[str(year) for year in grouped.groups.keys()])
plt.xticks(rotation=45)

# Show the plot
plt.show()
```

5.1.7. Boxplot

```
# List of variables to create separate box plots for
variables_to_plot = ['GNDVI_mean', 'NDRE_mean', 'NDVI_mean',␣
 ↪'OSAVI_mean', 'RECI_mean', 'TCARIOSAVI_mean',␣
 ↪'TCARI_mean', 'SLOPE_mean']
```

```
# Create a mapping of variable names without the "_mean" suffix
variable_labels = {var: var.replace('_mean', '') for var in variables_to_plot}

# Group the data by 'Año'
grouped = df.groupby('Año')

# Create an empty list to store the data for each variable
data_to_plot = {var: [] for var in variables_to_plot}

# Iterate over each group (each year) and extract the required columns for each variable
for year, group_data in grouped:
    for var in variables_to_plot:
        variable_data = group_data[var]
        combined_data = pd.df({'Year': year, var: variable_data})
        data_to_plot[var].append(combined_data)

# Create and save separate figures for each variable
for var in variables_to_plot:
    fig, ax = plt.subplots(figsize=(10, 6))
    ax.set_title(f'{variable_labels[var]} de cada año')
    ax.set_xlabel('Año')
    ax.set_ylabel(variable_labels[var])  # Use the variable label without "_mean"

    # Use the boxplot function to create the box plot
    ax.boxplot([data[var] for data in data_to_plot[var]], labels=[str(year) for year in grouped.groups.keys()])
    ax.set_xticklabels([str(year) for year in grouped.groups.keys()], rotation=45)

    # Save the figure with a unique name
    fig.savefig(f'{variable_labels[var]}_boxplot.png')
    plt.close(fig)  # Close the current figure to avoid overlap

# Show the plots if needed
#plt.show()
```

5.1.8. Histograma todas las variables

```
df = df.drop(columns = ["Año", "Lote","DSM", "EVAPO", "PP",␣
 ↪"TEMP", "REND_mean", "REND_stde", "REND_range"])

# Define una función para generar los gráficos con la curva␣
 ↪de distribución normal y el test de Shapiro-Wilk
def generate_histograms_with_normal_curve(group_df):
    # Selecciona las columnas del grupo
    group_columns = group_df.columns

    # Crea un nuevo subplot para cada columna
    fig, axes = plt.subplots(nrows=1,␣
 ↪ncols=len(group_columns), figsize=(15, 5))

    # Genera un histograma y la curva de distribución normal␣
 ↪para cada columna
    for i, col in enumerate(group_columns):
        data = group_df[col]

        # Histograma
        axes[i].hist(data, bins=20, color='skyblue',␣
 ↪edgecolor='black', alpha=0.7, density=True,␣
 ↪label='Histograma')


        # Curva de distribución normal
        #mean, std_dev = data.mean(), data.std()
        #x = np.linspace(data.min(), data.max(), 100)
        #pdf = stats.norm.pdf(x, mean, std_dev)
        #axes[i].plot(x, pdf, 'r-', label='Distribuion␣
 ↪normal')

        # Test de Shapiro-Wilk
        stat, p_value = stats.shapiro(data)
        axes[i].set_title(col)
        #axes[i].set_title(f'{col}\nShapiro p-value:␣
 ↪{p_value:.4f}')
        axes[i].set_xlabel("")
        axes[i].set_ylabel("Frecuencia")
        axes[i].legend()

    # Ajusta el espacio entre los subplots
    plt.tight_layout()

# Agrupa las variables en grupos de tres
```

```
grouped_columns = [df.columns[i:i+3] for i in range(0,␣
 ↪len(df.columns), 3)]

# Ordena cada grupo por la primera palabra del nombre de la␣
 ↪variable
sorted_grouped_columns = []
for group_columns in grouped_columns:
    sorted_group_columns = sorted(group_columns, key=lambda␣
 ↪x: x.split('_')[0])
    sorted_grouped_columns.append(sorted_group_columns)

# Genera los gráficos para cada grupo de tres variables con␣
 ↪la curva de distribución normal y el test de Shapiro-Wilk
for group_columns in sorted_grouped_columns:
    group_df = df[group_columns]
    generate_histograms_with_normal_curve(group_df)

# Muestra los gráficos
plt.show()
```

5.1.9. Histograma - variables climáticas

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
selected_columns = ['PP', 'EVAPO', 'TEMP']
df = df[selected_columns]
# Define una función para generar los gráficos con la curva␣
 ↪de distribución normal y el test de Shapiro-Wilk
def generate_histograms_with_normal_curve(group_df):
    # Selecciona las columnas del grupo
    group_columns = group_df.columns

    # Crea un nuevo subplot para cada columna
    fig, axes = plt.subplots(nrows=1,␣
 ↪ncols=len(group_columns), figsize=(15, 5))

    # Genera un histograma y la curva de distribución normal␣
 ↪para cada columna
    for i, col in enumerate(group_columns):
        data = group_df[col]

        # Histograma
        axes[i].hist(data, bins=20, color='skyblue',␣
 ↪edgecolor='black', alpha=0.7, density=True,␣
 ↪label='Histograma')
```

```
        # Curva de distribución normal
        #mean, std_dev = data.mean(), data.std()
        #x = np.linspace(data.min(), data.max(), 100)
        #pdf = stats.norm.pdf(x, mean, std_dev)
        #axes[i].plot(x, pdf, 'r-', label='Distribuion␣
 ↪normal')

        # Test de Shapiro-Wilk
        stat, p_value = stats.shapiro(data)
        axes[i].set_title(col)
        #axes[i].set_title(f'{col}\nShapiro p-value:␣
 ↪{p_value:.4f}')
        axes[i].set_xlabel("")
        axes[i].set_ylabel("Frecuencia")
        axes[i].legend()

    # Ajusta el espacio entre los subplots
    plt.tight_layout()

# Agrupa las variables en grupos de tres
grouped_columns = [df.columns[i:i+3] for i in range(0,␣
 ↪len(df.columns), 3)]

# Ordena cada grupo por la primera palabra del nombre de la␣
 ↪variable
sorted_grouped_columns = []
for group_columns in grouped_columns:
    sorted_group_columns = sorted(group_columns, key=lambda␣
 ↪x: x.split('_')[0])
    sorted_grouped_columns.append(sorted_group_columns)

# Genera los gráficos para cada grupo de tres variables con␣
 ↪la curva de distribución normal y el test de Shapiro-Wilk
for group_columns in sorted_grouped_columns:
    group_df = df[group_columns]
    generate_histograms_with_normal_curve(group_df)

# Muestra los gráficos
plt.show()
```

5.1.10. Histograma - variable pendiente

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
#df.info()
# Select the independent variable (Año) for analysis
pendiente = 'DSM'

# Filter out rows with missing values in the selected␣
 ↪independent variable
data = df.dropna(subset=[pendiente])

# Convert the independent variable column to the appropriate␣
 ↪data type (float)
data[pendiente] = data[pendiente].astype(float)

# Descriptive statistics
statistics = data[pendiente].describe()

# Distribution plot
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x=pendiente, bins=30,␣
 ↪color='skyblue', edgecolor='black', alpha=0.7, kde=False)
plt.title(f'{pendiente}')
#plt.title(f'Distribucion variable independiente')
plt.xlabel('')
plt.ylabel('Frecuencia')
plt.show()
```

5.1.11. Test de Shapiro-wilk de la distribucion de las variables

```
# Crea un df para almacenar los resultados
results_df = pd.df(columns=['Variable', 'Shapiro-Wilk',␣
 ↪'Comentario'])

# Itera a través de las columnas del df
for col in df.columns:
    data = df[col]

    # Realiza el test de Shapiro-Wilk
    stat, p_value = stats.shapiro(data)

    # Comentario basado en el p-valor
    if p_value < 0.05:
```

```
        comentario = 'No pasa el test (no es normal)'
    else:
        comentario = 'Pasa el test (aproximadamente normal)'

    # Agrega los resultados a la tabla
    results_df = results_df.append({'Variable': col,␣
 ↪'Shapiro-Wilk': p_value, 'Comentario': comentario},␣
 ↪ignore_index=True)

# Ordena la tabla por el valor del test de mayor a menor
results_df = results_df.sort_values(by='Shapiro-Wilk',␣
 ↪ascending=False)

# Muestra la tabla
#print(results_df)
```

5.1.12. Matriz de correlación

- Las matrices de correlación tienen el inconveniente de tener un tamaño notable cuando se dispone de muchas variables. Para facilitar la #identificación de pares de variables con correlaciones altas, es conveniente convertirlas en formato de tabla larga (tidy).

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df = df.drop(columns = ["Año", "Lote","DSM", "PP", "EVAPO",␣
 ↪"TEMP"])
#df.info()
```

```
#Function to plot the correlation matrix of a df.

def plot_corr(df,size=10):
    '''Plot a graphical correlation matrix for a df.

    Input:
        df: pandas df
        size: vertical and horizontal size of the plot'''

    %matplotlib inline
    import matplotlib.pyplot as plt

    # Compute the correlation matrix for the received df
    corr = df.corr()

    # Plot the correlation matrix with columns sorted by␣
 ↪their names
```

```
    sorted_columns = sorted(corr.columns)
    sorted_corr = corr[sorted_columns].
 ↪reindex(sorted_columns)

    # Plot the correlation matrix
    fig, ax = plt.subplots(figsize=(size, size))
    cax = ax.matshow(corr, cmap='RdYlGn')
    plt.xticks(range(len(corr.columns)), sorted_corr.
 ↪columns, rotation=90);
    plt.yticks(range(len(corr.columns)), sorted_corr.
 ↪columns);

    # Add the colorbar legend
    cbar = fig.colorbar(cax, ticks=[-1, 0, 1], aspect=40,␣
 ↪shrink=.8)

    # Highlight lines every 3 cells
    for i in range(0, len(sorted_corr.columns), 3):
        plt.axvline(x=i - 0.5, color='k', linewidth=2)
        plt.axhline(y=i - 0.5, color='k', linewidth=2)


X = df.corr().values
d = sch.distance.pdist(X)    # vector of ('55' choose 2)␣
 ↪pairwise distances
L = sch.linkage(d, method='complete')
ind = sch.fcluster(L, 0.5*d.max(), 'distance')
columns = [df.columns.tolist()[i] for i in list((np.
 ↪argsort(ind)))]
df = df.reindex(columns, axis=1)

plot_corr(df, size=18)
```

```
# Two pass clustering
# 1-We cluster the corr matrix
#   We sort the survey data according to this clustering
# 2-For cluster bigger than a threshold we cluster those␣
 ↪sub-clusters

cluster_th = 4
#df = df.reindex(columns=columns)
X = df.corr().values
d = sch.distance.pdist(X)
L = sch.linkage(d, method='complete')
ind = sch.fcluster(L, 0.5*d.max(), 'distance')
```

```
columns = [df.columns.tolist()[i] for i in list(np.
 ↪argsort(ind))]
df = df.reindex(columns=columns)

#df = df.reindex_axis(columns, axis=1)

unique, counts = np.unique(ind, return_counts=True)
counts = dict(zip(unique, counts))

i = 0
j = 0
columns = []
for cluster_l1 in set(sorted(ind)):
    j += counts[cluster_l1]
    sub = df[df.columns.values[i:j]]
    if counts[cluster_l1]>cluster_th:
        X = sub.corr().values
        d = sch.distance.pdist(X)
        L = sch.linkage(d, method='complete')
        ind = sch.fcluster(L, 0.5*d.max(), 'distance')
        col = [sub.columns.tolist()[i] for i in list((np.
 ↪argsort(ind)))]
        sub = sub.reindex(col, axis=1)
    cols = sub.columns.tolist()
    columns.extend(cols)
    i = j
df = df.reindex(columns, axis=1)

plot_corr(df, 18)
```

5.1.13. Análisis de la correlación lineal entre las variables

```
# Calculate correlations with 'Rend_media' and sort by␣
 ↪absolute value
correlations = df.select_dtypes(include=['float64']).
 ↪drop('REND_mean', axis=1).apply(lambda x: x.
 ↪corr(df['REND_mean'])).abs().sort_values(ascending=False)

# Get the columns in the order of correlation (from high to␣
 ↪low)
columnas_numeric = correlations.index

#columnas_numeric = df.select_dtypes(include=['float64']).
 ↪columns
#columnas_numeric = columnas_numeric.drop('Rend_media')
```

```
# Calculate the number of plots needed
num_plots = len(columnas_numeric)

# Calculate the number of rows and columns for subplots
num_rows = (num_plots + 2) // 3  # 3 columns and variable␣
 ↪number of rows
num_cols = min(num_plots, 3)

# Create subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols,␣
 ↪figsize=(10, 22))
axes = axes.flat

# Loop through numeric columns and create scatter plots
for i, column in enumerate(columnas_numeric):
    row, col = divmod(i, num_cols)

    # Scatter plot
    sns.regplot(
        x=df[column],
        y=df['REND_mean'],
        color="gray",
        marker='.',
        scatter_kws={"alpha": 0.4},
        line_kws={"color": "r", "alpha": 0.7},
        ax=axes[i]
    )

    # Calculate the R-squared value
    slope, intercept, r_value, p_value, std_err =␣
 ↪linregress(df[column], df['REND_mean'])
    r_squared = r_value**2

    # Set x-ticks based on data range
    x_min, x_max = df[column].min(), df[column].max()
    x_tick_values = np.linspace(x_min, x_max, 5)  # Adjust␣
 ↪the number of ticks as needed

    # Configure x-ticks and labels
    axes[i].set_xticks(x_tick_values)
    axes[i].set_xticklabels(["{:.2f}".format(val) for val in␣
 ↪x_tick_values])

    # Format y-axis labels with commas
    axes[i].yaxis.set_major_formatter(plt.
 ↪FuncFormatter(lambda x, loc: "{:,}".format(int(x))))
```

```
    axes[i].xaxis.set_major_formatter(plt.
↪FuncFormatter(lambda x, loc: "{:.2f}".format(x)))
    axes[i].tick_params(labelsize=8)
    axes[i].set_xlabel("")
    axes[i].set_ylabel("Rendimiento (Tn/Ha)", fontsize=9,␣
↪fontweight="normal")

     # Plot title with R-squared value
    title = f"Rendimiento vs {column}"
    axes[i].set_title(title, fontsize=10,␣
↪fontweight="normal")

    # Add R2 value as text inside the subplot
    axes[i].text(
        0.75, 0.05,  # Adjust the position of the text as␣
↪needed
        f"R2: {r_squared:.2f}",
        transform=axes[i].transAxes,
        fontsize=8,
        color="k",
        fontweight="normal"
    )

# Remove any empty subplots
for i in range(num_plots, num_rows * num_cols):
    fig.delaxes(axes[i])

fig.tight_layout()
plt.subplots_adjust(top=1)
plt.show()
```

