Maestría en

Ciencia de Datos y Máquinas de Aprendizaje mención en Inteligencia Artificial

Trabajo de investigación previo a la obtención del título de

Magíster en Ciencia de Datos y Máquinas de Aprendizaje con mención en Inteligencia Artificial

AUTORES:

Juan Gabriel Bravo Guzmán

Reidel González Paz

Karla Estefanía Mora Cajas

Fernanda Paulina Vizcaino Imacaña

TUTORES:

Docente titulación

Alejandro Cortés

Uso de Modelos de Aprendizaje Automático para predecir eventos climáticos en Ecuador.

Quito, Diciembre 2024

Certificación de autoría

Nosotros, **Juan Gabriel Bravo Guzmán, Reidel González Paz, Karla Estefanía Mora Cajas, Fernanda Paulina Vizcaino Imacaña** declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.

Firma del graduando

Juan Gabriel Bravo Guzmán

Firma del graduando

Reidel González Paz

Firma del graduando

Karla Estefanía Mora Cajas

Firma del graduando

Fernanda Paulina Vizcaino Imacaña

Autorización de Derechos de Propiedad Intelectual

Nosotros, **Juan Gabriel Bravo Guzmán, Reidel González Paz, Karla Estefanía Mora Cajas, Fernanda Paulina Vizcaino Imacaña** en calidad de autores del trabajo de investigación titulado ***Titulo del trabajo de investigación "Uso de Modelos de Aprendizaje Automático para predecir eventos climáticos en Ecuador."***, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, diciembre de 2024

Firma del graduando

Juan Gabriel Bravo Guzmán

Firma del graduando

Reidel González Paz

Firma del graduando

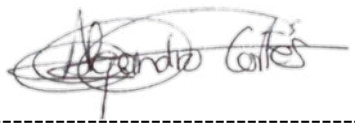
Karla Estefanía Mora Cajas

Firma del graduando

Fernanda Paulina Vizcaino Imacaña

Aprobación de dirección y coordinación del programa

Nosotros, **Alejandro Cortés e Iván Reyes,** declaramos que los graduandos: **Juan Gabriel Bravo Guzmán, Reidel González Paz, Karla Estefanía Mora Cajas, Fernanda Paulina Vizcaino Imacaña** son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés

Director/a de la
Maestría en Ciencia de Datos y Máquinas de Aprendizaje mención en Inteligencia Artificial

Iván Reyes

Coordinador/a de la
Maestría en Ciencia de Datos y Máquinas de Aprendizaje mención en Inteligencia Artificial

DEDICATORIA

Dedico este proyecto a mi familia, cuyo apoyo y sacrificio me han impulsado a superar cada obstáculo. Siempre han sido muestra de que si me esfuerzo por lo que quiero lo puedo conseguir.

También dedico este trabajo a mis compañeros de proyecto, quienes se convirtieron en amigos y aliados en este viaje. Su esfuerzo, colaboración y compromiso son parte esencial de este resultado, y estoy profundamente agradecido por haber compartido este camino con ustedes.

Juan Gabriel Bravo Guzmán

Dedicamos esta tesis a nuestras familias, quienes han sido nuestro pilar de fortaleza y motivación constante. A ustedes, que siempre creyeron en nuestras capacidades y nos brindaron el respaldo necesario para cumplir nuestros sueños.

También dedicamos este esfuerzo a todos los profesores que, con su pasión por la enseñanza, dejaron una huella en nuestro camino. Su ejemplo nos inspira a seguir creciendo y compartiendo conocimiento.

Finalmente, dedicamos este logro a nosotros mismos, como equipo: Juan, Karlita y Pau, por el trabajo, el compromiso y la determinación para alcanzar esta meta. Este es el resultado de nuestra unión, esfuerzo colectivo y perseverancia.

Reidel González Paz

Dedico este proyecto a mi familia, que siempre ha estado a mi lado brindándome su apoyo, ánimo y paciencia incondicional en cada etapa de este camino. También a mis profesores, por su valioso soporte y motivación, y a mis compañeros, por su esfuerzo y

colaboración. Este logro es un reflejo del compromiso colectivo y un paso significativo en nuestra formación profesional.

Karla Estefanía Mora Cajas

Dedico esta tesis a mí misma, por haber enfrentado cada reto con determinación y por cumplir el sueño de profundizar en un área de conocimiento que me apasiona profundamente. A mi familia en especial a Marcelo, por ser mi apoyo incondicional en cada desafío, brindándome amor y fortaleza para seguir adelante. Y a mis estudiantes, quienes son mi mayor inspiración para crecer como maestra y continuar mi preparación constante, buscando siempre ser una mejor guía en su aprendizaje

Fernanda Paulina Vizcaino Imacaña

AGRADECIMIENTOS

A mi familia, por ser el pilar fundamental en mi vida, por su amor incondicional, paciencia y apoyo constante en cada paso de este largo camino. Gracias por creer en mí y por motivarme a alcanzar mis metas, incluso en los momentos más desafiantes. Este logro es tanto mío como suyo.

A mis compañeros de proyecto, por compartir conmigo no solo el esfuerzo y la dedicación, sino también la pasión y las ganas de alcanzar la excelencia. Gracias por las largas horas de trabajo, las discusiones enriquecedoras y el compañerismo que convirtió este reto en una experiencia única e inolvidable.

Juan Gabriel Bravo Guzmán

Queremos expresar nuestro más profundo agradecimiento a todas las personas y que hicieron posible la realización de este trabajo de investigación, que marca el cierre de una etapa tan importante en nuestra formación profesional y personal.

En primer lugar, agradecemos a nuestros profesores y tutores, quienes, con su paciencia, experiencia y dedicación, nos guiaron en cada paso de este proyecto. Su compromiso con nuestra formación académica fue clave para superar los retos que enfrentamos y alcanzar nuestras metas.

A todo el equipo organizador de la maestría, queremos reconocer su arduo trabajo detrás de cada clase y actividad. Gracias por garantizar un ambiente académico de excelencia, por su esfuerzo en resolver nuestras dudas y por brindarnos herramientas que trascenderán esta etapa.

A nuestras familias y amigos, les agradecemos por el apoyo incondicional, el ánimo en los momentos difíciles y la confianza que depositaron en nosotros para seguir adelante. Sin su presencia en nuestras vidas, este logro no habría sido posible.

Finalmente, agradezco a nuestros compañeros de equipo Juan, Karlita y Pau. A lo largo de esta experiencia, aprendimos a trabajar juntos, apoyarnos mutuamente y enfrentar desafíos con resiliencia y creatividad. Cada uno de nosotros aportó algo único y valioso para culminar con éxito este proyecto.

Reidel González Paz

Quiero expresar mi más profundo agradecimiento a los docentes que, con su guía y conocimiento, hicieron posible el desarrollo de este proyecto. Agradezco especialmente a mis compañeros Juan, Reidel y Pauli, cuyo compromiso, dedicación y apoyo constante fueron fundamentales para alcanzar este logro. Este proyecto representa un paso importante en nuestro desarrollo profesional, y el trabajo en equipo ha sido clave para lograrlo.

Karla Estefanía Mora Cajas

A mi equipo de tesis, Karla, Reidel y Juan, por su dedicación, esfuerzo incansable y apoyo incondicional durante todo este proceso. Su compromiso y trabajo en equipo han sido fundamentales para alcanzar este logro. También extiendo mi más sincero agradecimiento a todos mis docentes, quienes con generosidad compartieron sus conocimientos y experiencias, inspirándome a crecer tanto personal como profesionalmente, así como a nuestro coordinador Iván Reyes por su excelente gestión. Este éxito no habría sido posible sin cada uno de ustedes.

Fernanda Paulina Vizcaino Imacaña

RESUMEN

Este proyecto aborda la predicción de eventos climáticos en Ecuador mediante modelos de aprendizaje automático. Se analizaron 45 años de datos históricos de 16 ciudades con diversos microclimas, obtenidos a través de la OpenWeather API. Utilizando la metodología KDD (Knowledge Discovery in Databases), se aplicaron técnicas avanzadas como la matriz de correlación para identificar relaciones entre variables y PCA (análisis de componentes principales) para reducir el conjunto de datos a las 7 características más relevantes. Estas técnicas permitieron manejar grandes volúmenes de información y optimizar el rendimiento de los modelos.

Durante la fase de desarrollo, se evaluaron 25 configuraciones de diferentes modelos para predicción y clasificación antes de seleccionar los modelos finales. Se diseñaron dos enfoques independientes: una red neuronal LSTM para predecir variables continuas en series temporales, como temperatura, humedad, presión entre otras, con la aplicación de secuencias, y un Random Forest Classifier para clasificar las variables predichas en 14 categorías discretas de tipo de clima. Ambos modelos fueron elegidos por su desempeño y capacidad para manejar patrones complejos en los datos.

Posteriormente, los modelos entrenados se almacenaron en formato Joblib, permitiendo su integración eficiente en una API. La API desarrollada permite realizar predicciones para los próximos 3 días, integrando ambos modelos para ofrecer tanto valores continuos como clasificaciones discretas. Este sistema proporciona una solución precisa y escalable que responde a la gran diversidad climática del Ecuador, sentando las bases para futuras investigaciones y aplicaciones en el ámbito de la predicción meteorológica.

Palabras Claves: Clima, Inteligencia artificial, Aprendizaje supervisado, Redes Neuronales, Random Forest.

ABSTRACT

This project addresses the prediction of climatic events in Ecuador using machine learning models. Forty-five years of historical data from 16 cities with various microclimates, obtained through the OpenWeather API, were analyzed. Using KDD (Knowledge Discovery in Databases) methodology, advanced techniques such as correlation matrix to identify relationships between variables and PCA (Principal Components Analysis) were applied to reduce the dataset to the 7 most relevant features. These techniques made it possible to handle large volumes of information and optimize the performance of the models.

During the development phase, 25 configurations of different models were evaluated for prediction and classification before getting to the final models. Two independent approaches were designed; an LSTM neural network to predict the variables in continuous time series, such as temperature, humidity, pressure, among others, with the application of sequences, and a Random Forest Classifier to classify the predicted variables into 14 discrete categories of climate types. Both models were chosen based on their performance and ability to handle complex patterns in the data.

Subsequently, the trained models were stored in Joblib format, allowing their efficient integration in an API. The API developed allows predictions for the next 3 next, integrating both models to provide both continuous values and discrete classifications.

This system provides an accurate and scalable solution that responds to the great climatic diversity of Ecuador, laying the groundwork for future research and applications in the field of weather forecasting.

Keywords: Weather, Artificial Intelligence, Supervised Learning, Neural Networks, Random Forest Classifier.

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

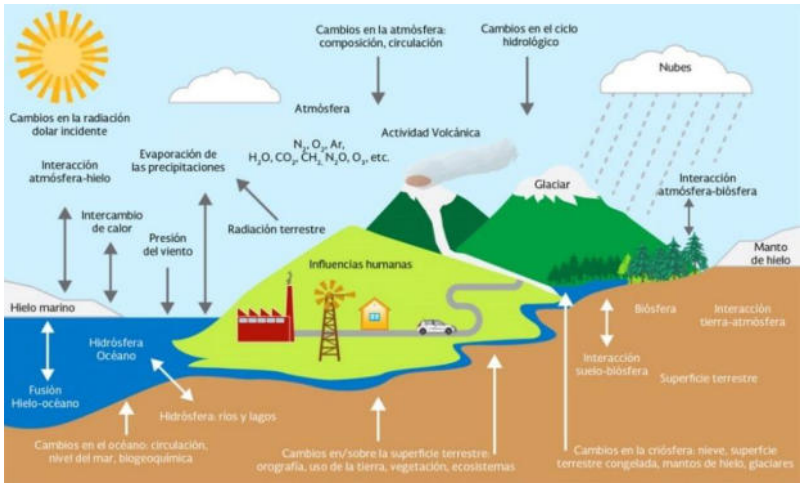
CAPITULO I

Introducción

El clima representa las características atmosféricas típicas de una región, observadas durante un extenso período. Su determinación requiere analizar datos recopilados por al menos tres décadas sobre diversos factores atmosféricos, incluyendo radiación, temperatura, lluvias, humedad del aire, vientos, presión atmosférica, evaporación y cobertura de nubes. En contraste, el tiempo atmosférico describe el estado momentáneo de la atmósfera. Se puede considerar como una manifestación puntual del clima. El clima actúa como un modelo o patrón del que derivan las condiciones meteorológicas específicas en un momento dado. (Gobierno de México, 2018)

Figura 1

Componentes del clima de la Tierra.



Nota. Esta imagen representa los distintos componentes que crean el clima de una región. (Gobierno de México, 2018)

El cambio climático se refiere a los cambios a largo plazo de las temperaturas y los patrones climáticos. Estos cambios pueden ser naturales, debido a variaciones en la actividad solar. Pero desde el siglo XIX, las actividades humanas han sido el principal motor del

cambio climático, debido principalmente a la quema de combustibles fósiles como el carbón, el petróleo y el gas. (United Nations, 2021)

El cambio climático ha intensificado la frecuencia y magnitud de eventos extremos en América Latina y el Caribe, lo que ha resultado en significativas pérdidas económicas, sociales y ambientales. La región ha experimentado un incremento progresivo en la temperatura promedio, alcanzando en 2022 un aumento de 1.7°C respecto al promedio del periodo 1901-1930. Este calentamiento ha contribuido a la ocurrencia de sequías severas, huracanes más destructivos y olas de calor extremas, como las inundaciones en Brasil y la sequía en Argentina, Brasil y Uruguay, que han tenido un impacto devastador en la agricultura y la disponibilidad de agua. (CEPAL, 2023)

En un planeta cada vez más desestabilizado por culpa de la emergencia climática provocada por la acción del hombre, disponer de sistemas capaces de pronosticar con fiabilidad la evolución de los fenómenos meteorológicos es cada vez más importante. Los modelos meteorológicos tradicionales hacen pronósticos basados en complejos cálculos matemáticos y ecuaciones, y requieren enormes cantidades de potencia informática, los modelos basados en AI llevan a cabo un enfoque diferente, mediante el reconocimiento de patrones en grandes cantidades de datos meteorológicos históricos que les permiten, posteriormente, generar pronósticos ingiriendo las condiciones actuales y aplicando lo que aprendieron de aquellos patrones históricos. El proceso requiere mucho menos procesamiento computacional y puede completarse en minutos o incluso segundos en ordenadores mucho más pequeños. Además, es capaz de mejorar la precisión del pronóstico al capturar patrones y escalas en los datos que no estaban representados en las ecuaciones explícitas manejadas anteriormente por los meteorólogos. (Dans, 2023)

En muchos artículos temáticos se pueden encontrar revisiones detalladas de los algoritmos de aprendizaje automático, como subgrupo más importante de los métodos de

inteligencia artificial en la ciencia atmosférica. En estas publicaciones se pueden encontrar detalles sobre muchos métodos y sus clasificaciones. Para los científicos atmosféricos, el grupo de técnicas más interesante resultó ser el aprendizaje supervisado, el grupo más dominante en las publicaciones recientes en este campo. En el caso de que se disponga de algunos datos etiquetados, se pueden utilizar como conjunto de datos de entrenamiento a partir de los cuales construir una función que asigne entradas dadas a salidas. Esa función se puede utilizar en un conjunto de datos diferente, denominado de prueba, para evaluar el modelo, y si los resultados son satisfactorios, se puede utilizar en la clasificación o regresión de cualquier tipo de aplicación que se necesite. En este grupo encontramos métodos como los Árboles de Decisión, por ejemplo, Random Forest (RF) o XGBoost (XGB), Redes Neuronales Artificiales (ANN), Deep Learning (DL) y Support Vector Machine (SVM). El segundo grupo en el aprendizaje automático es el aprendizaje no supervisado en el que los algoritmos no tienen datos etiquetados para entrenar, y deben decidir sobre otras formas de dividir un conjunto de datos dado, o reducir las dimensiones de este, para su posterior análisis. En este grupo, los métodos más populares entre los científicos atmosféricos son la agrupación de K-means y el análisis de componentes principales (PCA). (Bogdan & Ustrnul, 2022)

La predicción meteorológica, como procedimiento importante e indispensable en la vida cotidiana de las personas, evalúa la alteración que se está produciendo en el estado actual de la atmósfera. La analítica de datos es el proceso de análisis para extraer los patrones ocultos y la información aplicable que permita obtener mejores resultados. Hoy en día, varias partes de la sociedad están interesadas en los datos, y los institutos meteorológicos no están excluidos. Por lo tanto, el análisis de datos dará mejores resultados en la predicción meteorológica y ayudará a los meteorólogos a pronosticar el tiempo con mayor precisión.

