Universidad Zaragoza
1542

Trabajo Fin de Grado

Aplicación de técnicas de inteligencia artificial para la propuesta de fecha óptima de cosecha del maíz

Application of artificial intelligence techniques for the proposal of optimal corn harvest date

Autor

Ernesto Bielsa Gracia

Director

Juan López de Larrinzar Galdámez

Ponente

F. Javier Zarazaga Soria

Escuela de Ingeniería y Arquitectura

2022/2023

Aplicación de técnicas de inteligencia artificial para la propuesta de fecha óptima de cosecha del maíz

Resumen

El presente TFG se enfoca en el desarrollo de un sistema de predicción de la fecha óptima de cosecha de parcelas de maíz. Se emplea un enfoque basado en el análisis de datos agrícolas, como el NDVI, y datos climatológicos, como la precipitación, temperatura y presión atmosférica, obtenidos a través de teledetección. El trabajo se ha estructurado en iteraciones, donde se plantean hipótesis, se describe el proceso realizado para verificarlas y se presentan los resultados obtenidos. Las primeras iteraciones han sido enfocadas en la obtención de la información necesaria de manera precisa, utilizando peticiones a la API del cliente GeosLab y desarrollando un algoritmo de suavizado propio para limpiar los datos espurios causados por condiciones meteorológicas a las que es sensible la teledetección. Posteriormente, se han explorado distintas formas de tratar los datos junto a distintos modelos basados en inteligencia artificial, como redes neuronales, árboles de decisión y gradient boosting. Se han evaluado las hipótesis utilizando medidas de evaluación convencionales y medidas específicas creadas para este propósito. Como resultado, se ha obtenido un modelo final basado en Gradient Boosting, que, entrenado con más de 700 parcelas de dos años diferentes, ha logrado una precisión del 83% en los datos de test, prediciendo correctamente la fecha óptima de cosecha en hasta el 93.14% de las parcelas evaluadas. Todo el trabajo se ha documentado y recopilado en el repositorio
https://drive.google.com/drive/folders/1NvO70CVqZV-eoO9YlTlCpLJ53SAjt9u9?usp=drive_link
para su uso como base en futuros desarrollos y aplicaciones industriales en el campo de la predicción de cosechas.

DECLARACIÓN DE AUTORÍA

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Ernesto Bielsa Gracia ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de Grado en Ingeniería Informática (Título del Trabajo)

Aplicación de técnicas de inteligencia artificial para la propuesta de fecha óptima de cosecha del maíz / Application of artificial intelligence techniques for the proposal of optimal corn harvest date

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 06 de Junio de 2023

BIELSA GRACIA ERNESTO - 25203171R

Firmado digitalmente por BIELSA GRACIA ERNESTO - 25203171R
Fecha: 2023.06.06 12:43:35 +02'00'

Fdo: Ernesto

ÍNDICE

AGRADECIMIENTOS

Quiero expresar mis profundos agradecimientos a todas las personas que me han ayudado a sacar adelante este Trabajo de Fin de Grado.

En primer lugar, a Noelia, David y Juan de GeosLab, y a Mario y Javier del IAAA, agradecerles infinitamente su compromiso y colaboración a lo largo de todo este tiempo. No habría sido posible sin vosotros.

En segundo lugar, a mi familia, mis amigos y a todos aquellos que se han interesado en lo que he estado haciendo durante todo este tiempo. Gracias por confiar en mí.

1 INTRODUCCIÓN

1.1 Contexto del trabajo

GeoSpatiumLab S.L. (Geoslab) es una empresa de base tecnológica nacida en 2007 como spin-off del grupo de Sistemas de Información Avanzados de la Universidad de Zaragoza. Geoslab desarrolla tecnologías, proyectos y productos en los que los elementos geográficos de la información cobran un papel fundamental. En este sentido, y en el marco de una línea de trabajo vinculada a clientes del mundo agrario, la empresa se plantea abordar diferentes iniciativas destinadas a crear bases tecnológicas para el desarrollo de nuevos sistemas en el ámbito de la teledetección y la inteligencia artificial.

El maíz, junto con el arroz y el trigo, es uno de los tres cereales más consumidos en el mundo[1]. No obstante, y como una de las principales diferencias en el proceso industrial entre el maíz y los otros dos cereales, existen dos tipos de procesos diferentes para el maíz: la molienda en seco y la molienda en húmedo (el arroz y el trigo se procesan industrialmente sólo como molienda en seco). La primera produce, como productos primarios, sémola y varios tipos de harinas. La molienda húmeda produce, principalmente, almidón, gluten y piensos[2].

Uno de los principales problemas que tiene la molienda húmeda del maíz es la necesidad de minimizar los períodos de almacenamiento antes del procesamiento. El maíz húmedo es candidato para producir crecimiento de hongos y plagas de insectos[3]. En un escenario ideal, los granos deben introducirse en la cadena de proceso directamente desde su cosecha. En este sentido, la programación de las actividades de cosecha es fundamental para tener resultados óptimos.

1.2 Contexto tecnológico y herramientas utilizadas

Para el procesamiento y análisis de datos geoespaciales, se ha utilizado QGIS[4], una poderosa herramienta de Sistemas de Información Geográfica (SIG) de código abierto. QGIS permite la visualización, manipulación y análisis de datos espaciales, así como la generación de mapas temáticos y la realización de consultas espaciales. Además, se ha empleado GDAL[5] (*Geospatial Data Abstraction Library*), una biblioteca de código abierto para el manejo de datos geoespaciales en diferentes formatos. GDAL proporciona funciones esenciales para la lectura,

[1] Ramos Gamiño, Félix (2013) Maíz, trigo y arroz : los cereales que alimentan al mundo. 5 (5). Universidad Autónoma de Nuevo León, Monterrey, Nuevo León. ISBN 9786074339932. https://eprints.uanl.mx/3649/1/maiztrigoarroz.pdf

[2] Acuña, V. S. (2011). El maíz y su transformación en harina. Santa Fe, Argentina: El Cid. Disponible en https://www.monografias.com/trabajos16/maiz-harina/maiz-harina

[3] Angel Martín Gil, Santiago Cepeda Castro (Coordinadores). GUÍA DE GESTIÓN INTEGRADA DE PLAGAS: MAÍZ. Ministerio de Agricultura, Alimentación y Medio Ambiente. ISBN: 978-84-491-0037-6. 2015. https://www.mapa.gob.es/es/agricultura/temas/sanidad-vegetal/guiamaiz_tcm30-57958.pdf

[4] https://qgis.org/es/site/

[5] https://gdal.org/

escritura y transformación de datos raster y vectoriales, lo que ha facilitado la interoperabilidad entre diferentes fuentes de datos geográficos.

Python[6] ha sido el lenguaje de programación principal utilizado en el proyecto. Se aprovecha la amplia gama de librerías disponibles para análisis de datos y aprendizaje automático, como *Pandas*[7], *NumPy*[8], *Scikit-learn*[9] y *TensorFlow*[10]. Python ha permitido la automatización de tareas, el desarrollo de algoritmos personalizados y la implementación de modelos de machine learning. Además, se ha utilizado la librería *Requests*[11] para acceder a los servicios de la API de GeosLab.

Jupyter Notebook[12] ha sido la plataforma de desarrollo interactiva utilizada para la exploración y prototipado de código. Jupyter Notebook facilita la creación de documentos que combinan código, visualizaciones y explicaciones en un entorno colaborativo y reproducible.

Finalmente, se hizo uso de la línea de comandos (*Shell*) para ejecutar comandos y scripts, lo que permitió la automatización de tareas y la interacción con las herramientas mencionadas anteriormente.

1.3 Alcance, objetivos y limitaciones

El objetivo de este TFG ha sido determinar la capacidad de poder proponer una fecha de cosecha óptima para explotaciones de maíz haciendo uso de datos de teledetección y meteorológicos. Para ello, se han aplicado técnicas de inteligencia artificial para desarrollar un sistema que permita efectuar propuestas de fecha de recolección. Como se ha destacado, no se busca el desarrollo del sistema de ayuda a la toma de decisiones, sino demostrar que es viable poder desarrollarlo sobre la base de estos recursos de información.

Así mismo, el alcance del trabajo realizado también viene marcado por los datos utilizados. El trabajo se ha llevado a cabo haciendo uso de la información sobre la cosecha de maíz de todo un conjunto de parcelas vinculadas a un cliente de Geoslab. Concretamente, este cliente ha suministrado el identificador de las parcelas, así como la fecha de cosecha de las mismas. Todas estas parcelas se encuentran en el valle de Ebro por lo que la validez de los resultados que se presentan en este trabajo puede estar condicionada por las características bioclimáticas de esta zona geográfica.

Finalmente, ha existido una limitación fundamental vinculada a la propia dimensión de lo que es un TFG y los 12 ECTS asociados al mismo. Tal y como se presenta en la sección de

[6] https://www.python.org/
[7] https://pandas.pydata.org/
[8] https://numpy.org/
[9] https://scikit-learn.org/stable/
[10] https://www.tensorflow.org/learn
[11] https://pypi.org/project/requests/
[12] https://jupyter.org/

metodología, sería posible seguir avanzando en sucesivas iteraciones con el fin de poder identificar nuevas variables a considerar e inferir conocimiento a partir de las mismas en la búsqueda de una mayor precisión.

1.4 Esquema general de la memoria del proyecto

Además de este capítulo de introducción, esta memoria se compone de las siguientes secciones:

- Capítulo 2: Trabajo desarrollado: Explica la metodología utilizada en el proyecto, describe las iteraciones realizadas, aborda la dimensión del trabajo realizado y presenta los problemas encontrados durante el proceso.
- Capítulo 3: Conocimientos adquiridos y conclusiones: Expone los conocimientos adquiridos a lo largo del proyecto, plantea ideas futuras para continuar la investigación y presenta las conclusiones obtenidas.
- Anexo I: Teledetección y Datos Disponibles: Presenta algunos aspectos básicos de la teledetección, así como las fuentes de datos existentes y que se han usado para el proyecto.
- Anexo II: Términos: define terminología específica utilizada en el proyecto, como el SIGPAC, junto con un ejemplo de su enumeración.
- Anexo III: Detalles de las iteraciones 1 a 11: Proporciona la sección restante de trabajo desarrollado de las iteraciones 1 a 11, incluyendo de nuevo la sección de hipótesis y resultados para una lectura más fácil.

2 TRABAJO DESARROLLADO

2.1 Metodología de trabajo

El trabajo llevado a cabo es completamente elucubrativo en sus orígenes y puede ser calificado como fundamentado en investigación. En este sentido, se propuso abordarlo en sucesivas iteraciones que permitieran ir fijando las tareas a llevar a cabo en función de los resultados que fueran obteniéndose. Con carácter general, el trabajo ha necesitado de una identificación de variables a considerar, un acceso a valores para estas variables, operaciones de limpieza de datos, análisis para establecer dependencias y la realización de múltiples experimentos para determinar el modelo de IA que de mejores resultados.

Desde una aproximación más formal, se han ido proponiendo diferentes iteraciones que han contado con:

1. Hipótesis de partida de la iteración. En la fijación de cada iteración se ha discutido cuál podría ser esta hipótesis a partir de los resultados de la iteración anterior (excepto en la iteración 1) y de la visión del contexto que se tenía. En este punto se ha trabajado con el director, el ponente y la colaboración de técnicos de Geoslab.
2. Desarrollo del trabajo de validación de la hipótesis. Mediante tareas de limpieza de datos, análisis de correlaciones, y modelización de sistemas de inteligencia artificial, y su posterior evaluación, se ha buscado obtener una serie de resultados sobre los que construir el análisis del paso 3.
3. Análisis de los resultados obtenidos. A partir de los resultados de las tareas anteriores, se ha determinado la validez o no de la hipótesis planteada. En ocasiones, si los resultados no han sido concluyentes, ha sido necesario volver al punto 2 para complementar/rehacer/replantear tareas.

Adicionalmente, se ha incluido una iteración 0 destinada a recopilar y preparar todos los conjuntos de datos para los posteriores trabajos en el resto de iteraciones.

Seguidamente se detallan las 12 iteraciones que se han llevado a cabo en el proyecto. Tal y como se ha mencionado anteriormente, este número de iteraciones ha venido limitado por el esfuerzo de trabajo asociado a un TFG.

2.2 Iteraciones del proyecto

El detalle de todas las iteraciones llevadas a cabo es muy prolífico y su extensión hace que se exceda la longitud sugerida para el núcleo central de la memoria. Adicionalmente, en muchos casos el trabajo desarrollado ha conducido a demostrar que la hipótesis planteada no era correcta. Es por ello que se ha optado por relegar los detalles de las iteraciones 1 a 11 al Anexo III, incluyendo aquí solamente las hipótesis de partida y los resultados obtenidos.

Hipótesis general

En la hipótesis general del proyecto, se establece la posibilidad de distinguir, dentro de una misma parcela, la fecha óptima de cosecha del cultivo, así como las zonas con diferentes cultivos y el tipo de vegetal cultivado en cada una de las subparcelas identificadas. Para lograr esto, se utilizan como punto de partida la geometría de la parcela de la explotación agrícola y los datos de teledetección que registran la evolución de las plantas mediante el Índice de Vegetación de Diferencia Normalizada –también conocido como NDVI[13] (*Normalized Difference Vegetation Index*)–.

2.2.1 Iteración 0

Hipótesis de partida

En esta primera iteración, se considera factible recopilar la información de cada píxel de las parcelas de la zona de estudio utilizando las herramientas GDAL y QGIS. Estas herramientas son ampliamente utilizadas en el procesamiento y análisis de datos geoespaciales.

El objetivo de esta recopilación de información es la familiarización con las herramientas utilizadas, visualizar los datos y utilizarlos en Python, en línea con los objetivos establecidos.

Trabajo desarrollado

Para preparar los datos, se ha dispuesto, tanto de imágenes de satélite de la zona de estudio con el valor NDVI de cada píxel ('T30TXMyyyymmdd_NDVI.tiff'), como de un conjunto de 15 parcelas definidas por su geometría en un archivo shapefile ('doble_cultivo.shp').

Para preparar los datos se han utilizado las siguientes herramientas de software:

- QGIS 3.22.4-Białowieża: Es un Sistema de Información Geográfica (SIG) de software libre que permite manejar formatos raster (TIFF, JPG, etc.) y vectoriales (Shapefile, etc.) a través de la biblioteca GDAL (GADL/OGR)[14].
- GDAL 3.0.2: Es una biblioteca de software para la lectura y escritura de formatos de datos geoespaciales. Viene con una variedad de utilidades en línea de comando para la traducción y el procesamiento de datos geoespaciales[15].

Se ha generado en QGIS el archivo vectorial que contiene las geometrías de las parcelas ('doble_cultivo.shp') y se ha establecido su sistema de coordenadas en EPSG:32630, que corresponde a la zona UTM 30N en la cual se encuentra parte de España. Esto se ha realizado con el propósito de lograr la coincidencia de las coordenadas entre dicho archivo vectorial (ahora denominado 'doble_cultivo_32630.shp') y la imagen de satélite del terreno de Navarra ('T30TXMyyyymmdd_NDVI.tiff').

Posteriormente, con los rasters y las geometrías en el mismo sistema de coordenadas, se han abierto en QGIS (figura 1) para su visualización. Mediante este software, se han obtenido las

[13] https://es.wikipedia.org/wiki/Índice_de_vegetación_de_diferencia_normalizada
[14] https://es.wikipedia.org/wiki/QGIS
[15] https://es.wikipedia.org/wiki/GDAL

coordenadas de la esquina inferior izquierda (x1, y1) y la esquina superior derecha (x2, y2) del cuadrado que engloba cada parcela. Es importante destacar que estas coordenadas deben ser múltiplos de 10, ya que la resolución espacial del raster es de 10 metros.

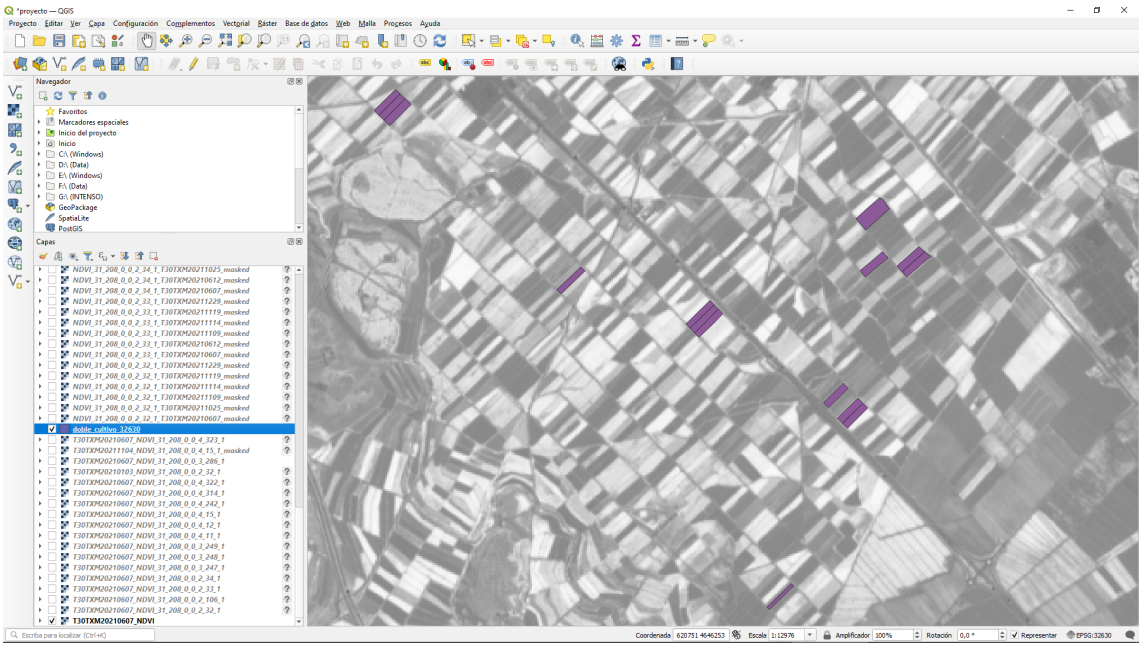


Figura 1: Proyecto QGIS que muestra la geometría de las parcelas superpuesta al raster de Navarra.

Ahora que se cuenta con la localización de cada parcela ('scripts/locations.txt') y un raster de la zona de estudio que abarca 71 fechas diferentes (ubicado en el disco duro externo), se ha utilizado el siguiente comando de GDAL para obtener un nuevo raster específico para cada fecha y cada parcela. El objetivo es generar sub-rasters que tengan el tamaño adecuado para contener cada parcela en cada fecha correspondiente, permitiendo trabajar con imágenes de menor tamaño. El comando utilizado ha sido el siguiente:

gdalwarp.exe -te <x1> <y1> <x2> <y2> <ruta raster in> <ruta raster out>

Este comando se ha ejecutado dentro del script 'scripts/recortar.sh', lo que ha permitido realizar el proceso de manera automática para cada uno de los rasters principales que contienen las 71 fechas diferentes y las 15 parcelas distintas. Como resultado, se han obtenido 1065 sub-rasters adicionales. A partir de este momento, se ha trabajado con rasters similares a los que se presentan en la figura 2.

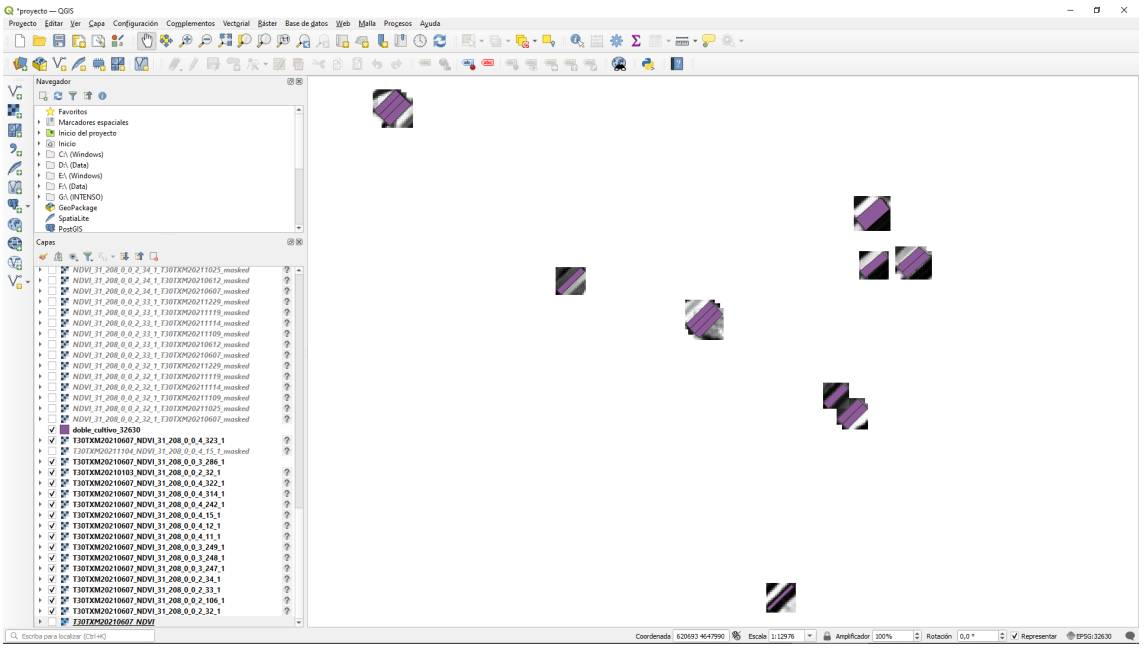


Figura 2: Proyecto QGIS que muestra la geometría de las parcelas superpuesta a los diferentes rasters de menor tamaño que las contienen.

El siguiente paso ha consistido en aplicar una máscara a la parcela para retener únicamente los píxeles del raster que se encuentren dentro de la geometría de la parcela. Esto se ha logrado utilizando el siguiente comando:

gdalwarp.exe -cutline <ruta doble_cultivo_32630.shx> -cwhere "parcela = '<Parcela>'" <ruta raster in> <ruta raster out>

Dicho comando se ha empleado en el script 'scripts/aplicarMascara.sh' para realizar este proceso de forma automática con los 1065 sub-rasters obtenidos en el paso anterior, correspondientes a sus respectivas parcelas. Como resultado, se obtienen otros 1065 sub-rasters adicionales, pero con valores nulos en las posiciones que se encuentren fuera de la máscara. Tras esta etapa, los rasters se asemejan a los mostrados en la figura 3.

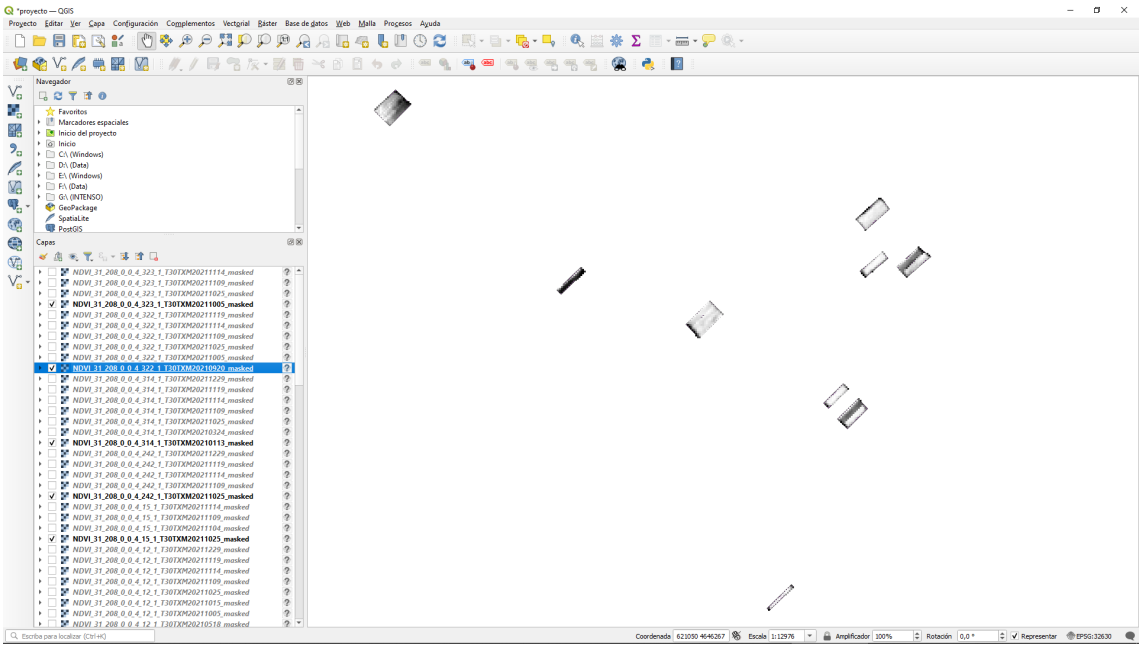


Figura 3: Proyecto QGIS que muestra los rasters ajustados a la máscara de cada geometría, superpuestos a la geometría de las parcelas que los contienen.

De manera opcional, es posible aumentar la resolución de los rasters para lograr una mejor coincidencia con la máscara de cada parcela. Para ello, se utiliza el siguiente comando:

gdalwarp.exe -tr <N1> <N2> <ruta raster in> <ruta raster out>

Este comando se ha empleado en el script 'scripts/aumentarResolucion.sh' con el propósito de convertir los píxeles de los rasters originales, que representan una escala de 10x10 metros en la realidad, a píxeles que representen una escala de N1xN2 metros en la realidad. Los nuevos rasters resultantes muestran una mejor adaptación a la geometría de la parcela, tal como se ilustra en la figura 4.

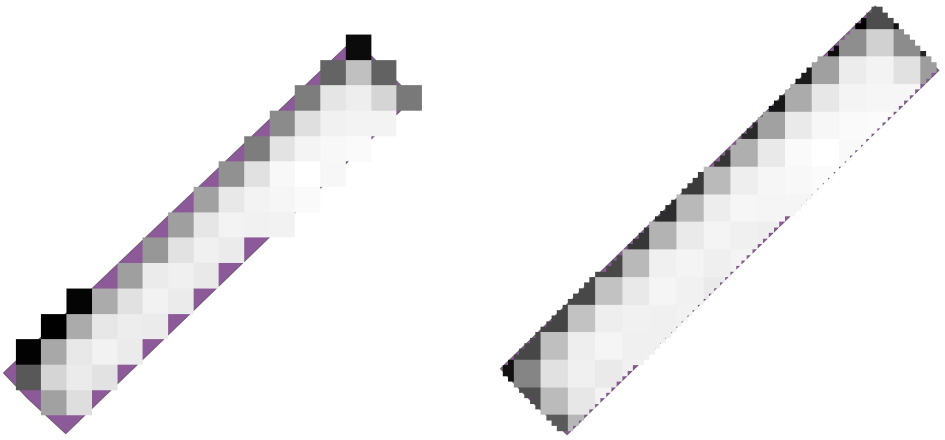


Figura 4: Comparación de la aplicación de la máscara de una parcela al raster en función de su resolución. Raster con una resolución de 10x10 a la izquierda y raster con una resolución de 2x2 a la derecha.

Con el fin de manejar estos datos, se han convertido a formato CSV siguiendo las columnas especificadas en la tabla 1. Este proceso se llevó a cabo mediante la ejecución del script 'scripts/obtenerDatosEntrenables3.sh', el cual ha utilizado el programa 'scripts/obtenerDatosEntrenables3.py' internamente.

fecha	ndvi	coordenadas	parcela
20210103	0.24740987	623060:4648100	31_208_0_0_2_32_1
20210103	0.2337043	623070:4648100	31_208_0_0_2_32_1
20210103	0.26550648	623050:4648110	31_208_0_0_2_32_1

Tabla 1: Formato del archivo 'csv/todasParcelas.csv'. Para cada combinación 'parcela'-'fecha', existen distintas coordenadas con valores de NDVI diferentes.

Resultados

Como resultado de esta iteración, se ha logrado obtener la información relevante de cada imagen de satélite asociada a los píxeles de las parcelas. Este proceso se ha realizado de forma escalable y reproducible mediante la utilización de scripts de shell y programas de Python que se integran mediante las funciones de línea de comandos de GDAL. Previo a esto, se ha llevado a cabo la visualización de las parcelas utilizando QGIS. De esta manera, se ha establecido un flujo de trabajo eficiente y automatizado para obtener y procesar los datos de forma sistemática.

La información recopilada ha sido el punto de partida para adentrarse en el estudio de la evolución de las parcelas a lo largo del año. A partir de estos datos, se han podido realizar suposiciones más complejas y trabajar con ellas en investigaciones posteriores.

2.2.2 Iteración 1

Hipótesis de partida

Utilizando los datos recopilados en la iteración anterior, se inicia el trabajo con Python y Jupyter Notebooks con el propósito de representar la información obtenida para obtener una mejor comprensión de sus características y determinar los pasos necesarios para alcanzar los objetivos.

Con el objetivo de predecir la fecha óptima de cosecha, se plantea la suposición de que el valor NDVI de las parcelas aumentaría gradualmente y, en algún momento, experimentaría un descenso abrupto, indicando la cosecha en una fecha específica, como se ilustra en la figura 5.

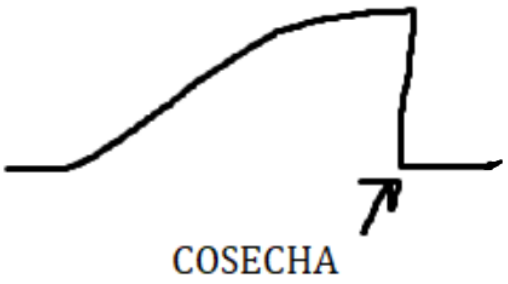


Figura 5: Concepto propio, previo a la investigación del desarrollo del NDVI para el cultivo del maíz, de la evolución del valor NDVI durante el ciclo de crecimiento y cosecha del cultivo, mostrando un aumento gradual seguido de un descenso abrupto al momento de la cosecha.

En relación a los objetivos de distinguir subparcelas de cultivos diferentes dentro de una misma parcela, se formulan las siguientes suposiciones iniciales.

Por un lado, se considera que distintos vegetales presentan características y períodos de cultivo y cosecha distintos. Por lo tanto, habría momentos del año en los que un cultivo alcanzaría su máximo valor de NDVI, mientras que otro mostraría un valor significativamente diferente.

Por otro lado, se plantea la hipótesis de que los distintos cultivos podrían tener temporadas de crecimiento diferentes, lo que implica que en diferentes períodos del año se cultivarán diferentes vegetales en las subparcelas, generando variaciones en los valores promedio de NDVI. Por ejemplo, en una parcela con dos subparcelas, A y B, podría suceder que durante los meses de invierno se cultive cereal en A y se deje en barbecho la subparcela B. Sin embargo, en los meses de verano se realizaría barbecho en A y se cultivaría maíz en B. Como resultado, los valores promedio de NDVI en cada mitad del año serían considerablemente diferentes, ya que una subparcela mantendría un valor bajo debido al barbecho, mientras que en la otra subparcela habría un valor más alto debido al cultivo de un vegetal.

Estas suposiciones iniciales son fundamentales para el análisis y la interpretación de los datos recopilados en el estudio.

Resultados

A lo largo de esta iteración, se han utilizado diferentes representaciones de los datos de las parcelas con el objetivo de verificar dos suposiciones previas y facilitar el análisis.

En cuanto a la primera hipótesis sobre la forma de la curva del NDVI, se ha observado un patrón distinto al esperado. En lugar de una curva continua, se han detectado numerosos días en los que el valor del NDVI es extremadamente bajo en todos los píxeles de la parcela. Se ha planteado la posibilidad de que estos resultados se deban a las condiciones meteorológicas a las que la teledetección satelital es sensible, a diferencia de otras como la radar, para capturar la información. Como resultado, se sugiere explorar la implementación de algoritmos u otros enfoques para suavizar la curva y reducir la presencia de datos anómalos.

La segunda hipótesis planteaba la posibilidad de identificar subparcelas con cultivos distintos dentro de una parcela basándose en los valores del NDVI de cada píxel. Las representaciones

realizadas han respaldado esta suposición, lo que sugiere que se podría continuar explorando este enfoque. Para avanzar en este objetivo, sería necesario obtener datos supervisados, es decir, datos etiquetados con la información correcta sobre los cultivos en cada subparcela. Esto permitiría comprender mejor cómo trabajar con estos datos y mejorar la precisión de las clasificaciones.

2.2.3 Iteración 2

Hipótesis de partida

Tras analizar los resultados anteriores, se toma la decisión de enfocarse exclusivamente en el objetivo de predecir la fecha óptima de cosecha. Además, se opta por trabajar únicamente con parcelas de maíz, basándose en la suposición de que este cultivo, al ser cosechado en seco y de manera completa en una sola operación, podría exhibir un comportamiento similar al representado en la figura 5 en cuanto a la evolución de su valor NDVI.

Con el fin de alcanzar este objetivo, se cuenta con la colaboración del cliente de GeosLab, que proporcionará información de numerosas parcelas de maíz en Aragón. Combinando esta información con las solicitudes a la API de datos geográficos y meteorológicos (desarrollo propio de Geoslab sobre sistemas abiertos de datos), se dispondrá de un amplio conjunto de datos para entrenar el modelo y permitirle predecir la fecha óptima de cosecha. Con esta nueva dirección, se descarta definitivamente cualquier exploración relacionada con la detección de zonas de cultivo dentro de una misma parcela.

Entre las parcelas proporcionadas, será interesante examinar las diferencias entre aquellas ubicadas cerca de la ribera del Ebro, y las parcelas de la provincia de Huesca, que se encuentran a mayor altitud. Estos cambios de localización podrían tener un impacto significativo en las fechas de recepción de las cosechas y en la duración de la temporada de alto NDVI del maíz en cada parcela.

Resultados

En esta iteración se ha tomado la decisión de enfocar el trabajo en la predicción exclusiva de la fecha óptima de cosecha para parcelas de maíz. Se ha considerado que, al tratarse de un cultivo que se cosecha cuando está seco y de forma repentina, es probable que exhiba un comportamiento similar al representado en la figura 5 en términos de la evolución de su valor NDVI.

A través de las peticiones a la API de GeosLab, se ha representado la curva del valor NDVI junto con las fechas de cosecha de distintas parcelas ubicadas cerca de la ribera de Zaragoza y en la provincia de Huesca, con el fin de comparar diferentes suposiciones relacionadas con la influencia de la humedad o altitud de las parcelas en las fechas de recepción, el período de alto valor NDVI y el valor máximo del NDVI de cada parcela.

Sin embargo, tras analizar los resultados obtenidos, no ha sido posible llegar a una conclusión definitiva sobre estas tres hipótesis, ya que se han observado casos variados en todos los

escenarios considerados. Por lo tanto, se confirma que las parcelas no se ven fuertemente afectadas por las condiciones de cada localización.

No obstante, lo que sí se ha observado es que todas las parcelas siguen un patrón de evolución del valor NDVI similar al mostrado en la figura 6, donde el NDVI aumenta rápidamente, se mantiene en niveles altos durante un tiempo y luego disminuye gradualmente. Además, la fecha de recepción tiende a ubicarse al final de la disminución, cuando el valor NDVI se estabiliza en torno a 0.2-0.3.