5.1.14. Cálculo del coeficiente de Spearman

```
df = pd.read_csv(r'D:
↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
↪csv')
#df.info()

# Seleccionar las columnas relevantes para el análisis (por␣
↪ejemplo, GNDVI_mean y Rend_media)
x = df['GNDVI_mean']
y = df['REND_mean']

# Calcular el coeficiente de correlación de Spearman y el␣
↪valor p
corr_spearman, p_value = spearmanr(x, y)
```

```
# Graficar la relación entre las dos variables
plt.scatter(x, y)
plt.xlabel('GNDVI_mean')
plt.ylabel('REND_mean')
plt.title(f'Correlación de Spearman: {corr_spearman:.2f}, Valor p: {p_value:.2f}')
plt.show()

# Interpretación del resultado
if p_value < 0.05:
    print(f'Hay una correlación significativa de Spearman de {corr_spearman:.2f}.')
else:
    print('No se encontró una correlación significativa.')
```

5.1.15. Matriz de correlación de Spearman de las variables medias

```
df = pd.read_csv(r'D:\Documents\Documents\academico\tesis_maie\data_sets\df\df.csv')
df = df.drop(columns = ['Lote', 'Año',
                        'REND_stde', "REND_range",
                        'NDRE_stde', 'NDRE_range',
                       'SLOPE_stde','SLOPE_range',
                       'GNDVI_stde', 'GNDVI_range',
                        'NDVI_stde', 'NDVI_range',
                       'OSAVI_stde', 'OSAVI_range',
                       'RECI_stde', 'RECI_range',
                       'TCARIOSAVI_stde', 'TCARIOSAVI_range',
                       'TCARI_stde', 'TCARI_range', ])
df.columns = df.columns.str.replace('_mean', '')
column_names = df.columns


# Obtener la lista de nombres de las columnas
#column_names = df.columns
num_columns = len(column_names)

# Calcular el número de filas y columnas para la disposición de subplots
num_rows = num_columns
num_cols = num_columns

# Crear una figura grande con subplots
```

```
fig, axes = plt.subplots(num_rows, num_cols, figsize=(20,␣
↪20))

# Eliminar etiquetas y títulos de los subplots
for ax in axes.ravel():
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_xlabel('')
    ax.set_ylabel('')

# Crear scatter plots y calcular correlaciones en los␣
↪subplots
for i in range(num_rows):
    for j in range(num_cols):
        if i == j:
            # No hacer scatter plot en la diagonal principal
            continue
        if i < j:
            # Crear scatter plot en la mitad inferior␣
↪izquierda
            x = df[column_names[j]]
            y = df[column_names[i]]
            axes[i, j].scatter(x, y, alpha=0.6)
        else:
            # Calcular la correlación de Spearman en la␣
↪mitad superior derecha
            x = df[column_names[j]]
            y = df[column_names[i]]
            corr_spearman, _ = spearmanr(x, y)
            # Crear heatmap con gradiente de colores basado␣
↪en la correlación
            cmap = plt.get_cmap('RdYlGn')
            im = axes[i, j].imshow([[corr_spearman]],␣
↪cmap=cmap, vmin=-1, vmax=1)
            # Añadir una barra de color a la derecha
            #cbar = fig.colorbar(im, ax=axes[i, j],␣
↪orientation='vertical')
            #cbar.set_label('Correlación de Spearman')
            axes[i, j].text(0, 0, f'{corr_spearman:.2f}',␣
↪color='black',
                            fontsize=12, ha='center',␣
↪va='center')

# Establecer etiquetas de los ejes para la figura completa
for i, col_name in enumerate(column_names):
    axes[num_rows - 1, i].set_xlabel(col_name)
    axes[i, 0].set_ylabel(col_name)
```

```
# Ajustar el espaciado entre los subplots
plt.tight_layout (rect=[0, 0, 0, 0.95])

# Título del gráfico
#plt.suptitle('Matriz de Correlación de Spearman',␣
 ↪fontsize=16)

# Mostrar la figura
plt.show()
```

6. Anexo VI

6.1. Código Modelos Regresión Lineal Múltiple.

6.1.1. Importar librerías y configuraciones

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.stats import shapiro
from scipy import stats
import seaborn as sns
import statsmodels.formula.api as smf
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.outliers_influence import␣
 ↪variance_inflation_factor
from statsmodels.tools.eval_measures import rmse
from typing import Union
```

6.1.2. RLM - 1 - Índices espectrales Climáticas Topográficas

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde',␣
 ↪'Año'])
X = df.select_dtypes(include=['float64', 'int64'])

X = df.drop(columns = ['REND_mean',
                       'NDRE_stde', 'NDRE_range',
                       'SLOPE_stde','SLOPE_range',
                       'GNDVI_stde', 'GNDVI_range',
                        'NDVI_stde', 'NDVI_range',
                       'OSAVI_stde', 'OSAVI_range',
                       'RECI_stde', 'RECI_range',
                       'TCARIOSAVI_stde', 'TCARIOSAVI_range',
                       'TCARI_stde', 'TCARI_range'])
```

6.1.2.1. Modelado

```
y = df['REND_mean']
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.
 ↪8,random_state=1234,shuffle=True) #.values.reshape(-1,1),
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.OLS(endog=y_train, exog=X_train,)
modelo = modelo.fit()
print(modelo.summary())

#El error (rmse) del test
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)
rmse = mean_squared_error(y_true  = y_test,y_pred  =␣
 ↪predicciones,squared = False)
print(f"El error (rmse) de test es: {rmse:.2f}")

#Test de normalidad de los reiduos -  Shapiro Wilk
prediccion_train= modelo.predict(exog = X_train)
residuos_train= prediccion_train-y_train

shapiro_test = stats.shapiro(residuos_train)

if shapiro_test.pvalue > 0.05:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪son normalmente distribuidos (p > 0.05)")
else:
```

```
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪no son normalmente distribuidos (p <= 0.05)")

print("Estadística de prueba:", shapiro_test.statistic)
print("Valor p:", shapiro_test.pvalue)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              REND_mean   R-squared:                       ␣
 ↪           0.680
Model:                            OLS   Adj. R-squared:                  ␣
 ↪           0.384
Method:                 Least Squares   F-statistic:                     ␣
 ↪           2.301
Date:                Thu, 28 Sep 2023   Prob (F-statistic):              ␣
 ↪          0.0752
Time:                        13:16:39   Log-Likelihood:                  ␣
 ↪         -45.867
No. Observations:                  26   AIC:                             ␣
 ↪           117.7
Df Residuals:                      13   BIC:                             ␣
 ↪           134.1
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
===
                 coef    std err          t      P>|t|   ␣
 ↪   [0.025
0.975]
------------------------------------------------------------------------------
---
const         39.7584     29.792      1.335      0.205   ␣
 ↪  -24.604
104.120
DSM            0.0153      0.023      0.680      0.509   ␣
 ↪   -0.033
0.064
EVAPO          6.1256     13.064      0.469      0.647   ␣
 ↪  -22.097
34.348
PP            -0.0016      0.007     -0.232      0.820   ␣
 ↪   -0.016
0.013
TEMP          -1.1787      0.933     -1.264      0.228   ␣
 ↪   -3.193
0.836
```

```
GNDVI_mean         -67.0999     92.700     -0.724       0.482
 ↪ -267.366
133.166
NDRE_mean          -27.2944     61.324     -0.445       0.664
 ↪ -159.776
105.187
NDVI_mean           43.0146     85.594      0.503       0.624
 ↪ -141.900
227.929
OSAVI_mean          76.9884    103.380      0.745       0.470
 ↪ -146.351
300.328
RECI_mean           -0.6998      4.049     -0.173       0.865
 ↪   -9.448
8.048
SLOPE_mean           1.0406      1.941      0.536       0.601
 ↪   -3.152
5.233
TCARIOSAVI_mean     45.0343     84.360      0.534       0.602
 ↪ -137.215
227.284
TCARI_mean        -321.6798    200.344     -1.606       0.132
 ↪ -754.497
111.138
==============================================================================
Omnibus:                        1.373   Durbin-Watson:
 ↪           1.990
Prob(Omnibus):                  0.503   Jarque-Bera (JB):
 ↪           0.834
Skew:                           0.438   Prob(JB):
 ↪           0.659
Kurtosis:                       2.962   Cond. No.
 ↪        5.38e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the
 ↪errors is correctly
specified.
[2] The condition number is large, 5.38e+05. This might
 ↪indicate that there are
strong multicollinearity or other numerical problems.
El error (rmse) de test es: 1.83
El test de Shapiro-Wilk indica que los residuos son
 ↪normalmente distribuidos (p
> 0.05)
Estadística de prueba: 0.9734340310096741
```

```
Valor p: 0.7134025692939758
```

6.1.2.2. Grafico de los residuos

```
# Importar la librería Seaborn y establecer un esquema de␣
 ↪color azul (celeste)

sns.set_palette("ocean")
# Diagnóstico errores (residuos) de las predicciones de␣
 ↪entrenamiento
#␣
 ↪==============================================================================

prediccion_train = modelo.predict(exog = X_train)
residuos_train   = prediccion_train - y_train
# Gráficos
#␣
 ↪==============================================================================
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(9, 8))

axes[0, 0].scatter(y_train, prediccion_train, edgecolors=(0,␣
 ↪0, 0), alpha=0.4)
axes[0, 0].plot([y_train.min(), y_train.max()], [y_train.
 ↪min(), y_train.max()], 'k--', color='black', lw=2)
axes[0, 0].set_title('Valor predicho vs valor real',␣
 ↪fontsize=10)
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize=7)

# 2 - Gráfico Residuos del Modelo
axes[0, 1].scatter(list(range(len(y_train))),␣
 ↪residuos_train, edgecolors=(0, 0, 0), alpha=0.4)
axes[0, 1].axhline(y=0, linestyle='--', color='black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize=10)
axes[0, 1].set_xlabel('id')
axes[0, 1].set_ylabel('Residuos')
axes[0, 1].tick_params(labelsize=7)

# 3 - Gráfico Distribución Residuos del Modelo
sns.histplot(data=residuos_train, stat="count", kde=True,␣
 ↪line_kws={'linewidth': 1},alpha=0.3, ax=axes[1, 0])
axes[1, 0].set_title('Distribución residuos del modelo',␣
 ↪fontsize=10)
axes[1, 0].set_xlabel("Residuos")
axes[1, 0].tick_params(labelsize=7)
```

```
# 4 - Gráfico Q-Q Residuos del Modelo
sm.qqplot(residuos_train, fit=True, line='45', ax=axes[1,␣
 ↪1], color='black', alpha=0.4, lw=2)
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize=10)
axes[1, 1].tick_params(labelsize=7)

# 5 - Gráfico Residuos del Modelo vs Predicción
axes[2, 0].scatter(prediccion_train, residuos_train,␣
 ↪edgecolors=(0, 0, 0), alpha=0.4)
axes[2, 0].axhline(y=0, linestyle='--', color='black', lw=2)
axes[2, 0].set_title('Residuos del modelo vs predicción',␣
 ↪fontsize=10)
axes[2, 0].set_xlabel('Predicción')
axes[2, 0].set_ylabel('Residuos')
axes[2, 0].tick_params(labelsize=7)

# Eliminar los axes vacíos
fig.delaxes(axes[2, 1])

# Ajustar diseño de la figura y título
fig.tight_layout()
plt.subplots_adjust(top=0.9)
#fig.suptitle('Diagnóstico residuos RLM - MEDIAS + clima ',␣
 ↪fontsize=12)

# Mostrar los gráficos
plt.show()
```

6.1.3. RLM - 2- PCA

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde',␣
 ↪'Año'])
df = df.drop(columns = [
                        'NDRE_stde', 'NDRE_range',
                        'SLOPE_stde','SLOPE_range',
                        'GNDVI_stde', 'GNDVI_range',
                         'NDVI_stde', 'NDVI_range',
                        'OSAVI_stde', 'OSAVI_range',
                        'RECI_stde', 'RECI_range',
                        'TCARIOSAVI_stde', 'TCARIOSAVI_range',
                        'TCARI_stde', 'TCARI_range'])
df = df.astype(float)
#df.info()
```

```
scaler = StandardScaler()
X_std = scaler.fit_transform(df)
n_components = 3  # Número de componentes principales a␣
 ↪conservar
pca = PCA(n_components=n_components)
principal_components = pca.fit_transform(X_std)
principal_df = pd.df(data=principal_components,␣
 ↪columns=[f'PC{i}' for i in range(1, n_components + 1)])
#principal_df.info()

explained_variance_ratio = pca.explained_variance_ratio_
print("Varianza explicada por cada componente principal:",␣
 ↪explained_variance_ratio)
print("Varianza total explicada:",␣
 ↪sum(explained_variance_ratio))
```

```
Varianza explicada por cada componente principal: [0.47614894␣
 ↪0.21919704
0.1113091 ]
Varianza total explicada: 0.8066550785957869
```

```
# Concatenar las componentes principales con tu DataFrame␣
 ↪original
df_with_pca = pd.concat([df['REND_mean'], principal_df],␣
 ↪axis=1) #['REND_mean']
X = df_with_pca.drop(columns = ['REND_mean'])
y = df_with_pca['REND_mean']

X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   PC1     33 non-null     float64
 1   PC2     33 non-null     float64
 2   PC3     33 non-null     float64
dtypes: float64(3)
memory usage: 920.0 bytes
```

6.1.3.1. Modelado

```
X_train, X_test, y_train, y_test = train_test_split(X, y,␣
 ↪train_size=0.8, random_state=1234, shuffle=True)
X_train = sm.add_constant(X_train, prepend=True)
modelo = sm.OLS(endog=y_train, exog=X_train,)
```

```
modelo = modelo.fit()
print(modelo.summary())

#El error (rmse) del test
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo.predict(exog = X_test)
rmse = mean_squared_error(y_true  = y_test,y_pred  =␣
 ↪predicciones,squared = False)
print(f"El error (rmse) de test es: {rmse:.2f}")

#Test de normalidad de los reiduos -  Shapiro Wilk
prediccion_train = modelo.predict(exog = X_train)
residuos_train   = prediccion_train - y_train

shapiro_test = stats.shapiro(residuos_train)

if shapiro_test.pvalue > 0.05:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪son normalmente distribuidos (p > 0.05)")
else:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪no son normalmente distribuidos (p <= 0.05)")

print("Estadística de prueba:", shapiro_test.statistic)
print("Valor p:", shapiro_test.pvalue)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              REND_mean   R-squared:                       ␣
 ↪           0.639
Model:                            OLS   Adj. R-squared:                  ␣
 ↪           0.590
Method:                 Least Squares   F-statistic:                     ␣
 ↪           13.00
Date:                Thu, 28 Sep 2023   Prob (F-statistic):              ␣
 ↪        4.24e-05
Time:                        13:16:41   Log-Likelihood:                  ␣
 ↪         -47.416
No. Observations:                  26   AIC:                             ␣
 ↪           102.8
Df Residuals:                      22   BIC:                             ␣
 ↪           107.9
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      ␣
 ↪[0.025      0.975]
```