Para lograr esto y recomendar soluciones favorables, se han sugerido varias técnicas y tecnologías de datos, como el Machine Learning, para gestionar y analizar el enorme volumen de datos meteorológicos procedentes de diferentes recursos. Al emplear el análisis de datos en la previsión meteorológica, pueden resolverse los retos relacionados con las técnicas y tecnologías tradicionales de gestión de datos. (Marzieh Fathi, 2021)

Las técnicas de aprendizaje automático se han convertido en una herramienta fundamentalmente eficaz en la forma de modelar y extraer patrones de big data de manera (semi) automática. La ciencia de la tierra ha sido también afectada por dicha revolución de muchas formas diferentes. Por ejemplo, los modelos de aprendizaje automático ahora son utilizados habitualmente para predecir y comprender los componentes del sistema terrestre encaminados a la:

- Clasificación de la cobertura terrestre
- Modelización del intercambio tierra-atmósfera y océano-atmósfera de gases de efecto invernadero
- Detección de anomalías y eventos extremos

(Julio & Armando, 2022)

Planteamiento Del Problema E Importancia Del Estudio

Tema Del Proyecto

Uso de Modelos de Aprendizaje Automático para predecir eventos climáticos en Ecuador.

Alcance Del Proyecto

Este proyecto tiene como objetivo desarrollar un modelo de aprendizaje automático capaz de predecir datos climáticos en Ecuador y clasificar eventos climáticos utilizando entradas proporcionadas por los usuarios a través de una interfaz web diseñada específicamente para este propósito. No se incluirán mapas climatológicos relacionados con

los datos. El análisis se enfocará en evaluar la efectividad del aprendizaje automático para predecir cambios climáticos en dieciséis ciudades representativas de las cuatro regiones geográficas del Ecuador.

Naturaleza o tipo de proyecto

Los cambios climáticos actualmente se presentan de manera brusca e inmediata siendo impredecibles por la población, ocasionando daños y pérdidas materiales, pero con el apoyo de las tecnologías presentes, como lo es la inteligencia artificial: machine learning, nos va a ayudar a anticipar estos hechos. (Meneses, Brescia, & Deyvis, 2023)

La geografía ecuatoriana es vulnerable a los efectos del cambio climático por su variedad de ecosistemas, que incluyen la Amazonía, los Andes, la costa y las Islas Galápagos. Cada una de estas regiones enfrenta amenazas específicas y tiene un rol crucial en la biodiversidad, agricultura y economía del país y para mitigar y adaptarse a estos desafíos, es esencial que Ecuador desarrolle políticas de gestión de riesgos climáticos y estrategias sostenibles, apoyadas en datos de predicción climática precisos. (World Bank Group, 2024)

En varios proyectos de análisis climático, se ha abordado la recolección y análisis de datos para estudiar los diversos cambios climáticos de Ecuador. A continuación, se presentan algunos trabajos relevantes.

El Instituto Nacional de Meteorología e Hidrologías emite el Boletín de Predicción Climática, con información del pronóstico de precipitación para los próximos tres meses: septiembre, octubre y noviembre del 2024. Los datos que les permiten realizar el pronóstico climático de precipitación son resultados del modelo numérico de predicción CWRF1 el mismo que hace uso de datos del Sistema de Pronóstico Climático (CFS). (INAMHI, 2024)

La Universidad Internacional de Valencia recalca la importancia y explora cómo la inteligencia artificial (IA) en el contexto de Big Data puede anticipar fenómenos climáticos. En su investigación menciona que Modelos predictivos y de detección de anomalías permiten

generar escenarios de riesgo, mejorando la capacidad de respuesta ante eventos extremos como inundaciones. Este enfoque puede ser clave para sectores sensibles como la agricultura y la gestión de recursos hídricos, que en Ecuador dependen en gran medida de condiciones climáticas específica. (Ehijo, 2024)

En este contexto existen varios proyectos o estudios relacionados con la predicción climática, como el estudio de "Aplicación de Redes Neuronales Artificiales para la Estimación de Precipitaciones, el cual analiza la cuenca del Río Pastaza en Ecuador", presenta la aplicación de redes neuronales artificiales (RNA) para abordar las deficiencias en los datos pluviométricos de esta cuenca. Implementando un modelo optimizado con 5000 iteraciones, donde se logró una fiabilidad del 95% en la estimación de datos pluviométricos. En este trabajo se analizaron datos de múltiples estaciones meteorológicas, ajustando el modelo según las distancias entre estaciones, y se demostró una mejora en precisión y coherencia en comparación con métodos tradicionales. Los resultados destacan la capacidad de las RNA para adaptarse a variaciones significativas en los datos, mejorando la planificación hídrica y mitigando los efectos de eventos climáticos extremos mediante una mejor predicción pluviométrica. (Rogel Alexander, 2024)

En artículo científico "Patrones de comportamiento de temperatura en el Ecuador en modelos de circulación atmosférica mediante Clustering**"**, se emplearon técnicas de reducción de dimensionalidad (PCA, TSNE y UMAP) y algoritmos de agrupamiento (K-means, DBSCAN, Agglomerative Clustering) para identificar comportamientos similares en los datos. Se evaluó la confiabilidad mediante el coeficiente de Silhouette y se validaron los resultados con diferentes métricas y gráficas usando Python como lenguaje de programación.

En los resultados se implementó el algoritmo UMAP para reducir la dimensionalidad con 20 vecinos y distancias calculadas mediante Chebyshev, logrando el mejor modelo de clustering en K-means con 4 grupos y un índice de Silhouette del 67%, que fue el más

destacado. En este trabajo se concluyó que cuatro grupos identificados podrían asociarse con frecuencias de días con temperaturas altas, muy altas, bajas y normales, lo cual coincide con los resultados del análisis exploratorio. (Lema, Natalia, & Toapanta, 2023)

Las redes neuronales son muy útiles para predecir valores futuros en lo que respecta a series de tiempo, ejemplificando un modelo tradicional como es el modelo ARIMA, con una red neuronal simple, una red neuronal recurrente simple y una red neuronal de memoria a largo plazo y corto plazo, se observa diferencias en cuanto la capacidad predictiva de estos modelos, sin embargo se recalca que tanto los modelos tradicionales como las redes neuronales tienen ventajas y desventajas, los modelos tradicionales son muy útiles en problemas más sencillos o con pocos datos, mientras que las redes neuronales son ventajosas en contextos de grandes volúmenes de datos. Los modelos de media móvil integrada autorregresiva (ARIMA) han dominado muchas áreas de la predicción de series de tiempo, en finanzas, retail, clima, entre otros. (Fernández, 2021)

Bases conceptuales

Python

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. (Amazon Web Services, 2024)

Numpy

NumPy es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación. (Sánchez, 2022)

Pandas

Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos. Las principales características son definir nuevas estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades, leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL, acceder a los datos mediante índices o nombres para filas y columnas. ofrecer métodos para reordenar, dividir y combinar conjuntos de datos, trabajar con series temporales. (Sánchez, 2022)

Drop

El método drop de pandas elimina la columna especificada del DataFrame. (Morocho, 2024)

Replace

Utilizamos el método replace de pandas para reemplazar los valores en la columna. (Morocho, 2024)

Matplotlib

Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones. Permite crear y personalizar los tipos de gráficos como diagramas de barras, histogramas, diagramas de sectores, diagramas de caja y bigotes, diagramas de violín, diagramas de dispersión o puntos, diagramas de líneas, diagramas de áreas, diagramas de contorno, mapas de color y combinaciones de todos ellos. (Sánchez, 2022)

Seaborn

Seaborn es una biblioteca de visualización de datos de Python basada en matplotlib . Proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos. (Waskom, 2024)

Scikit-learn

Scikit-learn, también conocido como sklearn, es una biblioteca de modelado de datos y aprendizaje automático de código abierto para Python. Incluye varios algoritmos de clasificación, regresión y agrupamiento, entre los que se incluyen máquinas de vectores de soporte, bosques aleatorios, potenciación de gradiente, k-means y DBSCAN, y está diseñado para interoperar con las bibliotecas de Python, NumPy y SciPy. (Domino Data Lab Inc, 2024)

Keras

Keras es una interfaz de programación de aplicaciones de alto nivel para ejecutar marcos de Machine Learning de bajo nivel como TensorFlow, Theano y otros marcos populares. Es fácil de usar, extensible y fácil de trabajar. (Hassan , y otros, 2023)

Tensorflow

TensorFlow facilita la creación de modelos de aprendizaje automático para computadoras de escritorio, dispositivos móviles, la web y la nube, sin importar si eres principiante o experto. (TensorFlow, 2024)

Bases de datos

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. (Oracle, 2020)

API

Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. API significa “interfaz de programación de aplicaciones”. En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones. Este contrato define cómo se comunican entre sí mediante solicitudes y respuestas. (Amazon Web Services, 2024)

CVS

La abreviatura "CSV" significa "valores separados por comas" (del inglés ‘comma separated value’), se trata de un formato y describe la estructura de un archivo de texto con el que se intercambian y almacenan datos estructurados de forma sencilla.

JSON

JavaScript Object Notation, se trata de un formato para guardar e intercambiar información que cualquier persona pueda leer. Los archivos json contienen solo texto y usan la extensión.json. (Deyimar, 2023)

Joblib

Es una biblioteca de Python que proporciona una interfaz fácil de usar para ejecutar programación/computación paralela en Python. La biblioteca de aprendizaje automático scikit-learn también utiliza joblib en segundo plano para ejecutar sus algoritmos en paralelo. (Solanky, 2022)

Matriz de correlación entre características

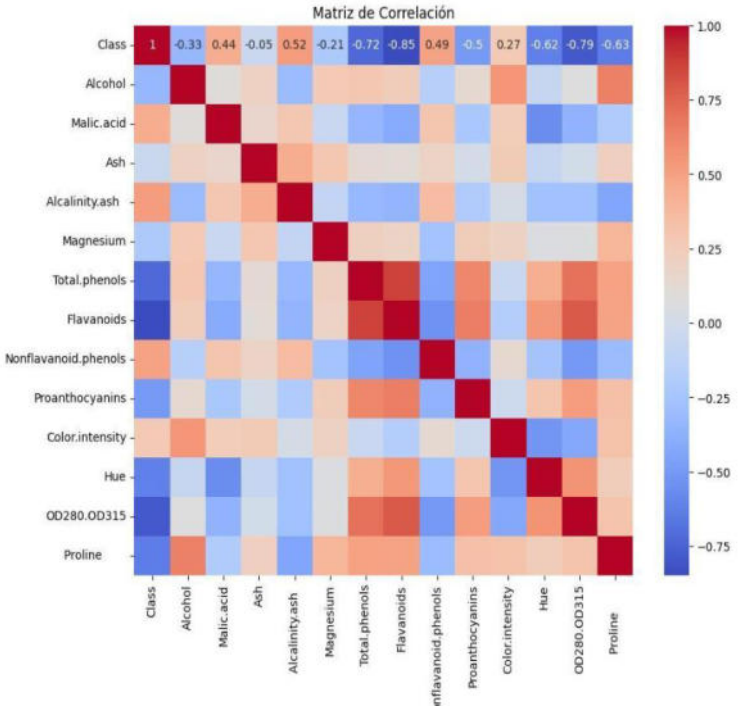
La matriz de correlación te permite ver las correlaciones entre las diferentes características un conjunto de datos. Los valores en la matriz de correlación varían de -1 a 1, donde:

- Un valor cercano a 1 indica una fuerte correlación positiva: si una característica aumenta, la otra también tiende a aumentar.
- Un valor cercano a -1 indica una fuerte correlación negativa: si una característica aumenta, la otra tiende a disminuir.
- Un valor cercano a 0 indica que no hay una correlación lineal entre las dos características.

(Morocho, 2024)

Figura 2

Matriz de correlación entre características.



Nota. Ejemplo de una matriz de correlación del conjunto de datos “vinos Piedmont”. (Morocho, 2024)

Profile Report

Es una clase que genera un informe de perfilado de datos. El informe de perfilado de datos proporciona un análisis rápido de los datos en el DataFrame, incluyendo el número de valores faltantes, la distribución de los datos, la correlación entre las características, y más. Es una herramienta útil para la exploración inicial de los datos. (Morocho, 2024)

Análisis de componentes principales (PCA)

El PCA es una técnica de machine learning no supervisadas que se utiliza para el análisis de datos, para reducir la dimensionalidad de los datos. El PCA crea nuevas variables, como componentes principales, que son combinaciones lineales de las variables originales. El PCA toma un conjunto de datos con múltiples variables como entrada y produce un conjunto

de datos en un subespacio inferior, es decir, un conjunto de datos reducido con menos variables. Aplicar el PCA puede ayudar a preprocesar o extraer las funciones más informativas de los conjuntos de datos con muchas variables. El preprocesamiento reduce la complejidad al tiempo que preserva la información relevante. (IBM, 2021)

Varianza Acumulada

Es una medida de la proporción de la varianza total en el conjunto de datos original que se explica por cada componente principal. La varianza acumulada total de un conjunto de componentes principales es simplemente la suma de las razones de varianza explicada de esos componentes. (SaturnCloud, 2023)

Multicolinealidad

La multicolinealidad se produce cuando las variables independientes de una ecuación de regresión están correlacionadas. Las variables multicolineales pueden afectar negativamente a las predicciones del modelo sobre datos no observados. Hay varias técnicas de regularización que pueden detectar y corregir la multicolinealidad y un método de diagnóstico es calcular una matriz de correlación para todas las variables independientes y las soluciones sencillas para la multicolinealidad van desde diversificar o ampliar el tamaño de la muestra de datos de entrenamiento hasta eliminar por completo los parámetros. (IBM, 2021)

Machine Learning

El machine learning es la ciencia de desarrollo de algoritmos y modelos estadísticos que utilizan los sistemas de computación con el fin de llevar a cabo tareas sin instrucciones explícitas, en vez de basarse en patrones e inferencias. Los sistemas de computación utilizan algoritmos de machine learning para procesar grandes cantidades de datos históricos e identificar patrones de datos. (Amazon Web Services, 2024)

En el aprendizaje automático (ML), una tarea fundamental es el desarrollo de modelos de algoritmos que analizan escenarios y realizan predicciones. Durante este trabajo, los

analistas incorporan diversos ejemplos a conjuntos de datos de entrenamiento, validación y prueba. (Kili Technology, 2023)

Clase Objetivo

Una clase objetivo también se conoce como variable dependiente. Una variable objetivo es el resultado que se pretende predecir o explicar mediante el modelo de aprendizaje automático. Una variable objetivo es la variable que se desea estimar o clasificar en función de los datos disponibles. (Smolic, 2024)

Variables Independientes

Las variables independientes (llamadas variables predictoras) son aquellas que se utilizan para generar predicciones sobre la variable dependiente (el objetivo) o para explicar la variación en ella. Las variables pueden ser cuantitativas o cualitativas y pueden tener una estructura continua o categórica. Los datos pueden modificarse o escalarse para aumentar su valor predictivo. (Williamson, 2024)

Train_test_split

La función train_test_split de la biblioteca sklearn en Python tiene la funcionalidad de dividir el conjunto de datos en conjuntos de entrenamiento y prueba. (Morocho, 2024)

Conjunto de entrenamiento

Los datos de entrenamiento son conjuntos de ejemplos o muestras que se utilizan para "enseñar" o "entrenar" el modelo de aprendizaje automático. El modelo utiliza un conjunto de datos de entrenamiento para comprender los patrones y las relaciones dentro de los datos, aprendiendo así a hacer predicciones o tomar decisiones sin estar programado explícitamente para realizar una tarea específica. (Kili Technology, 2023)

Conjunto de validación

El conjunto de datos de validación contiene diferentes muestras para evaluar los modelos de ML entrenados. En esta etapa, todavía es posible ajustar y controlar el modelo. El

trabajo con los datos de validación se utiliza para evaluar el rendimiento del modelo y ajustar los parámetros del modelo. Esto se convierte en un proceso iterativo en el que el modelo aprende de los datos de entrenamiento y luego se valida y ajusta en el conjunto de validación. (Kili Technology, 2023)

Conjunto de prueba

El conjunto de datos de prueba es una muestra separada, un conjunto de datos no vistos, que proporciona una evaluación final imparcial del ajuste del modelo. Los datos de entrada en los datos de prueba son similares a los de las etapas anteriores, pero no son los mismos datos. El conjunto de datos de prueba refleja datos del mundo real que el modelo de aprendizaje automático nunca ha visto antes. Su propósito principal es ofrecer una evaluación justa y final de cómo se desempeñaría el modelo cuando encuentre nuevos datos. (Kili Technology, 2023)

Tipos de algoritmos de machine learning

Los algoritmos se pueden clasificar en cuatro estilos de aprendizaje distintos en función de la salida esperada y del tipo de entrada. (Amazon Web Services, 2024)

Aprendizaje supervisado

El aprendizaje supervisado es el tipo de algoritmo de Machine Learning más frecuente. Utiliza un conjunto de datos conocidos (denominado conjunto de datos de entrenamiento) para entrenar un algoritmo con un conjunto de datos de entrada conocidos (denominados características) y respuestas conocidas para realizar predicciones. El algoritmo de aprendizaje supervisado intenta crear un modelo estableciendo relaciones entre las características y los datos de salida para realizar predicciones acerca de los valores de respuesta para un nuevo conjunto de datos. (MathWorks, 2024)

Aprendizaje no supervisado

El aprendizaje no supervisado, también conocido como machine learning no supervisado, utiliza algoritmos de machine learning para analizar y agrupar en clústeres conjuntos de datos sin etiquetar. Estos algoritmos descubren agrupaciones de datos o patrones ocultos sin necesidad de ninguna intervención humana. Su capacidad de descubrir similitudes y diferencias en la información lo convierten en la solución ideal para el análisis de datos exploratorios, las estrategias de venta cruzada, la segmentación de clientes y el reconocimiento de imágenes. (IBM, 2021)