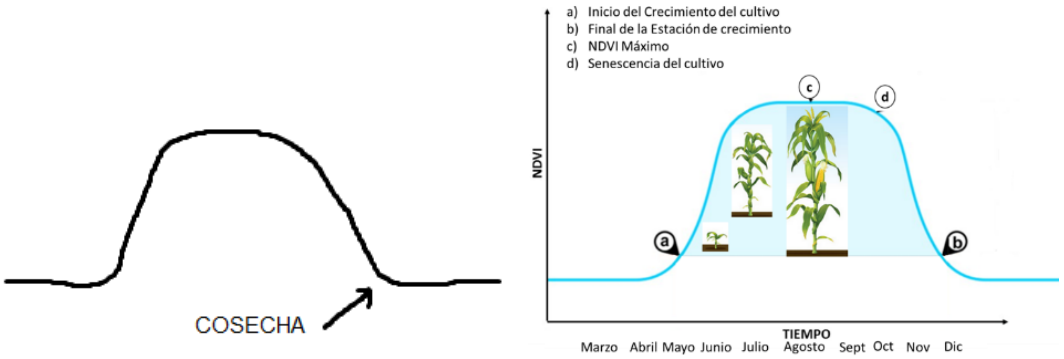


Figura 6: Esquema propio (a la izquierda) y teórico (a la derecha) de la evolución del NDVI observado en parcelas reales. Se ilustra un rápido incremento hasta alcanzar el valor máximo, seguido de un decrecimiento gradual, en contraposición a la suposición inicial de la figura 5. La fecha de cosecha suele coincidir con el final de la fase de decrecimiento, cuando la evolución se estabiliza en torno a valores de NDVI de 0.2-0.3. Fuente: Plataforma Geoweb para la Red de Desarrollo en Sustentabilidad Alimentaria. Disponible en: http://asam.centrogeo.org.mx/index.php/resultado-13?showall=1&limitstart=

2.2.4 Iteración 3

Hipótesis de partida

Para abordar las fluctuaciones en forma de picos que se observan en las gráficas de la evolución del NDVI, se plantea la posibilidad de desarrollar un algoritmo que permita eliminar estas fluctuaciones en forma de dientes de sierra y suavizar los puntos en las rectas, con el fin de obtener un valor más estable a lo largo del tiempo.

Resultados

Durante esta iteración se ha desarrollado un algoritmo de suavizado para corregir el comportamiento errático de los valores NDVI en las gráficas. El procedimiento seguido ha consistido en realizar pequeñas modificaciones, observar su impacto en las parcelas de observación seleccionadas, identificar nuevos problemas y continuar este proceso hasta que no se encontraran más errores.

Se ha desarrollado un algoritmo que ha demostrado obtener resultados satisfactorios, respaldando así la hipótesis planteada. A partir de ahora, este algoritmo se utilizará para

trabajar con los datos del NDVI, asegurando que el ruido presente no afectará los resultados futuros.

2.2.5 *Iteración 4*

Hipótesis de partida

Con el algoritmo actual que elimina errores y curvas inusuales de las parcelas, se puede comenzar a trabajar en la predicción de la fecha óptima de recepción utilizando diferentes modelos basados en Inteligencia Artificial, como las redes neuronales.

En esta iteración, se plantea la posibilidad de desarrollar un modelo de predicción que pueda determinar la fecha exacta de cosecha de un cultivo, utilizando el valor de NDVI de cada día a lo largo del año como datos de entrada y el día del año exacto (número del 1 al 365) como salida. Además, se considera la viabilidad de desarrollar tanto un modelo de clasificación como un modelo de regresión para este propósito.

Resultados

Como se ha mencionado a lo largo del trabajo realizado, se puede concluir que la hipótesis inicial no es válida. No se ha logrado desarrollar un modelo, ya sea de regresión o clasificación, capaz de predecir el momento exacto de la cosecha utilizando como dato de entrada el valor del NDVI en cada día de la curva de evolución.

2.2.6 *Iteración 5*

Hipótesis de partida

Se decide cambiar los datos de entrada con los que trabajar debido a los resultados insatisfactorios y a la limitación de utilizarlos en diferentes países, donde el mismo día del año puede corresponder a distintas estaciones y las fechas de recogida pueden variar debido a los cambios en las temperaturas. Se plantea trabajar con los datos en forma de series temporales, donde cada dato de entrada contiene información diaria, y se intenta predecir si se debe recoger (1) o no (0) el cultivo ese día. Esta aproximación permite incluir información de días anteriores en cada dato. Se proponen los siguientes formatos de datos de entrada para predecir la fecha óptima de cosecha:

- df_ts1: cada dato de entrada contiene únicamente el valor NDVI del día actual.
- df_ts2_X: cada dato de entrada contiene el valor NDVI del día actual 'i' y la diferencia de valor NDVI con cada uno de los X días anteriores. Es decir, en cada dato, el valor de la columna 'pendiente día - X' es: NDVI(i - x) - NDVI(i).
- df_ts3_X: cada dato de entrada contiene el valor NDVI del día actual 'i' y el valor NDVI de cada uno de los X días anteriores.

Se considera que se puede abordar el problema tanto como un problema de clasificación como de regresión. Además, se plantea reducir el desbalance entre las muestras de cada clase. Las muestras hasta 7 días antes de la fecha oficial de recogida se convertirían a la clase 1. En el caso de clasificación, esto equilibraría las clases. Si se trata de regresión, la salida de estas muestras convertidas aumentaría gradualmente hasta alcanzar el valor 1 en la muestra de recogida oficial.

Resultados

Durante esta iteración se han realizado tres pruebas de clasificación binaria y tres pruebas de regresión con el objetivo de verificar la hipótesis de si es posible crear un modelo de clasificación o regresión que prediga la fecha óptima de cosecha utilizando series temporales como datos de entrada.

En las pruebas de regresión, ha sido más evidente que las hipótesis eran incorrectas. El desequilibrio significativo entre las clases puede llevar a resultados engañosos, donde parece que el error es muy bajo, pero esto se ha debido a que todas las muestras de validación se etiquetaron con un valor cercano a cero.

En las pruebas de clasificación, especialmente en la segunda, se ha estado cerca de confirmar la hipótesis. Sin embargo, negar por completo la hipótesis no ha sido posible, ya que en esta prueba se ha logrado predecir correctamente al menos un día de la clase 1 en 21 de las 40 parcelas de validación disponibles, lo cual ha representado más de la mitad. No obstante, al examinar las parcelas, se ha observado que algunas predicciones incorrectas del modelo deberían haber sido correctas según las hipótesis iniciales, pero no lo han sido debido a que se cosecharon más tarde de lo habitual debido a decisiones individuales de cada agricultor.

2.2.7 Iteración 6

Hipótesis de partida

Tras evaluar los resultados de la iteración anterior, en la cual solo se obtuvo un resultado positivo en una de las tres pruebas de clasificación, se plantea la posibilidad de mejorar el rendimiento del modelo aumentando la cantidad de datos de entrenamiento. En concreto, se propone agregar un mes adicional de información correspondiente a enero del año siguiente. Esta adición de datos tiene como objetivo proporcionar al modelo una mayor cantidad de información temporal para mejorar su capacidad de predicción.

Resultados

Después de observar que la adición de más datos ha empeorado los resultados, se puede concluir que la hipótesis inicial ha sido incorrecta.

2.2.8 ITERACIÓN 7

Hipótesis de partida

Se plantea una nueva hipótesis después de las hipótesis anteriores incorrectas. Se considera la posibilidad de predecir la fecha óptima de cosecha utilizando datos más simplificados. Estos datos consisten en dos variables para cada día i: el valor NDVI del propio día i y la diferencia de valor NDVI con el día i-5 (NDVI(i-5) - NDVI(i)). Además, se considera como clase 1 tanto los cinco días anteriores como los cinco días siguientes al día de recogida oficial, con el objetivo de reducir el desbalance entre clases.

Resultados

Tras analizar los resultados desfavorables y observar que a medida que avanzaban los epochs, la precisión con los datos de entrenamiento se alejaba del 100%, se ha podido corroborar que las hipótesis planteadas eran incorrectas. Ha resultado evidente que el modelo no ha sido capaz de distinguir adecuadamente ni los datos de entrenamiento, lo cual es de vital importancia.

2.2.9 ITERACIÓN 8

Hipótesis de partida

En la iteración actual, se plantea la posibilidad de utilizar un nuevo formato de datos de entrada para predecir la fecha óptima de cosecha. Este formato incluye el valor NDVI del día actual (i), así como la pendiente calculada utilizando los valores NDVI de los días i-15, i-10 y i-5. La pendiente se obtiene utilizando la fórmula de la raíz de la hipotenusa: (y1 - y0) / (x1 - x0), donde y0 representa el valor NDVI del día i y x0 es el día correspondiente.

Además, en este planteamiento se decide utilizar únicamente la información de 20 días de cada parcela. Estos 20 días incluyen el día de recogida oficial, los 5 días siguientes y los 14 días anteriores. Para evitar el desbalance de clases, se establece que los 4 días anteriores a la fecha de recogida oficial se clasifiquen como clase 1. Esto resulta en una proporción de 5 elementos de clase 1 por cada 15 elementos de clase 0. Debido a esta proporción, se considera que la peor precisión aceptable para un modelo sería de 0.75, lo cual corresponde a predecir todos los datos como clase 0.

Resultados

Durante la iteración, se han realizado 7 pruebas diferentes para investigar la hipótesis de que utilizando 20 datos de cada parcela, compuestos por el valor NDVI del día i y las pendientes con respecto a los días i-15, i-10 y i-5, se podría predecir de manera precisa la fecha óptima de cosecha.

Estas pruebas han involucrado diversas técnicas, como agregar una columna de unos, expandir las variables y agregar columnas con relaciones entre las variables. También se ha probado el

modelo como un problema de regresión. Sin embargo, ninguna de estas pruebas ha proporcionado resultados que permitan respaldar la hipótesis planteada.

2.2.10 ITERACIÓN 9

Hipótesis de partida

La hipótesis planteada en esta iteración consiste en que al combinar el valor NDVI del día i, las pendientes con respecto a los días i-15, i-10 y i-5, junto con la información de la humedad relativa, precipitación, temperatura media y presión del día, se puede lograr una predicción más precisa de la fecha óptima de cosecha. Al igual que en la hipótesis anterior, se utilizan 20 días de cada parcela, clasificando los 4 días previos a la fecha oficial de recogida como clase 1, y los otros 15 días como clase 0. Para obtener estas 4 nuevas características, se requiere realizar peticiones a la API del cliente GeosLab.

Resultados

Durante esta iteración, se ha recopilado información sobre la temperatura media, precipitación y presión diaria para ser utilizada en diversos *dataframes*, los cuales han sido comparados para diferentes valores de regularización.

Al analizar los resultados obtenidos en esta iteración, no se ha podido afirmar ni descartar la hipótesis planteada. Si bien los resultados no han superado un umbral considerado suficiente para ser evaluados como éxito (un valor intermedio entre 0.80 y 0.85), se han observado mejoras en comparación con cualquier otra hipótesis previa. Esto sugiere que se podría estar en la dirección correcta.

2.2.11 ITERACIÓN 10

Hipótesis de partida

Tras las iteraciones previas, se identifican varios problemas que podrían estar afectando los resultados, como el desequilibrio de clases y la falta de normalización de los datos. Además, se sospecha que los resultados pueden estar influenciados por la selección específica de las parcelas de validación, lo que podría llevar a una separación de datos que ocasionalmente sea mejor o peor de lo esperado. Por lo tanto, se plantea la posibilidad de utilizar los mismos *dataframes* utilizados en la iteración anterior, junto con la implementación de validación cruzada, la corrección del desequilibrio de clases y la normalización de los datos de entrada. De esta manera, se espera desarrollar un modelo capaz de predecir la fecha óptima de cosecha de manera más precisa y confiable.

Resultados

A lo largo de esta iteración se han llevado a cabo múltiples pruebas y comparaciones exhaustivas con el fin de establecer una posición inicial sólida que respalde la hipótesis inicial.

Para evaluar estas comparaciones, se han utilizado dos métricas nuevas: una basada en el área bajo la curva ROC (ROC-AUC) y otra basada en la medida de éxito, con una ligera modificación que ha penalizado los modelos que tienden a clasificar todo como clase 1. Además, se han explorado diferentes modelos, tanto basados en redes neuronales como en otros enfoques, ya que lo más complejo no siempre garantiza un mejor rendimiento. Utilizando estas métricas y modelos, se han comparado distintas técnicas de balanceo de muestras, así como diferentes métodos de normalización de los datos y distintos dataframes. Posteriormente, se ha utilizado la biblioteca *Optuna* de Python para optimizar los resultados de las mejores pruebas, lo que finalmente ha llevado a obtener un modelo con una precisión del 99% en los datos de validación, confirmando satisfactoriamente la hipótesis inicial.

2.2.12 ITERACIÓN 11

Hipótesis de partida

Una vez se ha obtenido un modelo altamente eficaz que muestra un rendimiento óptimo, se plantea la posibilidad de mejorar el modelo mediante la eliminación de variables para reducir su complejidad, sin comprometer los resultados. Además, se plantea la idea de que este nuevo modelo puede realizar predicciones precisas sobre datos completamente desconocidos del año 2022, a pesar de haber sido entrenado exclusivamente con datos del año 2021.

Resultados

Basándose en el análisis realizado, se ha concluido que la hipótesis inicial no se sostiene. A pesar de haber obtenido un modelo simplificado con una reducción del 45.45% en características sin comprometer su rendimiento, los resultados obtenidos con los datos de test han sido inferiores a los esperados.

2.2.13 ITERACIÓN 12

Hipótesis de partida

Tras analizar los resultados insatisfactorios de la iteración anterior, se formulan las siguientes hipótesis. La primera hipótesis plantea que si se utiliza el modelo con las 11 columnas (sin eliminar las características menos influyentes según SHAP), este modelo sería capaz de predecir correctamente la fecha óptima de cosecha de las parcelas de test del año 2022. Se considera la posibilidad de que estas características sean relevantes en datos desconocidos, aunque no lo sean en datos conocidos del año 2021.

A continuación, se decide seguir la recomendación de SHAP y eliminar las características menos importantes, a excepción de la característica de precipitación. A pesar de que en los datos del año 2021, al tratarse de un año seco, esta característica no se considera relevante, se reconoce que debería ser una característica importante en general y no debería ser eliminada. La segunda hipótesis se plantea con el objetivo de evaluar si el modelo falló debido a un sobreajuste. Se supone que al utilizar estas 7 características y configurar los hiperparámetros

del modelo para evitar la especialización excesiva, el modelo será capaz de acertar los datos de test del año 2022.

En caso de que ambas hipótesis resulten incorrectas, se sugiere que los datos de ambos años son significativamente diferentes. Por lo tanto, se plantea la tercera hipótesis, la cual postula que al entrenar un modelo con datos de ambos años, dicho modelo será capaz de predecir correctamente las parcelas desconocidas tanto del año 2021 como del año 2022.

Trabajo desarrollado

Test con modelo de 11 columnas original

La eliminación de las características menos influyentes según SHAP posiblemente no ha afectado al modelo durante el entrenamiento y la validación, pero podría haber tenido un impacto al predecir nuevos datos completamente desconocidos, como los datos de test. Por lo tanto, se han repetido las tres pruebas de la iteración anterior utilizando conjuntos de parcelas de test diferentes. La primera prueba se ha realizado con el 100% de los datos de test (538 parcelas), la segunda con las parcelas de test cuya información del año anterior es conocida (42 parcelas), y la tercera con las parcelas de test que no son conocidas previamente (496 parcelas). Sin embargo, se ha vuelto a utilizar el modelo que contaba con las 11 columnas antes de eliminar las 5 menos influyentes según SHAP.

En la figura 7 se presentan los resultados de las tres pruebas distintas. Estos resultados han sido ligeramente mejores que los obtenidos en la iteración anterior, ya que todos muestran una mejora en la métrica TestAUC. Sin embargo, no han sido lo suficientemente buenos.

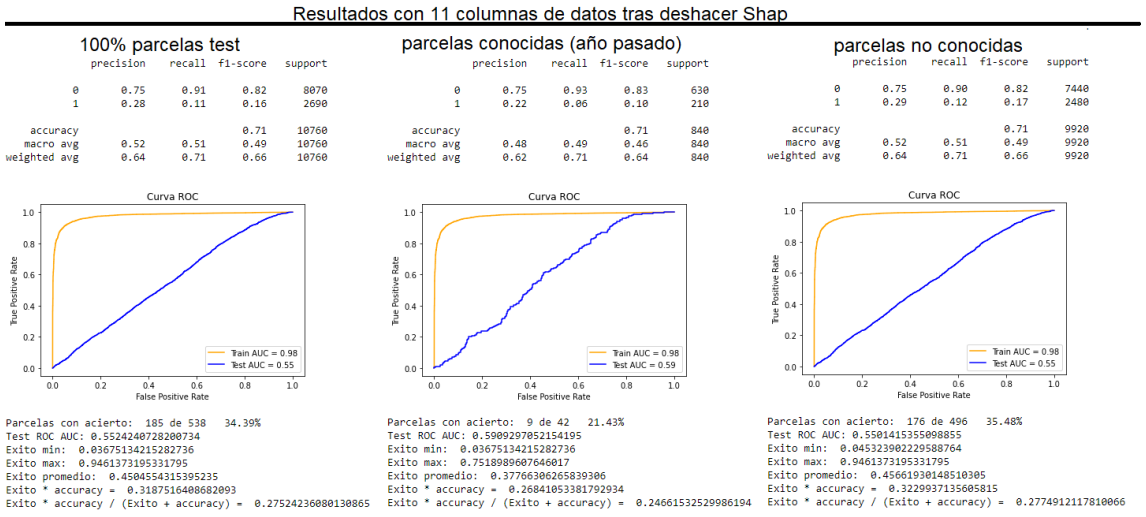


Figura 7: Resultados de los conjuntos de test del año 2022 y enero del año 2023 con el modelo inicial de 11 columnas, antes de utilizar SHAP para eliminar las 5 columnas menos influyentes. A la izquierda, prueba con 538 parcelas; en el centro, prueba con 42 parcelas conocidas del año anterior; a la derecha, prueba con 492 parcelas desconocidas.

Test con un modelo evitando el sobreajuste

También es posible que el modelo obtenido con *Optuna* haya estado demasiado ajustado a los datos de entrenamiento. Esto ha podido depender de diferentes parámetros del modelo, como *n_estimators* y *max_depth*. Por lo tanto, se ha buscado otro modelo utilizando *Optuna*, pero limitando estos dos hiperparámetros a valores no muy altos para evitar el sobreajuste.

De acuerdo con la hipótesis planteada, se ha mantenido la variable de precipitación a partir de ahora, a pesar de que SHAP la haya considerado de "poca influencia". Se ha considerado que esta variable es importante para predecir distintas parcelas en momentos secos o lluviosos.

Se ha repetido la prueba de la iteración anterior en la que se habían eliminado las columnas menos influyentes según SHAP, pero sin eliminar la de precipitación, y se ha añadido una restricción en los valores de los hiperparámetros *n_estimators* y *max_depth*, ya que se ha supuesto que son los dos más importantes para evitar la sobre-especialización del modelo.

Para determinar si el modelo anterior estaba sobre-especializado, se ha examinado el valor de todos sus parámetros. Ha resultado que este modelo tenía un valor de *max_depth* igual a -1, lo que indica que no había límite de profundidad en su árbol de decisión, lo que ha podido causar la sobre-especialización sin restricciones.

Por otro lado, el valor de *n_estimators* había sido de 100, lo cual se ha considerado un límite aceptable en este caso.

Por lo tanto, se ha añadido un rango de búsqueda en estos dos hiperparámetros y se ha buscado un nuevo modelo óptimo utilizando la misma metodología empleada en iteraciones anteriores. En la figura 8 se presentan los rangos de búsqueda y los valores obtenidos para cada modelo *LightGBM* utilizando *Optuna*.

		Modelo LightGBM					
		10 iteraciones (**max** TestAUC)		500 iteraciones (**max** TestAUC)		10 iteraciones (**media** TestAUC)	
	params	Rango búsqueda	Resultado	Rango búsqueda	Resultado	Rango búsqueda	Resultado
El número de árboles de decisión a construir durante el entrenamiento del modelo. Cuanto mayor sea este valor, más complejo será el modelo y mayor será el tiempo de entrenamiento.	n_estimators	[20 - 100]	84	[20 - 100]	82	[20 - 100]	82
Profundidad máxima de cada árbol de decisión. Controla la complejidad del modelo y evita el sobreajuste. Un valor más alto permitirá modelos más complejos, pero también aumenta el reisgo de sobreajuste. -1 equivale a que no hay límite de profundidad máxima.	max_depth	[1 - 10]	7	[1 - 10]	10	[1 - 10]	6
Función de pérdida a optimizar durante el entrenamiento. Puede ser de regresión, clasificación o ranking, como 'regression', 'binary', 'multiclass', 'lambdarank', entre otros.	objective	binary	binary	binary	binary	binary	binary
Métrica utilizada para evaluar el rendimiento del modelo durante el entrenamiento. Puede ser una métrica de regresión, clasificación o ranking, como 'rmse', 'auc', 'ndcg', entre otros.	metric	auc	auc	auc	auc	auc	auc
Determina la cantidad de información que se muestra durante el entrenamiento. 0 es el nivel más silencioso y 3 es el nivel más detallado. -1 evitará la visualización de información o mensajes por pantalla.	verbosity	-1	-1	-1	-1	-1	-1
Tipo de algoritmo de boosting a utilizar. Puede ser 'gbdt' para Gradient Boosting Decision Tree, 'dart' para Dropout Additive Regression Trees, 'goss' para Gradient-based One-Side Sampling, o 'rf' para Random Forest.	boosting_type	gbdt	gbdt	gbdt	gbdt	gbdt	gbdt
Términos de regularización que penalizan la complejidad del modelo. Ayudan a evitar el sobreajuste controlando el peso de los términos de regularización L1 y L2, respectivamente.	lambda_l1	[1e-8 - 10]	0.00023	[1e-8 - 10]	0.29406	[1e-8 - 10]	0.00044
	lambda_l2	[1e-8 - 10]	9.64370	[1e-8 - 10]	2.66640	[1e-8 - 10]	3.03151
Nº máximo de hojas en cada árbol.	num_leaves	[2 - 256]	192	[2 - 256]	115	[2 - 256]	40
Proporción de características (columnas) utilizadas para entrenar cada árbol. Un valor menor reduce la correlación entre los árboles y puede ayer a prevenir el sobreajuste.	feature_fraction	[0.1 - 1]	0.45479	[0.1 - 1]	0.97090	[0.1 - 1]	0.70406
Proporción de muestras a utilizar para cada iteración de bagging. Un valor menor reduce la varianza pero también puede llevar a un sesgo mayor.	bagging_fraction	[0.1 - 1]	0.49316	[0.1 - 1]	0.86668	[0.1 - 1]	0.68431
La frecuencia de uso de bagging. Controla la cantidad de iteraciones de bagging a realizar.	bagging_freq	[1 - 10]	8	[1 - 10]	9	[0 - 10]	4
Nº mínimo de muestras necesarias en un nodo terminal. Ayuda a controlar el crecimiento del árbol al evitar divisiones insignficantes	min_child_sample	[5 - 100]	25	[5 - 100]	5	[5 - 100]	74

Figura 8: Hiperparámetros utilizados en la optimización del modelo LightGBM para reducir el sobreajuste y recuperar la columna de precipitaciones después de aplicar SHAP para eliminar 5 columnas de datos. A la izquierda de cada hiperparámetro se encuentra una breve descripción

y posibles valores. A la derecha, se muestran los rangos de búsqueda en cada prueba junto con el valor utilizado en el mejor modelo obtenido.

En la figura 9 se muestran los mejores resultados. El mejor de los tres modelos ha sido el de enmedio, con un TestAUC de 0.99, que ha sido inferior al modelo previo a la reducción de sobre-especialización, pero sigue siendo muy bueno.

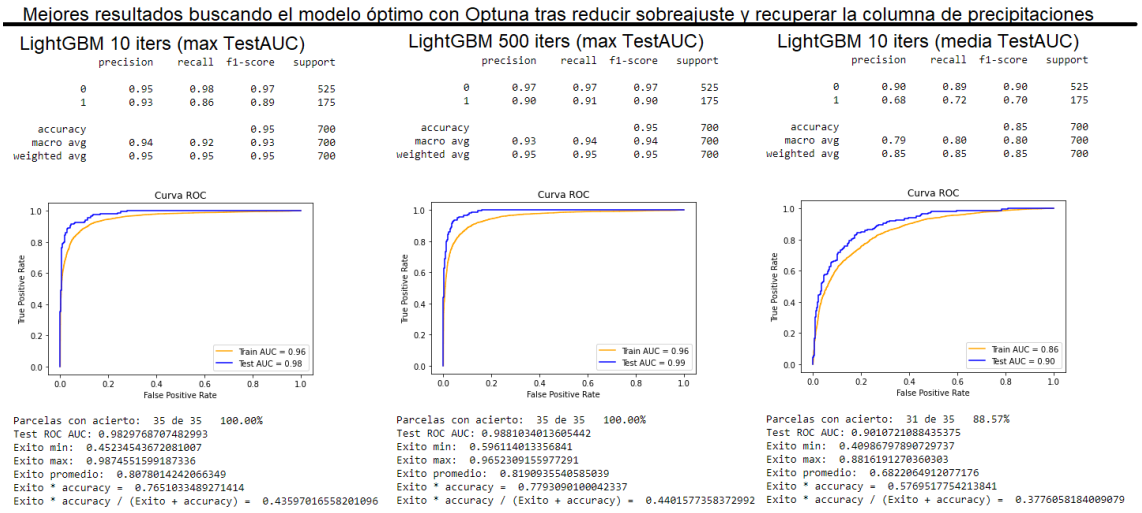


Figura 9: Informes con todas las métricas de los resultados obtenidos con el modelo LightGBM después de reducir el sobreajuste y recuperar la columna de precipitaciones, en comparación con las pruebas de la Figura 8. A la izquierda, mejor resultado de la prueba de 10 iteraciones; en el centro, mejor resultado de la prueba de 500 iteraciones; a la derecha, mejor resultado de la segunda prueba de 10 iteraciones.

Una vez obtenido otro modelo que funciona bien con los datos de entrenamiento y validación, se ha procedido a evaluarlo con los datos de test del año 2022 de la misma manera que en las iteraciones anteriores. El objetivo ha sido determinar si los malos resultados se debieron a la sobre-especialización del modelo con los datos del año 2021 o si los datos del año 2022 y 2021 han sido lo suficientemente distintos como para ser predecidos con solo un año de entrenamiento.

En la figura 10 se presentan estos resultados, los cuales han seguido siendo regulares o incluso malos. Aunque la prueba con las parcelas conocidas ha mostrado una ligera mejora en su métrica TestAUC, no ha sido suficiente para obtener conclusiones buenas.

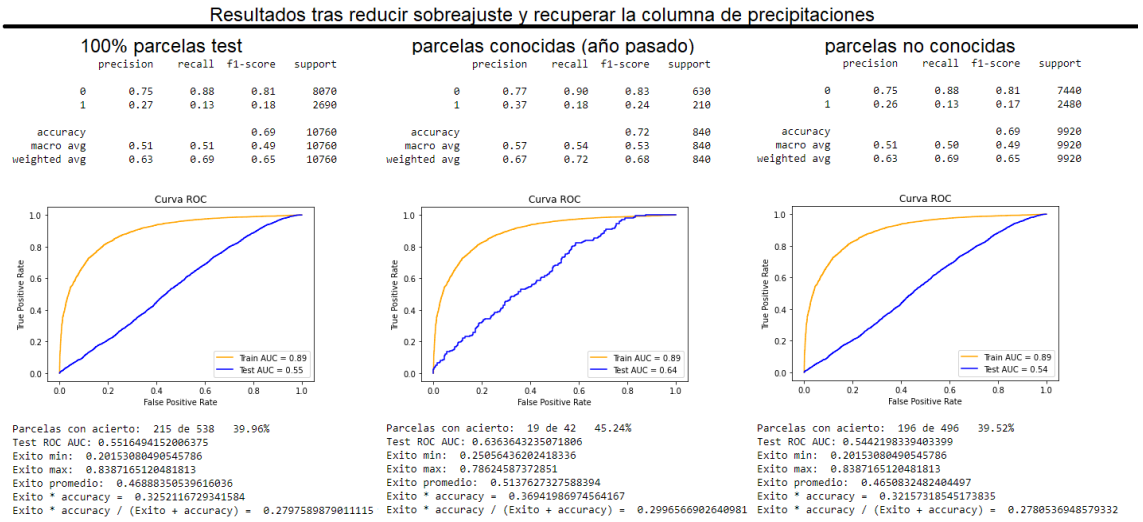


Figura 10: Resultados de los conjuntos de test del año 2022 y enero del año 2023 con el modelo después de reducir el sobreajuste y recuperar la columna de precipitaciones mediante SHAP. A la izquierda, prueba con 538 parcelas; en el centro, prueba con 42 parcelas conocidas del año anterior; a la derecha, prueba con 492 parcelas desconocidas.

Entrenamiento mezclado

Tras observar que las dos hipótesis anteriores eran incorrectas, se ha considerado que los datos del año 2022 no son adecuados para el modelo entrenado con los datos del año 2021, ya que las temperaturas y precipitaciones de cada año pueden haber sido muy diferentes. Por lo tanto, se ha decidido mezclar parcelas de ambos años en el entrenamiento para que el modelo pueda aprender de casos más diversos y adaptarse a datos muy distintos. Con el fin de validar este modelo, se han evaluado los resultados utilizando parcelas de ambos años juntos y por separado para determinar con qué conjunto de datos ha funcionado mejor.

Dado que se disponen de 346 parcelas del año 2021 y 538 parcelas del año 2022, se han seleccionado de forma aleatoria 192 parcelas del año 2022 para igualar el número de parcelas de cada año y evitar cualquier sesgo hacia un lado específico. Una vez se han obtenido estas 346 parcelas de cada año, se ha reservado un 10% de las parcelas de cada año para ser utilizadas como datos de test. Con el 90% restante de parcelas de cada año, se ha realizado un entrenamiento basado en *k-fold* con 10 pliegues, asegurando que tanto en los datos de entrenamiento como en los de validación hubiera el mismo número de parcelas de cada año en cada pliegue, manteniendo así una proporción constante del 50% de información de parcelas de cada año.

A continuación, se ha utilizado nuevamente *Optuna* en combinación con el modelo *LightGBM* para buscar el modelo óptimo dentro del rango de búsqueda de los diferentes hiperparámetros, que se detallan en la figura 11. Siguiendo la suposición anterior, en este entrenamiento se ha buscado reducir el sobreajuste y se ha mantenido la inclusión de la columna de precipitaciones, como se había hecho en la prueba anterior.

		Modelo LightGBM							
		10 iteraciones (**max** TestAUC)		500 iteraciones (**max** TestAUC)		10 iteraciones (**media** TestAUC)		500 iteraciones (**media** TestAUC)	
	params	Rango búsqueda	Resultado	Rango búsqueda	Resultado	Rango búsqueda	Resultado	Rango búsqueda	Resultado
El número de árboles de decisión a construir durante el entrenamiento del modelo. Cuanto mayor sea este valor, más complejo será el modelo y mayor será el tiempo de entrenamiento.	n_estimators	[20 - 100]	99	[20 - 100]	92	[20 - 100]	40	[20 - 100]	100
Profundidad máxima de cada árbol de decisión. Controla la complejidad del modelo y evita el sobreajuste. Un valor más alto permitirá modelos más complejos, pero también aumenta el reisgo de sobreajuste. -1 equivale a que no hay límite de profundidad máxima.	max_depth	[1 - 10]	5	[1 - 10]	10	[1 - 10]	8	[1 - 10]	10
Función de pérdida a optimizar durante el entrenamiento. Puede ser de regresión, clasificación o ranking, como 'regression', 'binary', 'multiclass', 'lambdarank', entre otros.	objective	binary	binary	binary	binary	binary	binary	binary	binary
Métrica utilizada para evaluar el rendimiento del modelo durante el entrenamiento. Puede ser una métrica de regresión, clasificación o ranking, como 'rmse', 'auc', 'ndcg', entre otros.	metric	auc	auc	auc	auc	auc	auc	auc	auc
Determina la cantidad de información que se muestra durante el entrenamiento. 0 es el nivel más silencioso y 3 es el nivel más detallado. -1 evitará la visualización de información o mensajes por pantalla.	verbosity	-1	-1	-1	-1	-1	-1	-1	-1
Tipo de algoritmo de boosting a utilizar. Puede ser 'gbdt' para Gradient Boosting Decision Tree, 'dart' para Dropout Additive Regression Trees, 'goss' para Gradient-based One-Side Sampling, o 'rf' para Random Forest.	boosting_type	gbdt	gbdt	gbdt	gbdt	gbdt	gbdt	gbdt	gbdt
Términos de regularización que penalizan la complejidad del modelo. Ayudan a evitar el sobreajuste controlando el peso de los términos de regularización L1 y L2, respectivamente.	lambda_l1	[1e-8 - 10]	0.00199	[1e-8 - 10]	1.76709	[1e-8 - 10]	0.00049	[1e-8 - 10]	0.00164
	lambda_l2	[1e-8 - 10]	0.97178	[1e-8 - 10]	1.90414	[1e-8 - 10]	1.53956	[1e-8 - 10]	3.11306
Nº máximo de hojas en cada árbol.	num_leaves	[2 - 256]	230	[2 - 256]	230	[2 - 256]	104	[2 - 256]	235
Proporción de características (columnas) utilizadas para entrenar cada árbol. Un valor menor reduce la correlación entre los árboles y puede ayer a prevenir el sobreajuste.	feature_fraction	[0.1 - 1]	0.64140	[0.1 - 1]	0.78847	[0.1 - 1]	0.49405	[0.1 - 1]	0.71018
Proporción de muestras a utilizar para cada iteración de bagging. Un valor menor reduce la varianza pero también puede llevar a un sesgo mayor.	bagging_fraction	[0.1 - 1]	0.62436	[0.1 - 1]	0.77594	[0.1 - 1]	0.70587	[0.1 - 1]	0.90991
La frecuencia de uso de bagging. Controla la cantidad de iteraciones de bagging a realizar.	bagging_freq	[1 - 10]	9	[1 - 10]	4	[0 - 10]	0	[1 - 10]	4
Nº mínimo de muestras necesarias en un nodo terminal. Ayuda a controlar el crecimiento del árbol al evitar divisiones insignficantes	min_child_sample	[5 - 100]	81	[5 - 100]	7	[5 - 100]	32	[5 - 100]	5

Figura 11: Hiperparámetros utilizados en la optimización del modelo LightGBM que mezcla datos del año 2021 y 2022, después de reducir el sobreajuste y recuperar la columna de precipitaciones utilizando SHAP. A la izquierda de cada hiperparámetro se encuentra una breve descripción y posibles valores. A la derecha, se muestran los rangos de búsqueda en cada prueba junto con el valor utilizado en el mejor modelo obtenido.

En la figura 12 se presentan los mejores resultados de cada prueba utilizando los datos de validación de ambos años. A diferencia de las pruebas anteriores, donde no se han observado diferencias significativas al variar entre 10 y 500 iteraciones, en este caso las pruebas de 500 iteraciones han mostrado resultados considerablemente mejores.