```
------------------------------------------------------------------------------
const          9.3159      0.323     28.833      0.000       ␣
 ↪8.646       9.986
PC1           -0.2965      0.133     -2.222      0.037       ␣
 ↪-0.573      -0.020
PC2           -1.1620      0.211     -5.510      0.000       ␣
 ↪-1.599      -0.725
PC3           -0.1535      0.260     -0.591      0.561       ␣
 ↪-0.692       0.385
==============================================================================
Omnibus:                        7.755   Durbin-Watson:              ␣
 ↪           1.707
Prob(Omnibus):                  0.021   Jarque-Bera (JB):           ␣
 ↪           5.782
Skew:                           0.930   Prob(JB):                   ␣
 ↪          0.0555
Kurtosis:                       4.370   Cond. No.                   ␣
 ↪           2.50
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the␣
 ↪errors is correctly
specified.
El error (rmse) de test es: 1.80
El test de Shapiro-Wilk indica que los residuos son␣
 ↪normalmente distribuidos (p
> 0.05)
Estadística de prueba: 0.9362507462501526
Valor p: 0.10926259309053421
```

6.1.3.2. Grafico de los residuos

```
# Importar la librería Seaborn y establecer un esquema de␣
 ↪color azul (celeste)

sns.set_palette("ocean")

# Gráficos
#␣
 ↪==============================================================================
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(9, 8))

axes[0, 0].scatter(y_train, prediccion_train, edgecolors=(0,␣
 ↪0, 0), alpha=0.4)
axes[0, 0].plot([y_train.min(), y_train.max()], [y_train.
 ↪min(), y_train.max()], 'k--', color='black', lw=2)
```

```
axes[0, 0].set_title('Valor predicho vs valor real',␣
 ↪fontsize=10)
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize=7)

# 2 - Gráfico Residuos del Modelo
axes[0, 1].scatter(list(range(len(y_train))),␣
 ↪residuos_train, edgecolors=(0, 0, 0), alpha=0.4)
axes[0, 1].axhline(y=0, linestyle='--', color='black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize=10)
axes[0, 1].set_xlabel('id')
axes[0, 1].set_ylabel('Residuos')
axes[0, 1].tick_params(labelsize=7)

# 3 - Gráfico Distribución Residuos del Modelo
sns.histplot(data=residuos_train, stat="count", kde=True,␣
 ↪line_kws={'linewidth': 1},alpha=0.3, ax=axes[1, 0])
axes[1, 0].set_title('Distribución residuos del modelo',␣
 ↪fontsize=10)
axes[1, 0].set_xlabel("Residuos")
axes[1, 0].tick_params(labelsize=7)

# 4 - Gráfico Q-Q Residuos del Modelo
sm.qqplot(residuos_train, fit=True, line='45', ax=axes[1,␣
 ↪1], color='black', alpha=0.4, lw=2)
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize=10)
axes[1, 1].tick_params(labelsize=7)

# 5 - Gráfico Residuos del Modelo vs Predicción
axes[2, 0].scatter(prediccion_train, residuos_train,␣
 ↪edgecolors=(0, 0, 0), alpha=0.4)
axes[2, 0].axhline(y=0, linestyle='--', color='black', lw=2)
axes[2, 0].set_title('Residuos del modelo vs predicción',␣
 ↪fontsize=10)
axes[2, 0].set_xlabel('Predicción')
axes[2, 0].set_ylabel('Residuos')
axes[2, 0].tick_params(labelsize=7)

# Eliminar los axes vacíos
fig.delaxes(axes[2, 1])

# Ajustar diseño de la figura y título
fig.tight_layout()
plt.subplots_adjust(top=0.9)
#fig.suptitle('Diagnóstico residuos PCA(medias + clima +␣
 ↪topo)', fontsize=12)
```

```
# Mostrar los gráficos
plt.show()
```

6.1.4. RLM - 3 - variables seleccionadas por stepwise (backward)

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde',␣
 ↪'Año'])
df = df.astype(float)
```

```
# Concatenar las componentes principales con tu DataFrame␣
 ↪original
X = df.drop(columns = ['REND_mean',
                       'NDRE_stde', 'NDRE_range',
                       'SLOPE_stde','SLOPE_range',
                       'GNDVI_stde', 'GNDVI_range',
                        'NDVI_stde', 'NDVI_range',
                       'OSAVI_stde', 'OSAVI_range',
                       'RECI_stde', 'RECI_range',
                       'TCARIOSAVI_stde', 'TCARIOSAVI_range',
                       'TCARI_stde', 'TCARI_range'])

y = df['REND_mean']
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   DSM              33 non-null     float64
 1   EVAPO            33 non-null     float64
 2   PP               33 non-null     float64
 3   TEMP             33 non-null     float64
 4   GNDVI_mean       33 non-null     float64
 5   NDRE_mean        33 non-null     float64
 6   NDVI_mean        33 non-null     float64
 7   OSAVI_mean       33 non-null     float64
 8   RECI_mean        33 non-null     float64
 9   SLOPE_mean       33 non-null     float64
 10  TCARIOSAVI_mean  33 non-null     float64
 11  TCARI_mean       33 non-null     float64
dtypes: float64(12)
memory usage: 3.2 KB
```

```
y = df['REND_mean']
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.
 ↪8,random_state=1234,shuffle=True) #.values.reshape(-1,1),
X_train = sm.add_constant(X_train, prepend=True)
```

```
# Funcion de selecciçon backward para modelos lineales de␣
 ↪statsmodels
#␣
 ↪=============================================================================
def backward_selection(
    X: pd.df,
    y: pd.Series,
    criterio: str='aic',
    add_constant: bool=True,
    verbose: bool=True
)-> list:

    """
    Realiza un procedimiento de selección de variables hacia␣
 ↪atrás (backward)
    utilizando como criterio de bondad la métrica␣
 ↪especificada. El procedimiento
    se detiene cuando no es posible mejorar más el modelo␣
 ↪eliminando variables.

    Parameters
    ----------
    X: pd.DataFrame
        Matriz de predictores
    y: pd.Series
        Variable respuesta
    metrica: str, default='aic'
        Métrica utilizada para seleccionar las variables.␣
 ↪Debe ser una de las
        siguientes opciones: 'aic', 'bic', 'rsquared_adj'.
    add_constant: bool, default=True
        Si `True` añade una columna de 1s a la matriz de␣
 ↪predictores con el
        con el nombre de intercept.
    verbose: bool, default=True
        Si `True` muestra por pantalla los resultados de cada␣
 ↪iteración.

    Returns
    -------
    seleccion: list
        Lista con las variables seleccionadas.
```

```
    """

    if add_constant:
        X = sm.add_constant(X, prepend=True).
↪rename(columns={'const':'intercept'})

    # Se inicia con todas las variables como predictores
    seleccion = X.columns.to_list()
    modelo  = sm.OLS(endog=y, exog=X[seleccion])
    modelo_res = modelo.fit()
    ultima_metrica = getattr(modelo_res, criterio)
    mejor_metrica = ultima_metrica
    if verbose:
            print(f'variables: {seleccion} | {criterio}:␣
↪{mejor_metrica:.3f}')

    while seleccion:
        metricas = []
        for candidata in seleccion:
            seleccion_temp = seleccion.copy()
            seleccion_temp.remove(candidata)
            modelo  = sm.OLS(endog=y, exog=X[seleccion_temp])
            modelo_res = modelo.fit()
            metrica = getattr(modelo_res, criterio)
            metricas.append(metrica)
        if criterio == 'rsquared_adj':
            mejor_metrica = max(metricas)
            if mejor_metrica > ultima_metrica:
                peor_variable = seleccion[np.
↪argmax(metricas)]
            else:
                break
        else:
            mejor_metrica = min(metricas)
            if mejor_metrica < ultima_metrica:
                peor_variable = seleccion[np.
↪argmin(metricas)]
            else:
                break

        seleccion.remove(peor_variable)
        ultima_metrica = mejor_metrica

        if verbose:
            print(f'variables: {seleccion} | {criterio}:␣
↪{mejor_metrica:.3f}')
```

```
    return sorted(seleccion)
```

```
# Selección de variables hacia backward
#␣
 ↪==========================================================================
predictores= backward_selection(
    X            = X_train,
    y            = y_train,
    criterio     = 'aic',
    add_constant = False, # Ya se le añadió anteriormente
    verbose      = True
)
predictores
```

```
variables: ['const', 'DSM', 'EVAPO', 'PP', 'TEMP',␣
 ↪'GNDVI_mean', 'NDRE_mean',
'NDVI_mean', 'OSAVI_mean', 'RECI_mean', 'SLOPE_mean',␣
 ↪'TCARIOSAVI_mean',
'TCARI_mean'] | aic: 117.734
variables: ['const', 'DSM', 'EVAPO', 'PP', 'TEMP',␣
 ↪'GNDVI_mean', 'NDRE_mean',
'NDVI_mean', 'OSAVI_mean', 'SLOPE_mean', 'TCARIOSAVI_mean',␣
 ↪'TCARI_mean'] | aic:
115.794
variables: ['const', 'DSM', 'EVAPO', 'TEMP', 'GNDVI_mean',␣
 ↪'NDRE_mean',
'NDVI_mean', 'OSAVI_mean', 'SLOPE_mean', 'TCARIOSAVI_mean',␣
 ↪'TCARI_mean'] | aic:
113.931
variables: ['const', 'DSM', 'EVAPO', 'TEMP', 'GNDVI_mean',␣
 ↪'NDRE_mean',
'NDVI_mean', 'OSAVI_mean', 'SLOPE_mean', 'TCARI_mean'] | aic:␣
 ↪112.460
variables: ['const', 'DSM', 'EVAPO', 'TEMP', 'GNDVI_mean',␣
 ↪'NDRE_mean',
'NDVI_mean', 'OSAVI_mean', 'TCARI_mean'] | aic: 110.835
variables: ['const', 'DSM', 'EVAPO', 'TEMP', 'GNDVI_mean',␣
 ↪'NDRE_mean',
'NDVI_mean', 'TCARI_mean'] | aic: 109.207
variables: ['const', 'DSM', 'EVAPO', 'TEMP', 'NDRE_mean',␣
 ↪'NDVI_mean',
'TCARI_mean'] | aic: 107.331
variables: ['const', 'DSM', 'EVAPO', 'TEMP', 'NDVI_mean',␣
 ↪'TCARI_mean'] | aic:
106.016
variables: ['const', 'EVAPO', 'TEMP', 'NDVI_mean',␣
 ↪'TCARI_mean'] | aic: 104.743
```

```
variables: ['const', 'TEMP', 'NDVI_mean', 'TCARI_mean'] | aic:
 ↪ 103.163

['NDVI_mean', 'TCARI_mean', 'TEMP', 'const']
```

```
# Entrenamiento del modelo con las variables seleccionadas
#␣
 ↪==========================================================================
modelo_final  = sm.OLS(endog=y_train,␣
 ↪exog=X_train[predictores])
modelo_final_res = modelo_final.fit()
print(modelo_final_res.summary())


#El error (rmse) del test
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo_final_res.predict(exog =␣
 ↪X_test[predictores])
rmse = mean_squared_error(y_true  = y_test,y_pred  =␣
 ↪predicciones,squared = False)
print(f"El error (rmse) de test es: {rmse:.2f}")

#Test de normalidad de los reiduos -  Shapiro Wilk
prediccion_train = modelo_final_res.
 ↪predict(X_train[predictores])

residuos_train= prediccion_train-y_train

shapiro_test = stats.shapiro(residuos_train)

if shapiro_test.pvalue > 0.05:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪son normalmente distribuidos (p > 0.05)")
else:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪no son normalmente distribuidos (p <= 0.05)")

print("Estadística de prueba:", shapiro_test.statistic)
print("Valor p:", shapiro_test.pvalue)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                REND_mean   R-squared:                       ␣
 ↪           0.635
Model:                            OLS   Adj. R-squared:                  ␣
 ↪           0.585
Method:                 Least Squares   F-statistic:                     ␣
 ↪           12.75
```

```
Date:                Tue, 10 Oct 2023   Prob (F-statistic):
 ↪        4.87e-05
Time:                        17:28:57   Log-Likelihood:
 ↪         -47.581
No. Observations:                  26   AIC:
 ↪           103.2
Df Residuals:                      22   BIC:
 ↪           108.2
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|
 ↪[0.025      0.975]
------------------------------------------------------------------------------
NDVI_mean     17.8847      3.599      4.970      0.000
 ↪10.422      25.348
TCARI_mean  -120.7217     28.371     -4.255      0.000
 ↪-179.559     -61.884
TEMP          -0.8089      0.326     -2.484      0.021
 ↪-1.484      -0.134
const         26.2496      7.942      3.305      0.003
 ↪9.779      42.720
==============================================================================
Omnibus:                        2.932   Durbin-Watson:
 ↪           1.972
Prob(Omnibus):                  0.231   Jarque-Bera (JB):
 ↪           1.818
Skew:                           0.639   Prob(JB):
 ↪           0.403
Kurtosis:                       3.209   Cond. No.
 ↪        2.05e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the
 ↪errors is correctly
specified.
[2] The condition number is large, 2.05e+03. This might
 ↪indicate that there are
strong multicollinearity or other numerical problems.
El error (rmse) de test es: 1.79
El test de Shapiro-Wilk indica que los residuos son
 ↪normalmente distribuidos (p
> 0.05)
Estadística de prueba: 0.952816903591156
Valor p: 0.269680440425872
```

Inspección visual

```
# Importar la librería Seaborn y establecer un esquema de␣
 ↪color azul (celeste)

sns.set_palette("ocean")

# Gráficos
#␣
 ↪==========================================================================
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(9, 8))

axes[0, 0].scatter(y_train, prediccion_train, edgecolors=(0,␣
 ↪0, 0), alpha=0.4)
axes[0, 0].plot([y_train.min(), y_train.max()], [y_train.
 ↪min(), y_train.max()], 'k--', color='black', lw=2)
axes[0, 0].set_title('Valor predicho vs valor real',␣
 ↪fontsize=10)
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize=7)

# 2 - Gráfico Residuos del Modelo
axes[0, 1].scatter(list(range(len(y_train))),␣
 ↪residuos_train, edgecolors=(0, 0, 0), alpha=0.4)
axes[0, 1].axhline(y=0, linestyle='--', color='black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize=10)
axes[0, 1].set_xlabel('id')
axes[0, 1].set_ylabel('Residuos')
axes[0, 1].tick_params(labelsize=7)

# 3 - Gráfico Distribución Residuos del Modelo
sns.histplot(data=residuos_train, stat="count", kde=True,␣
 ↪line_kws={'linewidth': 1},alpha=0.3, ax=axes[1, 0])
axes[1, 0].set_title('Distribución residuos del modelo',␣
 ↪fontsize=10)
axes[1, 0].set_xlabel("Residuos")
axes[1, 0].tick_params(labelsize=7)

# 4 - Gráfico Q-Q Residuos del Modelo
sm.qqplot(residuos_train, fit=True, line='45', ax=axes[1,␣
 ↪1], color='black', alpha=0.4, lw=2)
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize=10)
axes[1, 1].tick_params(labelsize=7)

# 5 - Gráfico Residuos del Modelo vs Predicción
axes[2, 0].scatter(prediccion_train, residuos_train,␣
 ↪edgecolors=(0, 0, 0), alpha=0.4)
```

```
axes[2, 0].axhline(y=0, linestyle='--', color='black', lw=2)
axes[2, 0].set_title('Residuos del modelo vs predicción',␣
 ↪fontsize=10)
axes[2, 0].set_xlabel('Predicción')
axes[2, 0].set_ylabel('Residuos')
axes[2, 0].tick_params(labelsize=7)

# Eliminar los axes vacíos
fig.delaxes(axes[2, 1])

# Ajustar diseño de la figura y título
fig.tight_layout()
plt.subplots_adjust(top=0.9)
#fig.suptitle('Diagnóstico residuos RLM MEDIAS + CLIMA +␣
 ↪INTERACCIONES', fontsize=12)

# Mostrar los gráficos
plt.show()
```