Aprendizaje semi supervisado

El aprendizaje semi supervisado combina el aprendizaje supervisado y el no supervisado. Para entrenar los sistemas, esta técnica se basa en el uso de una pequeña cantidad de datos etiquetados y de una gran cantidad de datos sin etiquetar. En primer lugar, los datos etiquetados se utilizan para entrenar parcialmente el algoritmo de machine learning. Después, el propio algoritmo entrenado parcialmente etiqueta los datos no etiquetados. (Amazon Web Services, 2024)

Aprendizaje por refuerzo

El aprendizaje por refuerzo es un método con valores de recompensa adjuntos a los diferentes pasos que debe dar el algoritmo. Así, el objetivo del modelo es acumular tantos puntos de recompensa como sea posible y alcanzar una meta final. Este método funciona mejor en entornos de datos inciertos y complejos, rara vez se aplica en contextos empresariales. No es eficiente para tareas bien definidas y el sesgo del desarrollador puede afectar los resultados. (Amazon Web Services, 2024)

Aprendizaje profundo

El Deep Learning (DL) o aprendizaje profundo es un campo de la inteligencia artificial (Artificial Intelligence-AI), y a su vez un subconjunto de algoritmos de aprendizaje

automático (Machine learning- ML). El DL es esencialmente una red neuronal (Neural network - NN) de gran complejidad, o lo que también definimos como una red neuronal con varias capas. (Chivatá, 2024)

El aprendizaje profundo es un tipo de técnica de machine learning que se basa en el cerebro humano. Los algoritmos de aprendizaje profundo analizan los datos con una estructura lógica similar a la que utilizan los humanos. El aprendizaje profundo utiliza sistemas inteligentes, denominados redes neuronales artificiales, para procesar información por capas. Los datos fluyen desde la capa de entrada a través de varias capas de redes neuronales “profundas” ocultas antes de llegar a la capa de salida. Las capas adicionales ocultas permiten un aprendizaje mucho más eficaz que el de los modelos estándar de machine learning. (Amazon Web Services, 2024)

Redes neuronales artificiales

Una Red Neuronal Artificial (RNA), es un modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Está compuesta por unidades de procesamiento llamadas neuronas artificiales o nodos, que están organizadas en capas interconectadas. Las redes neuronales artificiales son utilizadas para abordar una amplia variedad de tareas de aprendizaje automático, incluyendo clasificación, regresión, procesamiento de imágenes, procesamiento de lenguaje natural y más. (Morocho, 2024)

Red recurrente

Una red neuronal recurrente (RNN) es un tipo de red neuronal artificial que utiliza datos secuenciales o datos de series de tiempo. Estos algoritmos de aprendizaje profundo se utilizan comúnmente para problemas ordinales o temporales, como la traducción de idiomas, el reconocimiento de voz y subtítulos de imágenes. (IBM, 2021)

Series temporales

Una serie temporal es un conjunto de observaciones que se obtiene midiendo una variable única de manera regular a lo largo de un período de tiempo. (IBM, 2021)

Ventanas

Este método implica agregar gradualmente un valor del set de prueba al conjunto de entrenamiento a la vez, entrenar un nuevo modelo con el set de entrenamiento actualizado y usar el modelo para predecir el siguiente valor del set de prueba. Este proceso se repite hasta que se haya pronosticado todo el set de prueba. (Sanchez, 2023)

Hiperparámetros

Los hiperparámetros son variables de configuración externa que los científicos de datos utilizan para administrar el entrenamiento de modelos de machine learning. A veces llamados hiperparámetros de modelos, los hiperparámetros se configuran de manera manual antes de entrenar un modelo. (Amazon Web Services, 2024)

Hiperparámetros de la estructura del modelo

Antes de entrenar una red neuronal, debemos especificar varios hiperparámetros que afectan al proceso de aprendizaje. Algunos de los más importantes incluyen:

Capas

Las neuronas se organizan en capas. Una red neuronal típica consta de una capa de entrada, una o más capas ocultas y una capa de salida. Las capas ocultas permiten a la red aprender representaciones intermedias y abstraer características de los datos. (Morocho, 2024)

Tamaño de la capa oculta

Determina el número de neuronas en cada capa oculta de la red. (Heras, 2024)

Dense

Dense es una clase que permite crear capas de neuronas completamente conectadas, que son el tipo de capa más común en las redes neuronales. (Morocho, 2024)

Funciones de activación

Introduce no linealidad en la red, permitiendo que las redes neuronales aprendan relaciones y patrones complejos en los datos. Existen varias funciones de activación comunes utilizadas en las redes neuronales, entre las que se incluyen: Sigmoide, ReLU, Tangente Hiperbólica, etc (Morocho, 2024)

Tasa de aprendizaje

Establece la velocidad de ajuste de los pesos de la red durante el entrenamiento. (Heras, 2024)

Tasa de regularización

Controla la regularización aplicada a los pesos de la red para evitar el sobreajuste u overfiting. (Heras, 2024)

Función de pérdida

Especifica la función utilizada para calcular la pérdida entre las predicciones del modelo y los valores reales como puede ser la clasificación binaria, múltiple, regresión o clasificación múltiple con índice categórica. (Heras, 2024)

Algoritmo de optimización

Define el algoritmo utilizado para ajustar los pesos de un modelo de la red neuronal durante el proceso de entrenamiento cuyo objetivo es minimizar la función de pérdida mediante la actualización iterativa de los pesos del modelo. (Heras, 2024)

Tamaño del lote (batch size)

Determina el número de ejemplos de entrenamiento que se utilizan en cada época de actualización de los pesos. (Heras, 2024)

Hiperparámetros del algoritmo de aprendizaje

Epochs

Establece cuántas veces se va a iterar sobre el conjunto de datos de entrenamiento. (Heras, 2024)

Momentum

Ayuda a acelerar la convergencia y evita que el proceso de optimización quede atrapado en mínimos locales o puntos de silla. (Heras, 2024)

Learning rate

Establece la velocidad de ajuste de los pesos de la red durante el entrenamiento. (Heras, 2024)

Learning rate decay

Reduce gradualmente la tasa de aprendizaje a lo largo del tiempo durante el proceso de entrenamiento. Permitiendo una búsqueda más precisa del espacio de parámetros en cada época. Con lo cual conduce a una convergencia más rápida y a una mejor generalización del modelo. (Heras, 2024)

Batch size

Establece cuántas muestras se van a utilizar para calcular el gradiente y actualizar los pesos de la red en cada paso de entrenamiento. (Heras, 2024)

Estrategias de regularización se pueden utilizar para evitar el sobreajuste en redes neuronales

Regularización L1 y L2

Las técnicas de regularización L1 y L2 ayudan a reducir el sobreajuste y a mejorar la capacidad del modelo para generalizar a datos no vistos**.** (Code Labs Academy, 2024)

Dropout

Es una técnica de regularización que se aplica durante el entrenamiento de una red neuronal para prevenir el sobreajuste. (Heras, 2024)

Early Stopping

Monitorea la métrica de pérdida en el conjunto de validación (monitor='val_loss') y tiene con una paciencia de n épocas, lo que significa que el entrenamiento se detendrá si la pérdida en el conjunto de validación no mejora durante n épocas consecutivas. (Heras, 2024)

BatchNormalization

Es una técnica que normaliza las activaciones de cada capa oculta. Ayuda a estabilizar y acelerar el proceso de entrenamiento, permitiendo que la red aprenda más rápido y converja más rápido. (Heras, 2024)

Weight Decay

Consiste en penalizar los valores grandes de los pesos durante el entrenamiento al agregar una fracción del cuadrado de los pesos a la función de pérdida. (Heras, 2024)

Matriz de Confusión

La matriz de confusión es una tabla que muestra las predicciones correctas e incorrectas del modelo, divididas por clase. Los valores en la diagonal principal de la matriz representan las predicciones correctas, mientras que los otros valores representan las predicciones incorrectas (Morocho, 2024).

Objetivos

Objetivo general

Desarrollar modelos de aprendizaje automático para predecir eventos climatológicos en Ecuador utilizando datos históricos, con el objetivo de ayudar a las personas de grupos vulnerables a prepararse mejor ante condiciones adversas que puedan afectar sus actividades económicas y sociales.

Objetivos específicos

Evaluar diversas fuentes de información histórica de clima considerando fiabilidad de los datos, accesibilidad, antigüedad de los datos y calidad de información bajo los estándares de la industria.

Clasificar los datos climáticos seleccionados, de tal forma que sea posible identificar patrones que permitan predecir eventos climáticos empleando modelos de clasificación de aprendizaje automático.

Desarrollar modelos predictivos basados en aprendizaje automático para la predicción temprana de eventos climáticos.

Diseñar una interfaz de datos que permita el ingreso de parámetros para las predicciones y la presentación de datos en tiempo real sobre el clima

Justificación e importancia del trabajo de investigación

El cambio climático en nuestra región, aunque no es tan grave como en otras partes del mundo, tiene un grave y directo impacto en las vidas de sus habitantes, pues sectores vulnerables no industrializados al cien por ciento como la agricultura o la pesca se ven gravemente afectados, en la calidad de vida de propios y de quienes consumen sus productos.

En los últimos 50 años, los desastres naturales relacionados con el clima se han triplicado, teniendo efectos cada vez más adversos en términos de morbilidad, mortalidad, ecosistemas y economías. Estos desastres pueden reducir el PIB hasta en un 0,9 por ciento en los países de ingresos más bajos, mientras en el Caribe esta reducción puede llegar hasta el 3,6 por ciento. Además, se espera que el cambio climático genere la migración de aproximadamente 17 millones de personas para el año 2050. (BID, 2024)

Para enfrentar este problema, se realiza la investigación para probar nuevos enfoques y herramientas metodológicos, incluida la aplicación del aprendizaje automático (ML) aprovechando el potencial de la gran disponibilidad y variedad de big data espaciotemporal

para aplicaciones ambientales. Dada la creciente atención en la aplicación de métodos de ML a la evaluación del riesgo del cambio climático, el resultado del análisis en otras investigaciones que se han realizado toma una gran variedad de algoritmos de ML dentro de CCRA, entre ellos, los más recurrentes son Decisión Trees (Federica , et al., 2021), Random Forest y Artificial Neural Network. Estos algoritmos a menudo se aplican de forma conjunta o híbrida para analizar la mayoría de los eventos de riesgo de inundaciones y deslizamientos de tierra. Además, la aplicación del aprendizaje automático para manejar datos de teledetección es consistente y eficaz en todas las aplicaciones de CCRA analizadas, lo que permite la identificación y clasificación de objetivos y la detección de características ambientales y estructurales. (Federica , et al., 2021)

Los modelos de aprendizaje automático pueden mejorar la precisión en la predicción de eventos climáticos en Ecuador al aprovechar patrones complejos en datos climáticos históricos y actuales. (Markus, Gustau, Stevens, & Martin, 2019)

La variabilidad y el cambio climático pueden afectar prácticamente todos los aspectos de la sociedad, incluida la producción de alimentos, la salud, la vivienda, la energía, los recursos hídricos, la seguridad, el turismo, las finanzas y el transporte.

Monitorear las condiciones climáticas y predecir lo que traerá la próxima estación o cómo cambiará nuestro clima en los próximos años y décadas es fundamental para el desarrollo sostenible. (Organización Meteorológica Mundial, 2024)

En el contexto del cambio climático, la necesidad de utilizar estas herramientas para monitorear y gestionar los recursos naturales adquiere cada vez más relevancia, para dar respuesta a los retos impuestos al desarrollo sostenible y a la supervivencia de la especie humana, ya que las afectaciones negativas a los recursos naturales y la biodiversidad, por los impactos del cambio climático, inciden sobre los bienes y servicios ecosistémicos de los que se sirven los seres humanos para desarrollar sus actividades socioeconómicas y culturales. Es

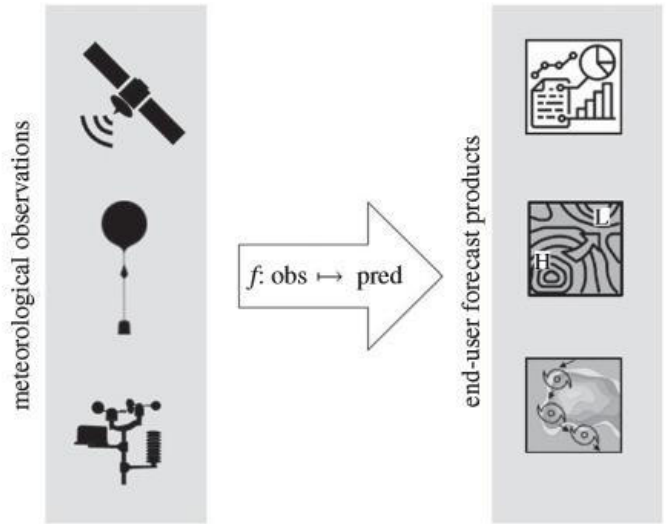
necesario, por lo tanto, buscar nuevas herramientas, más inteligentes y eficaces, que favorezcan el moni- toreo y gestión de estos recursos, para facilitar la toma de decisiones de científicos, directivos y población en general, en relación con su uso sostenible, conservación y mejoramiento. (Julio & Armando, 2022)

Por lo antes expuesto se entiende lo crucial de poder predecir o identificar factores que determinen si un evento climático puede ocurrir, a fin de prevenir a los habitantes de la región y mitigar el impacto de estos eventos en la economía y vida de estas personas.

En la Figura 3 se muestra la vista esquemática de la tarea principal de la predicción meteorológica, es decir, una correspondencia entre diversos datos de observación y productos de previsión específicos.

Figura 3

Arquitectura del proyecto



Nota. Conceptualización de la estructura funcional del sistema propuesto. (MG, Betancourt, Gong, & Kleinert, 2021)

CAPÍTULO II

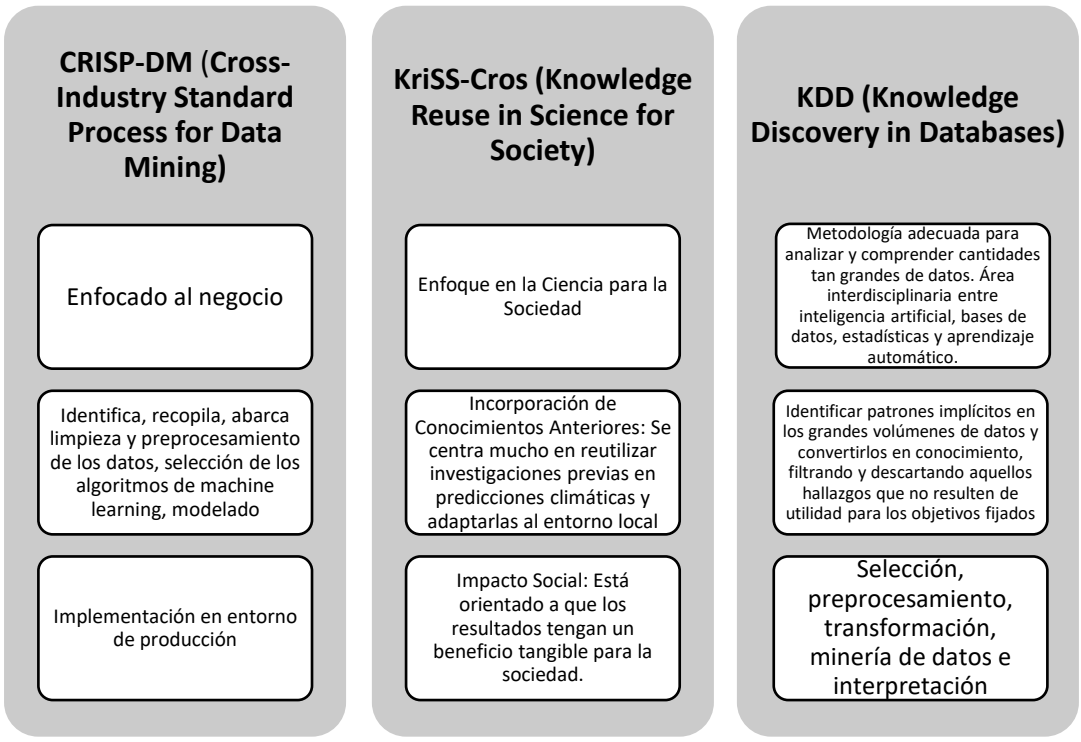
Metodología

En este tipo de proyecto, los datos recopilados se expresan generalmente en números y gráficos para confirmar teorías y predicciones. El método de investigación cuantitativa, la información fáctica se puede recopilar de muchas formas, como: Encuestas, Experimentos, Datos existentes, observaciones y análisis de contenido. (Thattamparambil, 2020)

En nuestro proyecto se evaluarán datos existentes y recopilaremos datos para incluirlos en nuestro análisis. Las metodologías CRISP-DM, KriSS-Cros y KDD son marcos de trabajo efectivos para desarrollar un proyecto de titulación en Data Science y Machine Learning. Ambas tienen un enfoque estructurado y te guían a través del proceso de solución de problemas. (Haya, 2021)

Figura 4

Características de las metodologías

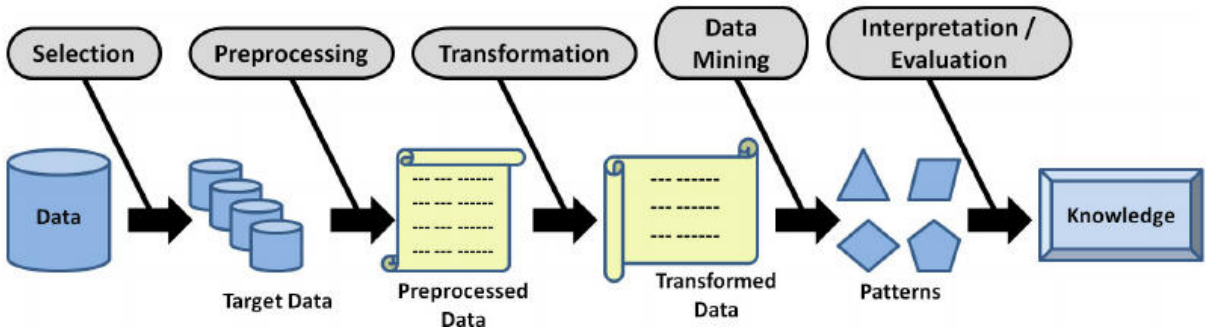


Nota. Características CRISP-DM, KriSS-Cros y KDD (Datahack Consulting, 2020)

La metodología KDD es una opción atractiva por el enfoque orientado hacia la creación del modelo y reutilización o adaptación de soluciones en nuestra data y la flexibilidad metodológica que brinda, es importante para la aplicación de los procesos para análisis y predicción es una metodología adecuada para analizar y comprender cantidades tan grandes de datos. Área interdisciplinaria entre inteligencia artificial, bases de datos, estadísticas y aprendizaje automático.