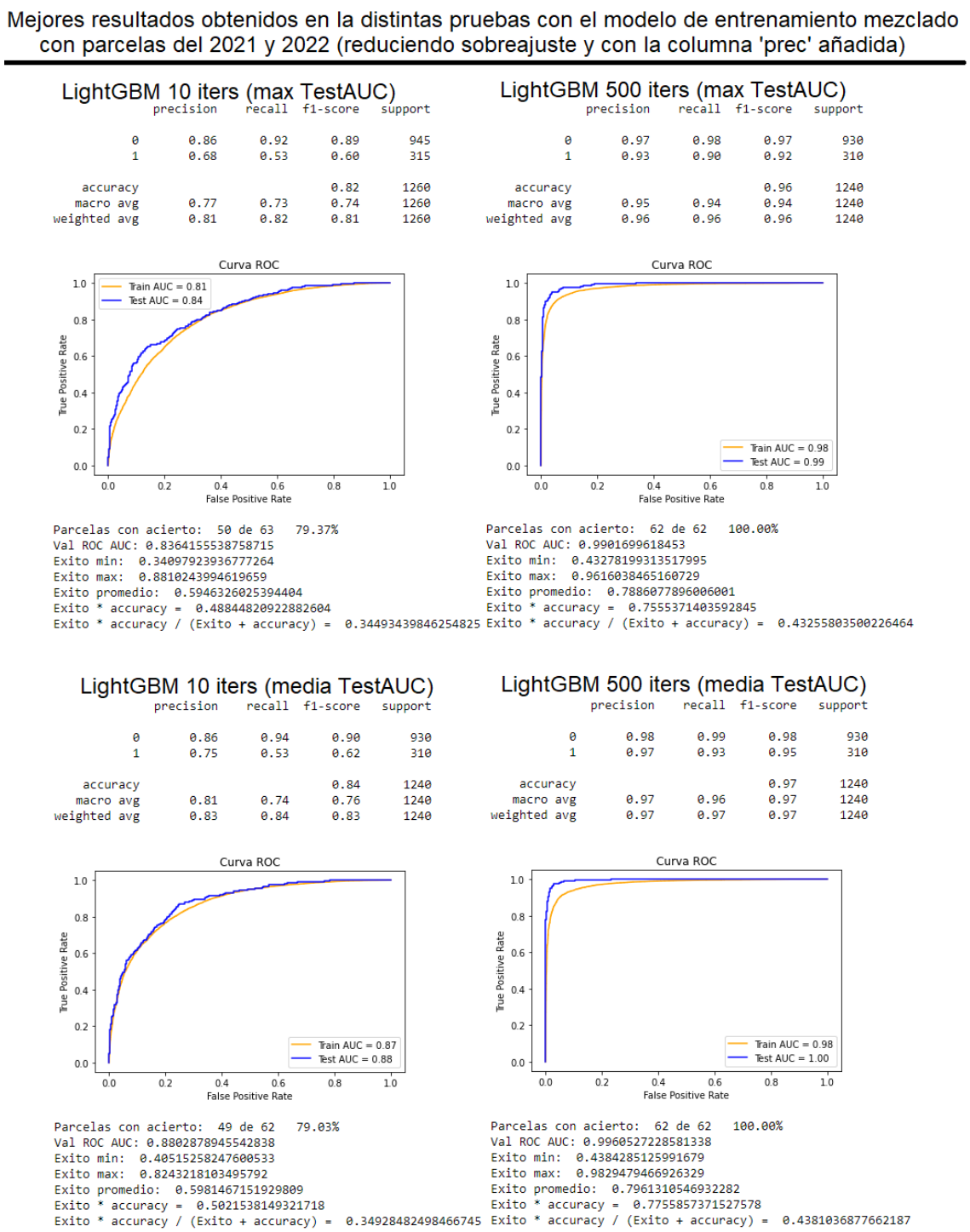


Figura 12: Informes con todas las métricas de los resultados obtenidos con el modelo LightGBM que utiliza datos mezclados del 2021 y 2022, después de reducir el sobreajuste y recuperar la columna de precipitaciones según la Figura 11. Los datos de validación contienen información de ambos años en igual proporción. Arriba a la izquierda, resultado de la prueba de 10 iteraciones; arriba a la derecha, resultado de la prueba de 500 iteraciones; abajo a la izquierda, resultado de la segunda prueba de 10 iteraciones; abajo a la derecha, resultado de la segunda prueba de 500 iteraciones.

Luego de obtener estos resultados, se ha evaluado el mejor modelo obtenido de estas cuatro pruebas utilizando el conjunto de datos de test que se había separado previamente, el cual ha contenido datos tanto del año 2021 como del año 2022. En la figura 13 se presentan los resultados globales, así como los resultados con solo los datos de test del año 2021 y del año 2022. Los resultados con las parcelas mezcladas han alcanzado un TestAUC de 0.81, lo cual ha sido significativamente mejor que todas las pruebas realizadas anteriormente. Aunque no predice todas las parcelas correctamente, ha acertado un número suficiente como para considerarse un buen modelo que obtiene resultados satisfactorios con datos totalmente desconocidos. En cuanto a las pruebas con datos de un solo año, parece funcionar mejor con las parcelas del año 2021, posiblemente debido a que es un año más seco y se ha dispuesto de más información sobre días secos (con "prec" igual a 0) que sobre días no secos (con "prec" distinto de 0). A pesar de esto, los resultados con parcelas de años separados también se pueden considerar buenos.

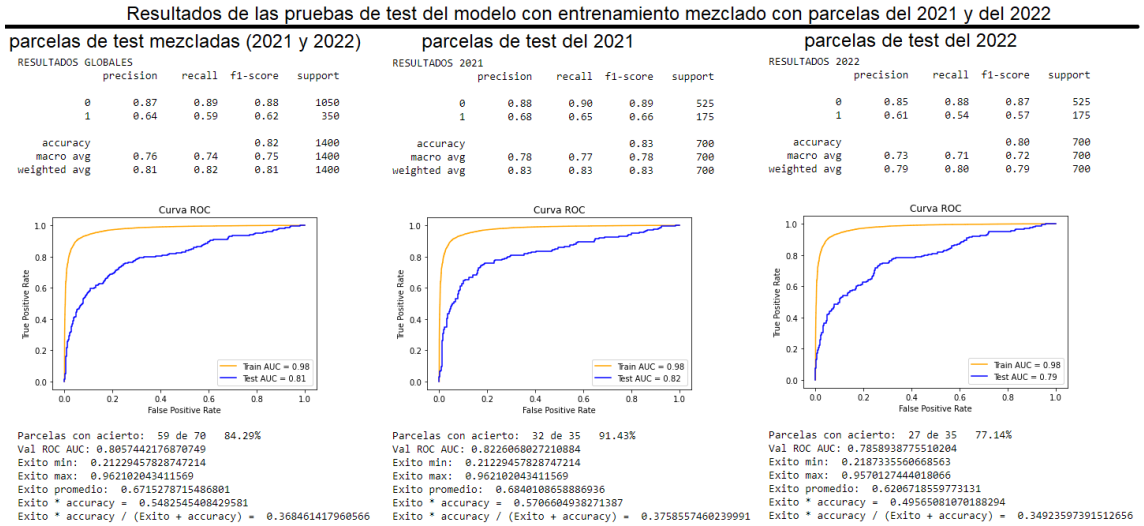


Figura 13: Resultados de los conjuntos de test con años mezclados y por separado. El modelo ha sido entrenado con datos de parcelas de ambos años en igual proporción, con el objetivo de reducir el sobreajuste y volver a incluir la columna de precipitaciones tras su eliminación en las pruebas con SHAP. A la izquierda, prueba con datos de test de ambos años; en el centro, prueba con datos de test solamente del año 2021; a la derecha, prueba con datos de test solamente del año 2022.

Resultados

Durante el proceso de esta iteración, se han llevado a cabo pruebas para validar tres hipótesis iniciales diferentes. La confirmación o refutación de estas hipótesis ha contribuido a comprender las características de los datos de cada año, lo que ha explicado los resultados obtenidos hasta el momento.

Tanto la primera como la segunda hipótesis han resultado ser incorrectas, lo que casi con certeza indica que los años 2021 y 2022 presentan diferencias significativas. Esto explica por qué el modelo entrenado con los datos del 2021 no ha podido distinguir los datos del 2022.

En anticipación a estos resultados, se había planteado la tercera hipótesis, la cual ha sido confirmada. Se han logrado obtener buenos resultados al entrenar un modelo utilizando datos equitativos de los años 2021 y 2022, lo que ha permitido realizar predicciones precisas en parcelas desconocidas de ambos años.

2.3 Dimensión del trabajo realizado

Tal y como se ha mostrado a lo largo de esta memoria, este trabajo se ha movido continuamente en la indefinición y la incertidumbre propia de las actividades de mayor carga de investigación. En este sentido, se propuso ir avanzando iteración a iteración mientras nos movíamos en la dimensión de un esfuerzo total acordado y soportado por una beca de colaboración. De este modo, resultaba posible parar el proyecto al final de cualquier iteración mostrando los resultados obtenidos hasta entonces. En la tabla 2 se presenta un resumen de los esfuerzos dedicados a cada una de las iteraciones.

Iteración	Fecha de inicio	Fecha de finalización	Esfuerzo (horas)
0	25/03/2022	05/05/2022	20 horas
1	05/05/2022	28/06/2022	25 horas
2	28/06/2022	21/07/2022	15 horas
3	21/07/2022	22/08/2022	25 horas
4	22/08/2022	20/09/2022	10 horas
5	20/09/2022	15/11/2022	80 horas
6	15/11/2022	22/11/2022	15 horas
7	22/11/2022	28/11/2022	20 horas
8	28/11/2022	10/01/2023	230 horas
9	10/01/2023	14/02/2023	125 horas
10	14/02/2023	15/03/2023	40 horas
11	15/03/2023	25/04/2023	20 horas
12	25/04/2023	02/05/2023	20 horas
Total	25/03/2022	02/05/2023	645 horas

Tabla 2: Resumen de los esfuerzos dedicados a cada iteración.

2.4 Problemas encontrados

Durante el desarrollo de este proyecto, se han encontrado diversos desafíos que han requerido soluciones específicas. Uno de los problemas iniciales ha sido la falta de información real sobre la división de las parcelas en subparcelas con diferentes cultivos, lo cual hizo inviable la primera hipótesis planteada debido a la necesidad de contar con datos supervisados. Para superar esta limitación, se ha tomado la decisión de descartar dicha hipótesis y enfocarse en otras áreas de investigación que disponían de datos adecuados.

Otro obstáculo identificado ha sido la presencia de fechas en los datos de teledetección donde los valores del NDVI eran inesperadamente bajos en lugar de seguir el patrón esperado. Para abordar esta situación, se ha implementado un algoritmo de suavizado que ha permitido corregir las anomalías y obtener resultados más coherentes y confiables. No obstante, se descubrió que un sexto paso en el algoritmo de suavizado, que funcionaba bien con las parcelas de observación, presentaba dificultades al ser aplicado a todos los datos. En consecuencia, se ha decidido revertir este paso y se ha optado por mantener los cinco pasos que se detallan en la memoria del proyecto.

Además, se han experimentado dificultades con los dataframes en ciertas iteraciones debido a errores en las peticiones a la API de GeosLab, ya sea por formulaciones incorrectas o la falta de resultados esperados. Para superar estos obstáculos, se ha repetido el proceso de recolección de datos, llevando a cabo verificaciones más exhaustivas en las solicitudes y realizando comprobaciones adicionales para asegurar que los datos obtenidos sean correctos.

La superación de estos desafíos ha demostrado la capacidad de adaptación y resolución de problemas en el desarrollo del proyecto. Cada obstáculo enfrentado ha proporcionado aprendizajes valiosos y ha contribuido a mejorar la calidad de la investigación.

3 CONOCIMIENTOS ADQUIRIDOS Y CONCLUSIONES

3.1 Conocimientos adquiridos

Durante mi carrera, he tenido la oportunidad de cursar asignaturas que considero clave para el desarrollo de este TFG como son 'Inteligencia Artificial', 'Aprendizaje Automático' y 'Visión por computador'. Estas asignaturas me proporcionaron una base sólida en los principios y técnicas fundamentales de la Inteligencia Artificial y el aprendizaje automático. En particular, he aplicado mis conocimientos a la hora de conocer distintos métodos de aprendizaje automático como las redes neuronales, además de hacerme consciente de conceptos adicionales, como la regularización, la normalización, las validaciones cruzadas y distintas maneras de evaluación para medir el éxito y el error de mis resultados. Asimismo, estas asignaturas me han dotado de diversas formas de representar datos, lo cual ha sido fundamental para llevar a cabo análisis significativos en mi trabajo.

Durante mi trabajo realizado y la redacción de esta memoria, he desarrollado habilidades específicas que han sido fundamentales para el éxito del proyecto. Mi dominio de Python ha mejorado significativamente, especialmente en la implementación de modelos de predicción y análisis de datos utilizando la biblioteca *pandas*. He adquirido experiencia en el desarrollo de código escalable y replicable, adoptando enfoques sistematizados en lugar de soluciones ad-hoc, lo cual ha sido aplicado en los scripts y programas Python desarrollados para optimizar la eficiencia del trabajo. Respecto al tema específico del TFG, he adquirido conocimientos relacionados con el mundo de la agricultura y la teledetección, como el valor NDVI, los cultivos de 1º y 2º cosecha, así como los métodos de recolección para distintos cultivos estacionales, con un enfoque particular en el maíz y su ciclo completo de crecimiento y secado. Por último, he perfeccionado mi capacidad de redacción al elaborar un documento técnico que debe ser usado por profesionales más allá de mi trabajo en este TFG. En este sentido, he abordado un especial esfuerzo para garantizar la coherencia y profesionalidad en la presentación de los resultados en esta memoria.

Respecto a competencias transferibles, he mejorado mi capacidad de comunicación al participar en reuniones donde he presentado y discutido los resultados obtenidos. Esto me ha permitido mejorar mi capacidad de transmisión de ideas y conclusiones que quería comunicar sobre el trabajo realizado. Además he mejorado mi habilidad para gestionar el tiempo al planificar el trabajo con antelación para llegar puntualmente a las reuniones con todo el trabajo hecho.

3.2 Ideas Futuras

El trabajo realizado podría servir como punto de partida para futuras investigaciones y desarrollos industriales. A todos los efectos, este TFG se ha constituido como demostración de la viabilidad de esta aproximación de cara a poder desarrollar servicios y productos a clientes de la empresa. En este sentido, en esta sección se avanzan algunas líneas de trabajo en las que

poder seguir profundizando, tanto a nivel más elucubrativo, como dando pasos efectivos hacia la industrialización de resultados.

Una propuesta que considero interesante es seguir reentrenando el modelo final utilizando nueva información cada año. Esto permitiría perfeccionar su capacidad de predicción al considerar datos de distintos años y condiciones climáticas (como humedad, sequedad, temperatura, etc.). Esto posibilita hacer predicciones precisas en años futuros sin requerir información previa.

Además, resulta necesario desarrollar un modelo o algoritmo capaz de suavizar las curvas de las gráficas de NDVI sin depender de tener todos los datos del año. Actualmente, el algoritmo de suavizado depende de tener todos los datos del año para obtener resultados precisos. Sin embargo, sería valioso investigar y diseñar un modelo de predicción que pueda suavizar las curvas diarias del NDVI sin contar con información futura. Esto facilitaría la predicción diaria de la fecha óptima de cosecha sin la necesidad de tener los datos de todo el año, algo que no siempre es posible tener.

Finalmente, creo que sería muy interesante poder transferir esta experiencia a otros cultivos con patrones similares de crecimiento. Los candidatos más inmediatos son otros cereales (trigo, cebada, etc), pero también leguminosas y cultivos proteicos (guisantes, soja, etc). La problemática de cosechado no es tan crítica desde el punto de vista de la gestión de procesado del cultivo, pero facilitaría planificaciones de trabajos cuando hay que optimizar recursos de cooperativas y grandes empresas.

3.3 Conclusiones

Desde el inicio del proyecto, se han planteado dos hipótesis generales. La primera hipótesis plantea la posibilidad de distinguir las zonas con distintos cultivos y el tipo de vegetal cultivado en cada subparcela identificada dentro de una misma parcela. Sin embargo, resultó ser muy ambiciosa y se determinó que no se puede cumplir.

Por otro lado, la segunda hipótesis plantea la posibilidad de predecir la fecha óptima de cosecha de distintos cultivos, la cual sí se puede cumplir. Para llegar a afirmar esta segunda hipótesis, se ha operado durante 12 iteraciones en las que se han experimentado avances y retrocesos, debido a diversas hipótesis erróneas sobre combinaciones de modelos de predicción y *dataframes* que no han alcanzado los resultados esperados. Finalmente, se ha logrado desarrollar un modelo basado en Gradient Boosting, el cual ha sido entrenado con datos de alrededor de 700 parcelas de la zona de estudio durante dos años (2021 y 2022). Este modelo ha logrado alcanzar una precisión del 83% en los datos de test, prediciendo correctamente la fecha óptima de cosecha en hasta el 91.43% de las parcelas de test utilizadas. Esto evidencia su capacidad para predecir de manera efectiva la fecha óptima de cosecha en distintas parcelas, adaptándose a las diversas condiciones que presentan cada año.

Además de los logros técnicos, se ha dedicado un esfuerzo considerable en la documentación detallada de cada iteración del proyecto. Esto incluye programas, scripts, conjuntos de datos y

configuraciones de modelos, que se encuentran disponibles en el repositorio https://drive.google.com/drive/folders/1NvO70CVqZV-eoO9YlTlCpLJ53SAjt9u9?usp=drive_link. Todos los archivos CSV mencionados a lo largo del informe están disponibles en dicho repositorio, junto con los programas correspondientes para su creación y uso. Esta documentación exhaustiva no solo garantiza la transparencia y reproducibilidad de los resultados, sino que también proporciona una sólida base para futuros desarrollos y aplicaciones industriales en el campo de la predicción de cosechas.

Finalmente, a modo de conclusión más de carácter personal, el desarrollo de este TFG me ha permitido desafiarme a mí mismo al trabajar en colaboración con un equipo profesional en la formulación de objetivos a medio y largo plazo. Esta experiencia ha sido fundamental para adquirir habilidades de organización eficiente de tareas y para lidiar exitosamente con el estrés asociado a la entrega de resultados. Me siento enriquecido a nivel personal, ya que esta oportunidad me ha proporcionado herramientas sólidas para enfrentar con ánimo futuros desafíos profesionales.

4 Anexo I. Teledetección y datos disponibles

Anexo I.I. Índices de teledetección

Tal y como se explica en la Wikipedia[16], la teledetección o detección remota es la adquisición de información a pequeña o gran escala de un objeto o fenómeno, ya sea usando instrumentos de grabación o instrumentos de escaneo en tiempo real inalámbricos o que no están en contacto directo con el objeto (como por ejemplo aviones, satélites, astronave, boyas o barcos). En la práctica, la teledetección consiste en recoger información a través de diferentes dispositivos de un objeto concreto o un área. Por ejemplo, la observación terrestre o los satélites meteorológicos, las boyas oceánicas y atmosféricas, las imágenes por resonancia magnética, la tomografía por emisión de positrones, los rayos-X y las sondas espaciales son todos ejemplos de teledetección. Actualmente, el término se refiere, de manera general, al uso de tecnologías de sensores para adquisición de imágenes, incluyendo: instrumentos a bordo de satélites o aerotransportados, usos en electrofisiología, y difiere de otros campos relacionados con imágenes como, por ejemplo, el de la imagen médica.

La teledetección trabaja siguiendo el principio del problema inverso. Mientras que el objeto o fenómeno en cuestión (el estado) no se van a medir de manera directa, existen otras variables que se detectan y miden (la observación), que están intrínsecamente relacionadas con el objeto de interés, a través de un modelo creado por ordenador. Una analogía para entender esto es tratar de determinar el tipo de animal por sus pisadas. Así, por ejemplo, aunque es imposible medir directamente la temperatura en las capas altas de la atmósfera, sí es posible medir las emisiones de un cierto espectro de especies químicas conocidas (CO2) en esa región. La frecuencia de dicha emisión se puede relacionar con la temperatura de esa zona a través de varias relaciones termodinámicas.

La teledetección ha cobrado gran relevancia en el desarrollo de la agricultura de precisión ya que permite un diagnóstico de la situación de las explotaciones de manera conjunta y sectorizada por zonas. Para ello han cobrado gran relevancia los avances tecnológicos conceptuales y de herramientas de procesado. Esto ha asentado unas bases para el desarrollo de productos comerciales de relativamente fácil construcción, pero gran utilidad. Por ejemplo, el Índice de vegetación de diferencia normalizada, también conocido como NDVI por sus siglas en inglés[17], es un índice usado para estimar la cantidad, calidad y desarrollo de la vegetación en base a la medición, por medio de sensores remotos instalados comúnmente desde una plataforma espacial, de la intensidad de la radiación de ciertas bandas del espectro electromagnético que la vegetación emite o refleja. Se trata de un índice tan extendido que, incluso, existen ya sensores remotos, de superficie, o utilizados por drones y el software que calcula el NDVI directamente.

[16] https://es.wikipedia.org/wiki/Teledetección
[17] https://es.wikipedia.org/wiki/%C3%8Dndice_de_vegetaci%C3%B3n_de_diferencia_normalizada

Anexo I.II. Programa Copernicus y datos de los satélites Sentinel-2

El programa Copernicus (Copérnico)[18], anteriormente llamado "Global Monitoring for Environment and Security" (GMES), es un proyecto dirigido conjuntamente por la Agencia Espacial Europea (ESA) y por la Unión Europea a través de la Agencia Europea de Medio Ambiente, que pretende lograr una completa, continua y autónoma capacidad de observación terrestre de alta calidad cuyos resultados sean accesibles libremente por la comunidad científica o cualquier otra persona interesada. El objetivo general es proveer de información exacta, fiable y continua, para, entre otras cosas, mejorar la gestión y conservación del medio ambiente, comprender y mitigar los efectos del cambio climático y asegurar la seguridad civil. Pretende agrupar diferentes fuentes de información de satélites medioambientales y bases terrestres para proporcionar una visión global del "estado de salud" de la Tierra.

Dentro de Copernicus, destaca el programa Sentinel[19], proyecto multi-satélite, que comenzó en 2014 con el lanzamiento de los satélites Sentinel-1A en 2014, y que en la actualidad cuenta con 7 satélites en órbita (Sentinel-1A, Sentinel-1B, Sentinel-2A, Sentinel-2B, Sentinel-3A, Sentinel-3B y Sentinel-5P), y tiene pendiente el despliegue de otros 4 satélites más.

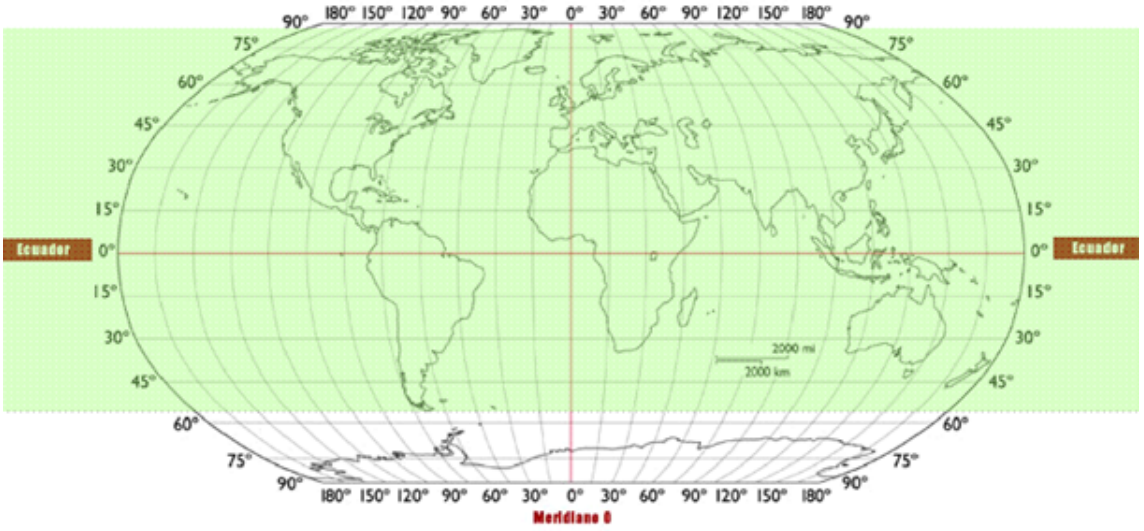


Figura 14: Cobertura global de Sentinel-2

Dentro del programa Sentinel, resultan especialmente interesantes para los trabajos de teledetección los datos proporcionados por los satélites de la constelación Sentinel-2. Éstos suministran imágenes multiespectrales con 13 bandas en el espectro visible, en el infrarrojo cercano e infrarrojos de onda corta, además del espectro electromagnético. Los dos satélites ofrecen cobertura global sistemática de la tierra entre los 56° S y los 84° N[20].

[18] https://es.wikipedia.org/wiki/Programa_Copérnico
[19] https://es.wikipedia.org/wiki/Sentinel_(satélite)
[20] https://www.esa.int/Space_in_Member_States/Spain/Datos_de_Sentinel

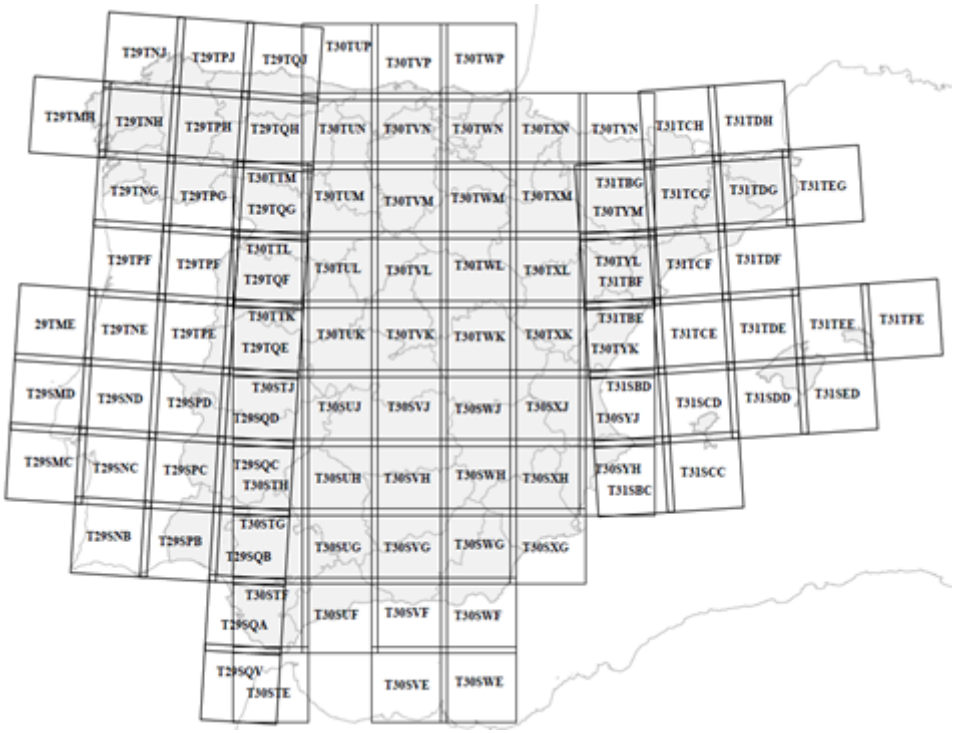


Figura 15: Distribución de la cobertura Sentinel-2 en la Península Ibérica (Usos 29, 30 y 31 UTM)[21]

La alternancia de los dos satélites permite tener datos cada 5 días, con resoluciones que van (según sensores) desde los 10 m/pixel a los 60 m/pixel. Y, posiblemente lo más relevante, con una política de publicación libre y en abierto en cuanto son procesados por el segmento tierra. Esto permite poder acceder a la información aproximadamente al día siguiente de ser capturada.

Geoslab dispone de una cadena de producción de información que se conecta a los servidores del programa Copernicus, descarga los datos que se establecen en un sistema de parametrización del proceso, y procede a calcular diferentes índices de teledetección entre los que se encuentra el NVDI usado en este proyecto. El sistema está en marcha desde mayo de 2020 y se ejecuta cada 5 días, en coordinación con la disponibilidad de nuevos datos de Sentinel-2. Esta cadena de producción oferta una API para la integración en productos y proyectos de la empresa que es la que se ha utilizado en este TFG.

Anexo I.III. Acceso a datos meteorológicos

Aunque en ocasiones no son considerados como datos meteorológicos, la realidad es que los datos recogidos por las agencias medioambientales relativos a temperaturas, precipitaciones, presión atmosférica, etc, son, a todos los efectos, datos de teledetección. Desde hace varios años, Geoslab dispone de un servicio que actúa de proxy y adaptador para poder recopilar datos meteorológicos de diferentes agencias, tanto nacional (AEMET) como de Comunidades Autónomas. Este servicio (cuyo acceso vía API ha sido usado en este TFG) permite determinar

[21] https://pnt.ign.es/acceso-a-imagenes

la estación meteorológica más cercana a un elemento geográfico (en este caso la geometría que representa a una parcela de maíz) para poder recuperar datos históricos de la misma.

5 ANEXO II. TÉRMINOS

Anexo II.I. SIGPAC

El Sistema de Información Geográfica de Parcelas Agrícolas, SIGPAC, permite identificar geográficamente las parcelas declaradas por los agricultores y ganaderos. Fue concebido con el propósito de facilitar a los agricultores la presentación de solicitudes, con soporte gráfico, así como facilitar los controles administrativos y sobre el terreno[22].

Este código está formado de 7 números, como en el siguiente ejemplo: "31_208_0_0_2_106_1". Cada número significa lo siguiente:

- Provincia: El código de la provincia española. En este caso, el código 31 se refiere a la provincia de Navarra.
- Municipio: El código del municipio dentro de la provincia. En este caso, el código 208 se refiere al municipio de Tudela.
- Agregado: Este 0 se refiere al agregado municipal al que pertenece la parcela, que es una unidad territorial menor que el municipio.
- Zona: Este otro 0 se refiere a la zona geográfica en la que se encuentra la parcela.
- Polígono: Este número 2 se refiere a la unidad de gestión de las parcelas agrícolas en la que se divide el territorio.
- Parcela: Este número 106 se refiere a la parcela agrícola individual dentro de un polígono.
- Recinto: Este número 1 se refiere a la unidad mínima de cultivo que se puede identificar en una parcela, que suele corresponder a una parcela o una parte de ella.

22 https://www.mapa.gob.es/es/agricultura/temas/sistema-de-informacion-geografica-de-parcelas-agricolas-sigpac-/default.aspx

6 Anexo III. Detalles de las iteraciones 1 a 11

Tal y como se ha indicado al inicio de la sección 2.2, en este anexo se procede a detallar todos los trabajos hechos en el marco de las iteraciones 1 a 11.

Anexo III.I. Iteración 1

Hipótesis de partida

Utilizando los datos recopilados en la iteración anterior, se inicia el trabajo con Python y Jupyter Notebooks con el propósito de representar la información obtenida para obtener una mejor comprensión de sus características y determinar los pasos necesarios para alcanzar los objetivos.

Con el objetivo de predecir la fecha óptima de cosecha, se plantea la suposición de que el valor NDVI de las parcelas aumentaría gradualmente y, en algún momento, experimentaría un descenso abrupto, indicando la cosecha en una fecha específica, como se ilustra en la figura 5.

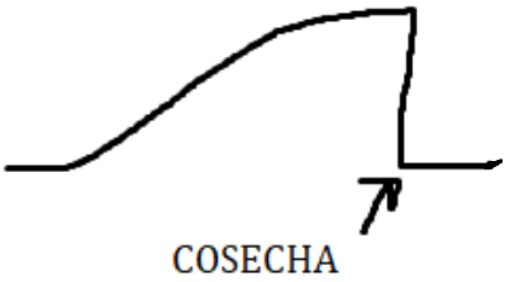


Figura 5: Concepto propio, previo a la investigación del desarrollo del NDVI para el cultivo del maíz, de la evolución del valor NDVI durante el ciclo de crecimiento y cosecha del cultivo, mostrando un aumento gradual seguido de un descenso abrupto al momento de la cosecha.

En relación a los objetivos de distinguir subparcelas de cultivos diferentes dentro de una misma parcela, se formulan las siguientes suposiciones iniciales.

Por un lado, se considera que distintos vegetales presentan características y períodos de cultivo y cosecha distintos. Por lo tanto, habría momentos del año en los que un cultivo alcanzaría su máximo valor de NDVI, mientras que otro mostraría un valor significativamente diferente.

Por otro lado, se plantea la hipótesis de que los distintos cultivos podrían tener temporadas de crecimiento diferentes, lo que implica que en diferentes períodos del año se cultivarán diferentes vegetales en las subparcelas, generando variaciones en los valores promedio de NDVI. Por ejemplo, en una parcela con dos subparcelas, A y B, podría suceder que durante los meses de invierno se cultive cereal en A y se deje en barbecho la subparcela B. Sin embargo, en los meses de verano se realizaría barbecho en A y se cultivaría maíz en B. Como resultado, los valores promedio de NDVI en cada mitad del año serían considerablemente diferentes, ya que una subparcela mantendría un valor bajo debido al barbecho, mientras que en la otra subparcela habría un valor más alto debido al cultivo de un vegetal.

Estas suposiciones iniciales son fundamentales para el análisis y la interpretación de los datos recopilados en el estudio.

Trabajo desarrollado

En el fichero ZonasDeCultivoTodasParcelas-Iter1.ipynb se ha iniciado el trabajo con los datos entrenables obtenidos en la iteración anterior, los cuales se encuentran en el fichero 'csv/todasParcelas.csv'. A continuación, se ha procedido a realizar visualizaciones de los datos de diversas formas con el objetivo de corroborar las hipótesis planteadas.

Representación 2D para distinguir posibles zonas de cultivo distintas dentro de la misma parcela:

Se ha realizado una representación 2D para distinguir posibles zonas de cultivo diferentes dentro de una misma parcela. En este proceso, se han comparado los valores de NDVI en fechas donde se obtiene el máximo valor para cada punto, con el objetivo de identificar momentos en los cuales una zona de la parcela alcanza su máximo valor anual, mientras que otras zonas presentan valores significativamente distintos. La representación se ha realizado mediante una gráfica que muestra la forma de cada parcela utilizando tres colores:

- Los colores amarillos/naranjas representan el valor de NDVI de las parcelas cuyo píxel se encuentra a una distancia umbral del valor máximo de la fecha, lo que indica que podrían formar parte del mismo cultivo.
- Las zonas granates (con valor 0) representan puntos cuya distancia al valor máximo de la fecha es mayor que el umbral establecido.
- Las zonas azules (con valor -1) representan áreas que están fuera de la geometría de la parcela.

En la figura 16 se muestra la representación 2D para la diferenciación de zonas de cultivo dentro de una misma parcela, usando un umbral de 0,15. Se ha utilizado como ejemplo la parcela 31_208_0_0_2_32_1, donde se identifican tres zonas distintas. La primera zona, delimitada por las líneas rojas, posiblemente corresponde al cultivo principal de la parcela debido a sus constantes valores altos de NDVI. La segunda zona, marcada en verde y ubicada fuera de la zona roja en las parcelas 2 y 4, podría ser un cultivo secundario, ya que su índice NDVI no sigue el patrón del supuesto cultivo principal en ciertos momentos. También es posible que pertenezca al mismo cultivo pero con un desarrollo menor o más lento, o que el valor del píxel se haya visto afectado por la vegetación del margen del campo debido a la resolución del píxel de 10 metros. Por último, la zona que se encuentra fuera de las líneas azules en todas las parcelas probablemente no es cultivada en ningún momento del año.

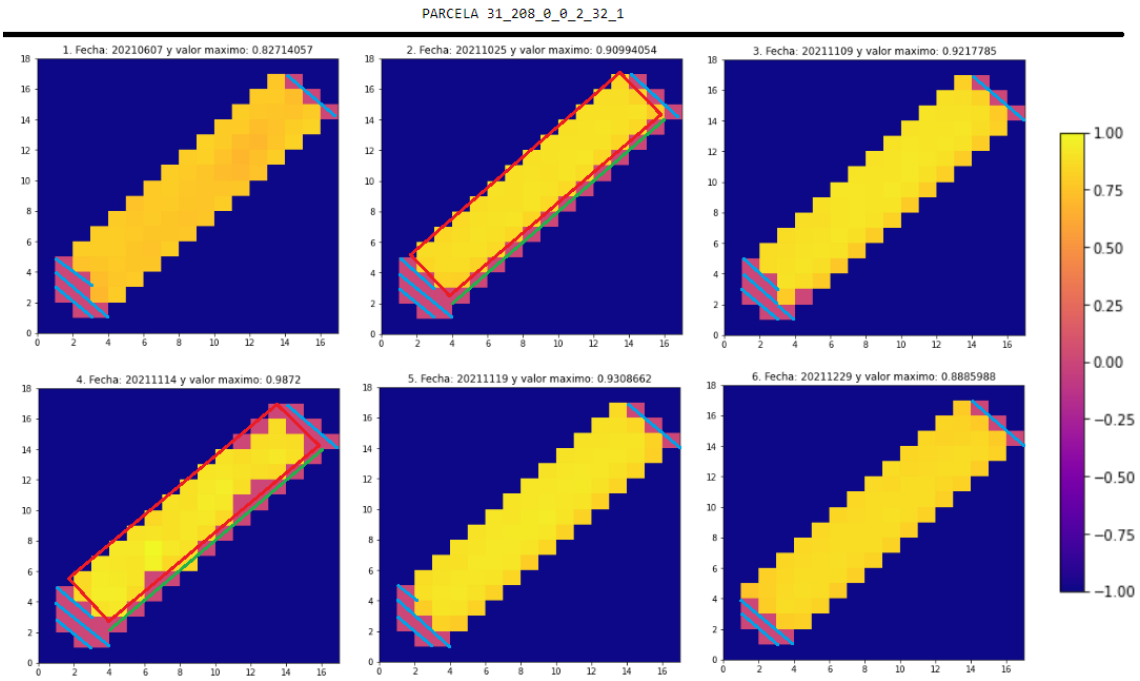


Figura 16: Representación 2D para la diferenciación de zonas de cultivo dentro de una misma parcela. Ejemplo con la parcela 31_208_0_0_2_32_1, que muestra dos posibles zonas distintas de cultivo (líneas roja y verde en las parcelas gráficas 2 y 4) y una zona sin cultivo durante todo el año (línea azul).