6.1.5. RLM - 4 - variables seleccionadas por stepwise (forward)

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde',␣
 ↪'Año'])
df = df.astype(float)
```

```
# Concatenar las componentes principales con tu DataFrame␣
 ↪original
X = df.drop(columns = ['REND_mean',
                       'NDRE_stde', 'NDRE_range',
                       'SLOPE_stde','SLOPE_range',
                       'GNDVI_stde', 'GNDVI_range',
                        'NDVI_stde', 'NDVI_range',
                       'OSAVI_stde', 'OSAVI_range',
                       'RECI_stde', 'RECI_range',
                       'TCARIOSAVI_stde', 'TCARIOSAVI_range',
                       'TCARI_stde', 'TCARI_range'])

y = df['REND_mean']
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 12 columns):
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   DSM               33 non-null     float64
 1   EVAPO             33 non-null     float64
 2   PP                33 non-null     float64
 3   TEMP              33 non-null     float64
 4   GNDVI_mean        33 non-null     float64
 5   NDRE_mean         33 non-null     float64
 6   NDVI_mean         33 non-null     float64
 7   OSAVI_mean        33 non-null     float64
 8   RECI_mean         33 non-null     float64
 9   SLOPE_mean        33 non-null     float64
 10  TCARIOSAVI_mean   33 non-null     float64
 11  TCARI_mean        33 non-null     float64
dtypes: float64(12)
memory usage: 3.2 KB
```

```
y = df['REND_mean']
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.
 ↪8,random_state=1234,shuffle=True) #.values.reshape(-1,1),
X_train = sm.add_constant(X_train, prepend=True)
```

```
# Funcion de selecciçon backward para modelos lineales de␣
 ↪statsmodels
#␣
 ↪=============================================================================

def forward_selection(
    X: pd.df,
    y: pd.Series,
    criterio: str='aic',
    add_constant: bool=True,
    verbose: bool=True
)-> list:

    """
    Realiza un procedimiento de selección de variables hacia␣
 ↪adelante (forward)
    utilizando como criterio de bondad la métrica␣
 ↪especificada. El procedimiento
    se detiene cuando no es posible mejorar más el modelo␣
 ↪añadiendo variables.

    Parameters
    ----------
    X: pd.DataFrame
        Matriz de predictores
```

```
    y: pd.Series
        Variable respuesta
    metrica: str, default='aic'
        Métrica utilizada para seleccionar las variables.␣
↪Debe ser una de las
        siguientes opciones: 'aic', 'bic', 'rsquared_adj'.
    add_constant: bool, default=True
        Si `True` añade una columna de 1s a la matriz de␣
↪predictores con el
        con el nombre de intercept.
    verbose: bool, default=True
        Si `True` muestra por pantalla los resultados de cada␣
↪iteración.

    Returns
    -------
    seleccion: list
        Lista con las variables seleccionadas.
    """

    if add_constant:
        X = sm.add_constant(X, prepend=True).
↪rename(columns={'const':'intercept'})

    restantes = X.columns.to_list()
    seleccion = []
    if criterio == 'rsquared_adj':
        mejor_metrica = -np.inf
        ultima_metrica = -np.inf
    else:
        mejor_metrica = np.inf
        ultima_metrica = np.inf

    while restantes:
        metricas = []
        for candidata in restantes:
            seleccion_temp = seleccion + [candidata]
            modelo  = sm.OLS(endog=y, exog=X[seleccion_temp])
            modelo_res = modelo.fit()
            metrica = getattr(modelo_res, criterio)
            metricas.append(metrica)
        if criterio == 'rsquared_adj':
            mejor_metrica = max(metricas)
            if mejor_metrica > ultima_metrica:
                mejor_variable = restantes[np.
↪argmax(metricas)]
            else:
```

```
                break
        else:
            mejor_metrica = min(metricas)
            if mejor_metrica < ultima_metrica:
                mejor_variable = restantes[np.
 ↪argmin(metricas)]
            else:
                break

        seleccion.append(mejor_variable)
        restantes.remove(mejor_variable)
        ultima_metrica = mejor_metrica

        if verbose:
            print(f'variables: {seleccion} | {criterio}:␣
 ↪{mejor_metrica:.3f}')

    return sorted(seleccion)
```

```
# Selección de variables hacia forward
#␣
 ↪==============================================================================
predictores = forward_selection(
    X            = X_train,
    y            = y_train,
    criterio     = 'aic',
    add_constant = False, # Ya se le añadió anteriormente
    verbose      = True
)
predictores
```

```
variables: ['GNDVI_mean'] | aic: 119.308
variables: ['GNDVI_mean', 'TCARI_mean'] | aic: 110.848
variables: ['GNDVI_mean', 'TCARI_mean', 'EVAPO'] | aic: 109.
 ↪814

['EVAPO', 'GNDVI_mean', 'TCARI_mean']
```

```
# Entrenamiento del modelo con las variables seleccionadas
#␣
 ↪==============================================================================
modelo_final  = sm.OLS(endog=y_train,␣
 ↪exog=X_train[predictores])
modelo_final_res = modelo_final.fit()
print(modelo_final_res.summary())
```

```
#El error (rmse) del test
X_test = sm.add_constant(X_test, prepend=True)
predicciones = modelo_final_res.predict(exog =␣
 ↪X_test[predictores])
rmse = mean_squared_error(y_true  = y_test,y_pred  =␣
 ↪predicciones,squared = False)
print(f"El error (rmse) de test es: {rmse:.2f}")

#Test de normalidad de los reiduos -  Shapiro Wilk
prediccion_train = modelo_final_res.
 ↪predict(X_train[predictores])

residuos_train= prediccion_train-y_train

shapiro_test = stats.shapiro(residuos_train)

if shapiro_test.pvalue > 0.05:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪son normalmente distribuidos (p > 0.05)")
else:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪no son normalmente distribuidos (p <= 0.05)")

print("Estadística de prueba:", shapiro_test.statistic)
print("Valor p:", shapiro_test.pvalue)
```

```
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:                REND_mean   R-squared␣
 ↪(uncentered):
0.967
Model:                            OLS   Adj. R-squared␣
 ↪(uncentered):
0.963
Method:                 Least Squares   F-statistic:
225.8
Date:                Tue, 10 Oct 2023   Prob (F-statistic):
3.38e-17
Time:                        17:29:35   Log-Likelihood:
-51.907
No. Observations:                  26   AIC:
109.8
Df Residuals:                      23   BIC:
113.6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
```

```
                 coef    std err          t      P>|t|      ␣
 ↪[0.025      0.975]
------------------------------------------------------------------------------
EVAPO          -5.3180      3.152     -1.687      0.105     ␣
 ↪-11.838       1.202
GNDVI_mean     28.0408      5.407      5.186      0.000      ␣
 ↪16.855       39.226
TCARI_mean    -92.8194     32.447     -2.861      0.009     ␣
 ↪-159.940     -25.698
==============================================================================
Omnibus:                        4.133   Durbin-Watson:                   ␣
 ↪           2.027
Prob(Omnibus):                  0.127   Jarque-Bera (JB):                ␣
 ↪           2.472
Skew:                           0.477   Prob(JB):                        ␣
 ↪           0.291
Kurtosis:                       4.172   Cond. No.                        ␣
 ↪            53.9
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the␣
 ↪model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the␣
 ↪errors is correctly
specified.
El error (rmse) de test es: 2.24
El test de Shapiro-Wilk indica que los residuos son␣
 ↪normalmente distribuidos (p
> 0.05)
Estadística de prueba: 0.9462195038795471
Valor p: 0.1888933777809143
```

Inspección visual

```
# Importar la librería Seaborn y establecer un esquema de␣
 ↪color azul (celeste)

sns.set_palette("ocean")

# Gráficos
#␣
 ↪==============================================================
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(9, 8))
```

```
axes[0, 0].scatter(y_train, prediccion_train, edgecolors=(0,␣
 ↪0, 0), alpha=0.4)
axes[0, 0].plot([y_train.min(), y_train.max()], [y_train.
 ↪min(), y_train.max()], 'k--', color='black', lw=2)
axes[0, 0].set_title('Valor predicho vs valor real',␣
 ↪fontsize=10)
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize=7)

# 2 - Gráfico Residuos del Modelo
axes[0, 1].scatter(list(range(len(y_train))),␣
 ↪residuos_train, edgecolors=(0, 0, 0), alpha=0.4)
axes[0, 1].axhline(y=0, linestyle='--', color='black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize=10)
axes[0, 1].set_xlabel('id')
axes[0, 1].set_ylabel('Residuos')
axes[0, 1].tick_params(labelsize=7)

# 3 - Gráfico Distribución Residuos del Modelo
sns.histplot(data=residuos_train, stat="count", kde=True,␣
 ↪line_kws={'linewidth': 1},alpha=0.3, ax=axes[1, 0])
axes[1, 0].set_title('Distribución residuos del modelo',␣
 ↪fontsize=10)
axes[1, 0].set_xlabel("Residuos")
axes[1, 0].tick_params(labelsize=7)

# 4 - Gráfico Q-Q Residuos del Modelo
sm.qqplot(residuos_train, fit=True, line='45', ax=axes[1,␣
 ↪1], color='black', alpha=0.4, lw=2)
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize=10)
axes[1, 1].tick_params(labelsize=7)

# 5 - Gráfico Residuos del Modelo vs Predicción
axes[2, 0].scatter(prediccion_train, residuos_train,␣
 ↪edgecolors=(0, 0, 0), alpha=0.4)
axes[2, 0].axhline(y=0, linestyle='--', color='black', lw=2)
axes[2, 0].set_title('Residuos del modelo vs predicción',␣
 ↪fontsize=10)
axes[2, 0].set_xlabel('Predicción')
axes[2, 0].set_ylabel('Residuos')
axes[2, 0].tick_params(labelsize=7)

# Eliminar los axes vacíos
fig.delaxes(axes[2, 1])

# Ajustar diseño de la figura y título
```

```
fig.tight_layout()
plt.subplots_adjust(top=0.9)
#fig.suptitle('Diagnóstico residuos RLM MEDIAS + CLIMA + INTERACCIONES', fontsize=12)

# Mostrar los gráficos
plt.show()
```

6.1.6. RLM - 5 - Interacciones Stepwise (forward)

```
df = pd.read_csv(r'D:\Documents\Documents\academico\tesis_maie\data_sets\df\df.csv')
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde', 'Año'])

X = df.drop(columns = ['REND_mean',

                       'NDRE_stde', 'NDRE_range',
                       'SLOPE_stde','SLOPE_range',
                       'GNDVI_stde', 'GNDVI_range',
                        'NDVI_stde', 'NDVI_range',
                       'OSAVI_stde', 'OSAVI_range',
                       'RECI_stde', 'RECI_range',
                       'TCARIOSAVI_stde', 'TCARIOSAVI_range',
                       'TCARI_stde', 'TCARI_range'])



y = df['REND_mean']
```

6.1.6.1. Modelado

```
# Divide los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=1234, shuffle=True)

# Se añade una nueva columna con la interacción
#['EVAPO', 'GNDVI_mean', 'TCARI_mean']

X_train['EVAPO_GNDVI_mean'] = X_train['EVAPO'] * X_train['GNDVI_mean']
X_test['EVAPO_GNDVI_mean'] = X_test['EVAPO'] * X_test['GNDVI_mean']
```

```
X_train['EVAPO_TCARI_mean']  = X_train['EVAPO'] *␣
 ↪X_train['TCARI_mean']
X_test['EVAPO_TCARI_mean'] = X_test['EVAPO'] *␣
 ↪X_test['TCARI_mean']

X_train['TCARI_GNDVI_mean'] = X_train['TCARI_mean'] *␣
 ↪X_train['GNDVI_mean']
X_test['TCARI_GNDVI_mean'] = X_test['TCARI_mean'] *␣
 ↪X_test['GNDVI_mean']

#X_train['OSAVI_mean_GNDVI_mean']  = X_train['OSAVI_mean'] *␣
 ↪X_train['GNDVI_mean']
#X_test['OSAVI_mean_GNDVI_mean'] = X_test['OSAVI_mean'] *␣
 ↪X_test['GNDVI_mean']

#X_train['TCARI_mean_NDRE_mean'] = X_train['TCARI_mean'] *␣
 ↪X_train['NDRE_mean']
#X_test['TCARI_mean_NDRE_mean'] = X_test['TCARI_mean'] *␣
 ↪X_test['NDRE_mean']


# Agrega una columna de 1s para el intercepto del modelo
X_train = sm.add_constant(X_train, prepend=True)
X_test = sm.add_constant(X_test, prepend=True)

# Ajusta el modelo de regresión lineal
modelo_interaccion = sm.OLS(endog=y_train, exog=X_train)
modelo_interaccion = modelo_interaccion.fit()
print(modelo_interaccion.summary())

# Calcula el error RMSE en el conjunto de prueba
predicciones = modelo_interaccion.predict(exog=X_test)
rmse = mean_squared_error(y_true=y_test,␣
 ↪y_pred=predicciones, squared=False)
print(f"El error (RMSE) en el conjunto de prueba es: {rmse:.
 ↪2f}")

# Prueba de normalidad de los residuos - Shapiro-Wilk
residuos_train = modelo_interaccion.resid
shapiro_test = stats.shapiro(residuos_train)

if shapiro_test.pvalue > 0.05:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪son normalmente distribuidos (p > 0.05)")
else:
    print("El test de Shapiro-Wilk indica que los residuos␣
 ↪no son normalmente distribuidos (p <= 0.05)")
```