Figura 5

Metodología KDD



Nota. Fases de la metodología KDD (Gullo, 2020)

Fases de la metodología KDD

Recopilación de datos

Recopilación de datos de diferentes fuentes integrándose en un único reposito de llamado data warehouse, se puede recopilar e integrar información de fuentes externas. Los datos deben guardarse de forma segura y confiable. (ALTER, 2024)

Selección, Limpieza, transformación

Una vez recopilados todos los datos, durante esta fase, se seleccionan los datos que se entienden como más importantes y se transforman para poder procesarse con mayor facilidad. el resultado de esta fase se le conoce como vista minable (ALTER, 2024). Esta fase se compone de 3 fases:

- Selección de datos

- Limpieza de datos
- Transformación de datos

Minería de datos

Esta fase es la más representativa del proceso KDD y es habitual utilizarla para referirse a todo el proceso. En ella se aplican algoritmos de minería de datos con el objeto de obtener modelos. (ALTER, 2024)

Interpretación y evaluación de los modelos obtenidos

Como entrada a esta fase, se utiliza el modelo o patrón obtenidos en la fase anterior, los cuales son analizados y evaluados para convertirse en conocimiento (ALTER, 2024).

Software

Se selecciona Visual Studio Code como interfaz para la programación, con lenguaje Python y la extracción de datos en formato JSON. Se elige Visual Studio Code por que es compatible con casi todos los lenguajes de programación principales como, JavaScript, TypeScript, CSS y HTML, pero se pueden encontrar extensiones (Visual Studio Code, 2024).

Recopilación de Datos

Fuentes de datos

Existen varias fuentes de datos que nos proporcionan información del clima como NASA EarthData, INAMHI, OpenWeather, debemos analizar cada una de las opciones para acceder a estos datos en tiempo real.

Instituciones locales como el Instituto Nacional de Meteorología e Hidrología de Ecuador (INAMHI) que puedan ofrecer datos más detallados a nivel regional, el INAMHI opera y mantiene la infraestructura nacional de estaciones meteorológicas e hidrológicas: recopila, estudia, procesa, publica, y difunde la información hidrometeorológica. Contiene los datos de las estaciones meteorológicas como: precipitación, temperatura ambiente, humedad relativa, presión atmosférica, radiación solar, velocidad y dirección del viento de fechas

determinadas con un enfoque de predicción climática, basándose en precipitaciones por regiones. Por lo cual dispone de datos, muy limitados para nuestro análisis climático (INAMHI, 2024).

Otra plataforma muy útil es NASA EarthData, esta plataforma proporciona acceso a datos satelitales de varias misiones, incluyendo el sistema MODIS (Moderate Resolution Imaging Spectroradiometer) y los satélites Landsat y Sentinel, que pueden ofrecer información sobre temperaturas, precipitaciones, nubosidad, humedad del suelo, vegetación, entre otros. Estos datos son útiles para monitorear y modelar eventos climáticos como sequías o patrones de lluvia. NASA EarthData permite descargar conjuntos de datos y visualizarlos en su herramienta EarthData Search. Pero al momento de buscar data para Ecuador en la plataforma, no cuenta con muchos datos climáticos disponibles y deben ser usados en la herramienta EarthData. NASA EarthData se centra en datos ambientales y climáticos a gran escala, como cambios en la superficie terrestre, composición atmosférica y monitoreo ambiental a largo plazo (Devadiga, s.f.).

OpenWeather es una plataforma que proporciona pronósticos hiperlocales por minuto, datos históricos, estado actual y datos meteorológicos a corto plazo, anuales y pronosticados. Todos los datos están disponibles a través de API estándar de la industrial, la información se puede proporcionar como un archivo continuo o solo para momentos específicos en el tiempo. Los datos están disponibles a través de API y en forma de exportación masiva única para sus ubicaciones (OpenWeather, 2024).

Se evaluaron varias opciones y se seleccionó OpenWeather por su capacidad de proporcionar datos específicos para países como Ecuador, con información detallada para cada ubicación. Dado que Ecuador cuenta con diversos microclimas, esta API permite obtener datos precisos en tiempo real para cada región. Además, ofrece flexibilidad en el uso de la información y cuenta con una amplia documentación que facilita su implementación.

Selección del dataset

Los datos fueron descargados en formato JSON desde OpenWeather y cargados en un dataframe para su posterior análisis en este proyecto, contiene datos climáticos históricos obtenidos de la API, con las siguientes características: temperatura, humedad, presión, velocidad del viento, y nos proporciona las condiciones climáticas como soleado, lluvioso, nublado, etc. El problema de clasificación que se aborda en el código es la predicción del tipo de clima basándose en diferentes variables meteorológicas.

Seleccionamos 16 ciudades para el análisis climático en Ecuador y obtuvimos datos de cada ciudad desde el 1 de enero de 1979 hasta el 19 de octubre de 2024. La elección de este número limitado de ciudades se debió a la capacidad de procesamiento de nuestras computadoras fue un factor determinante, ya que trabajar con un conjunto de datos tan extenso requiere recursos computacionales significativos y factores de presupuesto, ya que el acceso a datos históricos de múltiples años implica costos adicionales.

Tabla 1

Ciudades, rango de fechas y horas del dataset

N°	Ciudad	Fecha Inicio	Fecha Fin	Hora
1	Quito	1/1/1979	19/10/2024	24 horas
2	Guayaquil	1/1/1979	19/10/2024	24 horas
3	Cuenca	1/1/1979	19/10/2024	24 horas
4	Puyo	1/1/1979	19/10/2024	24 horas
5	Machala	1/1/1979	19/10/2024	24 horas
6	Lago Agrio	1/1/1979	19/10/2024	24 horas
7	Loja	1/1/1979	19/10/2024	24 horas
8	Manta	1/1/1979	19/10/2024	24 horas
9	Esmeraldas	1/1/1979	19/10/2024	24 horas
10	Ambato	1/1/1979	19/10/2024	24 horas
11	Ibarra	1/1/1979	19/10/2024	24 horas
12	Santo Domingo	1/1/1979	19/10/2024	24 horas
13	Zamora	1/1/1979	19/10/2024	24 horas
14	Puerto Morona	1/1/1979	19/10/2024	24 horas
15	Santa Cruz Island	1/1/1979	19/10/2024	24 horas
16	Quevedo	1/1/1979	19/10/2024	24 horas

Nota. Lista de ciudades del Ecuador seleccionadas para el análisis climático, fechas y horas de extracción de la data.

Características importantes

Estas características se consideran importantes porque se sabe que influyen en el tipo de clima. El modelo intentará aprender la relación entre estas características y las clases objetivo para poder predecir el tipo de clima con precisión. El código busca construir un modelo que pueda clasificar el clima en diferentes categorías utilizando diversas variables meteorológicas como características de entrada.

Tabla 2

Características del dataset

Característica	Descripción
Dt	fecha y hora
City name	ciudades
Lat	latitud
Lon	longitud
Temp	Temperatura
Dew_point	punto de rocío
feels_like	Sensación térmica.
temp.min	Temperatura mínima.
temp.max	Temperatura máxima.
pressure	Presión atmosférica
humidity	Humedad relativa.
Wind_speed	Velocidad del viento.
Wind_deg	Dirección del viento en grados.
Rain_1h	Cantidad de lluvia 1h.
Rain_3h	Cantidad de lluvia 3h.
Clouds_all	Nubosidad.
Weather id	tipo de clima por valor
Weather main	tipo de clima por nombre
Weather description	descripción de clima

Nota. Lista de las características de data set y la descripción de cada una.

Estrategia de Predicción del Clima

Inicialmente, enfrentamos el desafío de la predicción del clima utilizando un enfoque unificado que intentaba abordar tanto la predicción (de variables continuas como temperatura, humedad, etc.) como la clasificación (de eventos discretos como lluvia/no lluvia, nublado/despejado, etc.) con un único modelo.

Sin embargo, esta estrategia no produjo resultados efectivos. La complejidad inherente para modelar simultáneamente variables continuas y discretas, junto con las posibles interacciones complejas entre ellas, dificultó al modelo capturar patrones precisos y realizar predicciones confiables.

Para superar estas limitaciones, decidimos separar los procesos en dos modelos distintos:

Modelo de Predicción: Este modelo se centra exclusivamente en la predicción de variables meteorológicas continuas, como temperatura, humedad, presión atmosférica, velocidad del viento, etc. Al enfocarse en un tipo específico de datos y tarea, el modelo puede especializarse y aprender patrones más precisos para la predicción de valores numéricos.

Modelo de Clasificación: Este modelo se dedica a la clasificación de eventos meteorológicos discretos, como la ocurrencia de lluvia, el tipo de nubosidad, la presencia de tormentas, etc. Al separar la clasificación de la predicción, este modelo puede concentrarse en identificar patrones específicos que conducen a la ocurrencia de eventos discretos, mejorando así la precisión de la clasificación.

Esta separación de tareas permite que cada modelo se especialice en un tipo específico de problema, lo que facilita el aprendizaje de patrones relevantes y la realización de predicciones más precisas. Además, esta estrategia modular permite una mayor flexibilidad y control sobre cada proceso individual, lo que facilita la optimización y el ajuste fino de cada modelo para obtener un mejor rendimiento general

Análisis Exploratorio de Datos

Información General

En la tabla 3 se describe la información general de los datos utilizado para el desarrollo del modelo planteado en este documento.

Tabla 3

Características del dataset completo

Característica	Descripción
Número de columnas	28
Número de Filas	6’445.858
Cantidad de datos perdidos NaN	48’161.272
Columnas duplicadas	0
Memoria total	1,3 GB

Nota. Características generadas por dataprofile.

Preprocesamiento de los Datos

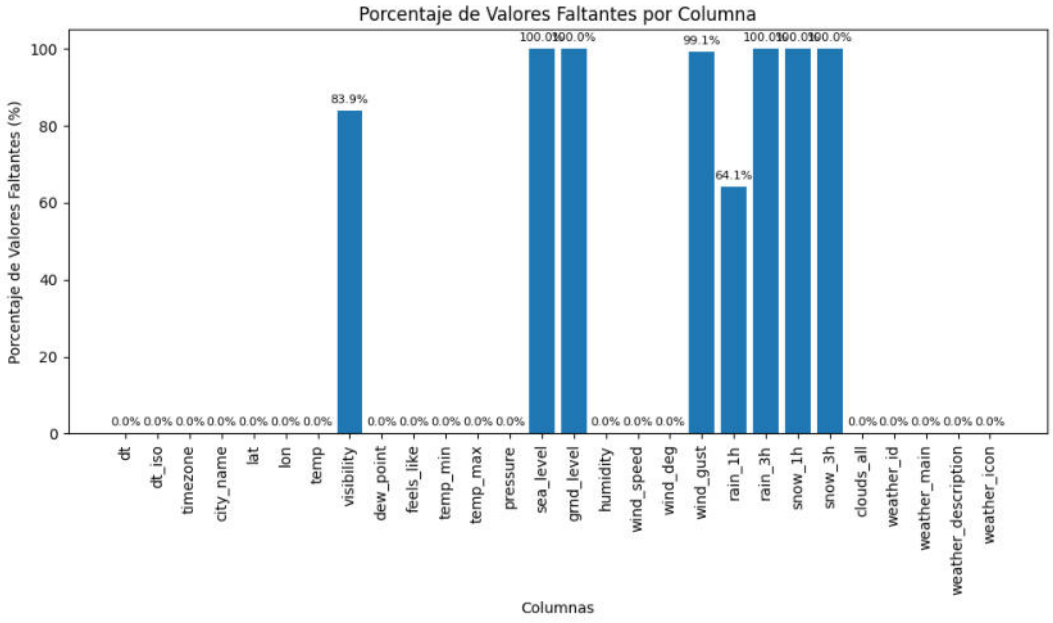
En este apartado se analizan los valores faltantes en las columnas de la data set, la matriz de correlación, la reducción de dimensionalidad y la selección de la data para cada modelo.

Identificación de valores faltantes

En el contexto de la manipulación y análisis de datos con Pandas, los valores faltantes, también conocidos como "missing values", representan datos que están ausentes en un conjunto de datos. Estos valores faltantes pueden surgir por diversas razones, como errores de entrada de datos, problemas de recolección de datos o simplemente porque la información no está disponible (CertiDevs, s.f.).

Figura 6

Valores faltantes en el dataset



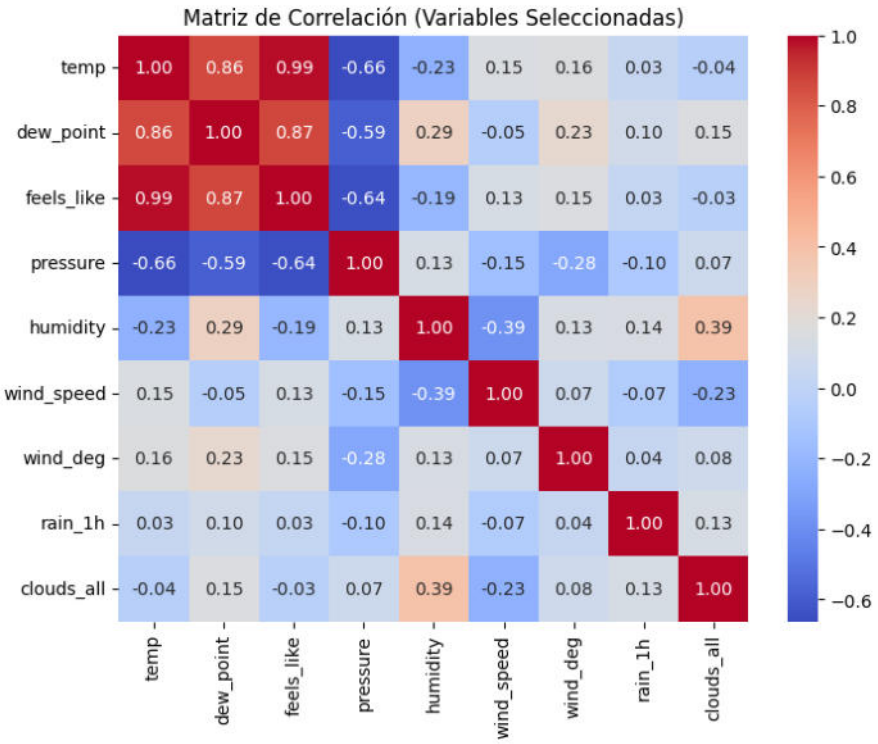
Nota. Porcentaje de valores faltantes de cada una de las variables.

Se determina que las variables, visibilidad, nivel del mar, nivel de la tierra, ráfaga de viento, lluvia de 3 horas, nieve de 1 hora y nieve de 3 horas, se deben eliminar debido a la falta de datos en estas características.

Matriz de correlación entre características

Figura 6

Matriz de correlación



Nota. Matriz de correlación de variables más relevantes del dataset.

La matriz representa una correlación positiva fuerte con los valores cercanos a 1, una correlación negativa fuerte con los valores cercamos a -1. Se visualizan correlaciones fuertes positivas entre temperatura, punto de rocío, y sensación térmica. Se visualizan correlaciones fuertes negativas entre presión, temperatura, punto de rocío y sensación térmica. Se puede considerar como una variable influye en otra directa o indirectamente pero también pueden estar involucrados otros factores como la multicolinealidad.