Representación 3D que ayude a visualizar la evolución del valor NDVI con el tiempo:

Se ha utilizado una representación 3D para visualizar la evolución del valor NDVI con el tiempo y analizar patrones predecibles. En cada parcela, se han graficado los valores NDVI de cada punto en función de la fecha (utilizando valores del 0 al 70 para representar la progresión de fechas a lo largo del año, desde principios de enero (0) hasta finales de diciembre (70)) y las coordenadas de cada parcela. El objetivo ha sido identificar si existe un comportamiento claro que indicara una fecha óptima de recepción, como se ha planteado en la figura 5. En el ejemplo de la figura 17, se observa que el valor NDVI no aumenta de manera gradual, como se esperaba inicialmente. En cambio, se han apreciado disminuciones abruptas en varias fechas. Este patrón se ha repetido en la mayoría, si no en todas, las parcelas. Se ha supuesto que estas disminuciones repentinas podrían ser resultado de la presencia de niebla o nubes captadas por los satélites durante esas fechas específicas, lo cual afecta el valor del NDVI. Además, se ha notado que los valores de cada píxel son similares en la mayoría de las fechas, excepto en aquellas situadas alrededor de la fecha 40, correspondientes a la mitad del año. Esto ha podido deberse a que durante ese período coincide con la preparación de la tierra para el cultivo y las primeras fases de crecimiento del maíz donde no tiene suficiente vigor vegetal, por lo que los valores de NDVI son bajos.

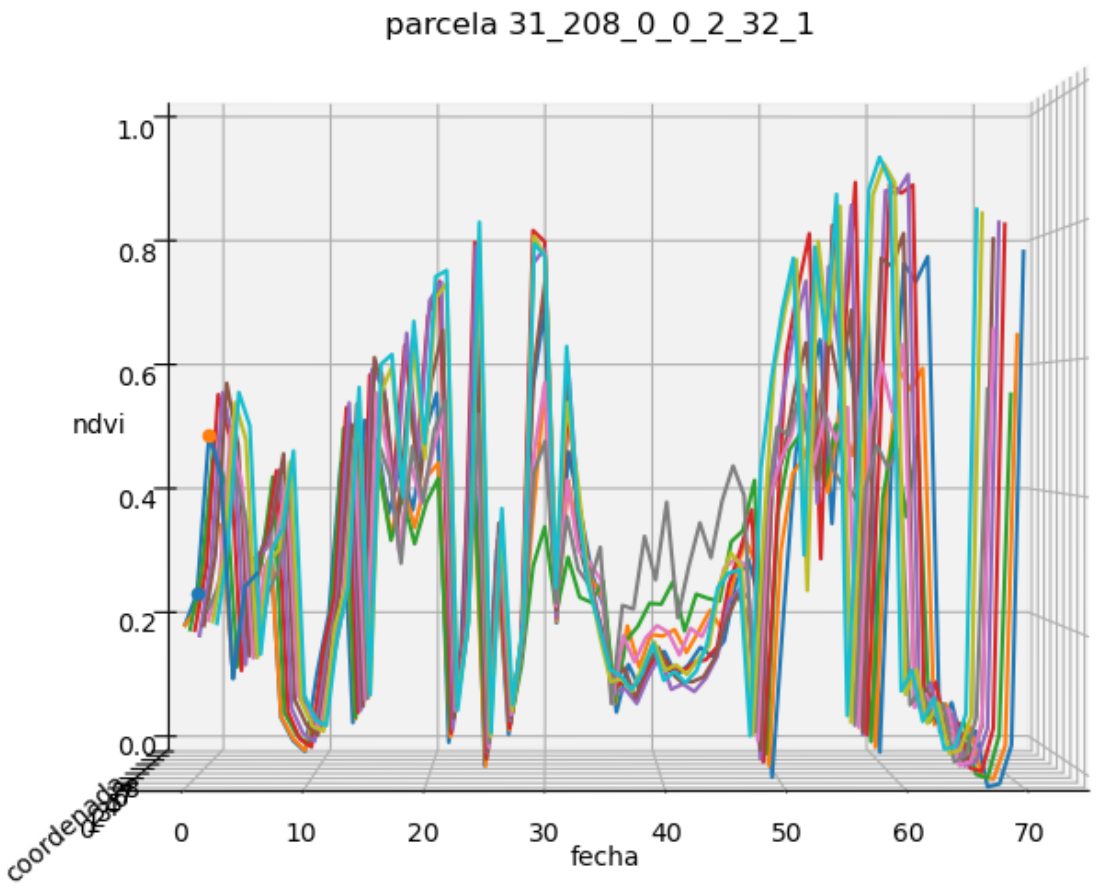


Figura 17: Representación 3D de la evolución del valor NDVI con el tiempo en cada píxel de una parcela. En este caso, se analiza la parcela 31_208_0_0_2_32_1. El eje "fecha" muestra valores del 0 al 70, que representan la progresión de fechas a lo largo del año, desde principios de enero (0) hasta finales de diciembre (70). El eje "coordenada" representa los distintos píxeles con valor NDVI en la parcela, utilizando valores del 0 al número total de píxeles menos 1. Por último, el eje "NDVI" muestra el valor NDVI de cada píxel en cada fecha distinta.

Representación 3D que ayude a visualizar el valor medio del NDVI del cultivo en la primera y segunda mitad del año:

Representación 3D del valor medio del NDVI del cultivo en la primera y segunda mitad del año para distinguir posibles grupos de píxeles con valores medios diferenciados. El objetivo ha sido identificar cultivos distintos dentro de la misma parcela, ya que algunos cultivos suelen cultivarse en épocas diferentes y podrían tener valores medios notablemente distintos.

En la figura 18, se observa la presencia de dos grupos claramente separados en la segunda mitad del año (representados por los círculos rojo y verde), los cuales están divididos por una zona vacía entre ellos. Como se ha comentado en la representación 2D anterior, estos grupos podrían mostrar el mismo cultivo con distinto desarrollo, o que debido a la resolución espacial de 10 metros, se haya visto afectado por la vegetación del margen de la parcela. Esta observación, junto con lo observado en la figura 16, ha sugerido la existencia de dos cultivos

diferentes en la parcela 31_208_0_0_2_32_1, así como dos pequeñas zonas sin cultivo durante todo el año.

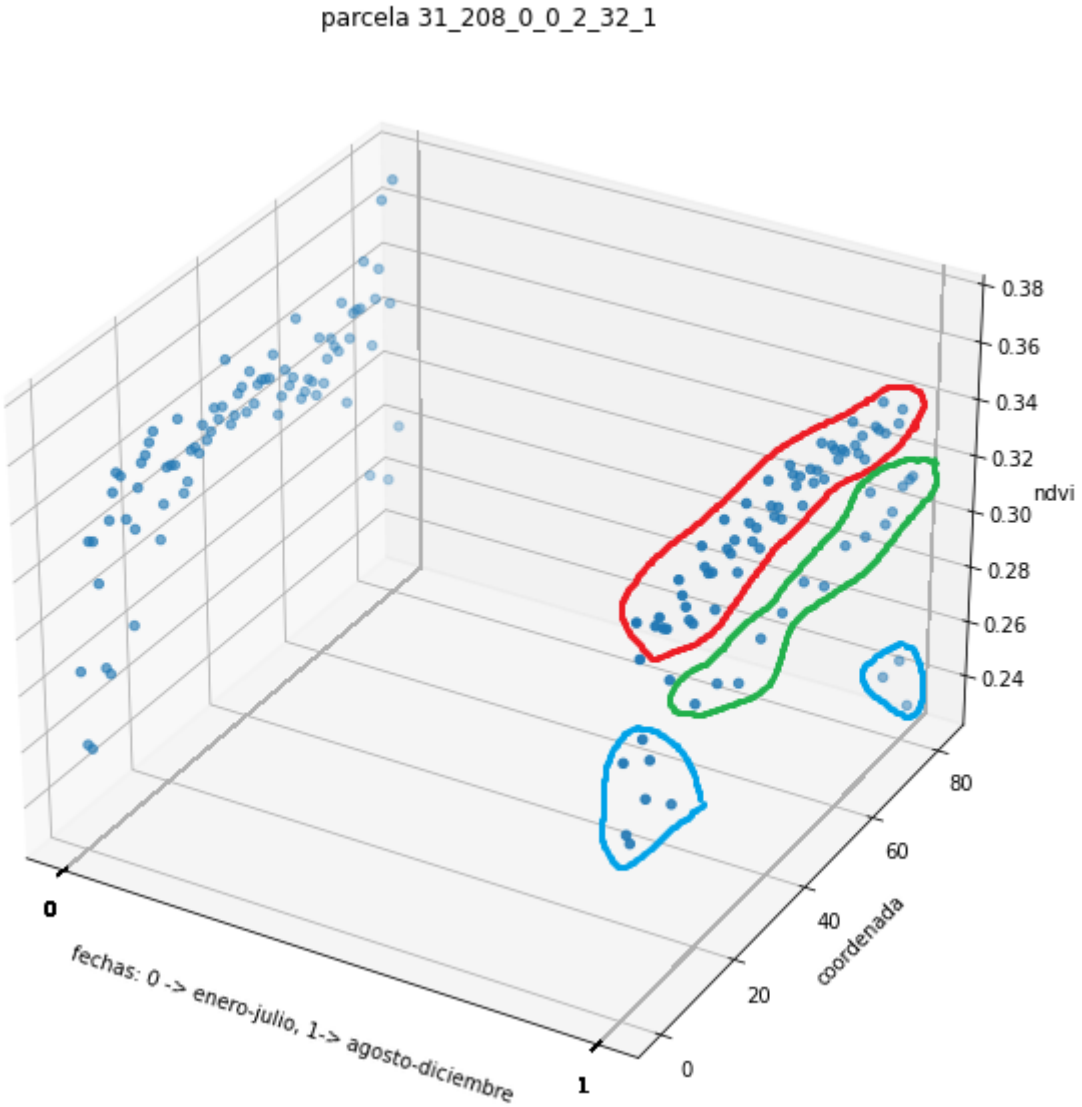


Figura 18: Representación 3D del valor medio del NDVI de cada píxel en cada mitad del año en una parcela. En este caso, se refiere a la parcela 31_208_0_0_2_32_1. Los círculos rojo y verde podrían representar desarrollos dispares del mismo cultivo. Los círculos azules podrían indicar áreas de la parcela que no son cultivadas en ningún momento del año.

Representación 2D de la clusterización de los valores medios del NDVI en cada mitad del año:

Finalmente, se ha llevado a cabo una representación 2D de la clusterización de los valores medios del NDVI en cada mitad del año, excluyendo los meses de junio y julio, bajo la suposición de que el cultivo, al encontrarse en fases iniciales de crecimiento, no ha adquirido todavía suficiente vigor vegetal. Esta representación en dos dimensiones se ha empleado para aclarar los casos en los que la separación no resultaba evidente en la representación 3D. Se ha utilizado un algoritmo de clusterización para agrupar los píxeles con valores similares con el fin

de identificar dos zonas distintas: una zona que representa el cultivo principal y otra zona secundaria que incluye subparcelas adicionales y áreas sin cultivar.

Esto ha permitido realizar una comparación con los resultados obtenidos en la primera prueba. Por ejemplo, en la figura 19 se observa cómo la clusterización de los valores de la segunda mitad del año muestra una zona de cultivo principal, coincidiendo con la suposición realizada en la primera prueba de esta iteración.

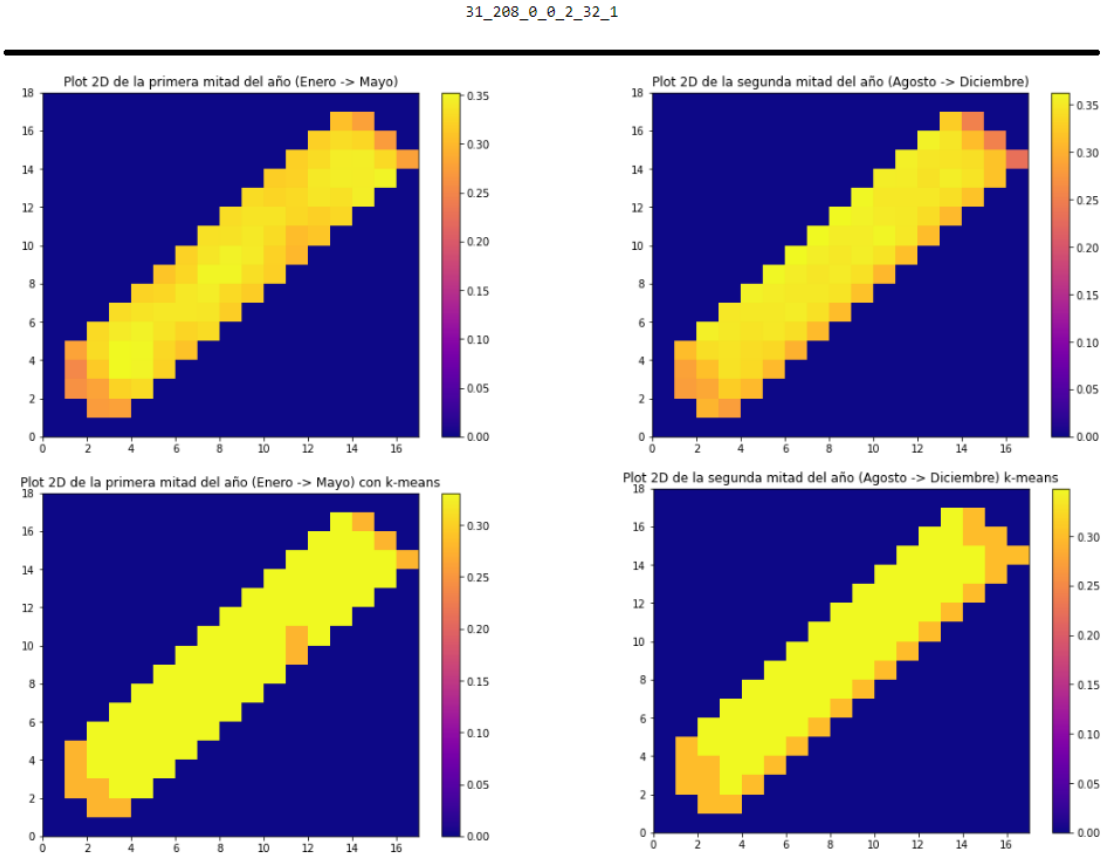


Figura 19: Representación 2D de zonas de cultivo principal y secundaria en una parcela. En este caso, se analiza la parcela 31_208_0_0_2_32_1. En la parte superior de la figura se muestra el valor medio de cada píxel de la parcela durante los primeros 5 meses del año (izquierda) y los últimos 5 meses del año (derecha). En la parte inferior se presenta la clusterización (utilizando k-means) de las figuras superiores.

Resultados

A lo largo de esta iteración, se han utilizado diferentes representaciones de los datos de las parcelas con el objetivo de verificar dos suposiciones previas y facilitar el análisis.

En cuanto a la primera hipótesis sobre la forma de la curva del NDVI, se ha observado un patrón distinto al esperado. En lugar de una curva continua, se han detectado numerosos días en los que el valor del NDVI es extremadamente bajo en todos los píxeles de la parcela. Se ha planteado la posibilidad de que estos resultados se deban a las condiciones meteorológicas a las que la teledetección satelital es sensible, a diferencia de otras como la radar, para capturar

la información. Como resultado, se sugiere explorar la implementación de algoritmos u otros enfoques para suavizar la curva y reducir la presencia de datos anómalos.

La segunda hipótesis planteaba la posibilidad de identificar subparcelas con cultivos distintos dentro de una parcela basándose en los valores del NDVI de cada píxel. Las representaciones realizadas han respaldado esta suposición, lo que sugiere que se podría continuar explorando este enfoque. Para avanzar en este objetivo, sería necesario obtener datos supervisados, es decir, datos etiquetados con la información correcta sobre los cultivos en cada subparcela. Esto permitiría comprender mejor cómo trabajar con estos datos y mejorar la precisión de las clasificaciones.

Anexo III.II. Iteración 2

Hipótesis de partida

Tras analizar los resultados anteriores, se toma la decisión de enfocarse exclusivamente en el objetivo de predecir la fecha óptima de cosecha. Además, se opta por trabajar únicamente con parcelas de maíz, basándose en la suposición de que este cultivo, al ser cosechado en seco y de manera completa en una sola operación, podría exhibir un comportamiento similar al representado en la figura 5 en cuanto a la evolución de su valor NDVI.

Con el fin de alcanzar este objetivo, se cuenta con la colaboración del cliente de GeosLab, que proporcionará información de numerosas parcelas de maíz en Aragón. Combinando esta información con las solicitudes a la API de datos geográficos y meteorológicos (desarrollo propio de Geoslab sobre sistemas abiertos de datos), se dispondrá de un amplio conjunto de datos para entrenar el modelo y permitirle predecir la fecha óptima de cosecha. Con esta nueva dirección, se descarta definitivamente cualquier exploración relacionada con la detección de zonas de cultivo dentro de una misma parcela.

Entre las parcelas proporcionadas, será interesante examinar las diferencias entre aquellas ubicadas cerca de la ribera del Ebro, y las parcelas de la provincia de Huesca, que se encuentran a mayor altitud. Estos cambios de localización podrían tener un impacto significativo en las fechas de recepción de las cosechas y en la duración de la temporada de alto NDVI del maíz en cada parcela.

Trabajo desarrollado

Se ha obtenido un archivo CSV ('csv/PARCELAS-MAIZ-2021-2022.csv') que contiene información sobre más de cinco mil parcelas de maíz en Aragón. Cada una de estas parcelas incluye datos como el tamaño de su superficie en hectáreas, la cosecha (1 o 2, indicando si es de primera o segunda cosecha del año) y su geometría.

Dado el amplio número de parcelas disponibles, se ha realizado una selección para descartar aquellas que pudieran generar ruido en los datos. Se ha optado por quedarse con aquellas parcelas que cumplen las siguientes características:

1. Parcelas de 1º cosecha, ya que son más comunes y su período de cultivo es más extenso, lo que proporciona una mayor cantidad de datos NDVI para su análisis. Se ha priorizado el estudio de estas parcelas en primer lugar.
2. Parcelas cuya superficie cultivada de maíz se asemeja bastante a la superficie total de la parcela. Esto se ha tenido en cuenta para garantizar una relación adecuada entre la zona cultivada de maíz y el conjunto de la parcela.

Una vez se ha obtenido el conjunto de parcelas con las que trabajar, se ha procedido a obtener los valores de NDVI a lo largo del año 2021 para cada una de ellas. Esto se ha realizado mediante una solicitud a la API de GeosLab, que ha devuelto el valor medio de NDVI de la parcela en cada fecha. La solicitud se formula de la siguiente manera:

- Direccion: https://teledeteccion.agroslab.com:9094/AgroslabHttpServlet_Prueba/AgroslabHttpServlet_Prueba
- Método: POST
- Parámetros de entrada (JSON): {“operation”: “getndviindexmeanvaluezone”, “initdate”: “01-04-2021”, “enddate”: “31-04-2021”, “id”: “50-120-0-0-501-105-2”}
- Respuesta (JSON): {“respuesta”: [{ “05-04-2021”: 0.9153699934869618}, { “10-04-2021”: 0.06536016439478763}, { “15-04-2021”: 0.038040428836648536}, { “20-04-2021”: 0.07304690998386253}, { “25-04-2021”: 0.7681393452085458}, { “30-04-2021”: 0.7415991274889929}] }

Todos los resultados de estas solicitudes se han guardado en el archivo 'csv/parcelasMaiz2.csv', el cual contiene las columnas "parcela", "fecha" y "ndvi".

Posteriormente, se ha facilitado otro archivo llamado 'csv/PARCELAS-MAIZ-2021-2022-CON-FECHAS.csv', el cual contiene la misma información que el primer archivo proporcionado, pero incluyendo una columna adicional con las fechas de cosecha correspondientes a cada parcela. Con el fin de investigar posibles variaciones en el día de cosecha y en la evolución del NDVI según la ubicación de las parcelas, se han seleccionado un total de 23 parcelas ubicadas en:

- La ribera del Ebro en Zaragoza, como Gallur (50_119_), Alagón (50_8_), Tauste (50_255_), Utebo (50_277_), Ejea (50_95_) y Almunia (50_25_).
- La provincia de Huesca, como Barbastro (22_61_), Sariñena (22_294_), Berbegal (22_72_), Monzón (22_218_), Huesca (22_901_) y Huerto (22_172_).

En las figuras 20 y 21 se muestra la evolución del NDVI de múltiples parcelas ubicadas en Zaragoza y Huesca, respectivamente, junto con sus fechas de recepción. Al analizar estas figuras, se ha podido observar que las evoluciones del NDVI no coinciden con las supuestas en la Figura 5 y presentan problemas similares de curvas erráticas, tal como se observó en la iteración anterior.

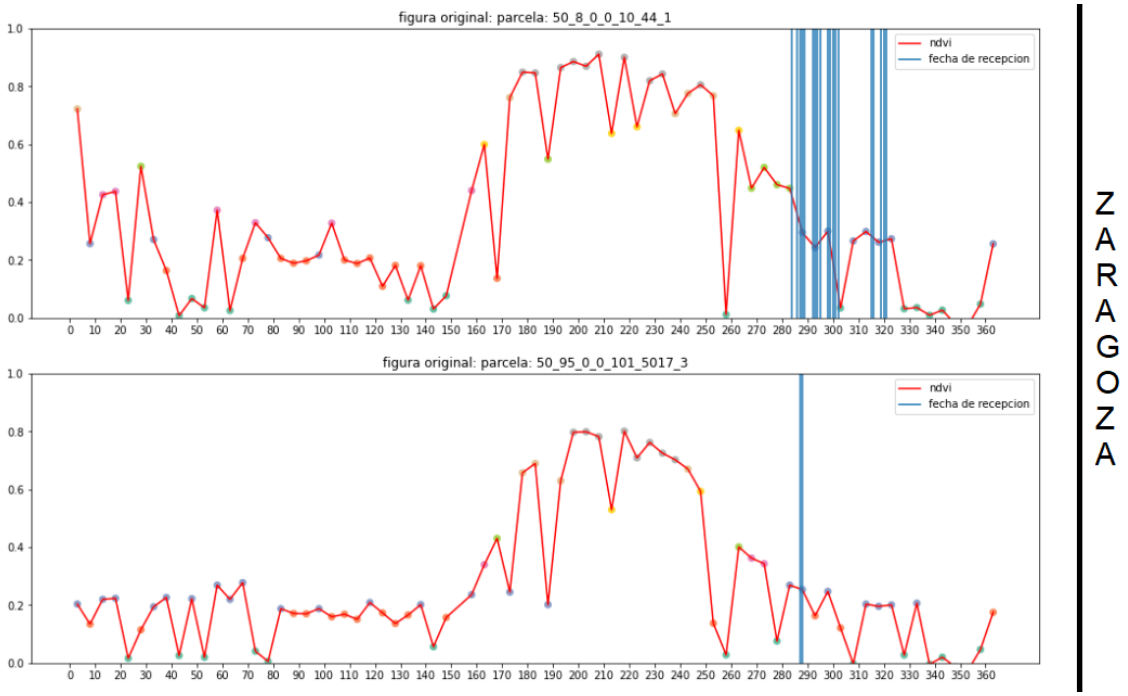


Figura 20: Evolución del NDVI (líneas rojas) y fechas de recepción (líneas azules) de 2 parcelas en la provincia de Zaragoza. Parcela de Alagón arriba y parcela de Ejea abajo.

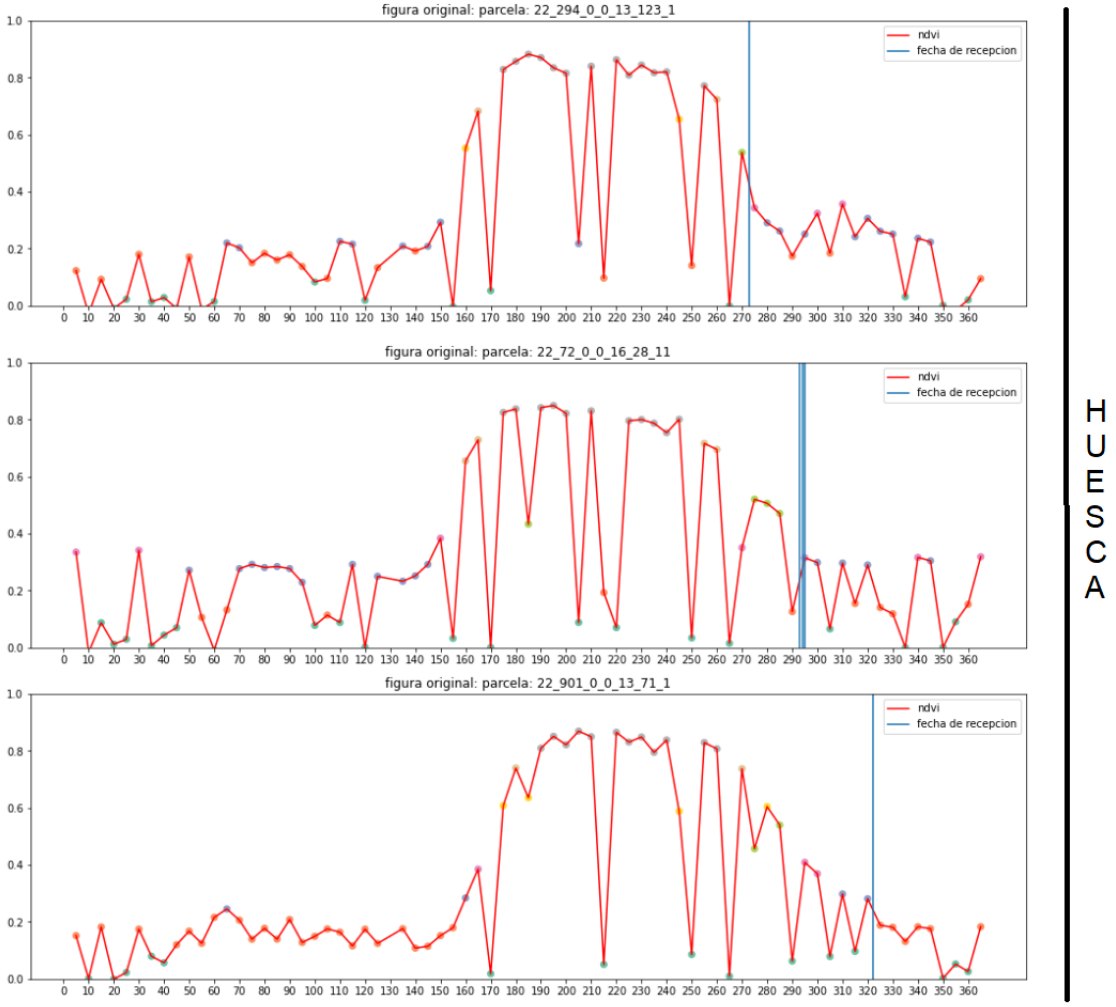


Figura 21: Evolución del NDVI (líneas rojas) y fechas de recepción (líneas azules) de 3 parcelas en la provincia de Huesca. Parcela de Sariñena arriba, parcela de Berbegal en medio y parcela de Huesca abajo.

Resultados

En esta iteración se ha tomado la decisión de enfocar el trabajo en la predicción exclusiva de la fecha óptima de cosecha para parcelas de maíz. Se ha considerado que, al tratarse de un cultivo que se cosecha cuando está seco y de forma repentina, es probable que exhiba un comportamiento similar al representado en la figura 5 en términos de la evolución de su valor NDVI.

A través de las peticiones a la API de GeosLab, se ha representado la curva del valor NDVI junto con las fechas de cosecha de distintas parcelas ubicadas cerca de la ribera de Zaragoza y en la provincia de Huesca, con el fin de comparar diferentes suposiciones relacionadas con la influencia de la humedad o altitud de las parcelas en las fechas de recepción, el período de alto valor NDVI y el valor máximo del NDVI de cada parcela.

Sin embargo, tras analizar los resultados obtenidos, no ha sido posible llegar a una conclusión definitiva sobre estas tres hipótesis, ya que se han observado casos variados en todos los

escenarios considerados. Por lo tanto, se confirma que las parcelas no se ven fuertemente afectadas por las condiciones de cada localización.

No obstante, lo que sí se ha observado es que todas las parcelas siguen un patrón de evolución del valor NDVI similar al mostrado en la figura 6, donde el NDVI aumenta rápidamente, se mantiene en niveles altos durante un tiempo y luego disminuye gradualmente. Además, la fecha de recepción tiende a ubicarse al final de la disminución, cuando el valor NDVI se estabiliza en torno a 0.2-0.3.

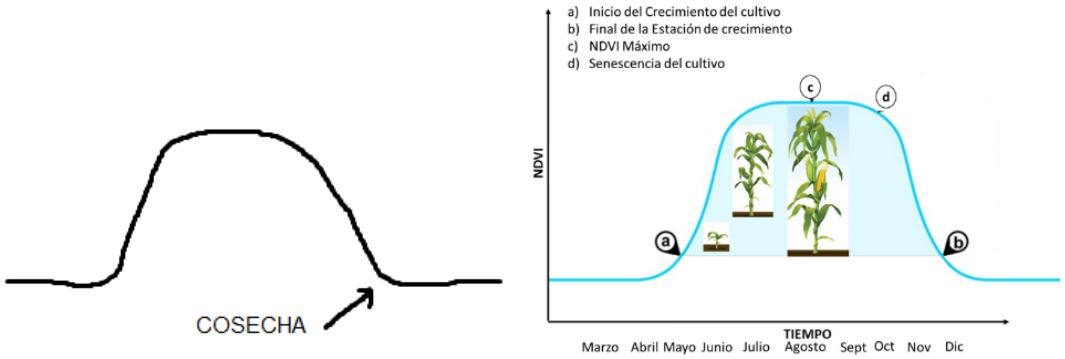


Figura 6: Esquema propio (a la izquierda) y teórico (a la derecha) de la evolución del NDVI observado en parcelas reales. Se ilustra un rápido incremento hasta alcanzar el valor máximo, seguido de un decrecimiento gradual, en contraposición a la suposición inicial de la figura 5. La fecha de cosecha suele coincidir con el final de la fase de decrecimiento, cuando la evolución se estabiliza en torno a valores de NDVI de 0.2-0.3. Fuente: Plataforma Geoweb para la Red de Desarrollo en Sustentabilidad Alimentaria. Disponible en: http://asam.centrogeo.org.mx/index.php/resultado-13?showall=1&limitstart=

Anexo III.III. Iteración 3

Hipótesis de partida

Para abordar las fluctuaciones en forma de picos que se observan en las gráficas de la evolución del NDVI, se plantea la posibilidad de desarrollar un algoritmo que permita eliminar estas fluctuaciones en forma de dientes de sierra y suavizar los puntos en las rectas, con el fin de obtener un valor más estable a lo largo del tiempo.

Trabajo desarrollado

Con el fin de desarrollar el algoritmo deseado, se ha optado por seguir una estrategia de ensayo y error. Se han realizado pequeñas modificaciones en el algoritmo, se ha observado los resultados y se han evaluado posibles problemas que puedan surgir en cada iteración. Este proceso se ha repetido hasta llegar a un punto en el que no se identifican más problemas.

Con el objetivo de garantizar que el algoritmo de suavizado mejore todas las gráficas, se ha seleccionado un conjunto de parcelas para evaluar los efectos de las modificaciones aplicadas. Este conjunto ha consistido en las mismas 23 parcelas analizadas en la iteración anterior. Estas

parcelas se han considerado lo suficientemente diversas entre sí como para que un algoritmo que se ajuste a ellas pueda funcionar adecuadamente con otras parcelas.

Para evaluar dichas modificaciones, se ha empleado un algoritmo de suavizado de gráficas y de interpolación de puntos utilizando las funciones *make_interp_spline*[23], *numpy.linspace*[24] y *X_Y_Spline*. Estos algoritmos han permitido estimar el valor NDVI en fechas no proporcionadas por la API utilizada en la iteración anterior, la cual registra información cada 5 días.

En la figura 22 se muestra el funcionamiento de la interpolación en comparación con los datos originales.

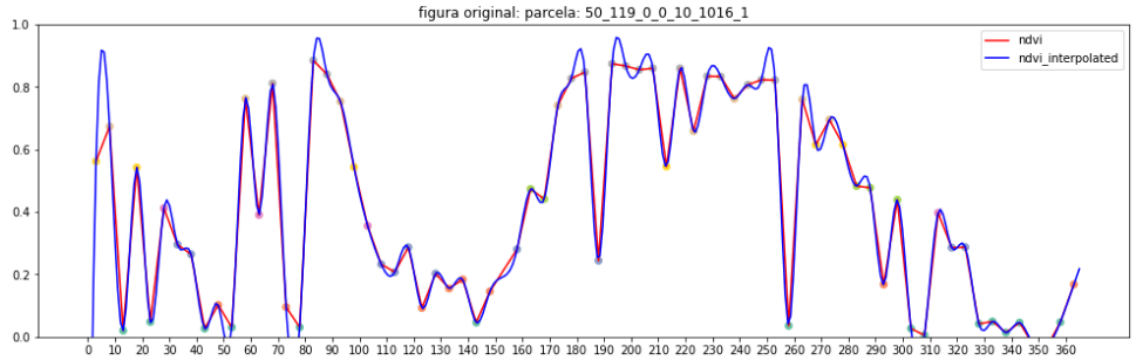
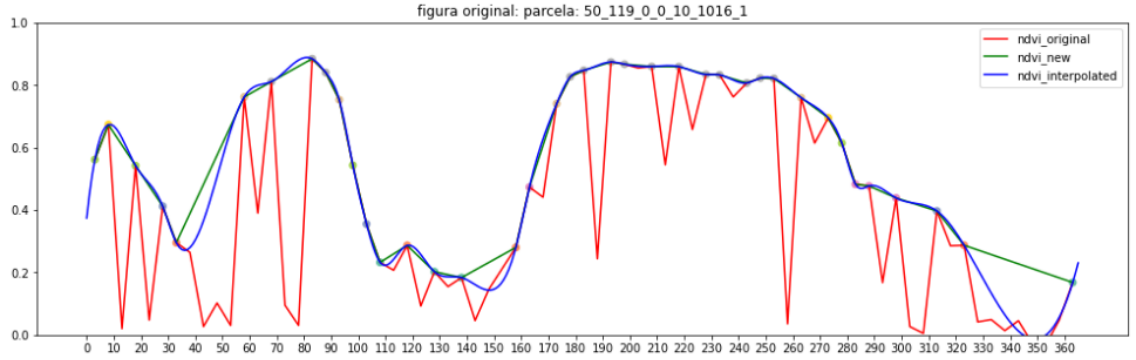


Figura 22: Evolución del NDVI de la parcela 50_119_0_0_10_1016_1 (en rojo) y curva obtenida mediante las funciones de interpolación (en azul oscuro) make_interp_spline, numpy.linspace y X_Y_Spline. Las líneas azules claras indican la fecha de recepción de la cosecha.

Primera prueba: Eliminación de datos menores a 0.15 y mínimos locales.

Se ha supuesto que cualquier dato por debajo de 0.15 indica la ausencia de cultivo, por lo que se han eliminado todos los valores por debajo de este umbral para retener sólo aquellos que contienen información relevante. Además, para eliminar los descensos irregulares (picos de sierra) que no tienen una explicación clara, se han eliminado los mínimos locales.