```
print("Estadística de prueba:", shapiro_test.statistic)
print("Valor p:", shapiro_test.pvalue)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              REND_mean   R-squared:                       ␣
 ↪           0.729
Model:                            OLS   Adj. R-squared:                  ␣
 ↪           0.323
Method:                 Least Squares   F-statistic:                     ␣
 ↪           1.796
Date:                Tue, 10 Oct 2023   Prob (F-statistic):              ␣
 ↪           0.176
Time:                        17:36:58   Log-Likelihood:                  ␣
 ↪         -43.687
No. Observations:                  26   AIC:                             ␣
 ↪           119.4
Df Residuals:                      10   BIC:                             ␣
 ↪           139.5
Df Model:                          15
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      ␣
 ↪    [0.025
0.975]
------------------------------------------------------------------------------
----
const         36.0407     38.983      0.925      0.377      ␣
 ↪   -50.819
122.901
DSM            0.0224      0.026      0.876      0.402      ␣
 ↪    -0.035
0.080
EVAPO        -57.3398     79.823     -0.718      0.489      ␣
 ↪  -235.197
120.518
PP             0.0011      0.007      0.147      0.886      ␣
 ↪    -0.016
0.018
TEMP          -1.7854      1.284     -1.391      0.194      ␣
 ↪    -4.645
1.074
GNDVI_mean   -87.3950    128.331     -0.681      0.511      ␣
 ↪  -373.334
198.544
```

```
NDRE_mean           -20.0867     78.710     -0.255       0.804 ␣
 ↪  -195.463
155.289
NDVI_mean            48.6019     90.002      0.540       0.601 ␣
 ↪  -151.934
249.138
OSAVI_mean          129.7035    123.475      1.050       0.318 ␣
 ↪  -145.417
404.824
RECI_mean            -2.5376      4.573     -0.555       0.591 ␣
 ↪   -12.727
7.652
SLOPE_mean            0.7014      2.096      0.335       0.745 ␣
 ↪    -3.969
5.372
TCARIOSAVI_mean      81.3906     95.933      0.848       0.416 ␣
 ↪  -132.361
295.142
TCARI_mean         -532.3520    403.084     -1.321       0.216 ␣
 ↪ -1430.480
365.776
EVAPO_GNDVI_mean    118.9189    139.097      0.855       0.413 ␣
 ↪  -191.009
428.847
EVAPO_TCARI_mean     96.4864    457.843      0.211       0.837 ␣
 ↪  -923.651
1116.624
TCARI_GNDVI_mean    214.2741    501.213      0.428       0.678 ␣
 ↪  -902.497
1331.046
==============================================================================
Omnibus:                        3.205   Durbin-Watson:                   ␣
 ↪           1.791
Prob(Omnibus):                  0.201   Jarque-Bera (JB):                ␣
 ↪           1.817
Skew:                           0.607   Prob(JB):                        ␣
 ↪           0.403
Kurtosis:                       3.449   Cond. No.                        ␣
 ↪        1.50e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the␣
 ↪errors is correctly
specified.
[2] The condition number is large, 1.5e+06. This might␣
 ↪indicate that there are
```

```
strong multicollinearity or other numerical problems.
El error (RMSE) en el conjunto de prueba es: 1.48
El test de Shapiro-Wilk indica que los residuos son␣
 ↪normalmente distribuidos (p
> 0.05)
Estadística de prueba: 0.9567442536354065
Valor p: 0.33153319358825684
```

6.1.6.2. Grafico de los residuos

```
# Importar la librería Seaborn y establecer un esquema de␣
 ↪color azul (celeste)

sns.set_palette("ocean")

# Gráficos
#␣
 ↪==========================================================================
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(9, 8))

axes[0, 0].scatter(y_train, prediccion_train, edgecolors=(0,␣
 ↪0, 0), alpha=0.4)
axes[0, 0].plot([y_train.min(), y_train.max()], [y_train.
 ↪min(), y_train.max()], 'k--', color='black', lw=2)
axes[0, 0].set_title('Valor predicho vs valor real',␣
 ↪fontsize=10)
axes[0, 0].set_xlabel('Real')
axes[0, 0].set_ylabel('Predicción')
axes[0, 0].tick_params(labelsize=7)

# 2 - Gráfico Residuos del Modelo
axes[0, 1].scatter(list(range(len(y_train))),␣
 ↪residuos_train, edgecolors=(0, 0, 0), alpha=0.4)
axes[0, 1].axhline(y=0, linestyle='--', color='black', lw=2)
axes[0, 1].set_title('Residuos del modelo', fontsize=10)
axes[0, 1].set_xlabel('id')
axes[0, 1].set_ylabel('Residuos')
axes[0, 1].tick_params(labelsize=7)

# 3 - Gráfico Distribución Residuos del Modelo
sns.histplot(data=residuos_train, stat="count", kde=True,␣
 ↪line_kws={'linewidth': 1},alpha=0.3, ax=axes[1, 0])
axes[1, 0].set_title('Distribución residuos del modelo',␣
 ↪fontsize=10)
axes[1, 0].set_xlabel("Residuos")
axes[1, 0].tick_params(labelsize=7)
```

```
# 4 - Gráfico Q-Q Residuos del Modelo
sm.qqplot(residuos_train, fit=True, line='45', ax=axes[1,␣
 ↪1], color='black', alpha=0.4, lw=2)
axes[1, 1].set_title('Q-Q residuos del modelo', fontsize=10)
axes[1, 1].tick_params(labelsize=7)

# 5 - Gráfico Residuos del Modelo vs Predicción
axes[2, 0].scatter(prediccion_train, residuos_train,␣
 ↪edgecolors=(0, 0, 0), alpha=0.4)
axes[2, 0].axhline(y=0, linestyle='--', color='black', lw=2)
axes[2, 0].set_title('Residuos del modelo vs predicción',␣
 ↪fontsize=10)
axes[2, 0].set_xlabel('Predicción')
axes[2, 0].set_ylabel('Residuos')
axes[2, 0].tick_params(labelsize=7)

# Eliminar los axes vacíos
fig.delaxes(axes[2, 1])

# Ajustar diseño de la figura y título
fig.tight_layout()
plt.subplots_adjust(top=0.9)
#fig.suptitle('Diagnóstico residuos RLM MEDIAS + CLIMA +␣
 ↪INTERACCIONES', fontsize=12)

# Mostrar los gráficos
plt.show()
```

7. Anexo VII

7.1. Código Modelo Random Forest Regression.

7.1.1. Importar librerías y configuraciones

```
import os
import re
import glob
import fiona
import joblib
import datetime
import numpy as np
import pandas as pd
import seaborn as sn
import rasterio.mask
```

```
import rasterio as rio
import geopandas as gpd
import matplotlib.pyplot as plt
import multiprocessing

from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.inspection import PartialDependenceDisplay
from sklearn.inspection import partial_dependence

# Tratamiento de datos
#␣
 ↪==============================================================================
import numpy as np
import pandas as pd

# Gráficos
#␣
 ↪==============================================================================
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Preprocesado y modelado
#␣
 ↪==============================================================================
from time import time
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import ParameterGrid
from sklearn.inspection import permutation_importance

# Configuración warnings
#␣
 ↪==============================================================================
#import warnings
#warnings.filterwarnings('once')
```

7.1.2. Importar base de datos

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde',␣
 ↪'Año'])
df = df.astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 29 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   DSM                33 non-null     float64
 1   EVAPO              33 non-null     float64
 2   PP                 33 non-null     float64
 3   TEMP               33 non-null     float64
 4   GNDVI_mean         33 non-null     float64
 5   GNDVI_stde         33 non-null     float64
 6   GNDVI_range        33 non-null     float64
 7   NDRE_mean          33 non-null     float64
 8   NDRE_stde          33 non-null     float64
 9   NDRE_range         33 non-null     float64
 10  NDVI_mean          33 non-null     float64
 11  NDVI_stde          33 non-null     float64
 12  NDVI_range         33 non-null     float64
 13  OSAVI_mean         33 non-null     float64
 14  OSAVI_stde         33 non-null     float64
 15  OSAVI_range        33 non-null     float64
 16  RECI_mean          33 non-null     float64
 17  RECI_stde          33 non-null     float64
 18  RECI_range         33 non-null     float64
 19  SLOPE_mean         33 non-null     float64
 20  SLOPE_stde         33 non-null     float64
 21  SLOPE_range        33 non-null     float64
 22  TCARIOSAVI_mean    33 non-null     float64
 23  TCARIOSAVI_stde    33 non-null     float64
 24  TCARIOSAVI_range   33 non-null     float64
 25  TCARI_mean         33 non-null     float64
 26  TCARI_stde         33 non-null     float64
 27  TCARI_range        33 non-null     float64
 28  REND_mean          33 non-null     float64
dtypes: float64(29)
memory usage: 7.6 KB
```

7.1.3. RFR 1 - Modelo de todas las variables (sin optimizar)

```
### División de los datos en train y test
#␣
 ↪==============================================================================
X_train, X_test, y_train, y_test = train_test_split(
                                        df.drop(columns =␣
 ↪["REND_mean"]),
                                        df['REND_mean'],
                                        random_state = 123
                                    )
# Creación del modelo
#␣
 ↪==============================================================================
modelo = RandomForestRegressor(
            n_estimators = 24,
            criterion    = 'squared_error', #'mse'
            max_depth    = None,
            max_features = 28,
            oob_score    = False,
            n_jobs       = -1,
            random_state = 123
         )

# Entrenamiento del modelo
#␣
 ↪==============================================================================
modelo.fit(X_train, y_train)
```

```
RandomForestRegressor(max_features=28, n_estimators=24,␣
 ↪n_jobs=-1,
                      random_state=123)
```

```
# Error de test del modelo inicial
# ========================================
predicciones = modelo.predict(X = X_test)

rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print(f"El error (rmse) de test es: {rmse}")

# Calcula el MAPE
mape = mean_absolute_percentage_error(
    y_true=y_test,
```

```
    y_pred=predicciones
)

print(f"El MAPE de test es: {mape}")
```

```
El error (rmse) de test es: 2.211849315637717
El MAPE de test es: 0.3030383384155824
```

```
# Gráfico de dispersión de los resultados predichos frente a␣
 ↪las observaciones reales
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predicciones, alpha=0.5)
plt.xlabel("Observaciones reales")
plt.ylabel("Resultados predichos")
plt.title("Resultados Predichos vs Observaciones Reales")

# Línea de referencia (y=x)
x = np.linspace(min(y_test), max(y_test), 100)
plt.plot(x, x, color='red', linestyle='--', linewidth=2,␣
 ↪label="Línea de Referencia (y=x)")

plt.legend()
plt.grid(True)
plt.show()
```

7.1.3.1. Importancia de predictores

7.1.3.2. Importancia por pureza de nodos

```
importancia_df = pd.DataFrame(
                              {'predictor': df.drop(columns =␣
 ↪"REND_mean").columns,
                               'importancia': modelo.
 ↪feature_importances_}
                              )
print("Importancia de los predictores en el modelo")
print("----------------------------------------")
df1 = importancia_df.sort_values('importancia',␣
 ↪ascending=False)
print(df1)
```

```
Importancia de los predictores en el modelo
----------------------------------------
           predictor  importancia
26        TCARI_stde     0.158606
7          NDRE_mean     0.134242
2                 PP     0.111280
```

```
4          GNDVI_mean     0.081966
22   TCARIOSAVI_mean      0.077997
3                TEMP     0.060988
1               EVAPO     0.056898
25         TCARI_mean     0.054269
27        TCARI_range     0.042476
6         GNDVI_range     0.028563
15        OSAVI_range     0.020686
23    TCARIOSAVI_stde     0.018530
11          NDVI_stde     0.016874
8           NDRE_stde     0.015845
14         OSAVI_stde     0.014090
9         NDRE_range     0.012332
24  TCARIOSAVI_range     0.012168
10          NDVI_mean     0.011444
16          RECI_mean     0.011402
0                 DSM     0.011184
21        SLOPE_range     0.010202
20         SLOPE_stde     0.009619
18         RECI_range     0.007356
19         SLOPE_mean     0.007211
13         OSAVI_mean     0.005640
17          RECI_stde     0.005225
12         NDVI_range     0.002661
5          GNDVI_stde     0.000245
```

```python
# Modificación del gráfico para feature_importances_
fig, ax = plt.subplots(figsize=(12.5, 15))
importancia_df = importancia_df.sort_values('importancia',␣
 ↪ascending=True)  # Ordenar de mayor a menor importancia

# Barras horizontales
ax.barh(
    importancia_df['predictor'],
    importancia_df['importancia'],
    align='center',
    alpha=0,
)

# Puntos de importancia
ax.plot(
    importancia_df['importancia'],
    importancia_df['predictor'],
    marker="D",
    linestyle="",
    alpha=0.8,
    color="r"
)
```

```
# Etiquetas en el eje x y el eje y
ax.set_xlabel('Importancia')
ax.set_ylabel('Características')
#ax.set_title('Importancia de las Características')

plt.show()
```

7.1.3.3. Importancia por permutación

```
importancia = permutation_importance(
                estimator    = modelo,
                X            = X_train,
                y            = y_train,
                n_repeats    = 5,
                scoring      = 'neg_root_mean_squared_error',
                n_jobs       = multiprocessing.cpu_count() -␣
 ↪1,
                random_state = 123
             )

# Se almacenan los resultados (media y desviación) en un␣
 ↪dataframe
df_importancia = pd.DataFrame(
                    {k: importancia[k] for k in␣
 ↪['importances_mean', 'importances_std']}
                 )
df_importancia['feature'] = X_train.columns
df1 = df_importancia.sort_values('importances_mean',␣
 ↪ascending=False)
#df_importancia.sort_values('importancia', ascending=False)
print(df1)
```

	importances_mean	importances_std	feature
26	0.525354	0.109272	TCARI_stde
7	0.295387	0.035778	NDRE_mean
22	0.156412	0.066755	TCARIOSAVI_mean
2	0.136987	0.102862	PP
4	0.128203	0.036484	GNDVI_mean
25	0.089772	0.063268	TCARI_mean
3	0.081067	0.021508	TEMP
1	0.069825	0.044942	EVAPO
27	0.058098	0.018075	TCARI_range
0	0.054788	0.015278	DSM
8	0.054594	0.007820	NDRE_stde
14	0.047838	0.017274	OSAVI_stde
23	0.044261	0.015632	TCARIOSAVI_stde
16	0.040504	0.010034	RECI_mean

```
18          0.032666          0.008645         RECI_range
24          0.027185          0.012605  TCARIOSAVI_range
11          0.020785          0.004831          NDVI_stde
17          0.019819          0.003168          RECI_stde
19          0.018043          0.012037         SLOPE_mean
15          0.014683          0.015328        OSAVI_range
20          0.013282          0.010254          SLOPE_stde
6           0.012863          0.012302        GNDVI_range
9           0.011549          0.008433         NDRE_range
21          0.005346          0.007142        SLOPE_range
10          0.005305          0.010709          NDVI_mean
12          0.001239          0.002450         NDVI_range
5          -0.001329          0.000708         GNDVI_stde
13         -0.008146          0.013483         OSAVI_mean
```

```
# Gráfico
fig, ax = plt.subplots(figsize=(12.5, 15))
df_importancia = df_importancia.
 ↪sort_values('importances_mean', ascending=True)
ax.barh(
    df_importancia['feature'],
    df_importancia['importances_mean'],
    xerr=df_importancia['importances_std'],
    align='center',
    alpha=0
)
ax.plot(
    df_importancia['importances_mean'],
    df_importancia['feature'],
    marker="D",
    linestyle="",
    alpha=0.8,
    color="r"
)
# Etiquetas en el eje x y el eje y
ax.set_xlabel('Importancia')
ax.set_ylabel('Características')
#ax.set_title('Importancia de las Características')

plt.show()
```

7.1.4. RFR 2 - Modelo de todas las variables (optimizado)

7.1.4.1. Optimización de hiperparámetros - Número de árboles En Random Forest, el número de árboles no es un hiperparámetro crítico en cuanto que, añadir árboles, solo puede hacer que mejorar el resultado. En Random Forest no se produce overfitting por exceso de árboles. Sin embargo, añadir árboles una vez que la mejora se estabiliza es una perdida te

recursos computacionales.