Reducción de dimensionalidad

Debido al gran volumen de datos que se deben procesar, es fundamental eliminar las variables con alta o baja correlación, un fenómeno conocido como multicolinealidad. Como menciona IBM, al incluir o excluir variables independientes en el modelo, los coeficientes estimados de cualquier predictor pueden cambiar drásticamente, lo que vuelve poco fiables e imprecisas las estimaciones de los coeficientes. La correlación entre dos o más predictores dificulta determinar el impacto individual de una variable en la salida del modelo. Esto se

debe a que un coeficiente de regresión mide el efecto de una variable predictora específica en la salida, asumiendo que las demás permanecen constantes. Sin embargo, si los predictores están correlacionados, puede ser imposible aislarlos completamente. (IBM, 2021).

Tabla 4

Selección de Variables para el PCA

Variable	Descripción	Consideraciones para PCA
temp	Temperatura	**Incluir:** Factor meteorológico importante
visibility	Visibilidad (distancia)	**Considerar:** Podría estar correlacionada con otros factores (humedad, nubes)
dew_point	Temperatura del punto de rocío	**Considerar:** Relacionada con la humedad, podría introducir redundancia
feels_like	Temperatura aparente	**Considerar:** Calculada a partir de otras variables, podría introducir redundancia
temp_min	Temperatura mínima	**Considerar:** Si temp representa el promedio, temp_min y temp_max podrían agregar información
temp_max	Temperatura máxima	**Considerar:** Similar a temp_min
pressure	Presión atmosférica	**Incluir:** Factor meteorológico importante
sea_level	Presión atmosférica a nivel del mar	**Considerar:** Si los datos son de una región costera, podría ser relevante
grnd_level	Presión atmosférica a nivel del suelo	**Considerar:** Similar a sea_level, pero podría ser más relevante para áreas del interior
humidity	Humedad relativa	**Incluir:** Factor meteorológico importante
wind_speed	Velocidad del viento	**Incluir:** Factor meteorológico importante
wind_deg	Dirección del viento (grados)	**Incluir:** Factor meteorológico importante
wind_gust	Velocidad de ráfaga de viento	**Considerar:** Podría estar correlacionada con la velocidad del viento
rain_1h	Volumen de lluvia en la última hora	**Incluir:** Importante para el análisis de precipitación

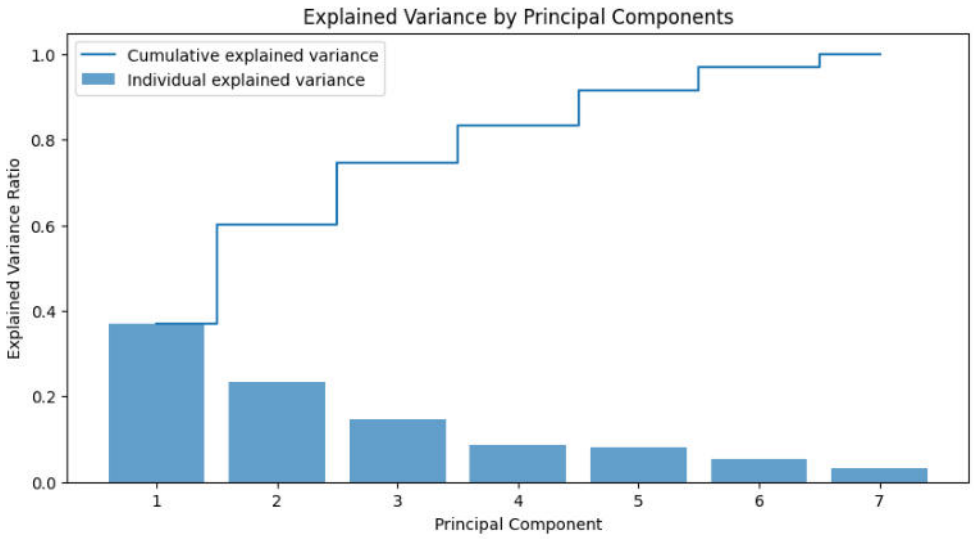
Variable	Descripción	Consideraciones para PCA
rain_3h	Volumen de lluvia en las últimas 3 horas	**Considerar:** Podría estar correlacionada con rain_1h
snow_1h	Volumen de nieve en la última hora	**Considerar:** Importante para el análisis de nieve, pero no tenemos este clima en Ecuador
snow_3h	Volumen de nieve en las últimas 3 horas	**Considerar:** Podría estar correlacionada con snow_1h
clouds_all	Nubosidad (porcentaje)	**Incluir:** Factor meteorológico importante

Nota. Análisis de variables para aplicar en el PCA de acuerdo con la matriz de correlación.

En este contexto tanto para el modelo clasificador como para el modelo predictor se eliminan las variables con alta y baja correlación para evitar la multicolinealidad. Adicional se aplica el análisis de componentes principales (PCA). Esta técnica permite eliminar el ruido y la información redundante, enfocándose en los componentes principales que capturan los patrones más relevantes de los datos (IBM, 2021).

Figura 8

Varianza de los Componentes principales



Nota. Análisis de los componentes principales.

- **Varianza acumulada después de 4 componentes: 83.40%** Esto significa que los primeros 4 componentes principales juntos capturan el 83.40% de la variación total en tu conjunto de datos original.
- **Varianza acumulada después de 5 componentes: 91.43%** Agregar el quinto componente principal aumenta la varianza explicada al 91.43%.
- **Varianza acumulada después de 6 componentes: 96.88%** Incluir el sexto componente principal aumenta aún más la varianza explicada al 96.88%.
- **Varianza acumulada después de 7 componentes: 100.00%** El uso de los 7 componentes principales explica el 100% de la varianza en tus datos originales, lo cual es de esperar ya que tienes 7 características originales

Se determinan el PCA para los 7 componentes de decidimos incluir en el análisis de los modelos.

Figura 9

PCA de 7 componentes

	daily_avg_temp	daily_avg_pressure	daily_avg_humidity	daily_avg_wind_speed	daily_avg_wind_deg	daily_avg_rain_1h	daily_avg_clouds_all
PC1	0.127820	-0.140937	0.525918	-0.505843	0.020980	0.451240	0.476733
PC2	0.664031	-0.626753	-0.264436	-0.115445	0.168122	0.031952	-0.231743
PC3	-0.195998	0.038156	-0.125221	0.039847	0.956851	0.045874	0.158722
PC4	0.121423	0.358756	-0.162986	0.129630	0.008127	0.816609	-0.382448
PC5	-0.302659	-0.552864	-0.035727	0.616968	-0.137014	0.328081	0.307252
PC6	0.571579	0.387705	-0.191283	0.344458	-0.034427	-0.065971	0.601835
PC7	0.266948	0.034395	0.757214	0.461631	0.188964	-0.120076	-0.301585

Nota. Análisis numérico de los componentes principales.

- **PC1**: Este componente está influenciado principalmente por 'daily_avg_humidity', 'daily_avg_rain_1h' y 'daily_avg_clouds_all' con cargas positivas, y 'daily_avg_wind_speed' con una carga negativa. Esto sugiere que PC1 podría representar una dimensión relacionada con los niveles generales de humedad y precipitación.

- **PC2**: Este componente está fuertemente asociado con 'daily_avg_temp' y 'daily_avg_pressure', con cargas opuestas. Podría representar una dimensión relacionada con las variaciones de temperatura y presión.
- **PC3:** Este componente está definido principalmente por 'daily_avg_wind_deg' con una alta carga positiva, lo que indica que captura información sobre la dirección del viento.
- **PC4:** Este componente está fuertemente influenciado por 'daily_avg_rain_1h' con una alta carga positiva y 'daily_avg_clouds_all' con una carga negativa, lo que podría representar una dimensión relacionada con la intensidad de la lluvia.
- **PC5:** Este componente tiene cargas significativas para 'daily_avg_wind_speed' y 'daily_avg_pressure', lo que sugiere que podría representar una dimensión relacionada con las interacciones entre la velocidad del viento y la presión atmosférica.
- **PC6:** Este componente está definido principalmente por 'daily_avg_temp' y 'daily_avg_clouds_all', posiblemente representando una dimensión relacionada con la temperatura y la nubosidad.
- **PC7:** Este componente está fuertemente influenciado por 'daily_avg_humidity' con una alta carga positiva y 'daily_avg_wind_speed' y 'daily_avg_temp' con cargas positivas, lo que podría representar una dimensión relacionada con la humedad y temperatura del aire con una influencia secundaria de la velocidad del viento.

Selección de la data para cada modelo

Datos para algoritmo de predicción.

Se toman las variables más relevantes para la predicción de acuerdo con el análisis previo.

Figura 10

Selección de características

```
columns = ['daily_avg_temp','daily_avg_pressure', 'daily_avg_humidity', 'daily_avg_wind_speed',
           'daily_avg_wind_deg', 'daily_avg_rain_1h', 'daily_avg_clouds_all']
```

Nota. Características para análisis de predicción.

Para el modelo de predicción, se tomas las 7 variables que se determinaron previamente en el análisis de PCA, adicional se considera tomar las horas debido a que nos proporciona la variabilidad diaria y permite realizar predicciones a corto plazo, mientras que los días son necesarios para identificar patrones a largo plazo y realizar predicciones a mediano y largo plazo.

Inicialmente adoptamos un enfoque amplio, utilizando todas las horas del día y todos los días de varios años como datos de entrada para nuestro modelo. La idea era proporcionar al modelo la mayor cantidad de información posible para que pudiera identificar patrones complejos y realizar predicciones precisas.

Sin embargo, este enfoque inicial presentó un desafío significativo: el modelo no lograba predecir ni identificar patrones de manera efectiva. La abundancia de datos resultaba en un problema de "ruido", donde las fluctuaciones y las variaciones a corto plazo oscurecían los patrones climáticos a largo plazo.

Ante esta dificultad, decidimos reenfocar nuestra estrategia y utilizar únicamente horas específicas de cada día como datos de entrada. Esta decisión se basó en la hipótesis de que ciertas horas del día son más representativas de las condiciones meteorológicas generales y pueden proporcionar información suficiente para realizar predicciones precisas sin la necesidad de procesar la totalidad de los datos.

En el sistema de predicción del clima, se ha optado por utilizar datos de horas específicas del día en lugar de la totalidad del conjunto de datos. Esta decisión se basa en las

siguientes consideraciones: reducción del procesamiento y carga computacional, eliminación de ruido, mejora de precisión. Seleccionamos las horas representativas que mejor reflejen las condiciones meteorológicas generales del día, para lo cual hicimos pruebas aleatorias para definir que horas nos dan una mejor precisión.

Tabla 5

Horas del dataset de predicción

Período del día	Horas seleccionadas
Mañana	06h00 - 08h00 -10h00
Tarde	11h00 - 13h00 -15h00
Noche	18h00 - 20h00 -22h00

Nota. Horas seleccionadas para el análisis de predicción del clima

Debido a que la relación de las variables meteorológicas y el tiempo son muy complejos porque son muy cambiantes en el clima de Ecuador, nos centramos también en calcular los promedios diarios para cada una de las variables meteorológicas durante las horas que definimos para la predicción.

Figura 11

Promedios para la data de predicción

```
# Group the DataFrame by date and get the first occurrence of each day
data_morn = calculate_averages(data_morn, columns_to_average=['temp', 'dew_point', 'feels_like', 'temp_min', 'temp_max', 'pressure', 'humidity',
                                                              'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all'], hours_to_average=hours_morning_to_average)
# Use pd.Grouper to specify the frequency
morning_data = data_morn.groupby([pd.Grouper(key='dt', freq='D'), 'city_name']).first().reset_index()
# Display the new DataFrame
morning_data.head(10)
```

Mostrar salida oculta

```
# Group the DataFrame by date and get the first occurrence of each day
data_aft = calculate_averages(data_aft, columns_to_average=['temp', 'dew_point', 'feels_like', 'temp_min', 'temp_max', 'pressure', 'humidity',
                                                            'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all'], hours_to_average=hours_aft_to_average)
# Use pd.Grouper to specify the frequency
aft_data = data_aft.groupby([pd.Grouper(key='dt', freq='D'), 'city_name']).first().reset_index()
# Display the new DataFrame
aft_data.head(10)
```

Mostrar salida oculta

```
# Group the DataFrame by date and get the first occurrence of each day
data_ngt = calculate_averages(data_ngt, columns_to_average=['temp', 'dew_point', 'feels_like', 'temp_min', 'temp_max', 'pressure', 'humidity',
                                                            'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all'], hours_to_average=hours_ngt_to_average)
# Use pd.Grouper to specify the frequency
ngt_data = data_ngt.groupby([pd.Grouper(key='dt', freq='D'), 'city_name']).first().reset_index()
# Display the new DataFrame
ngt_data.head(10)
```

Nota. Cálculo de promedios de las variables con las horas seleccionadas para el análisis de predicción

Se evalúan las características, se completan con ceros las variables que no poseen valores y se guardan las variables más importantes en el dataframe principal para el análisis de predicción.

Figura 12

Data sets para predicción

```
morning_data = drop_columns(morning_data, columns_drop, columns)
morning_data.head()
```

	daily_avg_temp	daily_avg_pressure	daily_avg_humidity	daily_avg_wind_speed	daily_avg_wind_deg	daily_avg_rain_1h	daily_avg_clouds_all
0	15.976667	1018.000000	98.333333	0.583333	236.333333	0.293333	99.666667
1	15.560000	1017.333333	97.333333	0.676667	231.000000	0.325000	100.000000
2	17.136667	1017.666667	97.000000	0.350000	140.000000	0.275000	98.333333
3	15.166667	1019.333333	96.000000	0.900000	105.666667	0.000000	89.333333
4	15.650000	1020.000000	98.666667	0.456667	302.000000	0.970000	99.333333

```
aft_data = drop_columns(aft_data, columns_drop, columns)
aft_data.head()
```

	daily_avg_temp	daily_avg_pressure	daily_avg_humidity	daily_avg_wind_speed	daily_avg_wind_deg	daily_avg_rain_1h	daily_avg_clouds_all
0	17.156667	1018.666667	90.333333	0.853333	265.333333	0.340000	99.666667
1	16.953333	1018.333333	91.333333	0.586667	125.000000	0.210000	98.666667
2	17.310000	1020.333333	93.000000	0.943333	100.666667	0.693333	98.333333
3	17.523333	1020.333333	78.666667	1.120000	89.000000	0.000000	82.666667
4	17.153333	1022.333333	96.666667	0.840000	275.333333	0.880000	100.000000

```
ngt_data = drop_columns(ngt_data, columns_drop, columns)
ngt_data.head()
```

	daily_avg_temp	daily_avg_pressure	daily_avg_humidity	daily_avg_wind_speed	daily_avg_wind_deg	daily_avg_rain_1h	daily_avg_clouds_all
0	21.350000	1015.333333	88.000000	1.630000	299.666667	1.776667	97.333333
1	19.943333	1015.333333	81.333333	1.786667	314.666667	1.976667	99.000000
2	21.506667	1015.666667	81.000000	1.620000	223.000000	0.996667	86.333333
3	21.660000	1017.000000	69.333333	1.906667	317.666667	0.810000	64.666667
4	20.293333	1018.000000	94.333333	2.236667	308.000000	1.056667	95.666667

Nota. Data set listo para análisis de predicción.

Clase objetivo para el predictor

Las clases objetivo para el predictor en nuestro código cambia dinámicamente en función de la característica actual que se está procesando. Se busca predecir los valores futuros de diferentes variables meteorológicas, y cada una de esas variables se convierte en la clase objetivo-numérica.

Tabla 6

Clase Objetivo Predictor

Característica	Descripción
Daily avg Temp	Temperatura promedio diaria
Daily avg Pressure	Presión promedio diaria
Daily avg Humidity	Humedad promedio diaria
Daily avg Wind Speed	Velocidad promedio del viento diario
Daily avg Wind deg	Grados del viento promedio diario
Daily avg Rain 1h	Lluvia de una hora promedio diario
Daily avg Clouds all	Nubes promedio diario

Nota. Descripción de cada una de las variables que deseamos predecir

Datos para algoritmo de clasificación.

Se eliminan las variables en el dataset, que no aportarán información para la clasificación y se crea una nueva lista con los parámetros para la clasificación, de acuerdo con el análisis previo de PCA.

Figura 13

Selección de características

```
[ ]  params = ['temp', 'pressure', 'humidity', 'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all', 'weather_main']

[21] df = df.drop(columns=['dt', 'dt_iso', 'timezone', 'city_name', 'lat', 'lon', 'visibility', 'dew_point', 'feels_like',
                           'temp_min', 'temp_max','sea_level', 'grnd_level','wind_gust','rain_3h',
                           'snow_1h', 'snow_3h', 'weather_id', 'weather_description', 'weather_icon'])
```

Nota. Selección de características para análisis de clasificación

Se evalúan las características, se completan con ceros las variables que no poseen valores y se guardan las variables más importantes en el dataframe principal para el análisis de clasificación.