En la figura 23 se muestra una notable mejora en la representación de la parcela 50_119_0_0_10_1016_1.



[23] https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.make_interp_spline.html
[24] https://numpy.org/doc/stable/reference/generated/numpy.linspace.html

Figura 23: Resultados del suavizado tras la primera prueba, que incluye la eliminación de valores menores a 0.15 y de los mínimos locales.

Sin embargo, se ha observado un efecto exagerado en las zonas donde no hay puntos, por lo tanto, se ha procedido a rellenar estas áreas con puntos intermedios para corregir este efecto no deseado.

Segunda prueba: Adición de valores intermedios en fechas con una separación de 10 días o más.

Con el objetivo de solucionar los espacios vacíos en las gráficas, se ha procedido a agregar el valor promedio entre dos puntos separados por 10 días o más. Tras aplicar esta técnica, en la figura 24 se observa una solución efectiva a este problema.

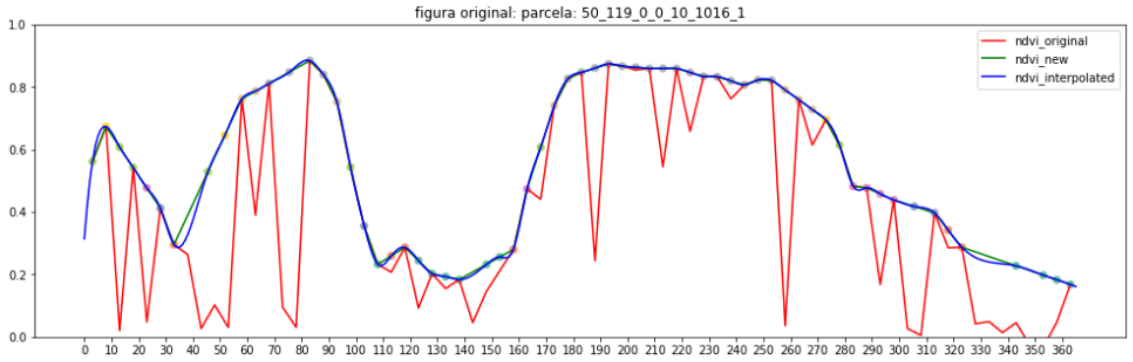


Figura 24: Resultados del suavizado tras la segunda prueba, que consiste en la adición de puntos en los espacios de 10 o más días sin valores.

Al analizar la figura, se han identificado dos problemas a solucionar. En primer lugar, se ha observado que en los extremos de las gráficas, donde no hay puntos disponibles, la curva puede presentar un efecto exagerado que no refleja de manera precisa el valor del NDVI en estas regiones extremas. El segundo problema se ha relacionado con la aparición de curvas excesivamente inclinadas, las cuales no parecen representar adecuadamente la evolución real del valor NDVI, como se ha evidenciado en la fecha 40 de la figura anterior.

Tercera prueba: Añadir información a los extremos de las gráficas.

Para solucionar este problema, se ha replicado el punto más cercano a cada extremo de la gráfica, evitando que las curvas tomen un efecto exagerado en esos puntos. En la figura 25 se muestra la comparativa entre la parcela 22_61_0_0_14_162_2 antes y después de aplicar esta modificación en el algoritmo. Se puede observar que la evolución del NDVI al principio del año presenta un mayor sentido tras esta modificación.

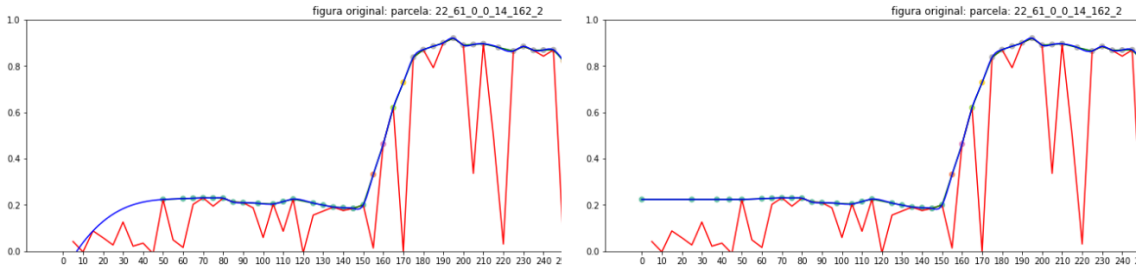


Figura 25: Evolución del NDVI en la parcela 22_61_0_0_14_162_2 antes y después de aplicar el algoritmo de suavizado en la tercera prueba. Se ha replicado el valor más cercano al inicio de la gráfica para evitar deformaciones debido a la falta de datos.

Cuarta prueba: Corrección de pendientes en intervalos de más de 20 días.

En este punto, las gráficas han mejorado significativamente en cuanto a la distorsión de la curva, pero aún se observan pendientes pronunciadas en ciertos tramos durante la subida o bajada del valor NDVI. Con el objetivo de abordar este problema, se ha identificado que los espacios de 20 días sin información son propensos a generar pendientes excesivas. Para solucionarlo, se han añadido puntos en estos espacios de manera que la pendiente resultante no sea tan inclinada. Estos puntos se sitúan a ⅓ de la distancia Y de ambos por encima del punto de menor valor Y, y a ⅓ de la distancia X de ambos cercano al de mayor valor Y. La figura 26 ilustra dos ejemplos para facilitar la comprensión de esta modificación.

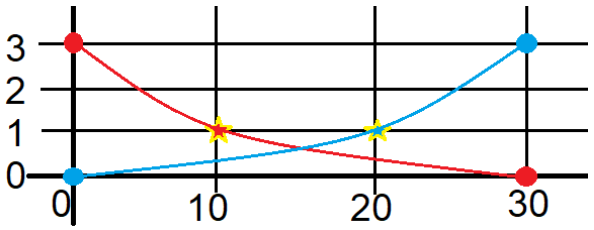


Figura 26: Reducción de las pendientes en las curvas mediante la introducción de un nuevo punto intermedio. La posición X del punto se encuentra a un tercio de la distancia entre los puntos originales, más cercano al punto de mayor valor. El valor Y del punto se sitúa a un tercio de la distancia de valor entre los puntos, tirando al punto de menor valor.

Por otro lado, en la figura 27 se muestra un ejemplo concreto de cómo esta modificación ha impactado en la parcela 22_172_0_0_19_137_1, eliminando la pendiente pronunciada de la curva y obteniendo un resultado más acorde con la evolución real del NDVI.

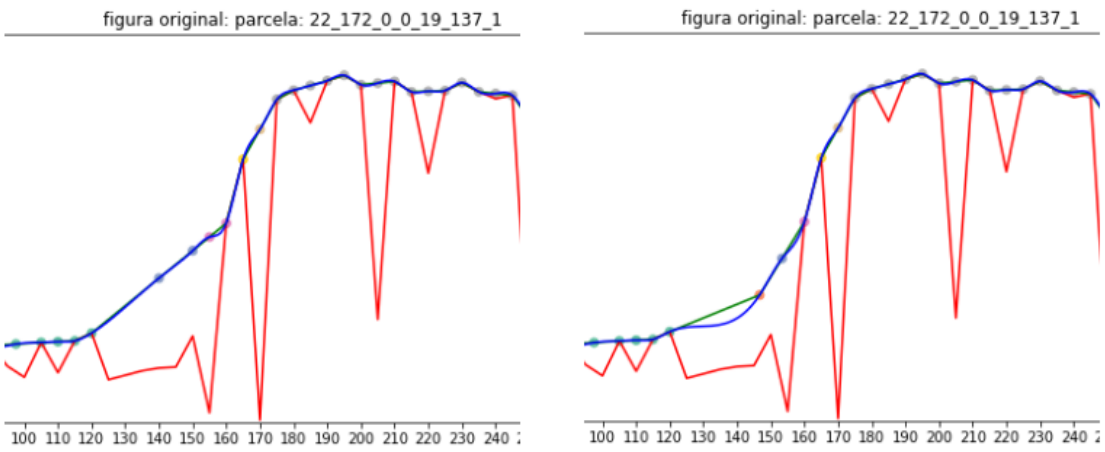


Figura 27: Parcela 22_172_0_0_19_137_1 antes y después del uso del algoritmo de suavizado en la cuarta prueba. Se ha realizado una interpolación de puntos intermedios en intervalos de 20 o más días para evitar pendientes exageradas, logrando un resultado más realista.

Quinta y última prueba: Eliminación de parejas de valores que forman mínimos locales.

A pesar de los esfuerzos realizados en la primera prueba, todavía persisten algunos picos de sierra en las zonas con valores altos de NDVI. Estos picos no fueron eliminados en la prueba anterior debido a que aparecían en parejas y no formaban un mínimo local por sí solos. Por lo tanto, en esta última prueba se han eliminado las parejas de valores que formen un mínimo local entre ellos. Por ejemplo, si se tiene una secuencia de valores ordenados como [3, 1, 2, 4], los valores 1 y 2 son ambos menores que los valores vecinos 3 y 4, respectivamente. En este caso, se eliminan tanto el valor 1 como el valor 2. Al aplicar esta eliminación de parejas de mínimos locales, se han logrado los resultados deseados, los cuales se pueden apreciar en la figura 28.

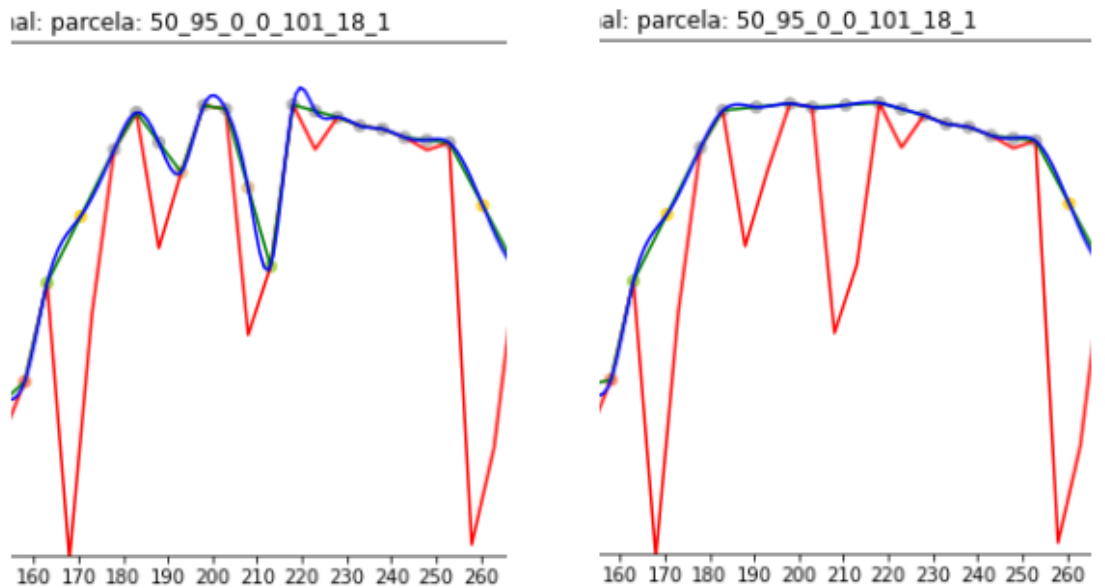


Figura 28: Parcela 50_95_0_0_101_18_1 antes y después de aplicar el algoritmo de suavizado tras la quinta prueba, en la cual se han eliminado parejas de valores que forman mínimos

locales para mitigar la presencia de picos de sierra en las regiones con valores elevados de NDVI.

Resultados

Durante esta iteración se ha desarrollado un algoritmo de suavizado para corregir el comportamiento errático de los valores NDVI en las gráficas. El procedimiento seguido ha consistido en realizar pequeñas modificaciones, observar su impacto en las parcelas de observación seleccionadas, identificar nuevos problemas y continuar este proceso hasta que no se encontraran más errores.

Se ha desarrollado un algoritmo que ha demostrado obtener resultados satisfactorios, respaldando así la hipótesis planteada. A partir de ahora, este algoritmo se utilizará para trabajar con los datos del NDVI, asegurando que el ruido presente no afectará los resultados futuros.

Anexo III.IV. Iteración 4

Hipótesis de partida

Con el algoritmo actual que elimina errores y curvas inusuales de las parcelas, se puede comenzar a trabajar en la predicción de la fecha óptima de recepción utilizando diferentes modelos basados en Inteligencia Artificial, como las redes neuronales.

En esta iteración, se plantea la posibilidad de desarrollar un modelo de predicción que pueda determinar la fecha exacta de cosecha de un cultivo, utilizando el valor de NDVI de cada día a lo largo del año como datos de entrada y el día del año exacto (número del 1 al 365) como salida. Además, se considera la viabilidad de desarrollar tanto un modelo de clasificación como un modelo de regresión para este propósito.

Trabajo desarrollado

Como se ha mencionado en la hipótesis, se ha procedido a evaluar el modelo tanto como un problema de clasificación como de regresión. En el caso de los problemas de regresión, se ha utilizado el Error Cuadrático Medio (RMSE) como medida de éxito. Por otro lado, para los problemas de clasificación se ha empleado la precisión como medida de éxito.

Prueba principal

En esta primera prueba se han utilizado datos de entrada que consistían en un arreglo con los valores NDVI de los 365 días, y la salida objetivo ha sido el día correspondiente (del 1 al 365) a la fecha de recepción.

Inicialmente, se ha utilizado un modelo de regresión sencillo como el *DecisionTreeRegressor*[25]. Para encontrar el modelo con el menor error de validación, se ha empleado el 80% de los datos para entrenamiento y validación, con 10 divisiones de validación cruzada (*folds*) y probando diferentes profundidades (parámetro *max_depth*) del *DecisionTreeRegressor*, desde 1 hasta 30.

[25] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

El mejor modelo encontrado ha tenido una profundidad de 3 y un RMSE de validación aproximado de 500 días. Sin embargo, el error con los datos de test (el 20% restante) ha sido superior a 10,000 días, lo que ha llevado a descartar este modelo debido a sus malos resultados.

Luego, se ha utilizado un modelo de clasificación basado en una red neuronal secuencial, utilizando la función *Sequential*[26] del módulo *keras.models*. Esta red neuronal ha constado de una capa con 365 neuronas (una por cada clase) y ha utilizado la función de activación *sigmoide*. La red se ha compilado utilizando la función de pérdida de entropía cruzada categórica, el optimizador *Adam* y se ha evaluado utilizando la métrica de precisión. En el código 1 se muestra el código de la red neuronal utilizada.

```
# Función para parar cuando ya no mejora el error en los datos de validacion
earlystop=EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='auto')

# Perceptron de un solo nivel
model = Sequential()
model.add(Dense(365, activation='sigmoid', input_shape=(365,)))
model.compile(loss='categorical_crossentropy', optimizer=Adam(),  metrics=['accuracy'])
model.summary()
```

Código 1: Arquitectura de la red neuronal secuencial con 1 capa.

Se ha asignado el 90% de los datos para entrenamiento y el 10% restante para validación. El modelo se ha entrenado durante 20 *epochs* (0.5 segundos), pero los resultados han seguido siendo insatisfactorios, ya que solo se obtuvo una precisión de validación (*val_accuracy*) de 0.2632.

Estas dos pruebas insatisfactorias han demostrado la invalidez de la hipótesis inicial.

Prueba alternativa

Dada la brevedad de las pruebas anteriores, se ha tomado la decisión de realizar otra prueba similar, pero con pequeñas modificaciones en los datos de entrada, alejándose de la propuesta inicial de la hipótesis. Los datos de entrada han seguido siendo 365, pero en lugar de ser únicamente el valor NDVI del día 'i', se han compuesto de 5 valores de la siguiente manera para cada día 'i':

[NDVI(i - 2) - NDVI(i), NDVI(i - 1) - NDVI(i), NDVI(i), NDVI(i) - NDVI(i + 1), NDVI(i) - NDVI(i + 2)]

Es decir, se ha considerado la diferencia de valor NDVI en relación a los 2 días anteriores y los 2 días siguientes, además del valor NDVI del propio día 'i'.

Posteriormente, se ha vuelto a evaluar el modelo *DecisionTreeRegressor*, pero se ha obtenido un RMSE de validación de aproximadamente 11000 en la profundidad 3, mucho peor que en el experimento anterior, por lo que no ha valido la pena probarlo con los datos de test.

Luego, se ha repetido la prueba del problema de clasificación utilizando otra red neuronal secuencial, pero esta vez se han utilizado 3 capas en lugar de solo 1. La primera capa ha

[26] https://keras.io/guides/sequential_model/

constado de 10000 neuronas, la segunda capa de 365 neuronas y la tercera capa (la capa de salida) tiene 365 neuronas. Se ha utilizado la función de activación *sigmoide* en la primera capa, la función *ReLU* en la segunda capa y la función *softmax* en la última capa. La red se ha compilado de la misma manera que la red anterior. En el código 2 se muestra el código de la red neuronal utilizada.

```
# Función para parar cuando ya no mejora el error en los datos de validacion
earlystop=EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='auto')

# Perceptron de un solo nivel
model = Sequential()
model.add(Dense(10000, activation='sigmoid', input_shape=(365,)))
model.add(Dense(365, activation='relu'))
model.add(Dense(53, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(),  metrics=['accuracy'])
model.summary()
```

Código 2: Arquitectura de la red neuronal secuencial con 3 capas.

En primer lugar, se ha repetido la prueba utilizando como salida el día exacto del año. El modelo ha sido entrenado durante 48 *epochs* (6.471s) y se ha obtenido una precisión del 34.09% en los datos de validación.

Posteriormente, se ha llevado a cabo una prueba adicional utilizando el mismo modelo pero con la semana del año (del 1 al 53) como salida en lugar del día, con la esperanza de obtener mejores resultados al simplificar el problema. El modelo ha sido entrenado durante 29 *epochs* (3.947s) y se ha obtenido una precisión del 36.36% en los datos de test, lo cual representa una mejora en comparación con la prueba anterior, pero aún es considerada baja.

Resultados

Como se ha mencionado a lo largo del trabajo realizado, se puede concluir que la hipótesis inicial no es válida. No se ha logrado desarrollar un modelo, ya sea de regresión o clasificación, capaz de predecir el momento exacto de la cosecha utilizando como dato de entrada el valor del NDVI en cada día de la curva de evolución.

Anexo III.V. Iteración 5

Hipótesis de partida

Se decide cambiar los datos de entrada con los que trabajar debido a los resultados insatisfactorios y a la limitación de utilizarlos en diferentes países, donde el mismo día del año puede corresponder a distintas estaciones y las fechas de recogida pueden variar debido a los cambios en las temperaturas. Se plantea trabajar con los datos en forma de series temporales, donde cada dato de entrada contiene información diaria, y se intenta predecir si se debe recoger (1) o no (0) el cultivo ese día. Esta aproximación permite incluir información de días anteriores en cada dato. Se proponen los siguientes formatos de datos de entrada para predecir la fecha óptima de cosecha:

- df_ts1: cada dato de entrada contiene únicamente el valor NDVI del día actual.

- df_ts2_X: cada dato de entrada contiene el valor NDVI del día actual 'i' y la diferencia de valor NDVI con cada uno de los X días anteriores. Es decir, en cada dato, el valor de la columna 'pendiente día - X' es: NDVI(i - x) - NDVI(i).
- df_ts3_X: cada dato de entrada contiene el valor NDVI del día actual 'i' y el valor NDVI de cada uno de los X días anteriores.

Se considera que se puede abordar el problema tanto como un problema de clasificación como de regresión. Además, se plantea reducir el desbalance entre las muestras de cada clase. Las muestras hasta 7 días antes de la fecha oficial de recogida se convertirían a la clase 1. En el caso de clasificación, esto equilibraría las clases. Si se trata de regresión, la salida de estas muestras convertidas aumentaría gradualmente hasta alcanzar el valor 1 en la muestra de recogida oficial.

Trabajo desarrollado

En un principio, se ha contemplado la idea de probar con todos los *dataframes* disponibles, tanto con un valor X de 7 días como de 180 días, dependiendo de si se ha considerado relevante la información de la última semana o de los últimos 6 meses. No obstante, debido a las limitaciones de tiempo para realizar todas las pruebas, se ha tomado la decisión de trabajar únicamente con el *dataframe* df_ts2_X. Esta elección se ha basado en considerar que su información era más relevante, ya que la posibilidad de tener diferencias negativas entre los valores NDVI podía aportar más información que los valores siempre positivos presentes en el df_ts3_X, donde solo se registra el valor NDVI (siempre positivo).

Además, se ha optado por utilizar un valor X de 180 días en este *dataframe*, ya que se considera que es un periodo de tiempo lo bastante largo como para agrupar todas las fechas en las que los valores NDVI de la gráfica son más altos. Esto puede proporcionar información adicional en comparación con un valor X más pequeño. El *dataframe* resultante se ha guardado en el archivo 'csv/df_ts2_180dias_bien.csv'.

En el caso de los modelos de clasificación binaria, se ha evaluado el éxito midiendo el porcentaje de parcelas que son correctamente clasificadas al menos en uno de los días marcados como 1 (*true positive*). En los problemas de regresión, se han utilizado diversas métricas de error medio para evaluar el rendimiento de los modelos.

Primera prueba de clasificación

En la siguiente prueba del problema de clasificación, se han utilizado los 7 días anteriores a la fecha de recogida, etiquetados como 1, con el fin de abordar el desequilibrio entre las clases.

Se ha repetido la prueba utilizando otra red neuronal secuencial, esta vez con 3 capas. La primera capa ha constado de 10,000 neuronas, la segunda capa de 365 neuronas y la tercera capa (la capa de salida) tiene 2 neuronas para el problema de clasificación binaria. Se ha empleado la función de activación *sigmoide* en la primera capa, la función *ReLU* en la segunda capa y la función *softmax* en la capa de salida. La red ha sido compilada utilizando la función de pérdida de entropía cruzada categórica, el optimizador *Adam* y se ha evaluado utilizando la métrica de precisión. Esta red ha sido similar a la utilizada en la segunda prueba de la iteración

anterior, con la diferencia de las neuronas en la capa de salida para adaptarse al problema de clasificación binaria. En el código 3 se muestra el código de la red neuronal utilizada.

```
# Perceptron de un solo nivel
model = Sequential()
model.add(Dense(10000, activation='sigmoid', input_shape=(181,)))
model.add(Dense(365, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(),  metrics=['accuracy'])
model.summary()
```

Código 3: Arquitectura de la red neuronal secuencial con 3 capas.

El modelo ha sido entrenado durante 2000 *epochs* (8 horas) utilizando el 90% de los datos y se ha obtenido una precisión del 91.91% en los datos de validación (10%). Sin embargo, es importante destacar que la precisión y el *recall* de los datos de la clase 1 han sido del 19.4% y 27.8%, respectivamente, lo cual indica un rendimiento muy bajo en la predicción de dicha clase.

Segunda prueba de clasificación

En el experimento actual, se ha repetido la prueba anterior agregando una capa de normalización a la entrada de datos de la red neuronal. El objetivo ha sido mejorar la robustez y generalización del modelo al reducir la covarianza de las características de entrada. La capa de normalización se ha implementado utilizando el método *layers.Normalization()*[27] del módulo *tensorflow.keras* de Python, y se ha utilizado una normalización por lotes que calcula nuevos valores de entrada basados en estadísticas de los datos, como la media y la varianza de cada característica.

El modelo ha sido entrenado durante 5600 *epochs* (22 horas) y se ha obtenido una precisión del 99.95% en los datos de entrenamiento y del 93.65% en los datos de validación. Sin embargo, al igual que en la prueba anterior, la precisión (28.07%) y el *recall* (30%) de los datos de la clase 1 han seguido siendo bajos. A pesar de esto, el modelo ha acertado al menos un día etiquetado como 1 en 21 de las 40 parcelas de validación (52.5%), que se había establecido como medida de éxito para el modelo.

Para analizar visualmente los resultados, se han mostrado las 40 parcelas de validación con sus curvas NDVI, indicando las fechas correspondientes a la clase 1 y las fechas predichas por el modelo como clase 1. En la figura 29 se presentan dos ejemplos de parcelas en las que no se acierta ningún día de la clase 1. Se puede observar que las fechas de recepción y las predicciones son completamente opuestas en ambas parcelas. En la figura 30 se muestran dos ejemplos de parcelas en las que se acierta al menos un día de la clase 1. En estas parcelas se puede observar que una de ellas acierta al mostrar muchos valores etiquetados como 1 antes de la fecha de recepción, mientras que en la otra parcela se predicen muchos valores 1 incluso después de la fecha de recepción.

[27] https://keras.io/api/layers/preprocessing_layers/numerical/normalization/

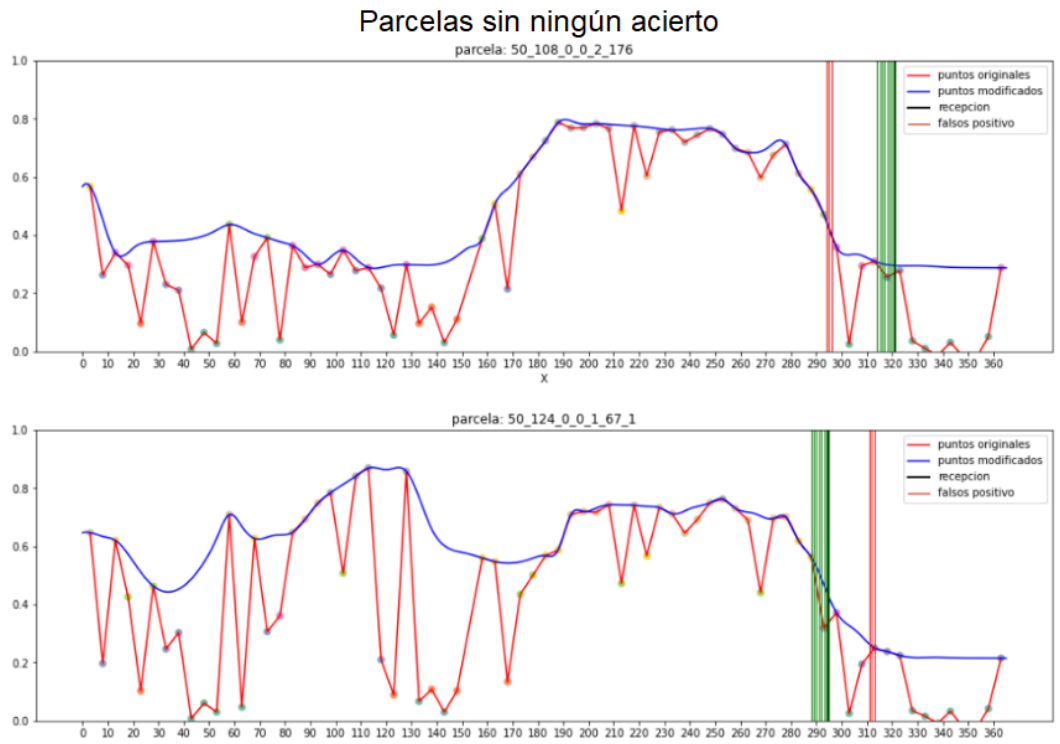


Figura 29: Ejemplos de parcelas sin predicciones correctas de cosecha por parte del modelo. Se muestran las curvas de valores NDVI originales (rojas) y las curvas de valores NDVI después de aplicar el algoritmo de suavizado (azules). Las líneas verticales verdes indican los días de la clase 1, mientras que las líneas rojas representan predicciones incorrectas del modelo para esos días.

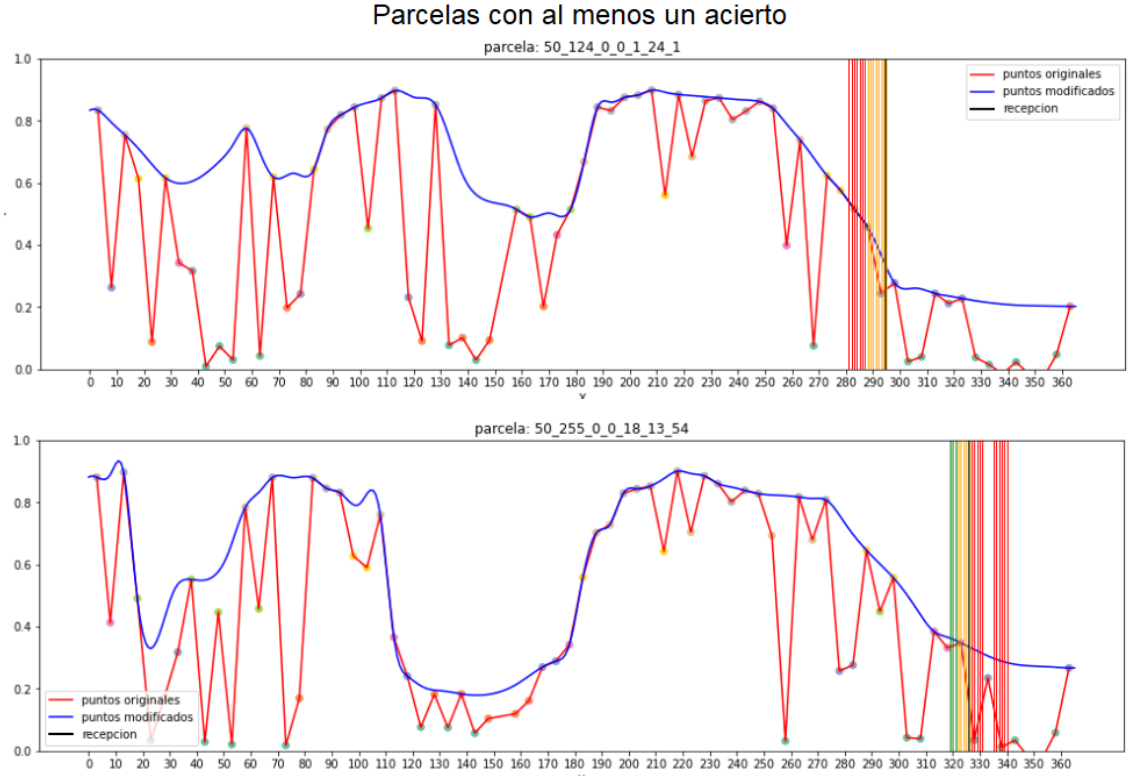


Figura 30: Ejemplos de parcelas con al menos 1 predicción correcta de cosecha por parte del modelo. Se muestran las curvas de valores NDVI originales (rojas) y las curvas de valores NDVI después de aplicar el algoritmo de suavizado (azules). Las líneas verticales verdes representan los días de la clase 1, que se muestran en amarillo si el modelo ha realizado una predicción correcta. Las líneas verticales rojas representan predicciones incorrectas del modelo para esos días.

Tercera prueba de clasificación

Después de la prueba anterior, se ha realizado un intento adicional repitiendo la prueba pero marcando como 1 solo los 5 días anteriores a la fecha de recolección. El objetivo ha sido determinar si ajustar aún más los días de clase 1 podría conducir a mejores resultados. El modelo ha sido entrenado durante 5400 *epochs* (21 horas), logrando una precisión del 99.56% en los datos de entrenamiento y del 94.66% en los datos de validación. Sin embargo, los resultados para la clase 1 han mostrado una precisión (18.6%) y un *recall* (19.1%) más bajos que los obtenidos en la prueba anterior. Además, el modelo ha acertado al menos un 1 en 13 de las 40 parcelas de validación (32.5%), lo cual también ha sido considerablemente inferior a los resultados anteriores. Por lo tanto, esta prueba no ha sido satisfactoria.

Primera prueba de regresión

Alternativamente a las pruebas de clasificación, se ha explorado la conversión del problema de clasificación a uno de regresión. Para lograr esto, se han modificado los valores de los 7 días anteriores a la fecha de recolección, que originalmente se habían establecido todos en 1, y se asignaron los siguientes valores: [0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9].

Se ha creado una red neuronal secuencial con múltiples capas. La primera capa ha constado de 10,000 neuronas y utiliza una función de activación *sigmoide*. Esta capa ha recibido una matriz de forma (181,), es decir, un vector de 181 características. A continuación, se ha agregado una capa oculta con 365 neuronas y una función de activación *ReLU*. Después de eso, se ha incluido otra capa con 10 neuronas y una función de activación *softmax*. Por último, se ha añadido una capa de salida con una sola neurona y una función de activación *lineal*. La función de pérdida utilizada ha sido el error cuadrático medio y el optimizador *RMSprop* con una tasa de aprendizaje de 0.001. Las métricas utilizadas para evaluar el modelo han sido el error absoluto medio (MAE) y el error cuadrático medio (MSE). En el código 4 se muestra el código de la red neuronal utilizada.

```
model = Sequential()

model.add(Dense(10000, activation='sigmoid', input_shape=(181,)))
model.add(Dense(365, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.add(Dense(1, activation='linear'))
optimizer = tf.keras.optimizers.RMSprop(0.001)
model.compile(loss='mse',
              optimizer=optimizer,
              metrics=['mae', 'mse'])
model.summary()
```

Código 4: Arquitectura de la red neuronal secuencial con 4 capas.

El modelo ha sido entrenado durante 1500 *epochs* (6 horas) utilizando el 90% de los datos. Los resultados obtenidos para los datos de entrenamiento han sido un MAE de 0.0534 y un MSE de 0.0198. Sin embargo, los resultados para los datos de validación (10%) han sido bastante malos, con un MAE de 0.6211 y un MSE de 0.4408.

Segunda prueba de regresión

En la segunda prueba de regresión, se han realizado algunos cambios en los valores asignados a los 7 días anteriores a la fecha de recolección. En lugar de los antiguos valores que tenían un paso de 0.1 y varían desde 0.3 hasta 1, se han utilizado nuevos valores con un paso de 0.5/7, abarcando desde 0.5 hasta 1: [0.5, 0.57142857, 0.64285714, 0.71428571, 0.78571429, 0.85714286, 0.92857143]. Se ha utilizado la misma red neuronal que en la prueba de regresión anterior, con la única diferencia de que se ha ajustado el número de neuronas en la tercera capa, incrementándolo de 10 a 80. El modelo ha sido entrenado durante 700 *epochs* (3 horas) utilizando el 90% de los datos. Los resultados obtenidos han sido un MAE de 0.0620 y un MSE de 0.0244 para los datos de entrenamiento. Sin embargo, los resultados para los datos de validación (10%) tampoco han sido satisfactorios, con un MAE de 0.7175 y un MSE de 0.5417.

Adicionalmente, se ha realizado una prueba utilizando un modelo que no se basa en redes neuronales. Específicamente, se ha utilizado el modelo *linear_model.LinearRegression()*[28] importado del módulo *sklearn* de Python. Esta prueba ha sido sencilla y no ha requerido mucho tiempo. Los resultados obtenidos para los datos de test han sido un MAE de 0.68 y un MSE de 0.49, los cuales han sido mejores que los obtenidos anteriormente en esta prueba.

Tercera prueba de regresión

En la prueba actual, se ha utilizado una red neuronal distinta con más capas utilizando los mismos datos que en la prueba anterior. Esta red neuronal ha constado de 5 capas, con la primera capa compuesta por 128 neuronas, las tres capas siguientes con 256 neuronas cada una y la capa final con 1 neurona. Las funciones de activación utilizadas en las cuatro primeras capas han sido *ReLU*, mientras que la función de activación de la última capa ha sido *lineal*. Todos los pesos de las capas se han inicializado utilizando el parámetro *kernel_initializer='normal'*[29], lo cual implica una inicialización aleatoria de los pesos con una distribución normal de media cero y varianza uno. Esto se ha hecho con el objetivo de evitar el sobreajuste de los datos. La red se ha compilado utilizando la función de pérdida de error absoluto medio, el optimizador *Adam* y se ha utilizado el error absoluto medio como métrica para evaluar el rendimiento del modelo. La forma de entrada a la red ha constado de 181 características y se ha definido en la capa de entrada. En el código 5 se muestra el código de la red neuronal utilizada.