7.1.4.2. Validación empleando el Out-of-Bag error

```
train_scores = []
oob_scores   = []

# Valores evaluados
estimator_range = range(1, 150, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪n_estimators y extraer su error
# de entrenamiento y de Out-of-Bag.
for n_estimators in estimator_range:
    modelo = RandomForestRegressor(
                n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 0.4,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )
    modelo.fit(X_train, y_train)
    train_scores.append(modelo.score(X_train, y_train))
    oob_scores.append(modelo.oob_score_)

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, oob_scores, label="out-of-bag␣
 ↪scores")
ax.plot(estimator_range[np.argmax(oob_scores)],␣
 ↪max(oob_scores),
        marker='o', color = "red", label="max score")
ax.set_ylabel("R^2")
ax.set_xlabel("n_estimators")
ax.set_title("Evolución del out-of-bag-error vs número␣
 ↪árboles")
plt.legend();
print(f"Valor óptimo de n_estimators: {estimator_range[np.
 ↪argmax(oob_scores)]}")
```

7.1.4.3. Validación empleando k-cross-validation y neg_root_mean_squared_error

```
train_scores = []
cv_scores    = []
```

```
# Valores evaluados
estimator_range = range(1, 150, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪n_estimators y extraer su error
# de entrenamiento y de k-cross-validation.
for n_estimators in estimator_range:

    modelo = RandomForestRegressor(
                n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 0.4,
                oob_score    = False,
                n_jobs       = -1,
                random_state = 123
             )

    # Error de train
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X = X_train)
    rmse = mean_squared_error(
            y_true  = y_train,
            y_pred  = predicciones,
            squared = False
           )
    train_scores.append(rmse)

    # Error de validación cruzada
    scores = cross_val_score(
                estimator = modelo,
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
                cv        = 5
             )
    # Se agregan los scores de cross_val_score() y se pasa a␣
 ↪positivo
    cv_scores.append(-1*scores.mean())

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, cv_scores, label="cv scores")
ax.plot(estimator_range[np.argmin(cv_scores)],␣
 ↪min(cv_scores),
```

```
        marker='o', color = "red", label="min score")
ax.set_ylabel("root_mean_squared_error")
ax.set_xlabel("n_estimators")
ax.set_title("Evolución del cv-error vs número árboles")
plt.legend();
print(f"Valor óptimo de n_estimators: {estimator_range[np.
 ↪argmin(cv_scores)]}")
```

7.1.4.4. Optimización de hiperparámetros - Max features El valor de máx_features es uno de los hiperparámetros más importantes de random forest, ya que es el que permite controlar cuánto se decorrelacionan los árboles entre sí.

7.1.4.5. Validación empleando el Out-of-Bag error

```
train_scores = []
oob_scores   = []

# Valores evaluados
max_features_range = range(1, X_train.shape[1] + 1, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪max_features y extraer su error
# de entrenamiento y de Out-of-Bag.
for max_features in max_features_range:
    modelo = RandomForestRegressor(
                n_estimators = 24,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = max_features,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )
    modelo.fit(X_train, y_train)
    train_scores.append(modelo.score(X_train, y_train))
    oob_scores.append(modelo.oob_score_)

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(max_features_range, train_scores, label="train␣
 ↪scores")
ax.plot(max_features_range, oob_scores, label="out-of-bag␣
 ↪scores")
ax.plot(max_features_range[np.argmax(oob_scores)],␣
 ↪max(oob_scores),
        marker='o', color = "red")
```

```
ax.set_ylabel("R^2")
ax.set_xlabel("max_features")
ax.set_title("Evolución del out-of-bag-error vs número de␣
 ↪predictores")
plt.legend();
print(f"Valor óptimo de max_features: {max_features_range[np.
 ↪argmax(oob_scores)]}")
```

7.1.4.6. Validación empleando k-cross-validation y neg_root_mean_squared_error

```
train_scores = []
cv_scores    = []

# Valores evaluados
max_features_range = range(1, X_train.shape[1] + 1, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪max_features y extraer su error
# de entrenamiento y de k-cross-validation.
for max_features in max_features_range:

    modelo = RandomForestRegressor(
                n_estimators = 150,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = max_features,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )

    # Error de train
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X = X_train)
    rmse = mean_squared_error(
            y_true  = y_train,
            y_pred  = predicciones,
            squared = False
           )
    train_scores.append(rmse)

    # Error de validación cruzada
    scores = cross_val_score(
                estimator = modelo,
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
```

```
                cv         = 5
            )
    # Se agregan los scores de cross_val_score() y se pasa a␣
 ↪positivo
    cv_scores.append(-1*scores.mean())

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(max_features_range, train_scores, label="train␣
 ↪scores")
ax.plot(max_features_range, cv_scores, label="cv scores")
ax.plot(max_features_range[np.argmin(cv_scores)],␣
 ↪min(cv_scores),
        marker='o', color = "red", label="min score")
ax.set_ylabel("root_mean_squared_error")
ax.set_xlabel("max_features")
ax.set_title("Evolución del cv-error vs número de␣
 ↪predictores")
plt.legend();
print(f"Valor óptimo de max_features: {max_features_range[np.
 ↪argmin(cv_scores)]}")
```

7.1.4.7. Optimización de hiperparámetros - Numero de arboles + Max Features (Grid search)

Aunque el análisis individual de los hiperparámetros es útil para entender su impacto en el modelo e identificar rangos de interés, la búsqueda final no debe hacerse de forma secuencial, ya que cada hiperparámetro interacciona con los demás. Es preferible recurrir a grid search o random search para analizar varias combinaciones de hiperparámetros.

7.1.4.8. Grid Search basado en out-of-bag error

```
# Grid de hiperparámetros evaluados
#␣
 ↪==============================================================================
param_grid = ParameterGrid(
                {'n_estimators': [150],
                 'max_features': [None, 7, 25],
                 'max_depth'   : [None, 3, 10, 20]
                }
            )

# Loop para ajustar un modelo con cada combinación de␣
 ↪hiperparámetros
#␣
 ↪==============================================================================
resultados = {'params': [], 'oob_r2': []}
```

```
for params in param_grid:

    modelo = RandomForestRegressor(
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123,
                ** params
             )

    modelo.fit(X_train, y_train)

    resultados['params'].append(params)
    resultados['oob_r2'].append(modelo.oob_score_)
    print(f"Modelo: {params} ")

# Resultados
#␣
 ↪==========================================================================
resultados = pd.DataFrame(resultados)
resultados = pd.concat([resultados, resultados['params'].
 ↪apply(pd.Series)], axis=1)
resultados = resultados.drop(columns = 'params')
resultados = resultados.sort_values('oob_r2',␣
 ↪ascending=False)
resultados.head(4)
```

```
Modelo: {'max_depth': None, 'max_features': None,␣
 ↪'n_estimators': 150}
Modelo: {'max_depth': None, 'max_features': 7, 'n_estimators':
 ↪ 150}
Modelo: {'max_depth': None, 'max_features': 25,␣
 ↪'n_estimators': 150}
Modelo: {'max_depth': 3, 'max_features': None, 'n_estimators':
 ↪ 150}
Modelo: {'max_depth': 3, 'max_features': 7, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 3, 'max_features': 25, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 10, 'max_features': None,␣
 ↪'n_estimators': 150}
Modelo: {'max_depth': 10, 'max_features': 7, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 10, 'max_features': 25, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 20, 'max_features': None,␣
 ↪'n_estimators': 150}
```

```
Modelo: {'max_depth': 20, 'max_features': 7, 'n_estimators':␣
↪150}
Modelo: {'max_depth': 20, 'max_features': 25, 'n_estimators':␣
↪150}
```

	oob_r2	max_depth	max_features	n_estimators
1	0.136262	NaN	7.0	150.0
7	0.136262	10.0	7.0	150.0
10	0.136262	20.0	7.0	150.0
4	0.126559	3.0	7.0	150.0

```
# Mejores hiperparámetros por out-of-bag error
#␣
↪==============================================================================
print("--------------------------------------------")
print("Mejores hiperparámetros encontrados (oob-r2)")
print("--------------------------------------------")
print(resultados.iloc[0,0:])
```

```
--------------------------------------------
Mejores hiperparámetros encontrados (oob-r2)
--------------------------------------------
oob_r2            0.136262
max_depth              NaN
max_features      7.000000
n_estimators    150.000000
Name: 1, dtype: float64
```

7.1.4.9. Grid Search basado en validación cruzada

```
# Grid de hiperparámetros evaluados
#␣
↪==============================================================================
param_grid = {'n_estimators': [150],
              'max_features': [None,7, 9],
              'max_depth'   : [None, 3, 10, 20]
             }

# Búsqueda por grid search con validación cruzada
#␣
↪==============================================================================
grid = GridSearchCV(
            estimator  = RandomForestRegressor(random_state␣
↪= 123),
            param_grid = param_grid,
            scoring    = 'neg_root_mean_squared_error',
            n_jobs     = multiprocessing.cpu_count() - 1,
```

```
            cv         = RepeatedKFold(n_splits=5,␣
 ↪n_repeats=3, random_state=123),
            refit      = True,
            verbose    = 0,
            return_train_score = True
       )

grid.fit(X = X_train, y = y_train)

# Resultados
#␣
 ↪=============================================================================
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param.*|mean_t|std_t)') \
    .drop(columns = 'params') \
    .sort_values('mean_test_score', ascending = False) \
    .head(4)
```

```
   param_max_depth param_max_features param_n_estimators ␣
 ↪mean_test_score  \
1             None                  7                150     ␣
 ↪   -2.149713
7               10                  7                150     ␣
 ↪   -2.149713
10              20                  7                150     ␣
 ↪   -2.149713
4                3                  7                150     ␣
 ↪   -2.164897

    std_test_score  mean_train_score  std_train_score
1         0.829797         -0.855338         0.104023
7         0.829797         -0.855338         0.104023
10        0.829797         -0.855338         0.104023
4         0.790265         -0.952929         0.115405
```

```
# Mejores hiperparámetros por validación cruzada
#␣
 ↪=============================================================================
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
```

```
----------------------------------------
Mejores hiperparámetros encontrados (cv)
----------------------------------------
{'max_depth': None, 'max_features': 7, 'n_estimators': 150} :␣
 ↪-2.149713016559
```

```
neg_root_mean_squared_error
```

```
# Error de test del modelo final
# ==============================================
modelo_final = grid.best_estimator_
predicciones = modelo_final.predict(X = X_test)
rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print(f"El error (rmse) de test es: {rmse}")

# Calcula el MAPE
mape = mean_absolute_percentage_error(
    y_true=y_test,
    y_pred=predicciones
)

print(f"El MAPE de test es: {mape}")
```

```
El error (rmse) de test es: 2.5518464741220352
El MAPE de test es: 0.3556865790827425
```

```
# Gráfico de dispersión de los resultados predichos frente a␣
 ↪las observaciones reales
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predicciones, alpha=0.5)
plt.xlabel("Observaciones reales")
plt.ylabel("Resultados predichos")
plt.title("Resultados Predichos vs Observaciones Reales")

# Línea de referencia (y=x)
x = np.linspace(min(y_test), max(y_test), 100)
plt.plot(x, x, color='red', linestyle='--', linewidth=2,␣
 ↪label="Línea de Referencia (y=x)")

plt.legend()
plt.grid(True)
plt.show()
```

7.1.5. RFR 3 - Modelo de las variables seleccionadas por importancia

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
```

```
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde',␣
 ↪'Año'])

y = df[['REND_mean']].to_numpy()
df = df.drop(columns = [
'DSM',
'NDVI_stde',
'NDRE_stde',
'OSAVI_stde',
'NDRE_range',
'TCARIOSAVI_range',
'NDVI_mean',
'RECI_mean',
'SLOPE_range',
'SLOPE_stde',
'RECI_range',
'SLOPE_mean',
'OSAVI_mean',
'RECI_stde',
'NDVI_range',
'GNDVI_stde'])

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   EVAPO             33 non-null     float64
 1   PP                33 non-null     float64
 2   TEMP              33 non-null     float64
 3   GNDVI_mean        33 non-null     float64
 4   GNDVI_range       33 non-null     float64
 5   NDRE_mean         33 non-null     float64
 6   OSAVI_range       33 non-null     float64
 7   TCARIOSAVI_mean   33 non-null     float64
 8   TCARIOSAVI_stde   33 non-null     float64
 9   TCARI_mean        33 non-null     float64
 10  TCARI_stde        33 non-null     float64
 11  TCARI_range       33 non-null     float64
 12  REND_mean         33 non-null     float64
dtypes: float64(13)
memory usage: 3.5 KB
```

```
# División de los datos en train y test
#␣
 ↪==========================================================================
```

```
X_train, X_test, y_train, y_test = train_test_split(
                                        df.drop(columns =␣
 ↪["REND_mean"]),
                                        y,
                                        random_state = 123
                                    )

# Creación del modelo
#␣
 ↪=============================================================================
modelo = RandomForestRegressor(
            n_estimators = 24,
            criterion    = 'squared_error', #'mse'
            max_depth    = None,
            max_features = 12,
            bootstrap    = True,
            oob_score    = False,
            n_jobs       = -1,
            random_state = 123
         )

# Entrenamiento del modelo
#␣
 ↪=============================================================================
modelo.fit(X_train, y_train)
```

```
C:
 ↪\ProgramData\Anaconda3\envs\miner-tool\lib\site-packages\sklearn\base.
 ↪py:1152:
DataConversionWarning: A column-vector y was passed when a 1d␣
 ↪array was
expected. Please change the shape of y to (n_samples,), for␣
 ↪example using
ravel().
  return fit_method(estimator, *args, **kwargs)

RandomForestRegressor(max_features=12, n_estimators=24,␣
 ↪n_jobs=-1,
                      random_state=123)
```

```
# Error de test del modelo inicial
# ========================================
predicciones = modelo.predict(X = X_test)

rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
```

```
        squared = False
       )
print(f"El error (rmse) de test es: {rmse}")

# Calcula el MAPE
mape = mean_absolute_percentage_error(
    y_true=y_test,
    y_pred=predicciones
)

print(f"El MAPE de test es: {mape}")
```

```
El error (rmse) de test es: 2.085557232656081
El MAPE de test es: 0.2835310208630302
```

7.1.5.1. Gráfico de dispersión de los resultados predichos frente a las observaciones reales

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predicciones, alpha=0.5)
plt.xlabel("Observaciones reales")
plt.ylabel("Resultados predichos")
plt.title("Resultados Predichos vs Observaciones Reales")

# Línea de referencia (y=x)
x = np.linspace(min(y_test), max(y_test), 100)
plt.plot(x, x, color='red', linestyle='--', linewidth=2,␣
 ↪label="Línea de Referencia (y=x)")

plt.legend()
plt.grid(True)
plt.show()
```

7.1.6. RFR 4 - Modelo de las variables seleccionadas por importancia - optimizado

7.1.6.1. Optimización de hiperparámetros - Número de árboles En Random Forest, el número de árboles no es un hiperparámetro crítico en cuanto que, añadir árboles, solo puede hacer que mejorar el resultado. En Random Forest no se produce overfitting por exceso de árboles. Sin embargo, añadir árboles una vez que la mejora se estabiliza es una perdida te recursos computacionales.