Figura 14

Data set para clasificación

	temp	pressure	humidity	wind_speed	wind_deg	rain_1h	clouds_all	weather_main
0	16.71	1015	92	1.03	339	0.0	68	0
1	16.77	1016	93	0.74	11	0.0	100	0
2	16.92	1017	93	0.46	49	0.0	95	0
3	14.51	1017	100	0.81	100	0.0	31	0
4	15.04	1017	99	1.14	124	0.0	37	0
...	...	...	...	...	...	...	...	...
6445853	28.33	1010	79	3.60	290	0.0	75	0
6445854	26.64	1008	78	3.60	270	0.0	75	0
6445855	27.36	1008	79	3.60	280	0.0	75	0
6445856	26.23	1008	82	5.35	232	0.0	100	0
6445857	25.22	1010	83	4.48	227	0.0	99	1

6445858 rows × 8 columns

Nota. Data set listo para análisis de clasificación.

Clase objetivo para el clasificador

Las clases objetivo para el clasificador son los diferentes tipos de clima representados por la columna weather_main en nuestro dataset. Cada valor único en esta columna representa una clase distinta de clima, por ejemplo, lluvia ligera, nublado, despejado, etc.

Se evalúa la variable objetivo-categórica y se determina que posee 14 clases, lo cual nos da la referencia el aplicar un modelo de clasificación multiclase.

Tabla 7

Tipos de clima

Clima principal	Descripción
Clouds	Nublado
Rain	Lluvia
Clear	Despejado
Haze	Calina
Fog	Niebla
Drizzle	Llovizna
Mist	Neblina
Thunderstorm	Tormenta eléctrica
Smoke	Humo

Clima principal	Descripción
Dust	Polvo
Ash	Ceniza
Tornado	Tornado
Squall	Chubasco Violento
Snow	Nieve

Nota. Lista de tipos de clima de OpenWeather

Análisis de Modelos

Algunos modelos pueden usarse tanto para clasificación como para predicción, es importante entender las diferencias entre ambos tipos de problemas y configurar el modelo adecuadamente para la tarea específica.

A continuación, se detallan los métodos de aprendizaje automático que se evaluaron, como regresión, árboles de decisión, o redes neuronales. Se considera un análisis minucioso de los modelos para la geografía montañosa y tropical del Ecuador, teniendo en cuenta los microclimas que existen dentro del país.

Los algoritmos como árboles de decisión y redes neuronales artificiales. Tienen un sin número de aplicaciones prácticas, abarcando numerosos sectores e industrias. Sin embargo, para obtener un mejor rendimiento de cada uno de ellos es importante experimentar con los parámetros de cada algoritmo. Además, hay que considerar las ventajas y desventajas que presentan cada uno. (Morocho, 2024). En la tabla 8 se presentan las ventajas y desventajas de cada uno.

Tabla 8

Criterios para la selección de un modelo

Algoritmo	Rapidez de Predicción	Rapidez de entrenamiento	Memoria	Ajuste Requerido	Evaluación general
Regresión Logística (SVM lineal)	Rápido	Rápido	Pequeño	Mínimo	Bueno para pequeños problemas y tiene límites para decisiones lineales
Arboles de Decisión	Rápido	Rápido	Pequeño	Medio	Bueno generalmente, pero propenso a sobreajuste
Regresión Logística (SVM no lineal)	Lento	Lento	Medio	Medio	Bueno para problemas binarios, y maneja bien datos de alta dimensión
Vecinos Cercanos	Moderado	Mínimo	Medio	Mínimo	Bajo en precisión, pero fácil de usar e interpretar
Naive Bayes	Rápido	Rápido	Medio	Medio	Ampliamente usado para texto, usado en el filtrado de spam
Aprendizaje en Conjunto	Moderado	Lento	Varios	Medio	Alta precisión y buen desempeño para bases de datos pequeños y medianos
Redes Neuronales	Moderado	Lento	Medio a Grande	Lotes	Popular para clasificación, reconocimiento, comprensión y pronóstico

Nota. Ventajas y desventajas de algoritmos de clasificación y predicción. (Morocho, 2024)

Dentro del análisis de cada algoritmo se puede evaluar ciertos modelos que se aplican a predicción y clasificación de clima en las tablas 9, 10 y 11 se detallan algunos modelos de aprendizaje automático, como regresión, árboles de decisión, o redes neuronales. Adicional se considera un análisis minucioso para cada modelo.

Tabla 9

Modelo Series Temporales y características

Descripción general	Algoritmo	Descripción de cada algoritmo
Para datos de series temporales y patrones de tendencias de predicción de climas	ARIMA	Es un modelo estadístico que utiliza valores pasados para predecir valores futuros de una serie temporal.
	SARIMA	Es una extensión de ARIMA diseñada para manejar datos de series temporales con patrones estacionales. Se rige por el mismo mecanismo que ARIMA, pero tiene en cuenta factores estacionales que pueden afectar los datos.
	LSTM	Es un modelo de deep learning con el que se pueden manejar datos de series temporales con dependencias a largo plazo. Es capaz de identificar patrones complejos en datos de series temporales. A diferencia de los modelos tradicionales como ARIMA, LSTM no requiere que los datos sean estacionarios.

Nota. Características de cada modelo en aplicaciones climáticas. (EducaOpen, 2023)

Tabla 10

Modelo Árboles de decisión y características

Descripción general	Algoritmo	Descripción de cada algoritmo
Manejan gran cantidad de datos, manejan cambios no lineales y no se aplica a datos temporales	Random Forest para Clasificación	Este tipo de árbol se utiliza cuando la variable objetivo es categórica, es decir, pertenece a un conjunto discreto de clases o categorías. El árbol de clasificación divide el conjunto de datos en función de las características para clasificar las instancias en diferentes categorías
	Random Forest para Regresión	Un árbol de regresión se utiliza cuando la variable objetivo es numérica o continua, es decir, puede tomar cualquier valor dentro de un rango específico. El árbol de regresión divide el conjunto de datos en función de las características para predecir un valor numérico.

Nota. Características de cada modelo en aplicaciones climáticas. (EducaOpen, 2023)

Tabla 11

Modelo Redes Neuronales y características

Descripción general	Algoritmo	Descripción de cada algoritmo
Patrones a lo largo del tiempo, patrones secuenciales, para climas variables Útiles para secuencias de cambio	Recurrentes	Las redes neuronales recurrentes tienen conexiones que forman ciclos, permitiéndoles mantener una especie de memoria de la información previa. Son esenciales para tareas que involucran secuencias de datos, como el reconocimiento de voz o la traducción de idiomas.
	LSTM	Es un modelo de deep learning con el que se pueden manejar datos de series temporales con dependencias a largo plazo. Es capaz de identificar patrones complejos en datos de series temporales. A diferencia de los modelos tradicionales como ARIMA, LSTM no requiere que los datos sean estacionarios.
	Perceptrón	El perceptrón es la forma más simple de red neuronal, ideal para la clasificación de patrones linealmente separables.
	Perceptrón multicapa	El perceptrón multicapa es una extensión del perceptrón simple. Consiste en múltiples capas de neuronas, lo que permite abordar problemas no lineales. La presencia de varias capas ocultas permite a esta red aprender características más complejas.
	Redes neuronales convolucionales	Son redes neuronales especializadas en el procesamiento de imágenes. Utilizan capas convolucionales para detectar características relevantes y reducir la dimensionalidad
	Redes neuronales generativas adversarias (GAN)	Son un enfoque especial de aprendizaje no supervisado que se compone de dos redes complementarias: un generador y un discriminador, que compiten entre sí. El generador crea muestras sintéticas, mientras que el discriminador intenta distinguir entre muestras reales y sintéticas

Nota. Características de cada modelo en aplicaciones climáticas. (EducaOpen, 2023)

Versiones

Para evaluar el mejor modelo, realizamos más de 25 pruebas con diferentes algoritmos, y en las siguientes tablas presentamos los avances más relevantes con son sus características avances, dificultades en clasificación y predicción para el proyecto.

Tabla 12

Pruebas de modelos para clasificación y predicción primera versión

Detalle	Descripción
Algoritmo	LSTM
Épocas	20
Tamaño de la data	1744 rows x 25 columns
Ciudad(es)	Quito
Años analizados	Json OpenWeather 16 días
Hiperparámetros	optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
Tiempo de respuesta	2s
Métrica principal	Accuracy=0.1189
Resultados	Baja precisión
Dificultades (errores)	Utilizar train_test_split que realiza de forma aleatoria la separación de conjunto de train y test, pero el proyecto es series temporales requiere orden en los conjuntos de datos Descargar la información de la aplicación Comprender la estructura del objeto de información json Entender de ventanas/secuencia en las series temporales
Ventajas	Comprender la importancia d de los conjuntos de datos Train, Test y Val en series temporales Usar de encoder para las variables categóricas Optimizar la data mediante el escalamiento
Desventajas	Baja precisión
Requisitos Técnicos	tensorflow.keras.layers sklearn.preprocessing Google Colab

Tabla 13

Pruebas de modelos para clasificación y predicción quinta versión

Detalle	Descripción
Algoritmo	MLPClassifier
Épocas	100
Tamaño de la data	1568 rows × 26 columns
Ciudad(es)	Capitales de América del Sur
Años analizados	Json OpenWeather 16 días
Hiperparámetros	loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), metrics=['accuracy']
Tiempo de respuesta	2s
Métrica principal	Adam -categorical_crossentropy : 86.25%
Resultados	El problema requiere generar predicciones de los valores a futuro en las series temporales y este modelo es limitado para este requerimiento
Dificultades (errores)	Estandarizar de la información en el objeto JSON porque se envían las coordenadas, pero no se encontraba la cuidad, pero si tiene una respuesta
Ventajas	Comprender la información para la toma decisión para la adquisición en formato CVS de 16 cuidades del Ecuador Entender el objetivo del estudio y el uso adecuado del modelo IA en las condiciones y resultados esperados
Desventajas	La funcionalidad de este modelo no aplica a series temporales
Requisitos Técnicos	MLPClassifier_keras Google Colab

Tabla 14

Pruebas de modelos para clasificación y predicción décima versión

Detalle	Descripción
Algoritmo	LSTM
Épocas	5
Tamaño de la data	422367 rows × 13 columns

Detalle	Descripción
Ciudad(es)	Quito Cuenca Guayaquil Puyo Machala Lago Agrio Loja Manta Esmeraldas Ambato Ibarra Santo Domingo Zamora Puerto Morona Santa Cruz Island Quevedo
Años analizados	Data OpenWeather desde 1999 hasta 2001
Hiperparámetros	optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
Tiempo de respuesta	420s
Métrica principal	Accuracy y Validation loss
Resultados	El modelo comete un error alto al hacer predicciones y obtiene una precisión media
Dificultades (errores)	Se dificulta ejecutar el código con todos los años que tenemos descargados de la aplicación Se dificulta ejecutar más épocas por el tiempo de respuesta Agregando las épocas no se eleva el accuraccy significativamente Escalamos las características para el preprocesamiento Explorar técnicas para reducir el sobreajuste, como aumentar la tasa de dropout, agregar regularización, reducir el número de unidades en tus capas LSTM y usar más datos
Ventajas	Se crean series temporales, cada secuencia representa una ventana de datos pasados, y las etiquetas representan los valores futuros a predecir
Desventajas	Costo computacional: tiempos de entrenamiento más largos y mayores requisitos de memoria, para grandes conjuntos de datos o secuencias largas
Requisitos Técnicos	Google Colab

Tabla 15

Pruebas de modelos para clasificación y predicción décima primera versión

Detalle	Descripción
Algoritmo	LSTM
Épocas	20
Tamaño de la data	3092219 rows × 13 columns
Ciudad(es)	Quito Cuenca Guayaquil Puyo Machala Lago Agrio Loja Manta Esmeraldas Ambato Ibarra Santo Domingo Zamora Puerto Morona Santa Cruz Island Quevedo
Años analizados	Data OpenWeather desde 1980 hasta 2001
Hiperparámetros	optimizer=Adam con una tasa de aprendizaje de 0.01, loss='categorical_crossentropy', metrics=['accuracy']
Tiempo de respuesta	665s
Métrica principal	Accuracy = 0.5413 y Validation Loss = 1.5410
Resultados	El modelo tiene una precisión moderada tanto en el conjunto de entrenamiento como en el de validación, alrededor del 54%. La pérdida en el conjunto de validación es ligeramente mayor que en el de entrenamiento, lo que podría indicar un ligero sobreajuste
Dificultades (errores)	Agregamos más data para evaluar el modelo con todas las ciudades, se evalúa que el error es analizar todas las ciudades en conjunto, pese a que se aplican ventanas
Ventajas	Más datos ayudan al modelo a aprender mejores patrones.
Desventajas	Costo computacional: tiempos de entrenamiento más largos y mayores requisitos de memoria, para grandes conjuntos de datos o secuencias largas
Requisitos Técnicos	Visual Studio & Python

Tabla 16

Pruebas de modelos para clasificación y predicción décima sexta versión

Detalle	Descripción
Algoritmo	PCA
Épocas	0
Tamaño de la data	422367 rows × 13 columns
Ciudad(es)	Quito Cuenca Guayaquil Puyo Machala Lago Agrio Loja Manta Esmeraldas Ambato Ibarra Santo Domingo Zamora Puerto Morona Santa Cruz Island Quevedo
Años analizados	Data OpenWeather desde 2020 hasta 2024
Hiperparámetros	Linear Regression, loss='mean_squared_error', metrics=['mean_absolute_error']
Tiempo de respuesta	0s
Métrica principal	Mean Absolute Error
Resultados	Se identificaron los componentes fundamentales y más importantes para nuestro propósito.
Dificultades (errores)	El análisis de PCA fungió en una herramienta fundamental para identificar y determinar si las variables escogidas eran las correctas. La Regresión Lineal no otorgó buenos resultados por lo que se descaró su uso.
Ventajas	Se identifican las variables más relevantes y se descartan las que afectaban al rendimiento del modelo.
Desventajas	Se probó con un modelo de Regresión Lineal que no fue muy fructífero.
Requisitos Técnicos	Visual Studio & Python

Tabla 17

Pruebas de modelos para clasificación y predicción vigesimoprimera versión

Detalle	Descripción
Algoritmo	Random Forest Classifier
Épocas	100 estimators
Tamaño de la data	422367 rows × 13 columns
Ciudad(es)	Quito Cuenca Guayaquil Puyo Machala Lago Agrio Loja Manta Esmeraldas Ambato Ibarra Santo Domingo Zamora Puerto Morona Santa Cruz Island Quevedo
Años analizados	Data OpenWeather desde 2020 hasta 2024
Hiperparámetros	estimators = 100 random_state = 42 accuracy = accuracy_score
Tiempo de respuesta	60s
Métrica principal	Random Forest Classifier
Resultados	El algoritmo es capaz de clasificar y etiquetar correctamente los parámetros.
Dificultades (errores)	Dividir el modelo en 2 partes, un clasificador y una red neuronal resultó ser la mejor opción para lo que se buscaba.
Ventajas	Mayor precisión y velocidad de ejecución.
Desventajas	Desarrollar 2 modelos.
Requisitos Técnicos	Visual Studio & Python

Selección de Modelo

Tras una profunda exploración y rigurosas pruebas con el conjunto de datos, se optó por emplear el algoritmo Random Forest Classifier para las tareas de clasificación y una combinación de redes LSTM (Long Short-Term Memory) con técnicas de Ventanas para las

tareas de predicción. Esta decisión se fundamentó en el objetivo de aprovechar las fortalezas de cada modelo para abordar los requerimientos específicos del problema.

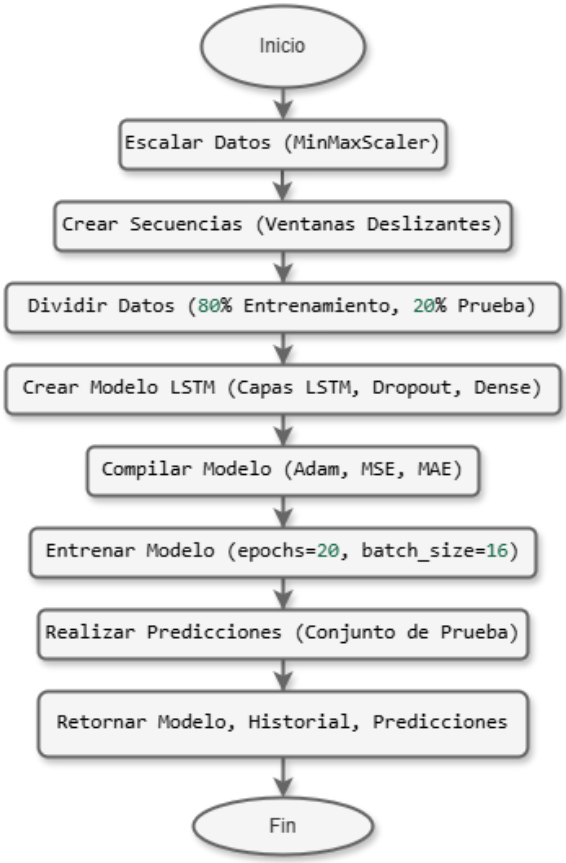
Predictor

Para abordar el problema de predicción de variables, se implementó un modelo LSTM (Long Short-Term Memory). Esta elección se fundamenta en la capacidad del LSTM para procesar eficientemente secuencias de datos, como se ha demostrado en la literatura y en estudios previos. Dicho modelo permitirá predecir el comportamiento futuro de las variables a partir de patrones identificados en las secuencias históricas.