[28] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
[29] https://keras.io/api/layers/initializers/

```
NN_model = Sequential()
NN_model.add(Dense(128, kernel_initializer='normal',
                   input_dim = dataframeTrain.drop(columns=['parcela','dia','recogido']).shape[1],
                   activation='relu'))
NN_model.add(Dense(256, kernel_initializer='normal',activation='relu'))
NN_model.add(Dense(256, kernel_initializer='normal',activation='relu'))
NN_model.add(Dense(256, kernel_initializer='normal',activation='relu'))
NN_model.add(Dense(1, kernel_initializer='normal',activation='linear'))
NN_model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mean_absolute_error'])
NN_model.summary()
```

Código 5: Arquitectura de la red neuronal secuencial con 5 capas.

En este entrenamiento, se ha utilizado un *callback* que guarda los modelos con el valor de *val_loss* más bajo entre todos los modelos entrenados en los *epochs* anteriores. Aunque se ha entrenado durante 500 *epochs*, el modelo evaluado con los datos de validación ha sido el obtenido en el *epoch* 78, que tuvo el mejor resultado según la métrica *val_loss*. Con este modelo, se ha obtenido un MAE de entrenamiento y validación (10% de los datos) de 0.0325 y 0.0324, respectivamente. Estos resultados han parecido ser significativamente mejores que los de las pruebas anteriores. Sin embargo, al observar las predicciones para los datos de validación, se ha notado que todas las predicciones son prácticamente cero (0.0000010555). Aparentemente, el modelo ha minimizado el MAE asignando un valor cercano a cero a todas las predicciones. Esto se ha debido a la gran desproporción entre las clases, ya que cada parcela tiene 177 datos con etiqueta 0 y solo 8 datos con etiquetas mayores a 0. Como resultado, el error medio no se ha visto muy afectado por esta desproporción.

Siguiendo la misma línea que en la prueba anterior, se han llevado a cabo pruebas adicionales utilizando modelos más sencillos. Al igual que en la prueba anterior de regresión con *LinearRegression()*, se han realizado dos pruebas simples. Una de ellas se ha realizado utilizando *RandomForestRegressor()*[30] del módulo *sklearn.ensemble* de Python, mientras que la otra se ha realizado utilizando *XGBRegressor()*[31] del módulo *xgboost* de Python.

Los resultados obtenidos con los datos de test (10%) han sido los siguientes: 0.05561 para *RandomForestRegressor()* y 0.05889 para *XGBRegressor()*. Estos resultados han sido considerablemente mejores que los obtenidos con las redes neuronales, a pesar de no haber requerido tiempo de entrenamiento adicional.

Resultados

Durante esta iteración se han realizado tres pruebas de clasificación binaria y tres pruebas de regresión con el objetivo de verificar la hipótesis de si es posible crear un modelo de clasificación o regresión que prediga la fecha óptima de cosecha utilizando series temporales como datos de entrada.

En las pruebas de regresión, ha sido más evidente que las hipótesis eran incorrectas. El desequilibrio significativo entre las clases puede llevar a resultados engañosos, donde parece que el error es muy bajo, pero esto se ha debido a que todas las muestras de validación se etiquetaron con un valor cercano a cero.

[30] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[31] https://xgboost.readthedocs.io/en/stable/python/python_api.html

En las pruebas de clasificación, especialmente en la segunda, se ha estado cerca de confirmar la hipótesis. Sin embargo, negar por completo la hipótesis no ha sido posible, ya que en esta prueba se ha logrado predecir correctamente al menos un día de la clase 1 en 21 de las 40 parcelas de validación disponibles, lo cual ha representado más de la mitad. No obstante, al examinar las parcelas, se ha observado que algunas predicciones incorrectas del modelo deberían haber sido correctas según las hipótesis iniciales, pero no lo han sido debido a que se cosecharon más tarde de lo habitual debido a decisiones individuales de cada agricultor.

Anexo III.VI. Iteración 6

Hipótesis de partida

Tras evaluar los resultados de la iteración anterior, en la cual solo se obtuvo un resultado positivo en una de las tres pruebas de clasificación, se plantea la posibilidad de mejorar el rendimiento del modelo aumentando la cantidad de datos de entrenamiento. En concreto, se propone agregar un mes adicional de información correspondiente a enero del año siguiente. Esta adición de datos tiene como objetivo proporcionar al modelo una mayor cantidad de información temporal para mejorar su capacidad de predicción.

Trabajo desarrollado

Se ha obtenido el valor NDVI del mes de enero de 2022 para cada parcela utilizando las mismas solicitudes a la API utilizadas en la iteración 2. Estos valores se han combinado con los datos originales del año 2021, que se han encontrado en el archivo 'csv/parcelasMaiz2.csv', y se han guardado en el archivo 'csv/datosNoEntrenables2122.csv'. Luego se ha aplicado nuevamente el algoritmo de suavizado utilizando los datos de los 13 meses en lugar de los 12 meses anteriores para obtener los valores correctos para cada uno de los 396 días de cada parcela. Con esta información, se ha generado de nuevo el *dataframe* ‘df_ts_180' incluyendo los datos de enero. Los datos de este *dataframe* se han guardado en el archivo 'csv/df_ts2_180dias_bien_enero.csv'.

A continuación, se ha preparado la misma red neuronal que se utilizó en la segunda prueba de clasificación de la iteración anterior, ya que ha sido la que proporcionó los mejores resultados. Esta red neuronal ha constado de 3 capas: la primera capa tiene 10,000 neuronas, la segunda capa tiene 365 neuronas y la tercera capa (capa de salida) tiene 2 neuronas. Se ha utilizado la función de activación *sigmoide* en la primera capa, la función *ReLU* en la segunda capa y la función *softmax* en la capa de salida. La red se ha compilado utilizando la función de pérdida de entropía cruzada categórica, el optimizador *Adam* y se ha evaluado utilizando la métrica de precisión. Además, se ha agregado una capa de normalización de entrada de datos a la red neuronal. En el código 6 se muestra el código de la red neuronal utilizada.

```
normalizer = layers.Normalization()
normalizer.adapt(X)

model = Sequential()
model.add(normalizer)
model.add(Dense(10000, activation='sigmoid', input_shape=(181,)))
model.add(Dense(365, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(),  metrics=['accuracy'])
model.summary()
```

Código 6: Arquitectura de la red neuronal secuencial con 3 capas, además de la de normalización.

Antes de entrenar el modelo, se han creado 4 *callbacks* que guardan los modelos con las mejores métricas de *loss*, *val_loss*, *accuracy* y *val_accuracy*, para mantener el mejor modelo obtenido durante el entrenamiento y no solo el de la última iteración. Así, después de entrenar durante 2000 *epochs* (9 horas), se ha encontrado que el modelo con el valor de *loss* más bajo (0.01434) corresponde a la iteración 1988. Con este modelo, se ha logrado una precisión del 99.55% en los datos de entrenamiento y del 94.07% en los datos de validación. Sin embargo, la precisión y el *recall* de los datos de la clase 1 han sido del 21.9% y 23.4% respectivamente, lo cual ha sido inferior a los resultados obtenidos en la prueba anterior. Además, este modelo solo ha acertado al menos un valor de la clase 1 en 19 de las 40 parcelas de validación (47.5%), lo cual también ha sido inferior a las 21 parcelas acertadas por el modelo de la iteración anterior.

Resultados

Después de observar que la adición de más datos ha empeorado los resultados, se puede concluir que la hipótesis inicial ha sido incorrecta.

Anexo III.VII. Iteración 7

Hipótesis de partida

Se plantea una nueva hipótesis después de las hipótesis anteriores incorrectas. Se considera la posibilidad de predecir la fecha óptima de cosecha utilizando datos más simplificados. Estos datos consisten en dos variables para cada día i: el valor NDVI del propio día i y la diferencia de valor NDVI con el día i-5 (NDVI(i-5) - NDVI(i)). Además, se considera como clase 1 tanto los cinco días anteriores como los cinco días siguientes al día de recogida oficial, con el objetivo de reducir el desbalance entre clases.

Trabajo desarrollado

En el archivo 'csv/datosEntrenables20221122.csv', se ha guardado el nuevo *dataframe* siguiendo la hipótesis planteada inicialmente. A continuación, se han realizado pruebas con dos redes neuronales diferentes:

Primera prueba

Se ha utilizado la misma red neuronal que se empleó en la iteración anterior, compuesta por 3 capas: la primera capa con 10,000 neuronas, la segunda capa con 365 neuronas y la tercera capa (capa de salida) con 2 neuronas. Se ha empleado la función de activación *sigmoide* en la primera capa, la función *ReLU* en la segunda capa y la función *softmax* en la capa de salida. La red se ha compilado utilizando la función de pérdida de entropía cruzada categórica, el optimizador *Adam* y se ha evaluado utilizando la métrica de precisión. Además, se ha incluido una capa de normalización de entrada de datos en la red neuronal. En el código 7 se muestra el código de la red neuronal utilizada.

```
normalizer = layers.Normalization()
normalizer.adapt(X)

model = Sequential()
model.add(normalizer)
model.add(Dense(10000, activation='sigmoid', input_shape=(3,)))
model.add(Dense(365, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(),  metrics=['accuracy'])
model.summary()
```

Código 7: Arquitectura de la red neuronal secuencial con 3 capas, además de la de normalización.

Se ha entrenado durante 1277 *epochs* (1 hora). El mejor modelo obtenido ha alcanzado una precisión del 97.19% tanto en los datos de entrenamiento como en los de validación, ya que ha predicho todos los datos como clase 0.

Segunda prueba

En esta prueba, se ha utilizado una red neuronal distinta con más capas utilizando los mismos datos de entrada. Esta red neuronal ha constado de 5 capas, con la primera capa compuesta por 128 neuronas, las tres capas siguientes con 256 neuronas cada una y la capa final con 2 neuronas. Las funciones de activación utilizadas en las cuatro primeras capas han sido *ReLU*, mientras que la función de activación de la última capa ha sido *softmax*. Todos los pesos de las capas se han inicializado utilizando el parámetro *kernel_initializer='normal'*, lo cual implica una inicialización aleatoria de los pesos con una distribución normal de media cero y varianza uno. Esto se ha hecho con el objetivo de evitar el sobreajuste de los datos. La red se ha compilado utilizando la función de pérdida de entropía cruzada categórica, el optimizador *Adam* y se ha evaluado utilizando la métrica de precisión. La entrada a la red ha constado de 2 características definidas en la capa de entrada. Además, se ha incluido una capa de normalización de entrada de datos en la red neuronal. En el código 8 se muestra el código de la red neuronal utilizada.

```
normalizer = layers.Normalization()
normalizer.adapt(X)

model = Sequential()
model.add(normalizer)
model.add(Dense(128, kernel_initializer='normal',
                input_dim = dataframeTrain.drop(columns=['parcela','dia','recogido']).shape[1],
                activation='relu'))
model.add(Dense(256, kernel_initializer='normal',activation='relu'))
model.add(Dense(256, kernel_initializer='normal',activation='relu'))
model.add(Dense(256, kernel_initializer='normal',activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(),  metrics=['accuracy'])
model.summary()
```

Código 8: Arquitectura de la red neuronal secuencial con 5 capas, además de la de normalización.

Después de entrenar durante 1700 *epochs* (1 hora), la precisión de entrenamiento ha empezado a diferenciarse de la prueba anterior, alcanzando un 97.21%, ya que no ha predicho todas las muestras como clase 0.

Continuando el entrenamiento hasta los 40100 *epochs* (14 horas), se ha obtenido una precisión de entrenamiento y validación del 94.89% y 94.94% respectivamente en las últimas instancias. La clase 1 en los datos de validación ha presentado una precisión y *recall* del 19.9% y 26.4% respectivamente.

Resultados

Tras analizar los resultados desfavorables y observar que a medida que avanzaban los epochs, la precisión con los datos de entrenamiento se alejaba del 100%, se ha podido corroborar que las hipótesis planteadas eran incorrectas. Ha resultado evidente que el modelo no ha sido capaz de distinguir adecuadamente ni los datos de entrenamiento, lo cual es de vital importancia.

Anexo III.VIII. Iteración 8

Hipótesis de partida

En la iteración actual, se plantea la posibilidad de utilizar un nuevo formato de datos de entrada para predecir la fecha óptima de cosecha. Este formato incluye el valor NDVI del día actual (i), así como la pendiente calculada utilizando los valores NDVI de los días i-15, i-10 y i-5. La pendiente se obtiene utilizando la fórmula de la raíz de la hipotenusa: (y1 - y0) / (x1 - x0), donde y0 representa el valor NDVI del día i y x0 es el día correspondiente.

Además, en este planteamiento se decide utilizar únicamente la información de 20 días de cada parcela. Estos 20 días incluyen el día de recogida oficial, los 5 días siguientes y los 14 días anteriores. Para evitar el desbalance de clases, se establece que los 4 días anteriores a la fecha de recogida oficial se clasifiquen como clase 1. Esto resulta en una proporción de 5 elementos de clase 1 por cada 15 elementos de clase 0. Debido a esta proporción, se considera que la peor precisión aceptable para un modelo sería de 0.75, lo cual corresponde a predecir todos los datos como clase 0.

Trabajo desarrollado

A lo largo de las pruebas, se han explorado diferentes configuraciones de redes neuronales y técnicas para tratar los datos de entrada, introduciendo nuevas columnas relacionadas con las existentes. Estas técnicas han incluido:

- Añadir una columna de unos para que la red neuronal aprenda un término constante que pueda resultar necesario para el modelo.
- Expandir las variables para aumentar la complejidad del modelo y capturar relaciones no lineales entre las variables. Esto implica agregar términos polinomiales de orden superior, como x^2, x^3, x^4, y así sucesivamente, a una variable x.
- Añadir columnas que representen la multiplicación de dos variables. Esto permitirá al modelo capturar relaciones complejas entre las variables.

Dado que estos ajustes pueden mitigar el problema del sobreajuste en los modelos, se ha aplicado regularización en algunos de ellos para evitar una especialización excesiva.

Además, en ciertos modelos se ha modificado el umbral utilizado para clasificar las observaciones como pertenecientes a la clase 1 o 0. El objetivo ha sido mejorar la precisión y la exhaustividad del modelo, garantizando que el modelo esté altamente seguro antes de asignar una predicción a la clase 1. Como alternativa, se ha explorado la posibilidad de tratar el problema como uno de regresión en lugar de clasificación.

Antes de comenzar con las pruebas, se han recopilado los datos con los que se trabajará y se han guardado en el archivo 'csv/datosEntrenables20221128.csv'.

Primera prueba

En la primera prueba, se ha utilizado un modelo de red neuronal de 3 capas, empleando las 4 columnas especificadas en la hipótesis sin agregar características adicionales. El modelo ha consistido en un modelo secuencial con una capa de entrada de 10,000 neuronas, una capa oculta de 365 neuronas y una capa de salida de 2 neuronas. Se ha utilizado la función de activación *sigmoide* en la capa de entrada, la función *ReLU* en la capa oculta y la función *softmax* en la capa de salida. El modelo se ha compilado utilizando la función de pérdida de entropía cruzada categórica, el optimizador *Adam* y la métrica de evaluación utilizada ha sido la precisión. Además, se ha incluido una capa de normalización antes de la capa de entrada. En el código 9 se muestra el código de la red neuronal utilizada.

```
normalizer = layers.Normalization()
normalizer.adapt(X)

model = Sequential()
model.add(normalizer)
model.add(Dense(10000, activation='sigmoid', input_shape=(4,)))
model.add(Dense(365, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(),  metrics=['accuracy'])
model.summary()
```

Código 9: Arquitectura de la red neuronal secuencial con 3 capas, además de la de normalización.

El modelo ha sido entrenado durante 13,890 *epochs* (5 horas), y se ha obtenido un mejor resultado de precisión global de 0.77110 en los datos de entrenamiento.

Segunda prueba

En la segunda prueba, se ha agregado una columna de unos a todos los datos utilizados en la prueba anterior y se han repetido las pruebas con el mismo modelo de red neuronal. El modelo ha sido entrenado durante 132,744 *epochs* (47 horas). El mejor modelo obtenido en esta prueba ha logrado una precisión del 82.79% en los datos de entrenamiento y del 65.10% en los datos de validación. El *recall* y la precisión para la clase 1 en los datos de validación han sido del 29.0% y 26.5% respectivamente. El modelo ha acertado al menos un 1 en el 46.5% de las parcelas de validación.

Tercera prueba

En la tercera prueba, se ha llevado a cabo la expansión de las variables 'Pendiente dia -15', 'Pendiente dia -10', 'Pendiente dia -5' y 'ndvi' hasta el grado 5, lo que ha generado 5 versiones de cada una de estas variables como datos de entrada. Además, se ha incluido la columna de unos que mostró buenos resultados en la prueba anterior. El modelo de red neuronal utilizado en esta prueba ha sido el mismo que en las dos pruebas anteriores, pero se ha eliminado la capa de normalización para evitar que afectara el funcionamiento de las nuevas variables. La red neuronal se ha entrenado durante 51,512 *epochs* (18 horas). Sin embargo, el mejor modelo obtenido en esta prueba solo ha logrado una precisión de entrenamiento del 74.5%, por lo que se ha descartado el uso de la expansión de variables, ya que no ha mejorado los resultados de las pruebas anteriores.

Cuarta prueba

En la cuarta prueba, se han añadido nuevas columnas a los datos de entrada que representan las relaciones entre todas las variables. Se han generado 6 columnas adicionales correspondientes a la multiplicación de cada pareja de variables a partir de las 4 variables originales. En esta prueba, se ha modificado la estructura de la red neuronal, la cual consta de 4 capas. La primera capa ha tenido n^3 neuronas con la función de activación *ReLU*. La segunda capa ha tenido n^2 neuronas también con la función de activación *ReLU*. La tercera capa ha tenido n neuronas con la función de activación *ReLU*. Por último, la cuarta capa ha constado de una única neurona con la función de activación *sigmoide*. La red se ha compilado utilizando la función de pérdida *binary_crossentropy*, el optimizador *Adam* y se han incluido dos métricas de evaluación: precisión binaria (*binary_accuracy*) y *recall*. Además, la precisión binaria se ha evaluado con un umbral de 0.6 en lugar del umbral predeterminado de 0.5. En el código 10 se muestra el código de la red neuronal utilizada.

```
model = Sequential()
model.add(Dense(n**3, activation='relu', input_shape=(n,)))
model.add(Dense(n**2, activation='relu'))
model.add(Dense(n, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=Adam(),
              metrics=[tf.keras.metrics.BinaryAccuracy(name='binary_accuracy', threshold=0.6),
                       keras.metrics.Recall(name='recall')])
model.summary()
```

Código 10: Arquitectura de la red neuronal secuencial con 4 capas.

Se ha entrenado durante 1,180,000 *epochs* (64 horas). El mejor modelo obtenido en esta prueba ha alcanzado una precisión del 89.52% en los datos de entrenamiento y del 67.38% en los datos de validación. La precisión y el *recall* de la clase 1 en los datos de validación han sido del 33.1% y 30% respectivamente. Además, se ha acertado al menos un 1 en 25 de las 40 parcelas de validación. Estos resultados han sido los mejores obtenidos hasta el momento.

Quinta prueba

En la quinta prueba, se ha repetido la prueba anterior tratando el problema como un caso de regresión. Se han asignado valores objetivos específicos a los datos de la siguiente manera: los 9º y 16º datos de cada parcela tienen un valor objetivo de 0.6, mientras que el 8º y 17º datos tienen un valor objetivo de 0.3. Esto se ha hecho con el propósito de crear una transición gradual de la clase 1 a la clase 0. Por lo tanto, las salidas de los datos se han representado de la siguiente forma: [0, 0, 0, 0, 0, 0, 0, 0, 0.3, 0.6, 1, 1, 1, 1, 1, 0.6, 0.3, 0, 0, 0].

Se ha utilizado la misma red neuronal que en la prueba anterior, pero se ha ajustado la compilación y las métricas para adaptarlas a un problema de regresión. La red se ha compilado utilizando la función de pérdida de error cuadrático medio (MSE) y el optimizador *Adam*. Se han utilizado varias métricas para evaluar el rendimiento del modelo durante el entrenamiento y la validación, que han incluido el error cuadrático medio (*mean_squared_error*), el error absoluto medio (*mean_absolute_error*), el porcentaje de error absoluto medio (*mean_absolute_percentage_error*) y la precisión categórica (*categorical_accuracy*). En el código 11 se muestra el código de la red neuronal utilizada.

```
model = Sequential()
model.add(Dense(n**3, activation='relu', input_shape=(n,)))
model.add(Dense(n**2, activation='relu'))
model.add(Dense(n, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='mse', optimizer='adam', metrics=[metrics.mean_squared_error,
                  metrics.mean_absolute_error,
                  metrics.mean_absolute_percentage_error,
                  metrics.categorical_accuracy])
model.summary()
```

Código 11: Arquitectura de la red neuronal secuencial con 4 capas.

Se ha entrenado durante 535,000 *epochs* (27 horas). El mejor modelo obtenido ha tenido un error cuadrático medio de 0.00846 en los datos de entrenamiento. Para comparar con el modelo de clasificación, se ha calculado la precisión del modelo utilizando un umbral de 0.1 para determinar si la clase predicha por el modelo ha estado lo suficientemente cerca de la clase real de los datos. Según esta métrica, si la clase de un dato es 0.3 y el modelo ha predecido un valor de 0.21, se ha considerado la predicción como correcta debido a que está entre 0.2 y 0.4. Bajo esta evaluación, la precisión global del modelo con los datos de entrenamiento ha sido del 87.64%, mientras que con los datos de evaluación ha sido del 35.38%. La precisión y el *recall* de la clase 1 en los datos de validación han sido del 18.4% y 12.7% respectivamente, lo cual ha sido un resultado muy pobre.

Sexta prueba

En la sexta prueba, tras las dos pruebas anteriores, se ha planteado la posibilidad de eliminar las columnas que multiplicaban el NDVI, ya que se ha considerado que no tenían una relación significativa con las variables de pendiente. Por lo tanto, se han eliminado estas columnas de los datos de entrada y se han probado tres redes neuronales diferentes con una regularización de 0.001 para evitar el sobreajuste en los modelos.

Las tres redes neuronales se han basado en el mismo modelo utilizado en la cuarta prueba de esta iteración. En la primera red neuronal se han mantenido todas las capas, en la segunda red se ha eliminado la primera capa y en la tercera red se han eliminado la primera y la segunda capa. Estos cambios se han realizado con el objetivo de encontrar un equilibrio entre el rendimiento del modelo y la prevención del sobreajuste.

Los modelos han sido entrenados durante 545000 *epochs* (21 horas), 310000 *epochs* (4 horas) y 155000 *epochs* (2 horas), respectivamente. Sin embargo, los dos últimos modelos no han mostrado indicios de alcanzar una precisión global en los datos de entrenamiento del 76%, por lo que no se ha considerado que valiera la pena seguir trabajando en ellos. El primer modelo, el cual ha conservado todas las capas de la red neuronal, ha logrado alcanzar una precisión del 79%. Aunque este modelo ha presentado un mejor desempeño en comparación con los otros dos, tampoco se ha considerado satisfactorio en términos generales.

Séptima prueba

En la séptima y última prueba, se han repetido los experimentos de la prueba anterior, pero se ha cambiado el término de regularización de 0.001 a 0.00001 con la esperanza de lograr un buen ajuste a los datos de entrenamiento sin llegar al sobreajuste.

Se han entrenado tres redes neuronales con todas las capas, con una capa menos y con dos capas menos durante 155000 *epochs* (6 horas), 540000 *epochs* (8 horas) y 310000 *epochs* (4 horas), respectivamente. Sin embargo, ninguno de los modelos ha destacado, ya que no han logrado mejorar los resultados de las pruebas anteriores.

Resultados

Durante la iteración, se han realizado 7 pruebas diferentes para investigar la hipótesis de que utilizando 20 datos de cada parcela, compuestos por el valor NDVI del día i y las pendientes con respecto a los días i-15, i-10 y i-5, se podría predecir de manera precisa la fecha óptima de cosecha.

Estas pruebas han involucrado diversas técnicas, como agregar una columna de unos, expandir las variables y agregar columnas con relaciones entre las variables. También se ha probado el modelo como un problema de regresión. Sin embargo, ninguna de estas pruebas ha proporcionado resultados que permitan respaldar la hipótesis planteada.

Anexo III.IX. Iteración 9

Hipótesis de partida

La hipótesis planteada en esta iteración consiste en que al combinar el valor NDVI del día i, las pendientes con respecto a los días i-15, i-10 y i-5, junto con la información de la humedad relativa, precipitación, temperatura media y presión del día, se puede lograr una predicción más precisa de la fecha óptima de cosecha. Al igual que en la hipótesis anterior, se utilizan 20 días de cada parcela, clasificando los 4 días previos a la fecha oficial de recogida como clase 1, y los otros 15 días como clase 0. Para obtener estas 4 nuevas características, se requiere realizar peticiones a la API del cliente GeosLab.

Trabajo desarrollado

En esta iteración se ha llevado a cabo la obtención de nuevos datos, que incluyen la humedad relativa, temperatura, precipitación y presión. Además, se ha implementado una nueva medida de éxito, la cual se comentará más adelante. Asimismo, se ha realizado la creación de diversos *dataframes* y se ha llevado a cabo comparaciones entre ellos.

Obtención de los nuevos datos

En la búsqueda de mejorar los resultados de la iteración anterior, se ha planteado la idea de incorporar información climática a cada dato. Para lograr esto, se ha dispuesto de dos APIs: una que proporciona las estaciones meteorológicas más cercanas a una parcela y otra que ofrece información diaria sobre el clima para una estación meteorológica específica.

La propuesta ha consistido en añadir datos sobre la humedad relativa, precipitación, temperaturas y presión atmosférica del día anterior a los registros. Esta elección se ha debido a que, si en un futuro se desea utilizar el modelo para predecir si el día actual es adecuado para la cosecha, no se tendría acceso a la información climática del mismo día, ya que aún no estaría disponible en la API. En cuanto a la precipitación, ha surgido la duda sobre qué enfoque sería más efectivo: utilizar únicamente la precipitación del día anterior o emplear la precipitación acumulada de los tres días anteriores. Para abordar esta incertidumbre, se han incluido ambas variables en los datos para su posterior comparación.

Las dos peticiones API que se han utilizado para obtener la información climática son las siguientes:

Primera petición: Obtención de la estación meteorológica más cercana a una parcela.

- Direccion: http://maps.agroslab.com/AgroslabHttpServlet/AgroslabHttpServlet
- Método: POST
- Parámetros de entrada (JSON): {“operation”: “aemetestacionesdwithin”, “id”: “22-27-0-1-503-5017-1”, “distanceinkilometers”: 50}
- Respuesta (JSON): {'provincia': 22, 'municipio': 27, 'agregado': 0, 'zona': 1, 'poligono': 503, 'parcela': 5017, 'recinto': 1, 'aemet_estaciones': [{'id': '9898', 'nombre': 'HUESCA, AEROPUERTO', 'distancia (km)': 32.73474482149}, {'id': '9201K', 'nombre': 'JACA',

'distancia (km)': 41.0159179976}, {'id': '9208E', 'nombre': 'ARAGÜÉS DEL PUERTO', 'distancia (km)': 46.16255470142}]}

Segunda petición: Obtención de información climática diaria para una estación meteorológica determinada.

- Direccion: http://maps.agroslab.com/AgroslabHttpServlet/AgroslabHttpServlet
- Método: POST
- Parámetros de entrada (JSON): {“operation”: “aemetclimatologiadiaria”, "initdate":"01-01-2022", "enddate":"03-01-2022", "idema" : 9898}
- Respuesta (JSON): [{'fecha': '2022-01-01', 'indicativo': '9898', 'nombre': 'HUESCA, AEROPUERTO', 'provincia': 'HUESCA', 'altitud': '546', 'tmed': '10,0', 'prec': '0,0', 'tmin': '5,2', 'horatmin': '23:40', 'tmax': '14,7', 'horatmax': '13:40', 'dir': '26', 'velmedia': '1,4', 'racha': '3,9', 'horaracha': '12:30', 'sol': '6,9', 'presMax': '964,8', 'horaPresMax': '11', 'presMin': '962,1', 'horaPresMin': '15'}, {'fecha': '2022-01-02', 'indicativo': '9898', 'nombre': 'HUESCA, AEROPUERTO', 'provincia': 'HUESCA', 'altitud': '546', 'tmed': '10,0', 'prec': '0,0', 'tmin': '5,3', 'horatmin': '23:59', 'tmax': '14,8', 'horatmax': '13:50', 'dir': '25', 'velmedia': '1,4', 'racha': '5,0', 'horaracha': '15:10', 'sol': '6,2', 'presMax': '964,7', 'horaPresMax': '10', 'presMin': '961,9', 'horaPresMin': '14'}, {'fecha': '2022-01-03', 'indicativo': '9898', 'nombre': 'HUESCA, AEROPUERTO', 'provincia': 'HUESCA', 'altitud': '546', 'tmed': '8,8', 'prec': '0,0', 'tmin': '4,1', 'horatmin': '06:20', 'tmax': '13,6', 'horatmax': '13:50', 'dir': '99', 'velmedia': '1,4', 'racha': '3,3', 'horaracha': 'Varias', 'sol': '6,3', 'presMax': '962,5', 'horaPresMax': '00', 'presMin': '954,1', 'horaPresMin': '24'}]

Una vez conocidas las dos peticiones API, se ha procedido a obtener los datos. Se ha observado que no se dispone de información sobre la humedad relativa, por lo que no ha sido posible utilizar este dato.

Para obtener la información de temperaturas y precipitaciones, se han agregado al *dataframe* más reciente las columnas "tmed", "prec", "precSum3" y "estacion" con valores no inicializados, que se han ido completando durante las peticiones. Luego, para cada parcela, se ha obtenido la estación meteorológica más cercana que contenga estos datos y se han rellenado las columnas correspondientes.

En cuanto a la información de presión atmosférica, se ha seguido un procedimiento similar, agregando al *dataframe* las columnas "pres" y "estacionPres" sin inicializar, ya que no todas las estaciones han contado con datos de presión. Además, como la respuesta de la petición ha incluido tanto la presión máxima del día ("presMax") como la presión mínima del día ("presMin"), se ha calculado el promedio de ambas y se ha almacenado en la nueva variable "pres".

Después de realizar todas las peticiones, se ha observado que muchas parcelas no han sido completadas, lo que ha resultado en un menor número de parcelas disponibles para el entrenamiento y la validación del modelo.

Nueva medida de éxito:

Se ha incorporado una nueva medida de éxito basada en la probabilidad de acierto dividida por la distancia respecto al día de cosecha oficial. Esta medida ha tenido como objetivo penalizar los resultados buenos que no se encuentren en el día exacto de recogida. Para calcular esta medida, se ha utilizado el logaritmo de la distancia entre el día de predicción y el día de recogida oficial, junto a la probabilidad predicha por el modelo.

La fórmula utilizada ha sido la siguiente: $\frac{ypred}{1+log(1+|distancia|)}$, donde 'ypred' representa la probabilidad predicha por el modelo para el día 'i' de la parcela 'j', y 'distancia' es la diferencia entre el día de predicción 'i' y el día de recogida oficial 'l' de esa parcela 'j'.

Se ha obtenido el valor máximo de éxito para cada parcela y se han analizado el mínimo, el máximo y el promedio del éxito obtenido entre todas las parcelas. Estos valores han proporcionado información sobre el desempeño del modelo en relación con la distancia respecto al día de cosecha oficial.

Dataframes usados:

Se han utilizado varios *dataframes* para realizar las comparaciones. A continuación se presenta un resumen de cada *dataframe* junto con su nombre abreviado para facilitar la comprensión de las explicaciones posteriores:

- *Dataframe* **TP**: Contiene los siguientes datos: NDVI, pendientes [-5, -10, -15] días, multiplicación de las columnas de NDVI y pendientes, temperatura media (tmed), precipitación (prec) y una columna de unos. En el *dataframe* 1 se muestra visualmente este formato de datos.

	pendiente dia -15	pendiente dia -10	pendiente dia -5	ndvi	col 1 * col 2	col 1 * col 3	col 1 * col 4	col 2 * col 3	col 2 * col 4	col 3 * col 4	tmed	prec	unos
0	0.013078	0.008582	-0.002236	0.456587	0.000112	-0.000029	0.005971	-0.000019	0.003919	-0.001021	17.4	0.0	1
1	0.011039	0.005081	-0.006042	0.467762	0.000056	-0.000067	0.005164	-0.000031	0.002377	-0.002826	18.0	0.0	1
2	0.008906	0.001723	-0.008617	0.478748	0.000015	-0.000077	0.004264	-0.000015	0.000825	-0.004126	17.6	0.0	1
3	0.006757	-0.001304	-0.009899	0.488632	-0.000009	-0.000067	0.003301	0.000013	-0.000637	-0.004837	15.5	0.0	1
4	0.004652	-0.003832	-0.010032	0.496736	-0.000018	-0.000047	0.002311	0.000038	-0.001903	-0.004983	13.7	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4367	0.005049	0.004059	0.004025	0.414865	0.000020	0.000020	0.002094	0.000016	0.001684	0.001670	16.8	0.0	1
4368	0.005029	0.004006	0.004614	0.409131	0.000020	0.000023	0.002057	0.000018	0.001639	0.001888	16.4	0.0	1
4369	0.004980	0.004131	0.005174	0.402966	0.000021	0.000026	0.002007	0.000021	0.001665	0.002085	17.8	0.2	1
4370	0.004917	0.004413	0.005670	0.396435	0.000022	0.000028	0.001949	0.000025	0.001750	0.002248	18.8	0.0	1
4371	0.004868	0.004802	0.006100	0.389605	0.000023	0.000030	0.001896	0.000029	0.001871	0.002377	18.8	0.0	1

Dataframe 1: Formato de los datos de entrada del dataframe TP.

- *Dataframe* **TPSN**: Contiene los siguientes datos: NDVI, pendientes [-5, -10, -15] días, multiplicación de las pendientes entre ellas, temperatura media (tmed), precipitación (prec) y una columna de unos. En el *dataframe* 2 se muestra visualmente este formato de datos.

	pendiente dia -15	pendiente dia -10	pendiente dia -5	col 1 * col 2	col 1 * col 3	col 2 * col 3	tmed	prec	ndvi	unos
0	0.013078	0.008582	-0.002236	0.000112	-0.000029	-0.000019	17.4	0.0	0.456587	1
1	0.011039	0.005081	-0.006042	0.000056	-0.000067	-0.000031	18.0	0.0	0.467762	1
2	0.008906	0.001723	-0.008617	0.000015	-0.000077	-0.000015	17.6	0.0	0.478748	1
3	0.006757	-0.001304	-0.009899	-0.000009	-0.000067	0.000013	15.5	0.0	0.488632	1
4	0.004652	-0.003832	-0.010032	-0.000018	-0.000047	0.000038	13.7	0.0	0.496736	1
...	...	...	...	...	...	...	...	...	...	...
4367	0.005049	0.004059	0.004025	0.000020	0.000020	0.000016	16.8	0.0	0.414865	1
4368	0.005029	0.004006	0.004614	0.000020	0.000023	0.000018	16.4	0.0	0.409131	1
4369	0.004980	0.004131	0.005174	0.000021	0.000026	0.000021	17.8	0.2	0.402966	1
4370	0.004917	0.004413	0.005670	0.000022	0.000028	0.000025	18.8	0.0	0.396435	1
4371	0.004868	0.004802	0.006100	0.000023	0.000030	0.000029	18.8	0.0	0.389605	1

Dataframe 2: Formato de los datos de entrada del dataframe TPSN.