7.1.6.2. Validación empleando el Out-of-Bag error

```
train_scores = []
oob_scores   = []

# Valores evaluados
```

```
estimator_range = range(1, 150, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪n_estimators y extraer su error
# de entrenamiento y de Out-of-Bag.
for n_estimators in estimator_range:
    modelo = RandomForestRegressor(
                n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 0.4,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )
    modelo.fit(X_train, y_train)
    train_scores.append(modelo.score(X_train, y_train))
    oob_scores.append(modelo.oob_score_)

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, oob_scores, label="out-of-bag␣
 ↪scores")
ax.plot(estimator_range[np.argmax(oob_scores)],␣
 ↪max(oob_scores),
        marker='o', color = "red", label="max score")
ax.set_ylabel("R^2")
ax.set_xlabel("n_estimators")
ax.set_title("Evolución del out-of-bag-error vs número␣
 ↪árboles")
plt.legend();
print(f"Valor óptimo de n_estimators: {estimator_range[np.
 ↪argmax(oob_scores)]}")
```

7.1.6.3. Validación empleando k-cross-validation y neg_root_mean_squared_error

```
train_scores = []
cv_scores    = []

# Valores evaluados
estimator_range = range(1, 150, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪n_estimators y extraer su error
# de entrenamiento y de k-cross-validation.
for n_estimators in estimator_range:
```

```
    modelo = RandomForestRegressor(
                n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 0.4,
                oob_score    = False,
                n_jobs       = -1,
                random_state = 123
             )

    # Error de train
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X = X_train)
    rmse = mean_squared_error(
            y_true  = y_train,
            y_pred  = predicciones,
            squared = False
           )
    train_scores.append(rmse)

    # Error de validación cruzada
    scores = cross_val_score(
                estimator = modelo,
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
                cv        = 5
             )
    # Se agregan los scores de cross_val_score() y se pasa a␣
 ↪positivo
    cv_scores.append(-1*scores.mean())

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, cv_scores, label="cv scores")
ax.plot(estimator_range[np.argmin(cv_scores)],␣
 ↪min(cv_scores),
        marker='o', color = "red", label="min score")
ax.set_ylabel("root_mean_squared_error")
ax.set_xlabel("n_estimators")
ax.set_title("Evolución del cv-error vs número árboles")
plt.legend();
print(f"Valor óptimo de n_estimators: {estimator_range[np.
 ↪argmin(cv_scores)]}")
```

7.1.6.4. Optimización de hiperparámetros - Max features El valor de máx_features es uno de los hiperparámetros más importantes de random forest, ya que es el que permite controlar cuánto se decorrelacionan los árboles entre sí.

7.1.6.5. Validación empleando el Out-of-Bag error

```
train_scores = []
oob_scores   = []

# Valores evaluados
max_features_range = range(1, X_train.shape[1] + 1, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪max_features y extraer su error
# de entrenamiento y de Out-of-Bag.
for max_features in max_features_range:
    modelo = RandomForestRegressor(
                n_estimators = 150,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = max_features,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )
    modelo.fit(X_train, y_train)
    train_scores.append(modelo.score(X_train, y_train))
    oob_scores.append(modelo.oob_score_)

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(max_features_range, train_scores, label="train␣
 ↪scores")
ax.plot(max_features_range, oob_scores, label="out-of-bag␣
 ↪scores")
ax.plot(max_features_range[np.argmax(oob_scores)],␣
 ↪max(oob_scores),
        marker='o', color = "red")
ax.set_ylabel("R^2")
ax.set_xlabel("max_features")
ax.set_title("Evolución del out-of-bag-error vs número de␣
 ↪predictores")
plt.legend();
print(f"Valor óptimo de max_features: {max_features_range[np.
 ↪argmax(oob_scores)]}")
```

7.1.6.6. Validación empleando k-cross-validation y neg_root_mean_squared_error

```
train_scores = []
cv_scores    = []

# Valores evaluados
max_features_range = range(1, X_train.shape[1] + 1, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪max_features y extraer su error
# de entrenamiento y de k-cross-validation.
for max_features in max_features_range:

    modelo = RandomForestRegressor(
                n_estimators = 150,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = max_features,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )

    # Error de train
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X = X_train)
    rmse = mean_squared_error(
            y_true  = y_train,
            y_pred  = predicciones,
            squared = False
           )
    train_scores.append(rmse)

    # Error de validación cruzada
    scores = cross_val_score(
                estimator = modelo,
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
                cv        = 5
             )
    # Se agregan los scores de cross_val_score() y se pasa a␣
 ↪positivo
    cv_scores.append(-1*scores.mean())

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(max_features_range, train_scores, label="train␣
 ↪scores")
```

```
ax.plot(max_features_range, cv_scores, label="cv scores")
ax.plot(max_features_range[np.argmin(cv_scores)],␣
 ↪min(cv_scores),
        marker='o', color = "red", label="min score")
ax.set_ylabel("root_mean_squared_error")
ax.set_xlabel("max_features")
ax.set_title("Evolución del cv-error vs número de␣
 ↪predictores")
plt.legend();
print(f"Valor óptimo de max_features: {max_features_range[np.
 ↪argmin(cv_scores)]}")
```

7.1.6.7. Optimización de hiperparámetros - Numero de arboles + Max Features (Grid search)

Aunque el análisis individual de los hiperparámetros es útil para entender su impacto en el modelo e identificar rangos de interés, la búsqueda final no debe hacerse de forma secuencial, ya que cada hiperparámetro interacciona con los demás. Es preferible recurrir a grid search o random search para analizar varias combinaciones de hiperparámetros.

7.1.6.8. Grid Search basado en out-of-bag error

```
### Grid de hiperparámetros evaluados
#␣
 ↪==============================================================================
param_grid = ParameterGrid(
                {'n_estimators': [150],
                 'max_features': [None, 7, 25],
                 'max_depth'   : [None, 3, 10, 20]
                }
             )

# Loop para ajustar un modelo con cada combinación de␣
 ↪hiperparámetros
#␣
 ↪==============================================================================
resultados = {'params': [], 'oob_r2': []}

for params in param_grid:

    modelo = RandomForestRegressor(
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123,
                ** params
             )

    modelo.fit(X_train, y_train)
```

```
    resultados['params'].append(params)
    resultados['oob_r2'].append(modelo.oob_score_)
    print(f"Modelo: {params} ")

# Resultados
#␣
 ↪==========================================================================
resultados = pd.DataFrame(resultados)
resultados = pd.concat([resultados, resultados['params'].
 ↪apply(pd.Series)], axis=1)
resultados = resultados.drop(columns = 'params')
resultados = resultados.sort_values('oob_r2',␣
 ↪ascending=False)
resultados.head(4)
```

	oob_r2	max_depth	max_features	n_estimators
1	0.050429	NaN	7.0	150.0
7	0.050429	10.0	7.0	150.0
10	0.050429	20.0	7.0	150.0
4	0.039434	3.0	7.0	150.0

```
# Mejores hiperparámetros por out-of-bag error
#␣
 ↪==========================================================================
print("--------------------------------------------")
print("Mejores hiperparámetros encontrados (oob-r2)")
print("--------------------------------------------")
print(resultados.iloc[0,0:])
```

```
--------------------------------------------
Mejores hiperparámetros encontrados (oob-r2)
--------------------------------------------
oob_r2             0.050429
max_depth               NaN
max_features       7.000000
n_estimators     150.000000
Name: 1, dtype: float64
```

7.1.6.9. Grid Search basado en validación cruzada

```
# Grid de hiperparámetros evaluados
#␣
 ↪==========================================================================
param_grid = {'n_estimators': [150],
              'max_features': [None,7, 9],
              'max_depth'   : [None, 3, 10, 20]
```

```
            }

# Búsqueda por grid search con validación cruzada
#␣
 ↪==============================================================================
grid = GridSearchCV(
            estimator  = RandomForestRegressor(random_state␣
 ↪= 123),
            param_grid = param_grid,
            scoring    = 'neg_root_mean_squared_error',
            n_jobs     = multiprocessing.cpu_count() - 1,
            cv         = RepeatedKFold(n_splits=5,␣
 ↪n_repeats=3, random_state=123),
            refit      = True,
            verbose    = 0,
            return_train_score = True
       )

grid.fit(X = X_train, y = y_train)

# Resultados
#␣
 ↪==============================================================================
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param.*|mean_t|std_t)') \
    .drop(columns = 'params') \
    .sort_values('mean_test_score', ascending = False) \
    .head(4)
```

```
   param_max_depth param_max_features param_n_estimators␣
 ↪mean_test_score  \
4                3                  7                150␣
 ↪   -2.162930
1             None                  7                150␣
 ↪   -2.168434
7               10                  7                150␣
 ↪   -2.168434
10              20                  7                150␣
 ↪   -2.168434

    std_test_score  mean_train_score  std_train_score
4         0.870399         -0.950330         0.106073
1         0.870372         -0.866078         0.101084
7         0.870372         -0.866078         0.101084
10        0.870372         -0.866078         0.101084
```

```
# Mejores hiperparámetros por validación cruzada
#␣
 ↪==========================================================================
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
```

```
----------------------------------------
Mejores hiperparámetros encontrados (cv)
----------------------------------------
{'max_depth': 3, 'max_features': 7, 'n_estimators': 150} : -2.
 ↪1629302882352177
neg_root_mean_squared_error
```

```
# Error de test del modelo final
# ==============================================
modelo_final = grid.best_estimator_
predicciones = modelo_final.predict(X = X_test)
rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print(f"El error (rmse) de test es: {rmse}")


# Calcula el MAPE
mape = mean_absolute_percentage_error(
    y_true=y_test,
    y_pred=predicciones
)

print(f"El MAPE de test es: {mape}")
```

```
El error (rmse) de test es: 2.3168393861441623
El MAPE de test es: 0.3214762552874096
```

```
# Gráfico de dispersión de los resultados predichos frente a␣
 ↪las observaciones reales
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predicciones, alpha=0.5)
plt.xlabel("Observaciones reales")
plt.ylabel("Resultados predichos")
plt.title("Resultados Predichos vs Observaciones Reales")

# Línea de referencia (y=x)
x = np.linspace(min(y_test), max(y_test), 100)
```

```
plt.plot(x, x, color='red', linestyle='--', linewidth=2,␣
 ↪label="Línea de Referencia (y=x)")

plt.legend()
plt.grid(True)
plt.show()
```

7.1.7. RFR 5 - Modelo de las variables seleccionadas por análisis exploratorio

```
df = pd.read_csv(r'D:
 ↪\Documents\Documents\academico\tesis_maie\data_sets\df\df.
 ↪csv')
df = df.drop(columns = ['Lote', 'REND_range', 'REND_stde',␣
 ↪'Año'])
y = df[['REND_mean']].to_numpy()
y = np.ravel(y)
df = df.drop(columns = [
                        'NDRE_stde', 'NDRE_range',
                        'SLOPE_stde','SLOPE_range',
                        'GNDVI_stde', 'GNDVI_range',
                         'NDVI_stde', 'NDVI_range',
                        'OSAVI_stde', 'OSAVI_range',
                        'RECI_stde', 'RECI_range',
                        'TCARIOSAVI_stde', 'TCARIOSAVI_range',
                        'TCARI_stde', 'TCARI_range'])


new_column_names = {
    'GNDVI_mean': 'GNDVI',
    'NDRE_mean': 'NDRE',
    'NDVI_mean': 'NDVI',
    'OSAVI_mean': 'OSAVI',
    'RECI_mean': 'RECI',
    'SLOPE_mean': 'SLOPE',
    'TCARIOSAVI_mean': 'TCARIOSAVI',
    'TCARI_mean': 'TCARI',
    'REND_mean': 'REND'
}
df.rename(columns=new_column_names, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33 entries, 0 to 32
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   DSM         33 non-null     float64
```

```
 1   EVAPO        33 non-null     float64
 2   PP           33 non-null     float64
 3   TEMP         33 non-null     float64
 4   GNDVI        33 non-null     float64
 5   NDRE         33 non-null     float64
 6   NDVI         33 non-null     float64
 7   OSAVI        33 non-null     float64
 8   RECI         33 non-null     float64
 9   SLOPE        33 non-null     float64
 10  TCARIOSAVI   33 non-null     float64
 11  TCARI        33 non-null     float64
 12  REND         33 non-null     float64
dtypes: float64(13)
memory usage: 3.5 KB
```

```
# División de los datos en train y test
#␣
 ↪==============================================================================
X_train, X_test, y_train, y_test = train_test_split(
                                        df.drop(columns =␣
 ↪["REND"]), #,  'Rend_std' ,'Rend_var', 'AÃ±o']),
                                        y,
                                        random_state = 123
                                    )

# Creación del modelo
#␣
 ↪==============================================================================
modelo = RandomForestRegressor(
            n_estimators = 24,
            criterion    = 'squared_error', #'mse'
            max_depth    = None,
            max_features = 11,
            bootstrap    = True,
            oob_score    = False,
            n_jobs       = -1,
            random_state = 123
         )

# Entrenamiento del modelo
#␣
 ↪==============================================================================
modelo.fit(X_train, y_train)
```

```
RandomForestRegressor(max_features=11, n_estimators=24,␣
 ↪n_jobs=-1,
                      random_state=123)
```

```
# Error de test del modelo
# ==============================================
predicciones = modelo.predict(X = X_test)

rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print(f"El error (rmse) de test es: {rmse}")

# Calcula el MAPE
mape = mean_absolute_percentage_error(
    y_true=y_test,
    y_pred=predicciones
)

print(f"El MAPE de test es: {mape}")
```

```
El error (rmse) de test es: 1.8773980547724698
El MAPE de test es: 0.24797314449770294
```

```
# Gráfico de dispersión de los resultados predichos frente a␣
 ↪las observaciones reales
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predicciones, alpha=0.5)
plt.xlabel("Observaciones reales")
plt.ylabel("Resultados predichos")
plt.title("Resultados Predichos vs Observaciones Reales")

# Línea de referencia (y=x)
x = np.linspace(min(y_test), max(y_test), 100)
plt.plot(x, x, color='red', linestyle='--', linewidth=2,␣
 ↪label="Línea de Referencia (y=x)")

plt.legend()
plt.grid(True)
plt.show()
```