Se crea la función para entrenar los datos con el modelo LSTM y se presenta el funcionamiento del modelo en el diagrama de flujo de la figura ##

Figura 16

Modelo LSTM



Nota. Proceso de funcionamiento del modelo LSTM para predicción.

- **Escalar Datos:** En el primer paso, se crea un objeto para escalar los datos entre 0 y 1, se aplica el escalado a las características de la data utilizando fit_transform. Se crea un nuevo dataframe con los datos escalados.
- **Llama a la función de ventanas:** Segundo, se llama a la función create_sequences_classification para generar secuencias de datos con las ventanas deslizantes y etiquetas para el entrenamiento del modelo LSTM.
- **Dividir datos:** Se divide el conjunto de datos en conjuntos de entrenamiento y prueba utilizando un 80% para entrenamiento y un 20% para prueba. Se crea un modelo secuencial utilizando la librería Keras, se añaden capas LSTM con 256 y 128 unidades, respectivamente, se incluyen capas Dropout para prevenir el sobreajuste, se añade una capa Dense con días a futuro las cuales son las unidades para la salida del modelo.
- **Compilación del Modelo:** Se compila el modelo utilizando el optimizador 'adam', la función de pérdida 'mse' (error cuadrático medio) y la métrica 'mae' (error absoluto medio).
- **Entrenamiento del Modelo:** Se entrena el modelo utilizando los datos de entrenamiento, especificando el número de épocas (20) y el tamaño del batch (16). Se utiliza el conjunto de validación para evaluar el rendimiento del modelo durante el entrenamiento.
- **Predicciones:** Se realizan predicciones utilizando el conjunto de prueba.
- **Retorno de Resultados:** La función retorna el modelo entrenado, el historial de entrenamiento y las predicciones realizadas.

Conjunto de entrenamiento, validación y prueba

Inicialmente, se intentó dividir los datos en conjuntos X e Y de forma tradicional. Sin embargo, este enfoque resultó en predicciones inconsistentes y fuera de rango. El problema

radica en que la separación tradicional elimina la relación temporal de los datos, fragmentando la información de continuidad y presentando al modelo datos aleatorios sin un orden temporal definido.

Para preservar la estructura temporal, se implementó un enfoque de ventanas deslizantes. Este método crea subsecuencias del dataset, donde cada ventana contiene una porción del historial como entrada para predecir un valor futuro.

Figura 17

Ventanas

```python
def create_sequences_classification(data, target, window_size, dias_futuro):
    sequences = []
    labels = []

    for i in range(len(data) - window_size - dias_futuro):
        # Agregar la ventana de datos como secuencia de entrada
        sequences.append(data[i:i + window_size])

        # Las etiquetas para los siguientes dias_futuro días
        labels.append(target[i + window_size + 1:i + window_size + dias_futuro + 1])

    return np.array(sequences), np.array(labels)
```

Nota. Función que crea las ventanas para integrarlo al modelo de predicción.

Hiperparámetros del modelo de Predicción

A continuación, se presentas los hiperparámetros del modelo LSTM y la configuración del entrenamiento.

Tabla 18

Parámetros para el modelo LSTM

Hiperparámetros	Valores
Dropout	0.2
Días a futuro	3
Capas LSTM	[256, 128]
Días Historia	16

Hiperparámetros	Valores
Features	Daily_avg_temp, daily_avg_pressure, daily_avg_humidity, daily_avg_wing_speed, daily_avg_deg, daily_avg_rain_1h, daily_avg_clouds
Optimizador	adam
Loss	Mse
Metrics	mae

Nota. Valores y parámetros para el modelo LSTM

Entrenamiento del modelo

Figura 18

Función para el entrenamiento

```
def predictions_scaled(df, model, features, target):
  scaler_features = MinMaxScaler()
  scaler_target = MinMaxScaler()
  data_features_scaled = scaler_features.fit_transform(df[features])
  data_target_scaled = scaler_target.fit_transform(df[target])
  target_scaler_filename = f'{target[0]}_target_scaler.joblib'
  joblib.dump(scaler_target, target_scaler_filename)
  features_scaler_filename = f'{target[0]}_features_scaler.joblib'
  joblib.dump(scaler_features, features_scaler_filename)
  sequences, labels = create_sequences_classification(
    data_features_scaled,
    data_target_scaled.flatten(),
    dias_historia,
    dias_futuro
  )
  train_size = int(0.8 * len(sequences))
  X_train, X_test = sequences[:train_size], sequences[train_size:]
  y_train, y_test = labels[:train_size], labels[train_size:]
  predictions = model.predict(X_test)
  predictions_rescaled = scaler_target.inverse_transform(predictions)
  y_test_rescaled = scaler_target.inverse_transform(y_test)

  return predictions_rescaled, y_test_rescaled
```

Nota. Función para el entrenamiento de la data

Esta función se encarga de realizar predicciones utilizando un modelo LSTM pre-entrenado y luego reescalar tanto las predicciones como los valores reales a su escala

original. Este proceso es crucial para obtener resultados interpretables y comparables con los datos originales, ya que el modelo fue entrenado con datos escalados.

- **Escalado Inicial:** Se escalan los datos de entrada para que estén en un rango entre 0 y 1 utilizando MinMaxScaler
- **Creación de Secuencias y División de Datos:** Se utiliza la función create_sequences_classification para generar secuencias de datos y etiquetas a partir de los datos escalados. Se divide el conjunto de datos en conjuntos de entrenamiento (80%) y prueba (20%) para evaluar el rendimiento del modelo.
- **Predicciones:**Se utiliza el modelo pre-entrenado para realizar predicciones sobre el conjunto de prueba (X_test). Las predicciones resultantes (predictions) estarán en la escala escalada (entre 0 y 1).
- **Reescalado:** Se utiliza el método inverse_transform del escalador de la variable objetivo (scaler_target) para reescalar las predicciones (predictions_rescaled) y los valores reales (y_test_rescaled) a su escala original. Esto permite interpretar las predicciones en el contexto de los datos originales.
- **Retorno:** La función retorna las predicciones reescaladas (predictions_rescaled) y los valores reales reescalados (y_test_rescaled).

Prueba

Figura 19

Función de prueba para la predicción.

```
targetOld = 'pepe'
for feature in features:
    target[0] = feature

    print(f'\nTARGET {feature.upper()}:')

    # mañana
    print(f'\nProcesando {feature} en mañana:')
    model, history, predictions = train_fit_predit(morning_data, features, target, num_clases, N, dias_historia, dias_futuro)
    accuracy_graf(history)
    df_results_mrn = show_results(morning_data, model, features, target)

    # tarde
    print(f'\nProcesando {feature} en tarde:')
    model, history, predictions = train_fit_predit(aft_data, features, target, num_clases, N, dias_historia, dias_futuro)
    accuracy_graf(history)
    df_results_aft = show_results(aft_data, model, features, target)

    # noche
    print(f'\nProcesando {feature} en noche:')
    model, history, predictions = train_fit_predit(ngt_data, features, target, num_clases, N, dias_historia, dias_futuro)
    accuracy_graf(history)
    df_results_ngt = show_results(ngt_data, model, features, target)

    typeAxis = 0
    if target[0] == targetOld:
        df_final_results = pd.concat([df_final_results, df_results_mrn, df_results_aft, df_results_ngt], axis=0)
    else:
        df_final_results = pd.concat([df_final_results, df_results_mrn, df_results_aft, df_results_ngt], axis=1)
```

Nota. Función de prueba de la data.

El código tiene como objetivo principal predecir con el modelo LSTM para las 7 características el conjunto de datos de la mañana, tarde y noche, guarda cada modelo entrenado en un archivo separado. Además, genera gráficos de precisión y pérdida para cada modelo.

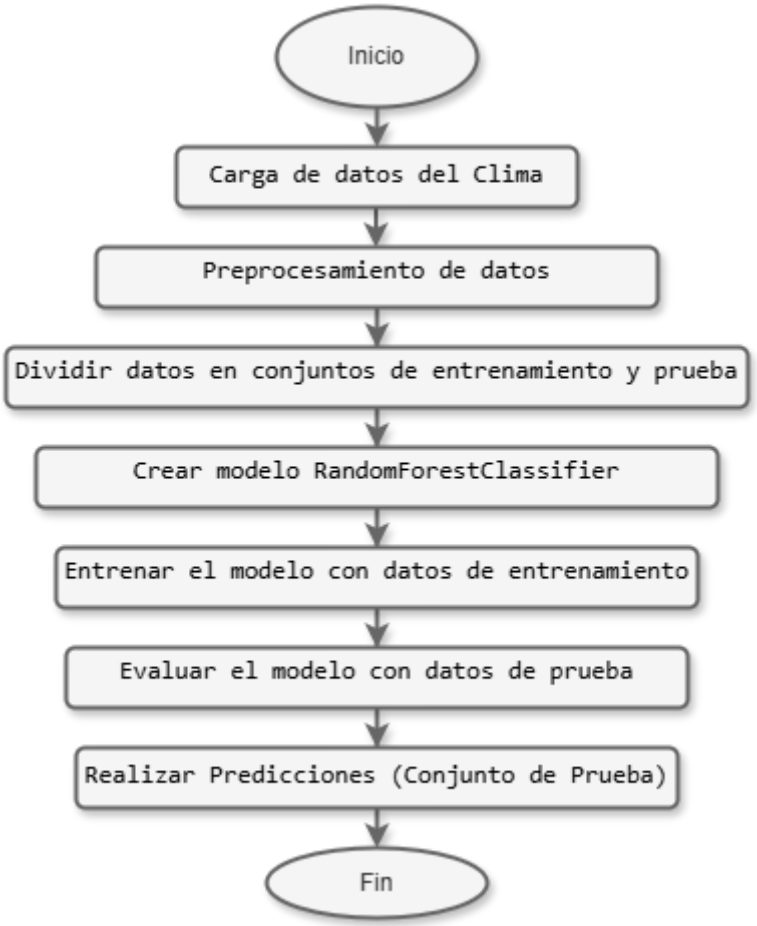
- El bucle for itera sobre cada característica en features.
- La función train_fit_predit se encarga de probar el modelo LSTM.
- La función accuracy_graf genera un gráfico de precisión y pérdida.
- La función show_results obtiene los resultados de las predicciones.
- joblib.dump se utiliza para guardar el modelo entrenado en un archivo.

Clasificador

Para abordar el problema de clasificación del clima, se implementó un modelo Random Forest para Clasificación, esta elección se fundamenta en la capacidad por su robustez frente al ruido y valores atípicos por lo que es común en datos meteorológicos, de acuerdo con el análisis previo.

Figura 20

Modelo Random Forest de Clasificación



Nota. Diagrama de flujo, del código para el análisis de Clasificación

- **Inicio:** El proceso comienza con la carga de los datos del clima desde un archivo o una fuente de datos.
- **Preprocesamiento de datos:** Se realizan tareas de preprocesamiento, como imputar valores faltantes, escalar/normalizar datos y codificar variables categóricas.
- **Dividir datos:** Los datos se dividen en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo.
- **Crear modelo:** Se crea una instancia del modelo RandomForestClassifier.

- **Entrenar modelo:** El modelo se entrena utilizando los datos de entrenamiento.
- **Evaluar modelo:** Se evalúa el rendimiento del modelo utilizando los datos de prueba, calculando métricas como la precisión y generando un reporte de clasificación.
- **Generar predicciones:** Se utiliza el modelo entrenado para generar predicciones con nuevos datos.

Conjunto de entrenamiento y prueba

En la figura # , se realiza una división de los datos en conjuntos de entrenamiento y prueba, utilizando la función train_test_split. X y Y de forma tradicional

Figura 21

División del dataset

```
X = df.drop('weather_main', axis=1)
y = df['weather_main']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=25)
```

Nota. División del dataset de la variable objetivo

Hiperparámetros del modelo de Clasificación

A continuación, se presentas los hiperparámetros del modelo Random Forest para Clasificación y la configuración del entrenamiento.

Tabla 19

Parámetros para el modelo Random Forest

Hiperparámetros	Valores
Número de árboles de decisión. (estimators)	100
Random State	25

Nota. Valores y parámetros para el modelo Random Forest de clasificación.

Entrenamiento del modelo

Figura 22

Código de entrenamiento

```
model = RandomForestClassifier(n_estimators=100, random_state=25)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=25)
```

Nota. Entrenamiento del modelo de clasificación

La línea model.fit(X_train, y_train) es donde se realiza el entrenamiento del modelo RandomForestClassifier. En este punto, el modelo "aprende" los patrones y relaciones entre las características (X_train) y las etiquetas (y_train) para poder realizar predicciones en el futuro.

Prueba

Una vez que el modelo ha sido entrenado, se puede utilizar para realizar predicciones en nuevos datos utilizando el método predict(). Donde X_test son los datos de prueba y y_pred son las predicciones del modelo para esos datos.

Figura 23

Prueba del clasificador

```
y_pred = model.predict(X_test)
```

Nota. Código para prueba de la data de clasificación.

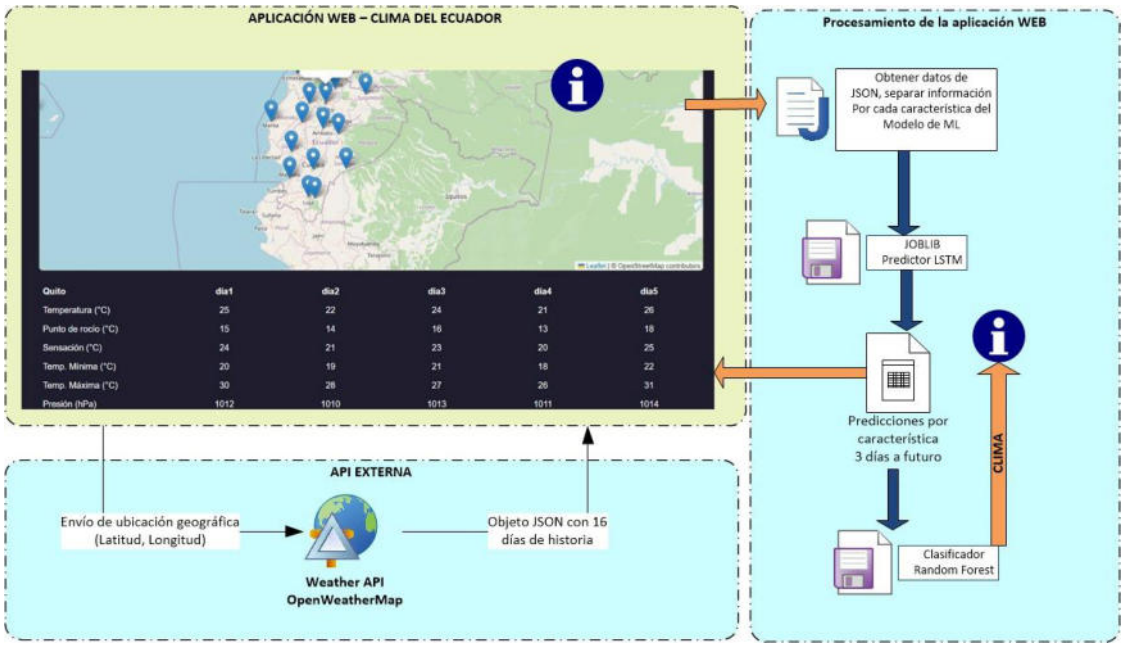
CAPITULO III

Esquema general del proyecto

En la figura 24 se detalla el esquema conceptual, primero se selecciona la ubicación deseada mediante una API, obteniendo las coordenadas necesarias para obtención de data histórica. Luego utilizando un modelo entrenado y almacenado con Joblib, se realiza la predicción para los próximos 3 días de cada característica climática individual, por ejemplo, temperatura, humedad y precipitación. Los resultados de las características climáticas son procesados mediante un modelo de clasificación que determina el estado climático general (soleado, lluvioso, nublado, etc.) para cada día. Este enfoque combina predicción de valores continuos y clasificación para ofrecer un panorama completo y preciso del clima en cualquier ubicación seleccionada.