- Dataframe **TP3**: Contiene los siguientes datos: NDVI, pendientes [-5, -10, -15] días, multiplicación de las columnas de NDVI y pendientes, temperatura media (tmed), precipitación acumulada de 3 días (precSum3) y una columna de unos. En el *dataframe* 3 se muestra visualmente este formato de datos.

	pendiente dia -15	pendiente dia -10	pendiente dia -5	ndvi	col 1 * col 2	col 1 * col 3	col 1 * col 4	col 2 * col 3	col 2 * col 4	col 3 * col 4	tmed	precSum3	unos
0	0.013078	0.008582	-0.002236	0.456587	0.000112	-0.000029	0.005971	-0.000019	0.003919	-0.001021	17.4	0.0	1
1	0.011039	0.005081	-0.006042	0.467762	0.000056	-0.000067	0.005164	-0.000031	0.002377	-0.002826	18.0	0.0	1
2	0.008906	0.001723	-0.008617	0.478748	0.000015	-0.000077	0.004264	-0.000015	0.000825	-0.004126	17.6	0.0	1
3	0.006757	-0.001304	-0.009899	0.488632	-0.000009	-0.000067	0.003301	0.000013	-0.000637	-0.004837	15.5	0.0	1
4	0.004652	-0.003832	-0.010032	0.496736	-0.000018	-0.000047	0.002311	0.000038	-0.001903	-0.004983	13.7	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8155	0.005049	0.004059	0.004025	0.414865	0.000020	0.000020	0.002094	0.000016	0.001684	0.001670	16.8	0.0	1
8156	0.005029	0.004006	0.004614	0.409131	0.000020	0.000023	0.002057	0.000018	0.001639	0.001888	16.4	0.0	1
8157	0.004980	0.004131	0.005174	0.402966	0.000021	0.000026	0.002007	0.000021	0.001665	0.002085	17.8	0.2	1
8158	0.004917	0.004413	0.005670	0.396435	0.000022	0.000028	0.001949	0.000025	0.001750	0.002248	18.8	0.2	1
8159	0.004868	0.004802	0.006100	0.389605	0.000023	0.000030	0.001896	0.000029	0.001871	0.002377	18.8	0.2	1

Dataframe 3: Formato de los datos de entrada del dataframe TP3.

- *Dataframe* **TP3SN**: Contiene los siguientes datos: NDVI, pendientes [-5, -10, -15] días, multiplicación de las pendientes entre ellas, temperatura media (tmed), precipitación acumulada de 3 días (precSum3) y una columna de unos. En el *dataframe* 4 se muestra visualmente este formato de datos.

	pendiente dia -15	pendiente dia -10	pendiente dia -5	col 1 * col 2	col 1 * col 3	col 2 * col 3	tmed	precSum3	ndvi	unos
0	0.013078	0.008582	-0.002236	0.000112	-0.000029	-0.000019	17.4	0.0	0.456587	1
1	0.011039	0.005081	-0.006042	0.000056	-0.000067	-0.000031	18.0	0.0	0.467762	1
2	0.008906	0.001723	-0.008617	0.000015	-0.000077	-0.000015	17.6	0.0	0.478748	1
3	0.006757	-0.001304	-0.009899	-0.000009	-0.000067	0.000013	15.5	0.0	0.488632	1
4	0.004652	-0.003832	-0.010032	-0.000018	-0.000047	0.000038	13.7	0.0	0.496736	1
...	...	...	...	...	...	...	...	...	...	...
8155	0.005049	0.004059	0.004025	0.000020	0.000020	0.000016	16.8	0.0	0.414865	1
8156	0.005029	0.004006	0.004614	0.000020	0.000023	0.000018	16.4	0.0	0.409131	1
8157	0.004980	0.004131	0.005174	0.000021	0.000026	0.000021	17.8	0.2	0.402966	1
8158	0.004917	0.004413	0.005670	0.000022	0.000028	0.000025	18.8	0.2	0.396435	1
8159	0.004868	0.004802	0.006100	0.000023	0.000030	0.000029	18.8	0.2	0.389605	1

Dataframe 4: Formato de los datos de entrada del dataframe TP3SN.

- Dataframe TPPSN: Contiene los siguientes datos: NDVI, pendientes [-5, -10, -15] días, multiplicación de las pendientes entre ellas, temperatura media (tmed), precipitación (prec), presión (pres) y una columna de unos. En el *dataframe* 5 se muestra visualmente este formato de datos.

	pendiente dia -15	pendiente dia -10	pendiente dia -5	col 1 * col 2	col 1 * col 3	col 2 * col 3	tmed	prec	ndvi	pres	unos
0	0.013078	0.008582	-0.002236	0.000112	-0.000029	-0.000019	17.4	0.0	0.456587	959.5	1
1	0.011039	0.005081	-0.006042	0.000056	-0.000067	-0.000031	18.0	0.0	0.467762	961.3499999999999	1
2	0.008906	0.001723	-0.008617	0.000015	-0.000077	-0.000015	17.6	0.0	0.478748	957.0	1
3	0.006757	-0.001304	-0.009899	-0.000009	-0.000067	0.000013	15.5	0.0	0.488632	953.3499999999999	1
4	0.004652	-0.003832	-0.010032	-0.000018	-0.000047	0.000038	13.7	0.0	0.496736	955.45	1
...	...	...	...	...	...	...	...	...	...	...	...
8155	0.005049	0.004059	0.004025	0.000020	0.000020	0.000016	16.8	0.0	0.414865	988.5	1
8156	0.005029	0.004006	0.004614	0.000020	0.000023	0.000018	16.4	0.0	0.409131	987.8	1
8157	0.004980	0.004131	0.005174	0.000021	0.000026	0.000021	17.8	0.2	0.402966	989.25	1
8158	0.004917	0.004413	0.005670	0.000022	0.000028	0.000025	18.8	0.0	0.396435	992.95	1
8159	0.004868	0.004802	0.006100	0.000023	0.000030	0.000029	18.8	0.0	0.389605	994.4000000000001	1

Dataframe 5: Formato de los datos de entrada del dataframe TPPSN.

Estos *dataframes* representan diferentes combinaciones de variables utilizadas en el modelo, incluyendo información sobre el NDVI, pendientes, temperatura, precipitación y presión.

Comparación:

En la comparación realizada, se ha utilizado el mismo modelo de red neuronal con 3 capas. La primera capa ha tenido n^2 neuronas y ha utilizado la función de activación *ReLU*, la segunda capa ha tenido n neuronas y también ha utilizado *ReLU* como función de activación, y la tercera capa ha consistido en una única neurona con función de activación *sigmoide*. El modelo se ha compilado utilizando la función de pérdida *binary_crossentropy*, el optimizador *Adam*, y se han incluido dos métricas de evaluación: precisión binaria (*binary_accuracy*) con un umbral de 0.6 y *recall*. En el código 12 se muestra el código de la red neuronal utilizada.

```
model = Sequential()
model.add(Dense(n**2, activation='relu', input_shape=(n,), kernel_regularizer=l2(regularizacion)))
model.add(Dense(n, activation='relu', kernel_regularizer=l2(regularizacion)))
model.add(Dense(1, activation='sigmoid', kernel_regularizer=l2(regularizacion)))
model.compile(loss='binary_crossentropy', optimizer=Adam(),
              metrics=[tf.keras.metrics.BinaryAccuracy(name='binary_accuracy', threshold=0.6),
                       keras.metrics.Recall(name='recall')])
model.summary()
```

Código 12: Arquitectura de la red neuronal secuencial con 3 capas.

A continuación se presentan los resultados obtenidos en las primeras pruebas (tabla 3).

Datos	Precisión datos validación	Parcelas acertadas (%)	Éxito (min,max, promedio)
TP	0.80952	14/21 (66.6%)	0.367, 0.779, 0.510
TP (reg = 0.0001)	0.8	13/21 (61.9%)	0.252, 0.717, 0.508
TPSN	0.80952	17/21 (80.9%)	0.254, 0.797, 0.569
TPSN (reg = 0.0001)	0.83571	18/21 (85.7%)	0.370, 0.807, 0.600

Tabla 3: Resultados de pruebas con los dataframes TP y TPSN con y sin regularización. Las parcelas incompletas se excluyeron debido a problemas en la obtención de precipitaciones y temperaturas.

En las pruebas realizadas, se observó que el porcentaje de parcelas clasificadas correctamente era alto, lo que sugiere buenos resultados.

Sin embargo, se ha notado que las solicitudes a la API a menudo han fallado debido a la sobrecarga. Para obtener información de todas las parcelas posibles, se han repetido las solicitudes, aunque algunas parcelas han seguido fallando (8 parcelas en total). Luego se han repetido las mismas pruebas con más datos disponibles (tabla 4).

Datos	Precisión datos validación	Parcelas acertadas (%)	Éxito (min,max, promedio)
TP	0.77692	21/39 (53.8%)	0.166, 0.762, 0.473
TP (reg = 0.0001)	0.78205	17/39 (43.5%)	0.111, 0.775, 0.463
TPSN	0.77564	23/39 (58.9%)	0.161, 0.786, 0.476
TPSN (reg = 0.0001)	0.78077	21/39 (53.8%)	0.250, 0.708, 0.491

Tabla 4: Resultados de pruebas con los dataframes TP y TPSN con y sin regularización. Se incluyeron todas las parcelas tras repetir las peticiones para obtener las precipitaciones y temperaturas.

A pesar de tener más datos para entrenar, estos nuevos resultados han sido peores que los anteriores. Esto puede indicar que el modelo se ha especializado demasiado o que no ha sido adecuado para este tipo de datos. También es posible que, al quedarse con menos datos, se haya coincidido en que todos los datos eran de Zaragoza o Huesca.

Luego se han realizado pruebas utilizando *dataframes* que incluían la precipitación acumulada de los 3 días anteriores en lugar de solo el día anterior. Los resultados se presentan en la tabla 5.

Datos	Precisión datos validación	Parcelas acertadas (%)	Éxito (min,max, promedio)
TP3	0.81154	30/39 (76.9%)	0.051, 0.999, 0.624
TP3 (reg = 0.0001)	0.78846	16/39 (41.0%)	0.230, 0.743, 0.499
TP3SN	0.80128	22/39 (58.9%)	0.237, 0.849, 0.502
TP3SN (reg = 0.0001)	0.8	20/39 (51.2%)	0.171 ,0.831, 0.493

Tabla 5: Resultados de pruebas con los dataframes TP3 y TP3SN con y sin regularización. Se incluyeron todas las parcelas en estas pruebas.

Excepto en el caso del *dataframe* TP3, los demás resultados han sido peores al usar la precipitación acumulada en lugar de la del día anterior.

Finalmente, se ha probado con el último *dataframe* que incluía la precipitación de un día y la presión en lugar de la precipitación de los últimos 3 días (tabla 6).

Datos	Precisión datos validación	Parcelas acertadas (%)	Éxito (min,max, promedio)
TPPSN (reg = 0.0001)	0.75	-	-
TPPSN	0.76429	-	-

Tabla 6: Resultados de pruebas con el dataframe TPPSN con y sin regularización. Se incluyeron todas las parcelas en estas pruebas.

La precisión ha sido tan baja que no se examinaron las parcelas clasificadas correctamente ni el éxito. Es posible que se requiera una normalización debido a que los datos de presión utilizados estaban en un rango estrecho de 950 a 960 hPa. Sin normalización, el modelo no ha logrado mejorar, por lo que se debería considerar la normalización al trabajar con datos de presión.

Resultados

Durante esta iteración, se ha recopilado información sobre la temperatura media, precipitación y presión diaria para ser utilizada en diversos *dataframes*, los cuales han sido comparados para diferentes valores de regularización.

Al analizar los resultados obtenidos en esta iteración, no se ha podido afirmar ni descartar la hipótesis planteada. Si bien los resultados no han superado un umbral considerado suficiente para ser evaluados como éxito (un valor intermedio entre 0.80 y 0.85), se han observado mejoras en comparación con cualquier otra hipótesis previa. Esto sugiere que se podría estar en la dirección correcta.

Anexo III.X. Iteración 10

Hipótesis de partida

Tras las iteraciones previas, se identifican varios problemas que podrían estar afectando los resultados, como el desequilibrio de clases y la falta de normalización de los datos. Además, se sospecha que los resultados pueden estar influenciados por la selección específica de las parcelas de validación, lo que podría llevar a una separación de datos que ocasionalmente sea mejor o peor de lo esperado. Por lo tanto, se plantea la posibilidad de utilizar los mismos *dataframes* utilizados en la iteración anterior, junto con la implementación de validación cruzada, la corrección del desequilibrio de clases y la normalización de los datos de entrada. De esta manera, se espera desarrollar un modelo capaz de predecir la fecha óptima de cosecha de manera más precisa y confiable.

Trabajo desarrollado

Modelos no basados en redes neuronales

Se ha considerado la posibilidad de utilizar modelos diferentes a las redes neuronales, ya que lo más complejo no siempre es necesariamente lo más efectivo. Se han evaluado modelos más simples que requieren poco tiempo de entrenamiento, tales como:

- *SVC*[32] (*Support Vector Classifier*): Un algoritmo de clasificación que busca encontrar el hiperplano óptimo para separar diferentes clases en un espacio de características utilizando vectores de soporte.
- *GaussianNB*[33] (*Naive Bayes Gaussiano*): Un modelo de clasificación probabilístico que asume una distribución gaussiana para las características y utiliza el teorema de Bayes para realizar la clasificación.
- *KNeighborsClassifier*[34]: Un algoritmo de clasificación que asigna una etiqueta a un punto de datos basándose en la mayoría de sus vecinos más cercanos en el espacio de características.
- *GradientBoostingClassifier*[35]: Un algoritmo de aprendizaje en conjunto que combina múltiples árboles de decisión débiles, corrigiendo iterativamente los errores de predicción de los árboles anteriores.
- *AdaBoostClassifier*[36]: Un algoritmo de aprendizaje en conjunto que entrena varios clasificadores débiles secuencialmente, asignando pesos a las instancias de entrenamiento para enfocarse en los casos más difíciles y combinando las predicciones de los clasificadores para obtener una predicción final.
- *MLPClassifier*[37] (*Multilayer Perceptron*): Un tipo de red neuronal artificial que consta de múltiples capas de neuronas, incluyendo una capa de entrada, una o más capas ocultas

[32] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[33] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[34] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[35] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[36] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
[37] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

y una capa de salida. Es capaz de aprender relaciones complejas entre las características de entrada y las etiquetas de salida.

- *RandomForest*: Un algoritmo de aprendizaje en conjunto basado en árboles de decisión, donde múltiples árboles se entrenan en subconjuntos aleatorios de características y se combina su predicción para obtener un resultado final.
- *Xgboost*[38] / *XGBClassifier*[39]: Una implementación optimizada y escalable del algoritmo de *Gradient Boosting* que utiliza árboles de decisión como clasificadores débiles.
- *Lgbm*[40] (*LightGBM*) / *LGBMClassifier*[41]: Una biblioteca de *Gradient Boosting* conocida por su velocidad y eficiencia. Utiliza un algoritmo de construcción de árbol basado en hojas y una estrategia de crecimiento basada en histogramas para mejorar el rendimiento.

Algunos de estos modelos tienen versiones simples que funcionan de manera similar a los demás, como *XGBClassifier* y *LGBMClassifier*. Sin embargo, otras versiones más complejas requieren la especificación de diferentes hiperparámetros y una manipulación diferente de los datos de entrada para obtener mejores resultados, como en el caso de *Xgboost* y *Lgbm*.

Métrica ROC-AUC

Se ha considerado necesaria la utilización de una métrica adicional a la precisión global para evaluar el rendimiento de los modelos, debido a la posible influencia de los datos utilizados. En este sentido, la métrica ROC-AUC (*Receiver Operating Characteristic - Area Under the Curve*) resulta relevante en la evaluación de modelos de clasificación binaria. Su cálculo se basa en la construcción de la curva ROC, que muestra la capacidad del modelo para distinguir correctamente entre las dos clases, y en la determinación del área bajo dicha curva AUC (figura 31).

La curva ROC muestra la tasa de verdaderos positivos (sensibilidad) en función de la tasa de falsos positivos (1 - especificidad) a medida que se varía el umbral de clasificación del modelo. Un modelo perfecto tendría una curva ROC que alcanza el punto (0, 1) en el gráfico, lo que indica una sensibilidad del 100% y una especificidad del 100%.

El valor del AUC (Area Under the Curve) varía entre 0 y 1, donde 1 indica un modelo perfecto, mientras que un valor de 0.5 indica que el modelo no tiene capacidad de discriminación y es equivalente a una clasificación aleatoria.

[38] https://xgboost.readthedocs.io/en/stable/
[39] https://xgboost.readthedocs.io/en/stable/python/python_api.html
[40] https://lightgbm.readthedocs.io/en/latest/Python-Intro.html
[41] https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html

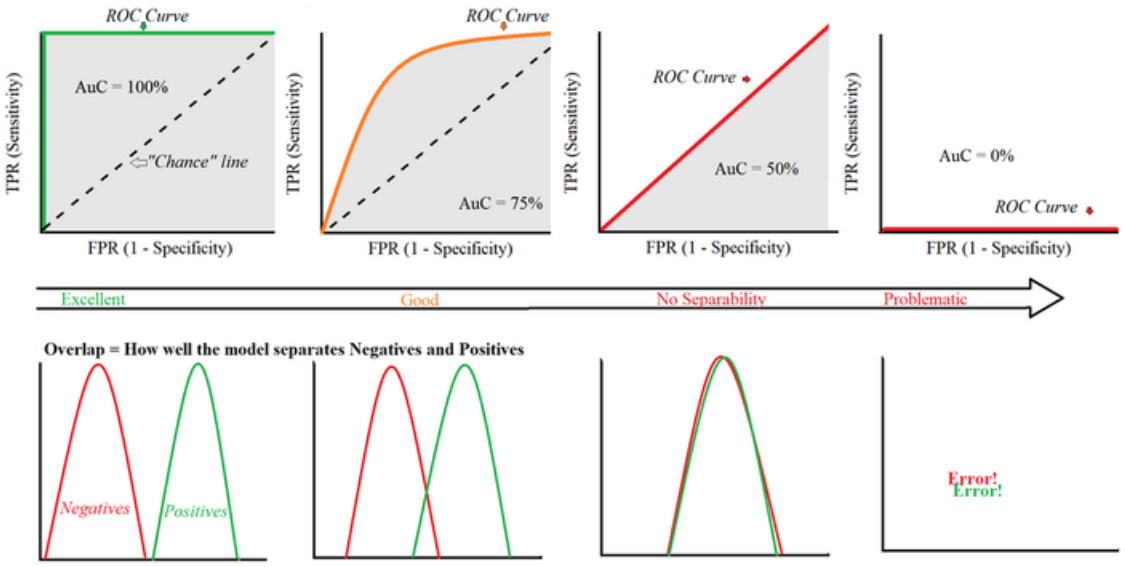


Figura 31: Representación gráfica de la métrica ROC-AUC.

Equilibrado de muestras

En esta etapa del proceso, aún se han enfrentado diversos problemas que pueden estar afectando los resultados. Uno de ellos ha sido el desequilibrio de las clases, lo que ha podido provocar que los modelos favorezcan a las clases mayoritarias. Para abordar este problema, se ha recurrido a bibliotecas que permiten equilibrar la cantidad de muestras entre las clases 1 y 0, asegurando tener igual número de muestras para ambas.

Existen dos enfoques principales para lograr el equilibrio de las muestras: el sobremuestreo (*oversampling*) y el submuestreo (*undersampling*). Para tratar el desequilibrio de clases en el conjunto de datos de clasificación, se ha contado con las siguientes tres bibliotecas:

- SMOTE[42] (*Synthetic Minority Over-sampling Technique*) del módulo *imblearn.over_sampling* de Python: Esta técnica de sobremuestreo se utiliza para aumentar la cantidad de muestras en la clase minoritaria. SMOTE genera nuevas instancias sintéticas al interpolar características entre muestras existentes de la clase minoritaria.
- *TomekLinks*[43] del módulo *imblearn.under_sampling* de Python: Esta técnica de submuestreo se utiliza para eliminar muestras ruidosas y redundantes de la clase mayoritaria. Consiste en identificar pares de instancias cercanas, una de la clase minoritaria y otra de la clase mayoritaria, y eliminar la instancia de la clase mayoritaria.
- *RandomUnderSampler*[44] del módulo *imblearn.under_sampling* de Python: Esta técnica de submuestreo selecciona aleatoriamente instancias de la clase mayoritaria para igualar el número de muestras de la clase minoritaria.

[42] https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
[43] https://imbalanced-learn.org/dev/references/generated/imblearn.under_sampling.TomekLinks.html
[44] https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html

Para determinar cuál de estas técnicas ha funcionado mejor para resolver el desequilibrio de clases, se han realizado comparaciones utilizando los modelos mencionados anteriormente. En la tabla 7 se presentan las proporciones actualizadas de las clases después de aplicar el equilibrado, así como los mejores resultados obtenidos con cada uno de los modelos. En cada prueba se han utilizado 5 conjuntos de validación cruzada para seleccionar el modelo con el mejor valor de Test AUC y su precisión global en los datos de validación. Inicialmente, se ha dispuesto de 384 parcelas, lo que proporciona 1920 muestras de clase 1 y 5760 muestras de clase 0.

Librería de equilibrado de clases	Muestras clase 0 / Muestras clase 1	RandomForestClassifier	SVC	GaussianNB	KNeighborsClassifier	GradientBoostingClassifier	AdaBoostClassifier	MLPClassifier
SMOTE	5760 / 5760	0.83 (83%)	0.55 (54%)	0.54 (54%)	0.79 (79%)	0.71 (71%)	0.65 (65%)	0.59 (59%)
TomekLinks	5253 / 1920	0.69 (81%)	0.5 (74%)	0.57 (50%)	0.69 (79%)	0.55 (76%)	0.52 (75%)	0.5 (74%)
RandomUnderSampling	1920 / 1920	0.75 (75%)	0.54 (54%)	0.54 (54%)	0.68 (68%)	0.68 (68%)	0.6 (60%)	0.59 (59%)

Tabla 7: Comparación de resultados de diferentes modelos después de aplicar diversas bibliotecas de equilibrado de clases. Se parte de un conjunto de datos desequilibrado, con 5760 muestras para la clase 0 y 1920 muestras para la clase 1. Debajo de cada modelo se muestra el valor de la métrica Test AUC, junto con la precisión global en los datos de validación, entre paréntesis. El color verde indica la mejor biblioteca para ese modelo, el color naranja indica la segunda mejor opción y el color rojo indica la peor. En estas pruebas, se utilizó un umbral de elección de 0.5.

Después de realizar estas pruebas, se ha podido concluir que la mejor forma de abordar el desequilibrio de clases es utilizando la biblioteca SMOTE, la cual ha añadido muestras de la clase minoritaria hasta igualar la cantidad de muestras de la clase mayoritaria.

Normalización

Otro problema identificado ha sido el de la normalización de los datos. Durante la iteración anterior, se ha observado que al entrenar con el *dataframe* TPPSN, los resultados base no han mejorado. Se ha atribuido esta falta de mejora a los valores de la variable 'pres', que oscilaban entre 950 y 960. Para abordar este problema, se ha considerado necesaria una normalización de los datos, ya sea en el rango de 0 a 1 o en el rango de -1 a 1.

Si todas las variables del *dataframe* fueran positivas, la opción adecuada sería la normalización en el rango de 0 a 1. Sin embargo, debido a la presencia de valores positivos y negativos en las variables de pendiente, ha surgido la duda sobre qué enfoque utilizar. Por lo tanto, se ha empleado el modelo *RandomForestClassifier* (el cual ofreció los mejores resultados en la

prueba anterior) en combinación con el oversampling de la librería SMOTE para comparar la normalización entre 0 y 1 y la normalización entre -1 y 1.

Para llevar a cabo estas normalizaciones, se ha utilizado la librería *MinMaxScaler*[45] del módulo *sklearn.preprocessing* de Python, ajustando el parámetro *feature_range* a los valores (0, 1) o (-1, 1) según corresponda.

En el caso de la normalización entre 0 y 1, el mejor pliegue ha obtenido un TestAUC de 0.83 (con una precisión del 83%), mientras que en el caso de la normalización entre -1 y 1, el mejor pliegue ha logrado un TestAUC de 0.82 (con una precisión del 82%). Dicha diferencia ha sido mínima para extraer conclusiones definitivas, por lo que se ha seguido considerando estos enfoques en futuras pruebas.

Modelos con parámetros complejos

Se han realizado pruebas más complejas con los modelos *XGBoost*, *LightGBM*, *RandomForestClassifier* y *GradientBoostingClassifier*, después de haber evaluado los enfoques sencillos para abordar el desequilibrio de clases (oversampling con SMOTE) y la normalización (entre 0 y 1).

Estas pruebas se han considerado complejas debido a la necesidad de ajustar los valores de varios hiperparámetros de cada modelo para obtener diferentes resultados. Los hiperparámetros definidos para cada modelo son los siguientes:

- XGBoost: params = { 'max_depth': 3, 'eta': 0.1, 'objective': 'binary:logistic', 'eval_metric': 'error' }
- LightGBM: params = { 'boosting_type': 'gbdt', 'objective': 'binary', 'metric': 'binary_logloss', 'num_leaves': 31, 'learning_rate': 0.1 }
- RandomForestClassifier: params = { 'boosting_type': 'gbdt', 'objective': 'binary', 'metric': 'binary_logloss', 'num_leaves': 31, 'learning_rate': 0.05 }
- GradientBoostingClassifier: n_estimators=100, learning_rate=1.0, max_depth=1.

En la tabla 8 se muestra la comparación del mejor pliegue (de 5) obtenido por cada modelo en términos de TestAUC. Además, se indica la precisión global de los datos de validación entre paréntesis.

XGBoost	LightGBM	RandomForestClassifier	GradientBoostingClassifier
0.78 (70%)	0.88 (80%)	0.6 (60%)	0.66 (66%)

Tabla 8: Comparación de resultados de distintos modelos. Se muestra el valor de la métrica Test AUC, junto con la precisión global en los datos de validación, para cada modelo (entre paréntesis). El color indica la mejor librería para cada modelo (verde para el mejor, naranja para el segundo mejor y rojo para los demás). El umbral de elección se ha establecido en 0.5 en estas pruebas.

[45] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Como se esperaba, los modelos *XGBoost* y *LightGBM* han ofrecido los mejores resultados, por lo que se les ha dado un gran peso en futuras pruebas.

Prueba con red neuronal

Ahora que se han resuelto dos problemas importantes como eran el desequilibrio de clases y la normalización, se han utilizado con el modelo de red neuronal de 3 capas que se ha usado hasta ahora. La primera capa oculta ha tenido n^2 neuronas y ha utilizado la función de activación *ReLU*, mientras que la segunda capa oculta ha contado con n neuronas y también ha utilizado *ReLU*. La capa final ha constado de una única neurona con función de activación *sigmoide* para la clasificación binaria. La red ha sido compilada con la función de pérdida *binary_crossentropy* y el optimizador *Adam*. Además, se han incluido métricas de evaluación como precisión binaria, con un umbral de 0.6, y *recall*. En el código 13 se muestra el código de la red neuronal utilizada.

```
model = Sequential()
model.add(Dense(n**2, activation='relu', input_shape=(n,) ))
model.add(Dense(n, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=Adam(),
              metrics=[tf.keras.metrics.BinaryAccuracy(name='binary_accuracy', threshold=0.6),
                       keras.metrics.Recall(name='recall')])
model.summary()
```

Código 13: Arquitectura de la red neuronal secuencial con 3 capas.

Para esta prueba, se ha decidido utilizar el *dataframe* TPSN, ya que había mostrado mejores resultados en las pruebas anteriores, a excepción del resultado excepcional del *dataframe* TP3 en la tabla 5.

Tras entrenar durante 180,000 *epochs* (3 horas), se ha obtenido un modelo que ya no mejoraba su precisión con los datos de entrenamiento, alcanzando un 81%. Utilizando *callbacks* que guardaban los modelos con las mejores métricas de *binary_accuracy*, *val_binary_accuracy*, *recall* y *val_recall*, se ha seleccionado el mejor modelo según *val_binary_accuracy*, obteniendo una precisión global del 76.92%, una precisión de la clase 1 del 57.2% y un *recall* del 30.2%. Sin embargo, en los datos de test, el modelo ha acertado 28 de las 39 parcelas (71.7%), en comparación con las 23 de 39 que ha acertado anteriormente. El éxito mínimo, máximo y promedio del modelo han sido de 0.206, 0.978 y 0.570 respectivamente, superando los resultados anteriores de la tabla 4, que habían sido de 0.161, 0.786 y 0.476.

En la figura 32 se presentan cuatro ejemplos de parcelas del conjunto de validación en las que el modelo ha acertado al menos un día de recogida. Estas parcelas son las mismas que se habían mostrado en las figuras 29 y 30 de la iteración 5, donde se ha observado el fallo y acierto. En esta figura, se puede apreciar que el modelo realiza predicciones más acertadas en más parcelas y comete menos errores, ya que apenas hay líneas verticales rojas que indican falsos positivos.

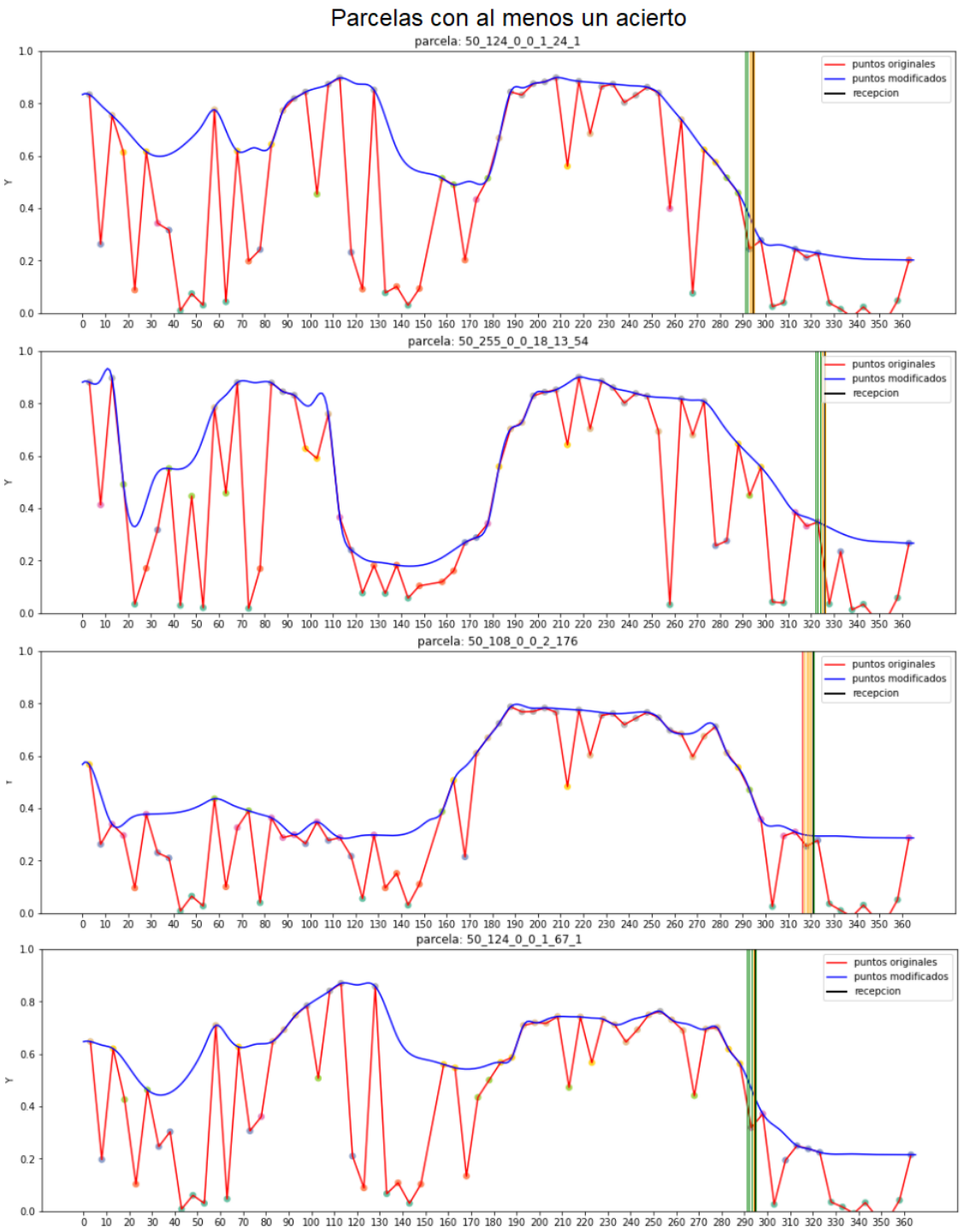


Figura 32: Ejemplos de parcelas con al menos 1 día de cosecha correctamente predicho por el modelo. Curvas NDVI originales (rojas) y curvas NDVI procesadas (azules) tras eliminar las irregularidades de la gráfica en iteraciones anteriores. Líneas verticales verdes indican días de la clase 1, y si el modelo los predice correctamente, serán amarillas; las líneas verticales rojas representan predicciones erróneas de recogida del cultivo.

En la figura 33 se muestran dos ejemplos de parcelas en las que no se acertó ningún día. Al igual que en las predicciones de la figura anterior y a diferencia de las de la figura 29, en estas predicciones apenas se observan falsos positivos.

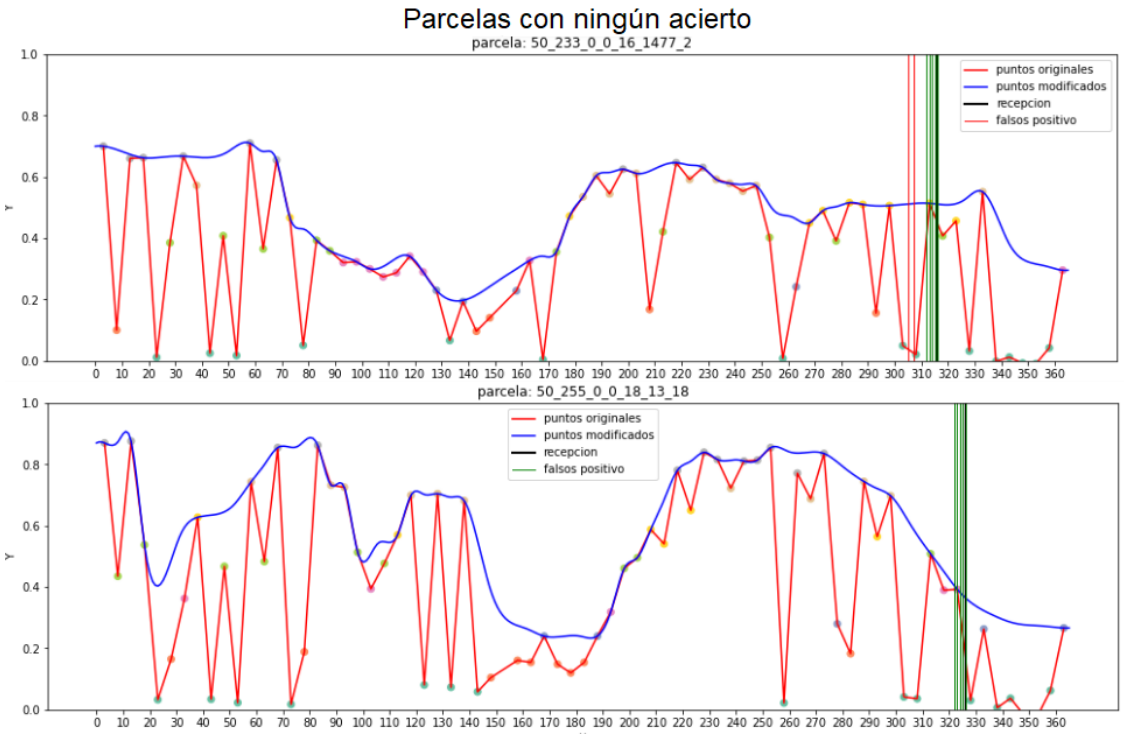


Figura 33: Ejemplos de parcelas sin días de cosecha correctamente predichos por el modelo. Curvas NDVI originales (rojas) y curvas NDVI procesadas (azules) tras eliminar las irregularidades de la gráfica en iteraciones anteriores. Líneas verticales verdes indican días de la clase 1, y las líneas verticales rojas representan predicciones erróneas de recogida del cultivo.

Aunque estos resultados han sido significativamente mejores que los obtenidos hasta ahora con cualquier modelo de redes neuronales, existe la esperanza de lograr resultados aún mejores con modelos más sencillos, los cuales no requieren un tiempo prolongado de entrenamiento. Por esta razón, se ha seguido probando con modelos como *RandomForest*, *XGBoost* y *LightGBM*.