7.1.7.1. Análisis de las Relaciones parciales entre las variables del modelo RFR -5

```
common_params = {
    "subsample": 50,
    "n_jobs": 2,
    "grid_resolution": 20,
    "random_state": 2,
    "n_cols": 4  # Ajusta el número de columnas para la␣
 ↪presentación de los gráficos
```

```
}

# Lista de las 28 variables que deseas analizar
variables_to_plot = [ 'DSM',
'EVAPO',
'PP',
'TEMP',
'GNDVI',
'NDRE',
'NDVI',
'OSAVI',
'RECI',
'SLOPE',
'TCARIOSAVI',
'TCARI']


print("Computing partial dependence plots...")
tic = time()

# Calcular el número total de gráficos a generar
total_plots = len(variables_to_plot)

# Calcular el número de filas necesarias para acomodar los␣
 ↪gráficos
n_rows = (total_plots - 1) // common_params["n_cols"] + 1

# Crear una figura con subplots
fig, axes = plt.subplots(n_rows, common_params["n_cols"],␣
 ↪figsize=(18, 15))


# Iterar a través de las variables y generar los gráficos de␣
 ↪dependencia parcial
for i, variable in enumerate(variables_to_plot):
    row = i // common_params["n_cols"]
    col = i % common_params["n_cols"]

    # Generar el gráfico para la variable actual
    display = PartialDependenceDisplay.from_estimator(
        modelo,
        X_train,
        features=[variable],
        kind="both",
        ax=axes[row, col],  # Usar el subplot correspondiente
        **common_params,
    )
```

```
    display.figure_.suptitle("")
```

7.1.8. RFR 6 - Modelo de las variables seleccionadas por análisis exploratorio - optimizado

7.1.8.1. Optimización de hiperparámetros - Número de árboles

En Random Forest, el número de árboles no es un hiperparámetro crítico en cuanto que, añadir árboles, solo puede hacer que mejorar el resultado. En Random Forest no se produce overfitting por exceso de árboles. Sin embargo, añadir árboles una vez que la mejora se estabiliza es una perdida te recursos computacionales.

7.1.8.2. Validación empleando el Out-of-Bag error

```
train_scores = []
oob_scores   = []

# Valores evaluados
estimator_range = range(1, 150, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪n_estimators y extraer su error
# de entrenamiento y de Out-of-Bag.
for n_estimators in estimator_range:
    modelo = RandomForestRegressor(
                n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 0.4,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )
    modelo.fit(X_train, y_train)
    train_scores.append(modelo.score(X_train, y_train))
    oob_scores.append(modelo.oob_score_)

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, oob_scores, label="out-of-bag␣
 ↪scores")
ax.plot(estimator_range[np.argmax(oob_scores)],␣
 ↪max(oob_scores),
        marker='o', color = "red", label="max score")
ax.set_ylabel("R^2")
ax.set_xlabel("n_estimators")
```

```
ax.set_title("Evolución del out-of-bag-error vs número␣
 ↪árboles")
plt.legend();
print(f"Valor óptimo de n_estimators: {estimator_range[np.
 ↪argmax(oob_scores)]}")
```

7.1.8.3. Validación empleando k-cross-validation y neg_root_mean_squared_error

```
train_scores = []
cv_scores    = []

# Valores evaluados
estimator_range = range(1, 150, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪n_estimators y extraer su error
# de entrenamiento y de k-cross-validation.
for n_estimators in estimator_range:

    modelo = RandomForestRegressor(
                n_estimators = n_estimators,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = 0.4,
                oob_score    = False,
                n_jobs       = -1,
                random_state = 123
             )

    # Error de train
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X = X_train)
    rmse = mean_squared_error(
            y_true  = y_train,
            y_pred  = predicciones,
            squared = False
           )
    train_scores.append(rmse)

    # Error de validación cruzada
    scores = cross_val_score(
                estimator = modelo,
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
                cv        = 5
             )
```

```
    # Se agregan los scores de cross_val_score() y se pasa a␣
 ↪positivo
    cv_scores.append(-1*scores.mean())

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(estimator_range, train_scores, label="train scores")
ax.plot(estimator_range, cv_scores, label="cv scores")
ax.plot(estimator_range[np.argmin(cv_scores)],␣
 ↪min(cv_scores),
        marker='o', color = "red", label="min score")
ax.set_ylabel("root_mean_squared_error")
ax.set_xlabel("n_estimators")
ax.set_title("Evolución del cv-error vs número árboles")
plt.legend();
print(f"Valor óptimo de n_estimators: {estimator_range[np.
 ↪argmin(cv_scores)]}")
```

7.1.8.4. Optimización de hiperparámetros - Max features El valor de máx_features es uno de los hiperparámetros más importantes de random forest, ya que es el que permite controlar cuánto se decorrelacionan los árboles entre sí.

7.1.8.5. Validación empleando el Out-of-Bag error

```
train_scores = []
oob_scores   = []

# Valores evaluados
max_features_range = range(1, X_train.shape[1] + 1, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪max_features y extraer su error
# de entrenamiento y de Out-of-Bag.
for max_features in max_features_range:
    modelo = RandomForestRegressor(
                n_estimators = 150,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = max_features,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )
    modelo.fit(X_train, y_train)
    train_scores.append(modelo.score(X_train, y_train))
    oob_scores.append(modelo.oob_score_)
```

```
# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(max_features_range, train_scores, label="train␣
 ↪scores")
ax.plot(max_features_range, oob_scores, label="out-of-bag␣
 ↪scores")
ax.plot(max_features_range[np.argmax(oob_scores)],␣
 ↪max(oob_scores),
        marker='o', color = "red")
ax.set_ylabel("R^2")
ax.set_xlabel("max_features")
ax.set_title("Evolución del out-of-bag-error vs número de␣
 ↪predictores")
plt.legend();
print(f"Valor óptimo de max_features: {max_features_range[np.
 ↪argmax(oob_scores)]}")
```

7.1.8.6. Validación empleando k-cross-validation y neg_root_mean_squared_error

```
train_scores = []
cv_scores    = []

# Valores evaluados
max_features_range = range(1, X_train.shape[1] + 1, 1)

# Bucle para entrenar un modelo con cada valor de␣
 ↪max_features y extraer su error
# de entrenamiento y de k-cross-validation.
for max_features in max_features_range:

    modelo = RandomForestRegressor(
                n_estimators = 150,
                criterion    = 'squared_error',
                max_depth    = None,
                max_features = max_features,
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123
             )

    # Error de train
    modelo.fit(X_train, y_train)
    predicciones = modelo.predict(X = X_train)
    rmse = mean_squared_error(
            y_true  = y_train,
            y_pred  = predicciones,
```

```
            squared = False
           )
    train_scores.append(rmse)

    # Error de validación cruzada
    scores = cross_val_score(
                estimator = modelo,
                X         = X_train,
                y         = y_train,
                scoring   = 'neg_root_mean_squared_error',
                cv        = 5
             )
    # Se agregan los scores de cross_val_score() y se pasa a␣
 ↪positivo
    cv_scores.append(-1*scores.mean())

# Gráfico con la evolución de los errores
fig, ax = plt.subplots(figsize=(10, 6.4))
ax.plot(max_features_range, train_scores, label="train␣
 ↪scores")
ax.plot(max_features_range, cv_scores, label="cv scores")
ax.plot(max_features_range[np.argmin(cv_scores)],␣
 ↪min(cv_scores),
        marker='o', color = "red", label="min score")
ax.set_ylabel("root_mean_squared_error")
ax.set_xlabel("max_features")
ax.set_title("Evolución del cv-error vs número de␣
 ↪predictores")
plt.legend();
print(f"Valor óptimo de max_features: {max_features_range[np.
 ↪argmin(cv_scores)]}")
```

7.1.8.7. Optimización de hiperparámetros - Numero de arboles + Max Features (Grid search) Aunque el análisis individual de los hiperparámetros es útil para entender su impacto en el modelo e identificar rangos de interés, la búsqueda final no debe hacerse de forma secuencial, ya que cada hiperparámetro interacciona con los demás. Es preferible recurrir a grid search o random search para analizar varias combinaciones de hiperparámetros.

7.1.8.8. Grid Search basado en out-of-bag error

```
### Grid de hiperparámetros evaluados
#␣
 ↪==============================================================================
param_grid = ParameterGrid(
                {'n_estimators': [150],
                 'max_features': [None, 7, 25],
```

```
                'max_depth'    : [None, 3, 10, 20]
               }
            )

# Loop para ajustar un modelo con cada combinación de␣
 ↪hiperparámetros
#␣
 ↪============================================================================
resultados = {'params': [], 'oob_r2': []}

for params in param_grid:

    modelo = RandomForestRegressor(
                oob_score    = True,
                n_jobs       = -1,
                random_state = 123,
                ** params
            )

    modelo.fit(X_train, y_train)

    resultados['params'].append(params)
    resultados['oob_r2'].append(modelo.oob_score_)
    print(f"Modelo: {params} ")

# Resultados
#␣
 ↪============================================================================
resultados = pd.DataFrame(resultados)
resultados = pd.concat([resultados, resultados['params'].
 ↪apply(pd.Series)], axis=1)
resultados = resultados.drop(columns = 'params')
resultados = resultados.sort_values('oob_r2',␣
 ↪ascending=False)
resultados.head(4)
```

```
Modelo: {'max_depth': None, 'max_features': None,␣
 ↪'n_estimators': 150}
Modelo: {'max_depth': None, 'max_features': 7, 'n_estimators':
 ↪ 150}
Modelo: {'max_depth': None, 'max_features': 25,␣
 ↪'n_estimators': 150}
Modelo: {'max_depth': 3, 'max_features': None, 'n_estimators':
 ↪ 150}
Modelo: {'max_depth': 3, 'max_features': 7, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 3, 'max_features': 25, 'n_estimators':␣
 ↪150}
```

```
Modelo: {'max_depth': 10, 'max_features': None,␣
 ↪'n_estimators': 150}
Modelo: {'max_depth': 10, 'max_features': 7, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 10, 'max_features': 25, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 20, 'max_features': None,␣
 ↪'n_estimators': 150}
Modelo: {'max_depth': 20, 'max_features': 7, 'n_estimators':␣
 ↪150}
Modelo: {'max_depth': 20, 'max_features': 25, 'n_estimators':␣
 ↪150}
```

	oob_r2	max_depth	max_features	n_estimators
1	0.154437	NaN	7.0	150.0
7	0.154437	10.0	7.0	150.0
10	0.154437	20.0	7.0	150.0
4	0.137227	3.0	7.0	150.0

```
# Mejores hiperparámetros por out-of-bag error
#␣
 ↪==============================================================================
print("--------------------------------------------")
print("Mejores hiperparámetros encontrados (oob-r2)")
print("--------------------------------------------")
print(resultados.iloc[0,0:])
```

7.1.8.9. Grid Search basado en validación cruzada

```
# Grid de hiperparámetros evaluados
#␣
 ↪==============================================================================
param_grid = {'n_estimators': [150],
              'max_features': [None,7, 9],
              'max_depth'   : [None, 3, 10, 20]
             }

# Búsqueda por grid search con validación cruzada
#␣
 ↪==============================================================================
grid = GridSearchCV(
            estimator  = RandomForestRegressor(random_state␣
 ↪= 123),
            param_grid = param_grid,
            scoring    = 'neg_root_mean_squared_error',
            n_jobs     = multiprocessing.cpu_count() - 1,
```

```
            cv         = RepeatedKFold(n_splits=5,␣
 ↪n_repeats=3, random_state=123),
            refit      = True,
            verbose    = 0,
            return_train_score = True
       )

grid.fit(X = X_train, y = y_train)

# Resultados
#␣
 ↪=============================================================================
resultados = pd.DataFrame(grid.cv_results_)
resultados.filter(regex = '(param.*|mean_t|std_t)') \
    .drop(columns = 'params') \
    .sort_values('mean_test_score', ascending = False) \
    .head(4)
```

```
   param_max_depth param_max_features param_n_estimators ␣
 ↪mean_test_score  \
4                3                  7                150     ␣
 ↪   -2.127466
1             None                  7                150     ␣
 ↪   -2.145209
7               10                  7                150     ␣
 ↪   -2.145209
10              20                  7                150     ␣
 ↪   -2.145209

    std_test_score  mean_train_score  std_train_score
4         0.995886         -0.939264         0.123077
1         0.993298         -0.841794         0.114452
7         0.993298         -0.841794         0.114452
10        0.993298         -0.841794         0.114452
```

```
# Mejores hiperparámetros por validación cruzada
#␣
 ↪=============================================================================
print("----------------------------------------")
print("Mejores hiperparámetros encontrados (cv)")
print("----------------------------------------")
print(grid.best_params_, ":", grid.best_score_, grid.scoring)
```

```
----------------------------------------
Mejores hiperparámetros encontrados (cv)
----------------------------------------
{'max_depth': 3, 'max_features': 7, 'n_estimators': 150} : -2.
 ↪127466004623812
```

```
neg_root_mean_squared_error
```

```
# Error de test del modelo final
# ==============================================
modelo_final = grid.best_estimator_
predicciones = modelo_final.predict(X = X_test)
rmse = mean_squared_error(
        y_true  = y_test,
        y_pred  = predicciones,
        squared = False
       )
print(f"El error (rmse) de test es: {rmse}")


# Calcula el MAPE
mape = mean_absolute_percentage_error(
    y_true=y_test,
    y_pred=predicciones
)

print(f"El MAPE de test es: {mape}")
```

```
El error (rmse) de test es: 1.9962755260731873
El MAPE de test es: 0.280593029578672
```

```
# Gráfico de dispersión de los resultados predichos frente a␣
 ↪las observaciones reales
plt.figure(figsize=(10, 6))
plt.scatter(y_test, predicciones, alpha=0.5)
plt.xlabel("Observaciones reales")
plt.ylabel("Resultados predichos")
plt.title("Resultados Predichos vs Observaciones Reales")

# Línea de referencia (y=x)
x = np.linspace(min(y_test), max(y_test), 100)
plt.plot(x, x, color='red', linestyle='--', linewidth=2,␣
 ↪label="Línea de Referencia (y=x)")

plt.legend()
plt.grid(True)
plt.show()
```