Repetición mejorada de las pruebas

En estas pruebas, se han aplicado diferentes enfoques para mejorar la evaluación de los modelos. En lugar de realizar *oversampling* en todos los datos antes de separarlos en pliegues, se ha asegurado que no hubiera datos de la misma parcela tanto en los conjuntos de entrenamiento como en los conjuntos de validación. Además, el *oversampling* solo se ha aplicado a los datos de entrenamiento, lo que ha permitido una mejor comparación de los resultados con los datos de validación.

Se han realizado pruebas con diferentes *dataframes*, modelos y tipos de normalización para determinar cuáles ofrecían los mejores resultados. Estas comparaciones se presentan en la figura 34. Se muestra el mejor resultado de cada modelo según su TestAUC, junto con la

precisión global en los datos de validación, el éxito y el número de parcelas de validación en las que se acierta al menos una fecha dentro de los días de recepción válidos.

Es importante destacar que el modelo *SVC* muestra un éxito excesivamente alto en comparación con sus métricas de TestAUC y precisión. Esto se ha debido a que tiende a etiquetar todas las muestras como clase 1, lo cual no es deseable. Para abordar este problema, se ha creado una nueva métrica de éxito que penaliza los modelos que tienen este comportamiento, similar al *f1-score*, con la siguiente fórmula: $\frac{éxito * prec}{(éxito + prec)}$. El valor óptimo de esta métrica es 0.5. Tras las primeras pruebas con los modelos *SVC*, *GradientBoostingClassifier*, *AdaBoostClassifier* y *MLPClassifier* utilizando el *dataframe* TPSN, se ha decidido no probar estos modelos con otros *dataframes* similares, ya que sus resultados no han sido tan buenos y no se espera una mejora significativa.

Dataframe (Normalización)	RandomForestClassifier				RandomForestRegression				XGBClassifier				LGBMClassifier			
	TestAUC (prec. global %)	Éxito (min,max, prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)	TestAUC (prec. global %)	Éxito (min,max, prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)	TestAUC (prec. global %)	Éxito (min,max ,prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)	TestAUC (prec. global %)	Éxito (min,max ,prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)
TPSN (0, 1)	0.77 (75%)	0.250, 0.91, 0.628	0.341	34 / 39 (87.18%)	0.8 (78%)	0.311, 0.94, 0.673	0.362	36 / 39 (92.3%)	0.81 (76%)	0.409, 0.974, 0.733	0.373	34 / 38 (89.47%)	0.8 (77%)	0.407, 0.952, 0.672	0.358	33 / 38 (86.84%)
TPSN (-1, 1)	0.78 (76%)	0.215, 0.92, 0.619	0.340	33 / 39 (84.62%)	0.8 (79%)	0.318, 0.96, 0.681	0.364	36 / 39 (92.3%)	0.82 (80%)	0.420, 0.989, 0.732	0.381	35 / 38 (92.11%)	0.82 (78%)	0.405, 0.952, 0.688	0.365	35 / 38 (92.11%)
TP3SN (0, 1)	0.8 (80%)	0.264, 0.93, 0.631	0.352	35 / 39 (89.74%)	0.83 (79%)	0.45, 0.98, 0.726	0.378	35 / 38 (92.11%)	0.83 (78%)	0.438, 0.992, 0.751	0.383	36 / 38 (94.74%)	0.83 (79%)	0.405, 0.958, 0.69	0.367	35 / 38 (92.11%)
TP3SN (-1, 1)	0.81 (77%)	0.433, 0.92, 0.697	0.365	34 / 38 (89.47%)	0.82 (76%)	0.449, 1, 0.707	0.367	34 / 38 (89.47%)	0.83 (78%)	0.339, 0.976, 0.73	0.377	34 / 38 (89.47%)	0.83 (79%)	0.413, 0.947, 0.687	0.368	37 / 38 (97.37%)
TPPSN (0, 1)	0.85 (83%)	0.26, 0.98, 0.629	0.358	29 / 35 (82.85%)	0.87 (83%)	0.26, 0.99, 0.677	0.372	29 / 35 (82.85%)	0.86 (84%)	0.183, 0.977, 0.728	0.389	30 / 35 (85.71%)	0.87 (84%)	0.362, 0.935, 0.676	0.374	30 / 35 (85.71%)
TPPSN (-1, 1)	0.85 (83%)	0.26, 1, 0.634	0.359	29 / 35 (82.85%)	0.87 (83%)	0.29, 0.97, 0.676	0.372	29 / 35 (82.85%)	0.87 (84%)	0.403, 0.988, 0.740	0.394	30 / 35 (85.71%)	0.88 (84%)	0.374, 0.941, 0.674	0.373	32 / 35 (91.43%)
	SVC				GradientBoostingClassifier				AdaBoostClassifier				MLPClassifier			
	TestAUC (prec. global %)	Éxito (min,max, prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)	TestAUC (prec. global %)	Éxito (min,max, prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)	TestAUC (prec. global %)	Éxito (min,max ,prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)	TestAUC (prec. global %)	Éxito (min,max ,prom)	Exito * Prec / (Exito + Prec)	Parcelas acertadas (%)
TPSN (0, 1)	0.61 (48%)	0.562, 1, 0.966	0.322	37 / 39 (94.87%)	0.75 (76%)	0.391, 0.817, 0.597	0.333	28 / 38 (73.68%)	0.68 (75%)	0.495, 0.506, 0.501	0.300	0 / 38 (0%)	0.69 (67%)	0.391, 0.765, 0.597	0.315	26 / 38 (68.42%)
TPSN (-1, 1)	0.62 (51%)	0, 1, 0.915	0.328	36 / 39 (92.3%)	0.76 (76%)	0.413, 0.79, 0.596	0.334	28 / 38 (73.68%)	0.68 (75%)	0.495, 0.505, 0.501	0.300	0 / 38 (0%)	0.71 (70%)	0.441, 0.838, 0.622	0.329	31 / 38 (81.58%)
TP3SN (0, 1)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figura 34: Comparación de modelos de inteligencia artificial con diferentes dataframes y normalizaciones. Se muestra el mejor modelo según la métrica TestAUC, junto con la precisión global en los datos de validación. También se presentan el éxito mínimo, máximo y promedio, el número de parcelas de validación con al menos una fecha acertada, y una nueva métrica de éxito que penaliza modelos que tienden a etiquetar las muestras como clase 1.

En conclusión, se ha observado que los modelos *XGBoost* y *LGBM* son los que ofrecen los mejores resultados, y el *dataframe* TPPSN, que incluye la temperatura media, las precipitaciones y la presión media del día anterior en cada muestra, ha sido el que mejor funciona. Sin embargo, aún no se ha identificado una diferencia significativa entre los dos tipos de normalización, por lo que no se ha tomado una conclusión definitiva sobre cuál debería usarse.

Optimización de los modelos con Optuna

Ahora que se ha determinado que el *dataframe* TPPSN, que incluye información sobre temperatura, precipitación y presión del día anterior en cada muestra, funciona mejor, y los modelos *XGBoost* y *LightGBM* han sido los más efectivos, se ha utilizado la versión de estos modelos que requiere hiperparámetros para obtener mejores resultados. Para optimizar estos

modelos, se ha empleado la biblioteca *Optuna*[46] de Python, que se utiliza para encontrar los mejores valores de hiperparámetros en algoritmos de aprendizaje automático. Optuna busca la combinación óptima de los hiperparámetros seleccionados para maximizar o minimizar una métrica de evaluación específica, en este caso, el TestAUC.

Con el fin de reducir el número de pruebas, se ha decidido utilizar la normalización entre 0 y 1, ya que se ha considerado que es la opción más adecuada, dado que en el *dataframe* hay más variables con valores positivos que negativos.

Se han realizado varias pruebas en cada modelo, variando el número de iteraciones y la métrica a maximizar. Las métricas a maximizar han sido el TestAUC obtenido en cualquiera de los 10 pliegues del modelo en cuestión y la media del TestAUC entre los 10 pliegues. En las figuras 35 y 36 se presentan los hiperparámetros que se optimizarán, junto con una breve explicación de su función, los rangos de búsqueda y los valores obtenidos para los mejores resultados.

		Modelo XGBoost			
		500 iteraciones (**max** TestAUC)		10 iteraciones (**media** TestAUC)	
	params	Rango búsqueda	Resultado	Rango búsqueda	Resultado
Nº de árboles de decisión a construir durante el entrenamiento del modelo. Cuanto mayor sea, mas complejo será el modelo y mayor será el tiempo de entrenamiento.	n_estimators	-	-	[100 - 1000]	327
Profundidad máxima de cada árbol de decisión.	max_depth	[2 - 10]	9	[3 - 200]	63
Tasa de aprendizaje.	learning_rate	[0.01 - 0.1]	0.07133	[0.01 - 0.5]	0.01257
Proporción de muestras utilizadas para entrenar cada árbol. Un valor menor reduce la varianza pero también puede llevar a un sesgo mayor.	subsample	[0.6 - 1]	0.98735	[0.5 - 1]	0.95125
Proporción de características (columnas) utilizadas para entrenar cada árbol. Un valor menor reduce la correlación entre los árboles y puede ayer a prevenir el sobreajuste.	colsample_bytree	[0.6 - 1]	0.95115	[0.5 - 1]	0.89937
Un valor de corte para dividir un nodo en un árbol basado en la ganancia esperada de reducción del error. Un valor más alto crea particiones más conservadores.	gamma	[0 - 10]	3.59740	[0 - 1]	0.75758
Cantidad minima de muestras necesarias en un nodo terminal. Ayuda a controlar el crecimiento del árbol al evitar divisiones insignificantes.	min_child_weight	[0 - 10]	0.91006	[1 - 10]	1
Términos de regularización que penalizan la complejidad del modelo. Ayudan a evitar el sobreajuste controlando el peso de los términos de regularización L1 y L2, respectivamente.	reg_alpha	[0 - 10]	0.29336	[0.001 - 10]	0.00123
	reg_lambda	[0 - 10]	0.34729	[0.001 - 10]	0.01727
Métrica utilizada para evaluar el rendimiento del modelo durante el entrenamiento. Puede ser una métrica de regresión o clasificación, como MSE o Logloss.	eval_metric	auc	auc	auc	auc
Función de pérdida a optimizar durante el entrenamiento. Puede ser de regresión o clasificación, como reg:linear o binary:logistic	objective	binary:logistic	binary:logistic	binary:logistic	binary:logistic
Tipo de modelo a utilizar, como 'gbtree' para árboles de decisión o 'gblinear' para modelos lineales.	booster	gbtree	gbtree	gbtree	gbtree
Método utilizado para construir los árboles de decisión, como 'auto', 'exact', 'approx' o 'hist'	tree_method	exact	exact	exact	exact
Nº de trabajos en paralelo para el entrenamiento y la predicción. Un valor de -1 significa que se usarán todos los núcleos disponibles.	n_jobs	-1	-1	-1	-1

Figura 35: Hiperparámetros optimizados del modelo XGBoost. A la izquierda se proporciona una breve descripción de la función y los posibles valores de cada hiperparámetro. A la derecha se muestran los rangos de búsqueda utilizados en cada prueba, junto con los valores seleccionados para el mejor modelo obtenido.

[46] https://optuna.org/

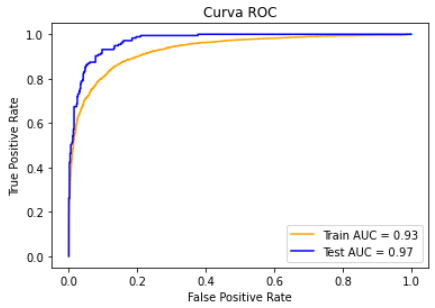
		Modelo LightGBM					
		10 iteraciones (**max** TestAUC)		500 iteraciones (**max** TestAUC)		10 iteraciones (**media** TestAUC)	
	params	Rango búsqueda	Resultado	Rango búsqueda	Resultado	Rango búsqueda	Resultado
Función de pérdida a optimizar durante el entrenamiento. Puede ser de regresión, clasificación o ranking, como 'regression', 'binary', 'multiclass', 'lambdarank', entre otros.	objective	binary	binary	binary	binary	binary	binary
Métrica utilizada para evaluar el rendimiento del modelo durante el entrenamiento. Puede ser una métrica de regresión, clasificación o ranking, como 'rmse', 'auc', 'ndcg', entre otros.	metric	auc	auc	auc	auc	auc	auc
Determina la cantidad de información que se muestra durante el entrenamiento. 0 es el nivel más silencioso y 3 es el nivel más detallado. -1 evitará la visualización de información o mensajes por pantalla.	verbosity	-1	-1	-1	-1	-1	-1
Tipo de algoritmo de boosting a utilizar. Puede ser 'gbdt' para Gradient Boosting Decision Tree, 'dart' para Dropout Additive Regression Trees, 'goss' para Gradient-based One-Side Sampling, o 'rf' para Random Forest.	boosting_type	gbdt	gbdt	gbdt	gbdt	gbdt	gbdt
Términos de regularización que penalizan la complejidad del modelo. Ayudan a evitar el sobreajuste controlando el peso de los términos de regularización L1 y L2, respectivamente.	lambda_l1	[1e-8 - 10]	0.12698	[1e-8 - 10]	0.00016	[1e-8 - 10]	0.00033
	lambda_l2	[1e-8 - 10]	5.21398	[1e-8 - 10]	2.94094	[1e-8 - 10]	0.77344
Nº máximo de hojas en cada árbol.	num_leaves	[2 - 256]	226	[2 - 256]	122	[2 - 256]	179
Proporción de características (columnas) utilizadas para entrenar cada árbol. Un valor menor reduce la correlación entre los árboles y puede ayer a prevenir el sobreajuste.	feature_fraction	[0.1 - 1]	0.56567	[0.1 - 1]	0.18647	[0.1 - 1]	0.68586
Proporción de muestras a utilizar para cada iteración de bagging. Un valor menor reduce la varianza pero también puede llevar a un sesgo mayor.	bagging_fraction	[0.1 - 1]	0.94927	[0.1 - 1]	0.94870	[0.1 - 1]	0.75145
La frecuencia de uso de bagging. Controla la cantidad de iteraciones de bagging a realizar.	bagging_freq	[1 - 10]	3	[1 - 10]	5	[0 - 10]	8
Nº mínimo de muestras necesarias en un nodo terminal. Ayuda a controlar el crecimiento del árbol al evitar divisiones insignficantes	min_child_sample	[5 - 100]	6	[5 - 100]	12	[5 - 100]	38

Figura 36: Hiperparámetros optimizados del modelo LightGBM. A la izquierda se presenta una breve descripción de la función y los posibles valores de cada hiperparámetro. A la derecha se detallan los rangos de búsqueda empleados en cada prueba, así como los valores utilizados en el mejor modelo obtenido.

En las figuras 37 y 38 se muestran los mejores resultados obtenidos en cada prueba. Como se puede observar, ambos modelos logran un TestAUC de 1, que ha sido el valor máximo posible. Además, el mejor resultado (el primero de la prueba con *LightGBM*) ha obtenido un *accuracy* del 99% en los datos de validación, ha acertado al menos una fecha de recolección correctamente en todas las parcelas de validación (35 de 35), ha presentado una precisión y *recall* de la clase 1 del 96% y 98% respectivamente, y ha tenido un éxito promedio de 0.944. Estos resultados han convertido a este modelo en el mejor obtenido en todas las iteraciones.

XGBoost 500 iters (max TestAUC)

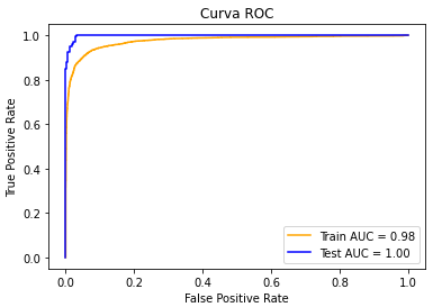
	precision	recall	f1-score	support
0	0.95	0.94	0.95	525
1	0.84	0.86	0.85	175
accuracy			0.92	700
macro avg	0.90	0.90	0.90	700
weighted avg	0.93	0.92	0.92	700




```
Parcelas con acierto:  35 de 35   100.00%
Test ROC AUC: 0.9723265306122448
Exito min:  0.4690346331654924
Exito max:  0.91531146
Exito promedio:  0.7509637304964724
Exito * accuracy =  0.6941050480445966
Exito * accuracy / (Exito + accuracy) =  0.414329370594027
```

XGBoost 10 iters (media TestAUC)

	precision	recall	f1-score	support
0	0.99	0.98	0.98	525
1	0.94	0.96	0.95	175
accuracy			0.98	700
macro avg	0.97	0.97	0.97	700
weighted avg	0.98	0.98	0.98	700

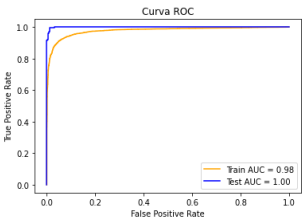



```
Parcelas con acierto:  35 de 35   100.00%
Test ROC AUC: 0.9981931972789116
Exito min:  0.6085927989065071
Exito max:  0.9732276
Exito promedio:  0.9037240076802442
Exito * accuracy =  0.8817764246365811
Exito * accuracy / (Exito + accuracy) =  0.4691701918257549
```

Figura 37: Informes con todas las métricas de los resultados obtenidos con el modelo XGBoost en relación con las pruebas presentadas en la Figura 35. A la izquierda se muestra el mejor resultado obtenido en la prueba de 500 iteraciones, y a la derecha se muestra el mejor resultado obtenido en la prueba de 10 iteraciones.

LightGBM 10 iters (max TestAUC)

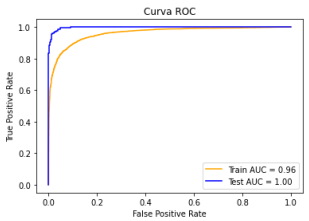
	precision	recall	f1-score	support
0	0.99	0.99	0.99	525
1	0.96	0.98	0.97	175
accuracy			0.99	700
macro avg	0.98	0.98	0.98	700
weighted avg	0.99	0.99	0.99	700




```
Parcelas con acierto:  35 de 35   100.00%
Test ROC AUC: 0.9992054421768708
Exito min:  0.6197778386580336
Exito max:  0.9935947965805236
Exito promedio:  0.94401275619152
Exito * accuracy =  0.9305268596744983
Exito * accuracy / (Exito + accuracy) =  0.48220646727088745
```

LightGBM 500 iters (max TestAUC)

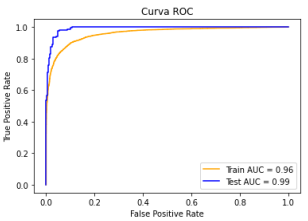
	precision	recall	f1-score	support
0	0.98	0.99	0.98	525
1	0.96	0.95	0.95	175
accuracy			0.98	700
macro avg	0.97	0.97	0.97	700
weighted avg	0.98	0.98	0.98	700




```
Parcelas con acierto:  35 de 35   100.00%
Test ROC AUC: 0.9974857142857143
Exito min:  0.6334424121964992
Exito max:  0.9779914192018835
Exito promedio:  0.8482092375750285
Exito * accuracy =  0.8288215978590279
Exito * accuracy / (Exito + accuracy) =  0.4540612193436166
```

LightGBM 10 iters (media TestAUC)

	precision	recall	f1-score	support
0	0.98	0.97	0.98	525
1	0.92	0.94	0.93	175
accuracy			0.96	700
macro avg	0.95	0.95	0.95	700
weighted avg	0.96	0.96	0.96	700




```
Parcelas con acierto:  35 de 35   100.00%
Test ROC AUC: 0.9916190476190476
Exito min:  0.6609386146583789
Exito max:  0.9865310454880655
Exito promedio:  0.8525444313350011
Exito * accuracy =  0.8208784953139868
Exito * accuracy / (Exito + accuracy) =  0.452174608077708
```

Figura 38: Informes con todas las métricas de los resultados obtenidos con el modelo LightGBM en relación con las pruebas presentadas en la Figura 36. A la izquierda se muestra el mejor resultado obtenido en la primera prueba de 10 iteraciones, en el centro se muestra el mejor resultado de la prueba de 500 iteraciones, y a la derecha se muestra el mejor resultado obtenido en la segunda prueba de 10 iteraciones.

Resultados

A lo largo de esta iteración se han llevado a cabo múltiples pruebas y comparaciones exhaustivas con el fin de establecer una posición inicial sólida que respalde la hipótesis inicial.

Para evaluar estas comparaciones, se han utilizado dos métricas nuevas: una basada en el área bajo la curva ROC (ROC-AUC) y otra basada en la medida de éxito, con una ligera modificación que ha penalizado los modelos que tienden a clasificar todo como clase 1. Además, se han explorado diferentes modelos, tanto basados en redes neuronales como en otros enfoques, ya que lo más complejo no siempre garantiza un mejor rendimiento. Utilizando estas métricas y modelos, se han comparado distintas técnicas de balanceo de muestras, así como diferentes métodos de normalización de los datos y distintos dataframes. Posteriormente, se ha utilizado la biblioteca *Optuna* de Python para optimizar los resultados de las mejores pruebas, lo que finalmente ha llevado a obtener un modelo con una precisión del 99% en los datos de validación, confirmando satisfactoriamente la hipótesis inicial.

Anexo III.XI. Iteración 11

Hipótesis de partida

Una vez se ha obtenido un modelo altamente eficaz que muestra un rendimiento óptimo, se plantea la posibilidad de mejorar el modelo mediante la eliminación de variables para reducir su complejidad, sin comprometer los resultados. Además, se plantea la idea de que este nuevo modelo puede realizar predicciones precisas sobre datos completamente desconocidos del año 2022, a pesar de haber sido entrenado exclusivamente con datos del año 2021.

Trabajo desarrollado

Influencia de las variables con SHAP

En busca de optimizar la eficiencia del modelo sin comprometer su rendimiento, se ha considerado el uso de la librería SHAP[47] (*SHapley Additive exPlanations*) en Python. Esta librería proporciona métodos para calcular la importancia e influencia de las características en las predicciones del modelo, lo que ha permitido comprender cómo el modelo toma decisiones y cuáles características son más influyentes. El objetivo ha sido identificar características que no sean esenciales en el modelo y puedan ser eliminadas sin afectar los resultados. Para ello, se ha añadido una instancia *Explainer*[48] de SHAP al mejor modelo obtenido en la iteración anterior, lo que ha permitido analizar la influencia de cada variable en los resultados. En la figura 39 se muestra un gráfico que representa la influencia de las diferentes variables en el modelo. Se observa que las variables menos importantes han sido la columna de unos, las precipitaciones y las tres columnas resultantes de la multiplicación de los valores de las pendientes.

[47] https://shap.readthedocs.io/en/latest/index.html
[48] https://shap.readthedocs.io/en/latest/generated/shap.Explainer.html

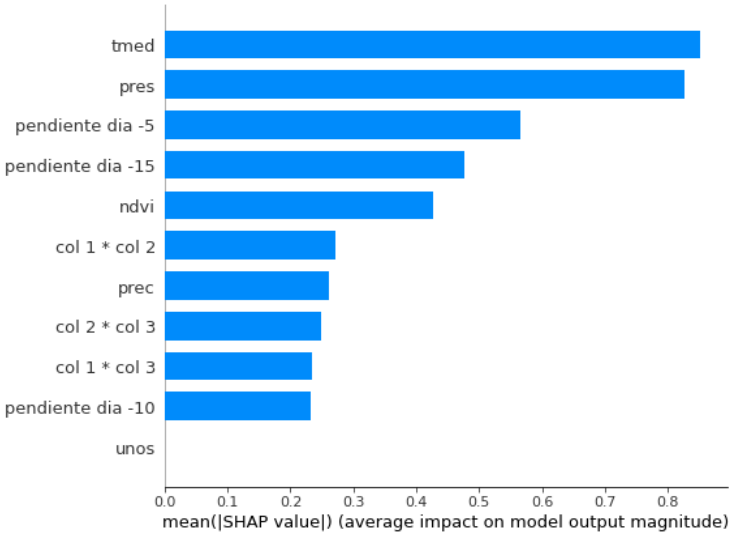


Figura 39: Gráfico de barras que muestra las contribuciones de las características según los valores SHAP en orden descendente.

Si se eliminan estas 5 variables de los datos de entrenamiento, reduciendo de 11 a 6 columnas, se logra una mejora aproximada del 45.45% en términos de simplicidad y eficiencia del *dataframe*. Para comprobar que estas 5 columnas pueden ser eliminadas sin afectar al modelo, se han realizado nuevamente pruebas utilizando *Optuna* y *LightGBM*, como en la iteración anterior, con el fin de obtener un nuevo modelo que siga mostrando un buen desempeño. En la figura 40 se presentan los hiperparámetros resultantes de cada prueba realizada con *Optuna*.

		Modelo LightGBM					
		10 iteraciones (**max** TestAUC)		500 iteraciones (**max** TestAUC)		500 iteraciones (**media** TestAUC)	
	params	Rango búsqueda	Resultado	Rango búsqueda	Resultado	Rango búsqueda	Resultado
Función de pérdida a optimizar durante el entrenamiento. Puede ser de regresión, clasificación o ranking, como 'regression', 'binary', 'multiclass', 'lambdarank', entre otros.	objective	binary	binary	binary	binary	binary	binary
Métrica utilizada para evaluar el rendimiento del modelo durante el entrenamiento. Puede ser una métrica de regresión, clasificación o ranking, como 'rmse', 'auc', 'ndcg', entre otros.	metric	auc	auc	auc	auc	auc	auc
Determina la cantidad de información que se muestra durante el entrenamiento. 0 es el nivel más silencioso y 3 es el nivel más detallado. -1 evitará la visualización de información o mensajes por pantalla.	verbosity	-1	-1	-1	-1	-1	-1
Tipo de algoritmo de boosting a utilizar. Puede ser 'gbdt' para Gradient Boosting Decision Tree, 'dart' para Dropout Additive Regression Trees, 'goss' para Gradient-based One-Side Sampling, o 'rf' para Random Forest.	boosting_type	gbdt	gbdt	gbdt	gbdt	gbdt	gbdt
Términos de regularización que penalizan la complejidad del modelo. Ayudan a evitar el sobreajuste controlando el peso de los términos de regularización L1 y L2, respectivamente.	lambda_l1	[1e-8 - 10]	0.00074	[1e-8 - 10]	1.42423	[1e-8 - 10]	3.60025
	lambda_l2	[1e-8 - 10]	4.86504	[1e-8 - 10]	2.98447	[1e-8 - 10]	0.00036
Nº máximo de hojas en cada árbol.	num_leaves	[2 - 256]	251	[2 - 256]	175	[2 - 256]	195
Proporción de características (columnas) utilizadas para entrenar cada árbol. Un valor menor reduce la correlación entre los árboles y puede ayer a prevenir el sobreajuste.	feature_fraction	[0.1 - 1]	0.94022	[0.1 - 1]	0.89105	[0.1 - 1]	0.90138
Proporción de muestras a utilizar para cada iteración de bagging. Un valor menor reduce la varianza pero también puede llevar a un sesgo mayor.	bagging_fraction	[0.1 - 1]	0.98150	[0.1 - 1]	0.91762	[0.1 - 1]	0.85036
La frecuencia de uso de bagging. Controla la cantidad de iteraciones de bagging a realizar.	bagging_freq	[1 - 10]	7	[1 - 10]	9	[0 - 10]	2
Nº mínimo de muestras necesarias en un nodo terminal. Ayuda a controlar el crecimiento del árbol al evitar divisiones insignficantes	min_child_sample	[5 - 100]	65	[5 - 100]	5	[5 - 100]	9

Figura 40: Hiperparámetros utilizados en la optimización del modelo basado en LightGBM después de eliminar 5 columnas de datos utilizando SHAP. Se presenta una breve descripción y los posibles valores de cada hiperparámetro, junto con los rangos de búsqueda especificados en cada prueba y el valor utilizado en el mejor modelo obtenido.

En la figura 41 se muestran los mejores resultados obtenidos en cada prueba. El mejor modelo se encuentra en la parte derecha, con un TestAUC de 1, al igual que el mejor modelo de la iteración anterior. En las demás métricas, este nuevo modelo ha sido ligeramente inferior al modelo anterior. Por ejemplo, presenta un -1% de *accuracy*, -2% de precisión de la clase 1, -1% de *recall* de la clase 1 y -0.01 de éxito promedio. Sin embargo, ha acertado el 100% de las parcelas y se considera que sigue siendo un modelo de alta calidad.

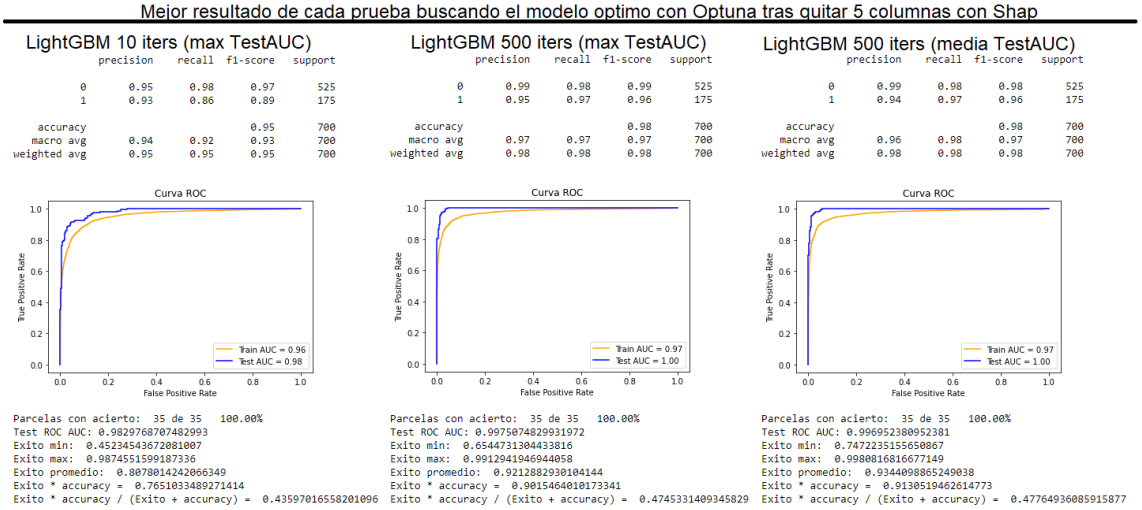


Figura 41: Informes con todas las métricas de los resultados obtenidos con el modelo LightGBM para las pruebas mencionadas en la figura 40. En la parte izquierda se muestra el mejor resultado obtenido en la primera prueba de 10 iteraciones, en el centro se muestra el mejor resultado de la prueba de 500 iteraciones y en la parte derecha se muestra el mejor resultado obtenido en la segunda prueba de 500 iteraciones.

Este modelo, con un 45.45% menos de información, solo ha sido un 1% peor que el modelo anterior. Esto se ha considerado un buen resultado y un modelo considerablemente mejor, ya que su desempeño sería mucho más rápido y eficiente en conjuntos de datos mucho más grandes.

Test con datos de 2022

Para realizar pruebas con los datos de test del año 2022, se ha llevado a cabo la recolección de datos de la misma manera en la que se obtuvieron los datos de 2021. El objetivo ha sido realizar una prueba final con un conjunto de datos totalmente desconocido para el modelo y evaluar su capacidad para predecir datos de otros años habiendo sido entrenado únicamente con datos de un año.

Se ha proporcionado un archivo llamado 'csv/PARCELAS-MAIZ-2022-2023-CON-FECHAS.csv', el cual contiene las fechas de cosecha de todas las parcelas del año 2022 y enero de 2023. Dado que en nuestro *dataframe* se tiene solo 5 muestras de la clase 1 para cada parcela, se han seleccionado las parcelas cuyas fechas de recepción se pueden agrupar en un intervalo de 5 días. Por ejemplo, una parcela con fechas de recepción en los días 300, 302 y 306 no se ha tenido en cuenta, ya que la diferencia entre la fecha máxima y mínima ha sido mayor a 5 y no se pueden agrupar en un intervalo de 5 días consecutivos.

Una vez identificadas las parcelas que cumplen con este criterio, se ha recopilado la información correspondiente a estas parcelas durante todo el año 2022 y enero de 2023 utilizando las APIs de GeosLab, de manera similar a como se ha hecho con las parcelas del año 2021 y enero de 2022 con las que se ha estado trabajando. Los datos resultantes de las 1080 parcelas se han guardado en el archivo 'csv/datosEntrenables2022_2023.csv'.

En los datos de entrenamiento y validación, solo se han utilizado parcelas con una única fecha de recepción, a diferencia de los datos de test donde se han permitido parcelas con hasta 5 días de recepción consecutivos. Por lo tanto, se ha contado con información de 346 parcelas para entrenamiento y validación, en comparación con las 1080 parcelas de test. Sin embargo, para realizar comparaciones "justas", se han eliminado las parcelas de test con más de una fecha de recepción, resultando en 538 parcelas que han cumplido con esta condición. Aunque sigue siendo un número considerablemente mayor que las parcelas disponibles para el año 2021, ahora están en la misma "condición".

Posteriormente, se han realizado tres pruebas distintas con estos datos de test: una con las 538 parcelas, otra con las parcelas conocidas cuyos datos del año anterior se han utilizado en el entrenamiento (42 parcelas) y otra con las parcelas desconocidas (496 parcelas). En la figura 42 se muestran los resultados de estas tres pruebas, los cuales son bastante pobres, ya que los valores de TestAUC apenas difieren de 0.5, que es el peor resultado posible en esta métrica.

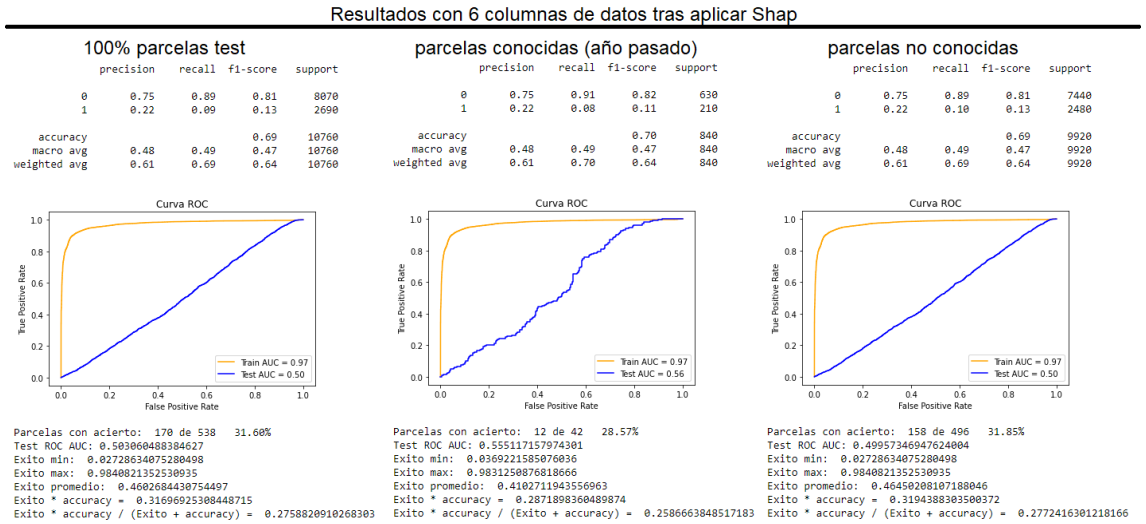


Figura 42: Informes de los resultados obtenidos con tres conjuntos distintos de parcelas de prueba correspondientes al año 2022 y enero de 2023. En la parte izquierda se presenta la prueba realizada con 538 parcelas, en el centro se muestra la prueba con las 42 parcelas que ya eran conocidas del año anterior, y en la parte derecha se muestra la prueba con las 492 parcelas que no se conocían previamente.

Los resultados obtenidos han sugerido que es posible que el modelo esté sobreajustado a los datos de entrenamiento o que los datos del año 2022 difieran significativamente de los del año 2021, posiblemente debido a variaciones en las temperaturas u otras razones desconocidas. Estas suposiciones han podido explicar los resultados incoherentes observados.

Resultados

Basándose en el análisis realizado, se ha concluido que la hipótesis inicial no se sostiene. A pesar de haber obtenido un modelo simplificado con una reducción del 45.45% en características sin comprometer su rendimiento, los resultados obtenidos con los datos de test han sido inferiores a los esperados.