Figura 24

Esquema conceptual del proyecto



Nota. Proceso de funcionamiento del proyecto

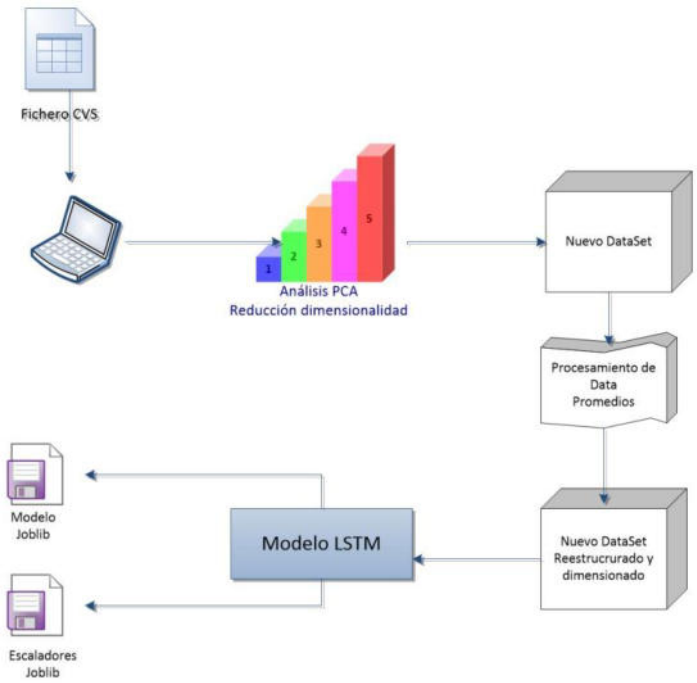
Resultados Predictor

Esquema conceptual

En la figura 25 se detalla el esquema conceptual del modelo para la predicción de valores de tres días a futuro, se realiza un preprocesamiento de datos usando promedio de la data con una reducción dimensional con PCA (22 características → 7 principales), para crear un nuevo dataset que permite la generación de ventanas de tiempo de 16 días para predecir los próximos 3 días. Se usa modelo LSTM considerando la importancia de secuencialidad para la definición de conjunto de datos para el entrenamiento y pruebas, finalmente se guardar el modelo en formato joblib junto con los escaladores por cada característica.

Figura 25

Esquema conceptual del modelo para predecir usando series temporales



Nota. Proceso del modelo predictor

Diagrama de funcionalidad

En la figura 26 se muestra el análisis funcional detalla el flujo de trabajo diseñado para el desarrollo de un modelo de predicción climática enfocado en 16 ciudades del Ecuador. Este modelo emplea una combinación de técnicas avanzadas de preprocesamiento de datos, reducción de dimensionalidad y redes neuronales recurrentes para lograr predicciones precisas y escalables. El sistema parte de un extenso dataset histórico, con más de 4 millones de registros y 22 características relacionadas con el clima, recopilados desde 1979 hasta la fecha actual. A través de un enfoque estructurado, los datos son refinados, analizados y transformados para adaptarse al modelo de predicción. El análisis principal incluye:

Preprocesamiento de datos: Los datos se dividen por franjas horarias (mañana, tarde y noche) para capturar patrones específicos del comportamiento climático a lo largo del día. Esto incluye cálculos de promedios y la eliminación de datos irrelevantes.

Reducción de dimensionalidad: Usando el Análisis de Componentes Principales (PCA), se seleccionan las 7 características más relevantes, permitiendo simplificar el modelo sin perder información significativa.

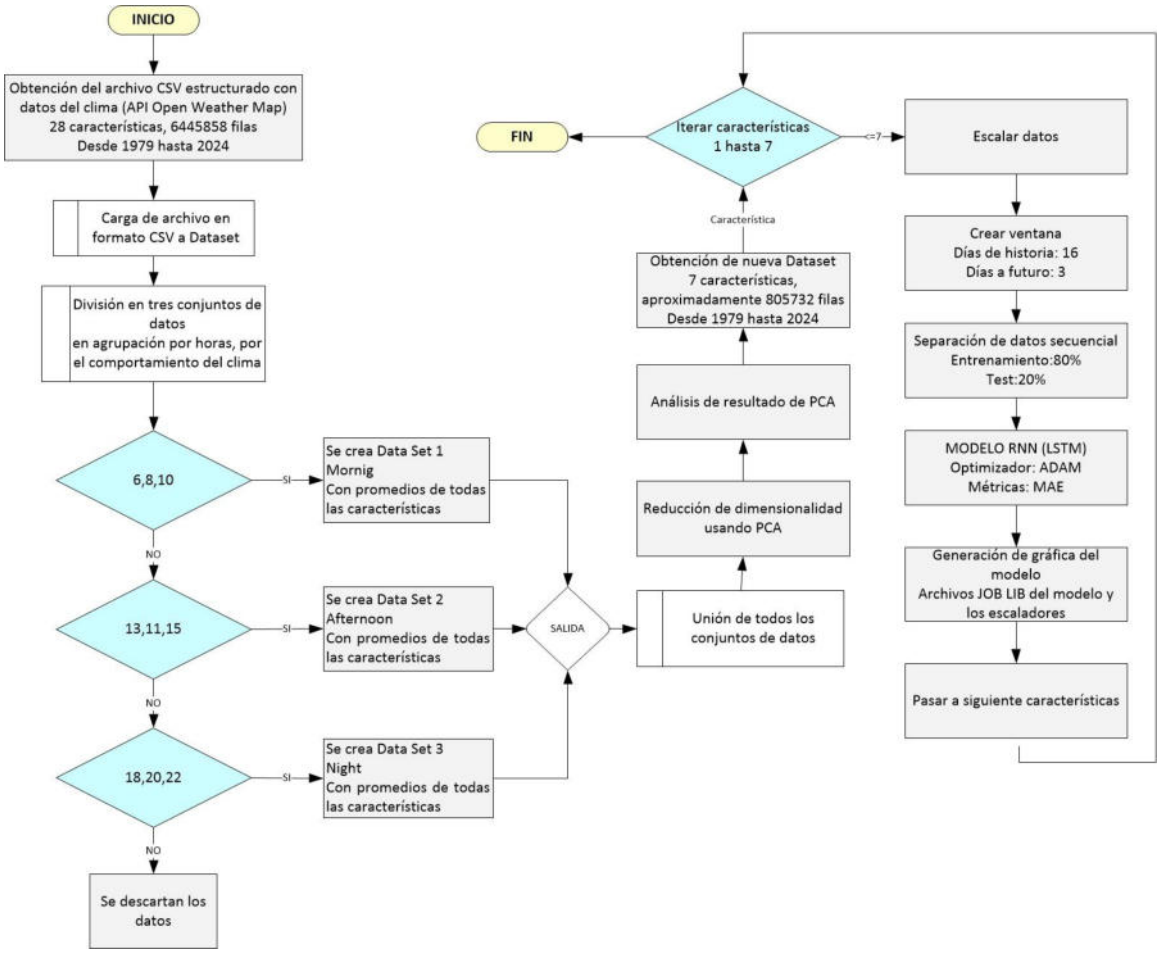
Creación de ventanas de tiempo: Se organizan los datos en ventanas de 16 días para predecir las condiciones climáticas de los 3 días siguientes, un enfoque que combina información histórica reciente para obtener predicciones a corto plazo.

Modelo LSTM (Long Short-Term Memory): Se utiliza una red neuronal recurrente para aprender de la secuencia de datos climáticos, con un optimizador ADAM y métricas de error MAE para evaluar el rendimiento del modelo.

Finalmente, el modelo entrenado, junto con los escaladores necesarios para la normalización de datos, se exporta en formato JOBLIB para facilitar su implementación en entornos web

Figura 26

Esquema conceptual del modelo para predecir usando series temporales



Nota. Proceso del predictor

Métricas del entrenamiento

En la tabla 20 se muestran las características y las métricas de entrenamiento del algoritmo de predicción.

Tabla 20

Métricas de entrenamiento

Característica	Tiempo	Loss	Mae	Val loss	Val mae
Promedio de temperatura diaria	Mañana	0,18	0,11	0,02	0,12
	Tarde	0,17	0,11	0,02	0,11
	Noche	0,02	0,12	0,03	0,13

Característica	Tiempo	Loss	Mae	Val loss	Val mae
Promedio de presión diaria	Mañana	0,02	0,11	0,02	0,12
	Tarde	0,02	0,11	0,02	0,12
	Noche	0,02	0,11	0,03	0,13
Promedio de Humedad diario	Mañana	0,01	0,08	0,02	0,11
	Tarde	0,01	0,08	0,02	0,08
	Noche	0,02	0,11	0,04	0,14
Velocidad promedio del viento diario	Mañana	0,01	0,08	0,02	0,11
	Tarde	0,01	0,08	0,02	0,08
	Noche	0,02	0,11	0,04	0,14
Grados del viento promedio diario	Mañana	0,01	0,07	0,01	0,07
	Tarde	0,016	0,08	0,01	0,07
	Noche	0,07	0,22	0,08	0,21
Lluvia de una hora promedio diario	Mañana	0,01	0,06	0,01	0,06
	Tarde	0,006	0,04	0,004	0,04
	Noche	0,01	0,08	0,01	0,08
Nubes promedio diario	Mañana	0,03	0,13	0,05	0,16
	Tarde	0,05	0,19	0,06	0,19
	Noche	0,04	0,14	0,05	0,17

Nota. Resultados de las métricas del entrenamiento para cada variable.

Interpretación General

Valores bajos: Indican un buen rendimiento del modelo para predecir esa variable en ese momento del día. Generalmente, valores por debajo de 0.1 se consideran buenos, y valores cercanos a 0 son excelentes.

Valores altos: Indican un rendimiento menos preciso del modelo para esa variable en ese momento del día. Valores por encima de 0.2 sugieren que el modelo podría tener dificultades para predecir esa variable.

A continuación, se detalla un análisis por variable:

Temperatura, Presión, Humedad, Velocidad del Viento: El modelo muestra un buen rendimiento en general para estas variables, con valores de pérdida y error relativamente bajos en la mayoría de los casos.

Dirección del Viento (Grados): El modelo tiene un rendimiento aceptable en la mañana y la tarde, pero su precisión disminuye en la noche, especialmente para la dirección del viento, donde los valores de pérdida y error son más altos.

Lluvia: El modelo muestra un excelente rendimiento para predecir la lluvia, con valores de pérdida y error muy bajos en todos los momentos del día.

Nubes: El modelo tiene un rendimiento moderado para predecir la nubosidad, con valores de pérdida y error intermedios.

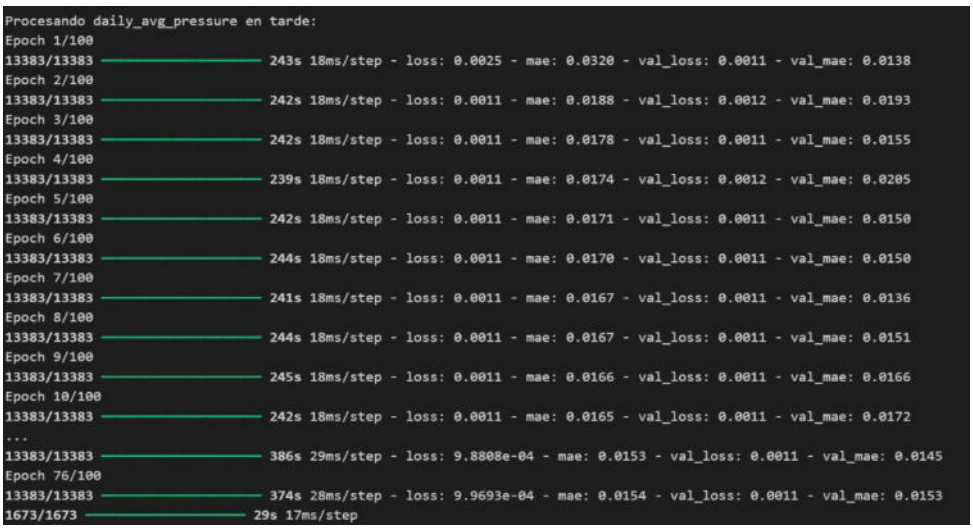
Interpretación especifica

Promedio de presión diaria

Las métricas (tanto 'loss' como 'mae) muestran poca variabilidad a lo largo de las épocas, especialmente después de las primeras 10-15 épocas. Esto sugiere que el modelo converge de manera estable y no presenta problemas significativos en las predicciones.

Figura 27

Métricas presión diaria

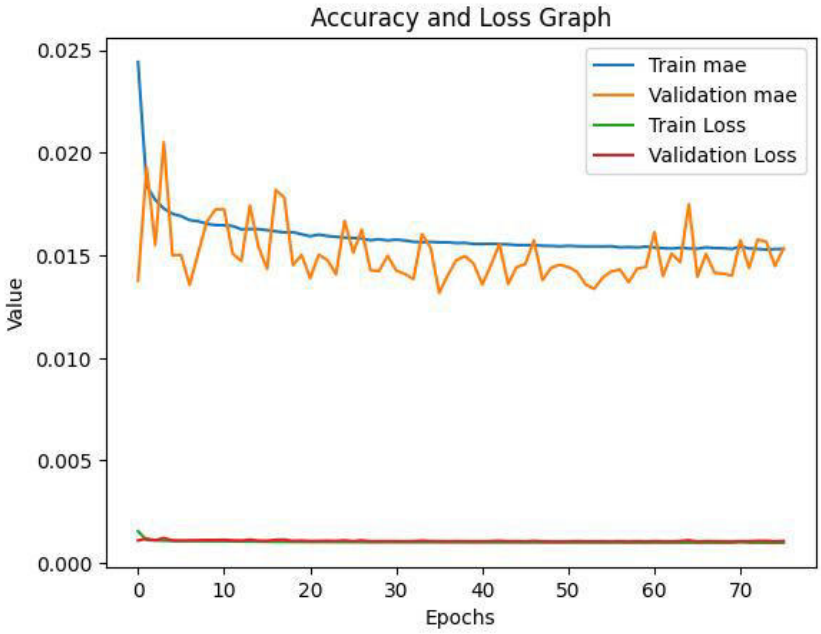


Nota. Evolución de la Pérdida y el Error Absoluto durante el Entrenamiento y Validación

El MAE en la etapa de entrenamiento disminuye rápidamente en las primeras épocas, estabilizándose alrededor de 0.015 después de la época 10. Luego en validación converge a valores cercanos al MAE, esto es un indicador de que el modelo aprende correctamente. Las variaciones pueden deberes a la variabilidad propia de los datos climáticos.

Figura 28

Precisión de presión diaria



Nota. Precisión y Pérdida del Modelo en Función del Tiempo de Entrenamiento

Promedio de Humedad diaria

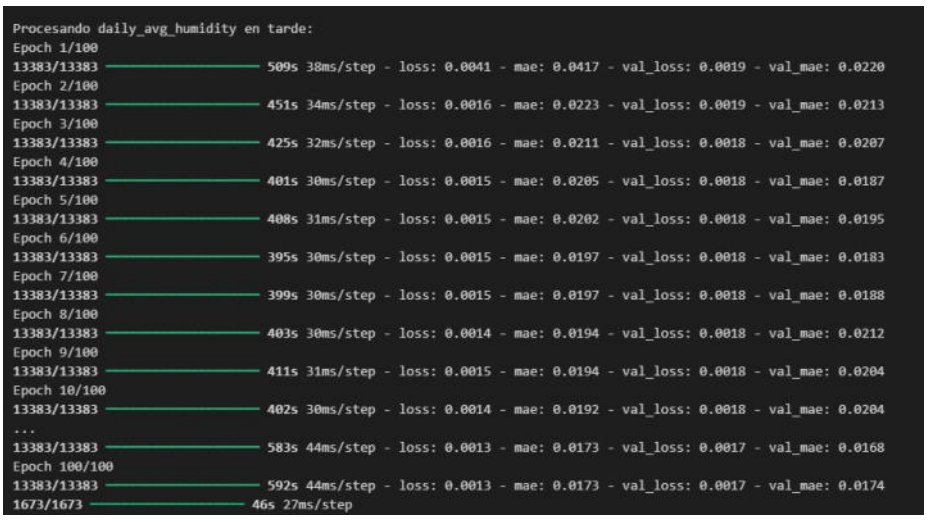
La pérdida y el MAE disminuyen de forma constante en las primeras épocas, lo que indica que el modelo está aprendiendo.

A partir de cierto punto (entre las épocas 10 y 20), las métricas se estabilizan tanto en entrenamiento como en validación, lo que indica que el modelo ha alcanzado un punto cercano al mínimo de la función de pérdida.

La diferencia entre ‘loss’ y ‘val_loss’, así como entre ‘mae’ y ‘val_mae’, es mínima, lo que sugiere que no hay sobreajuste significativo. Esto es un buen indicador de que el modelo está generalizando correctamente.

Figura 29

Métricas de humedad diaria



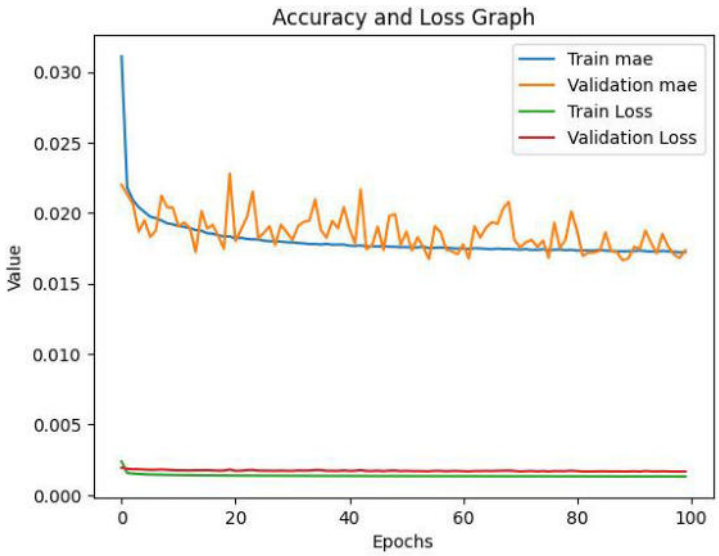
Nota. Evolución de la Pérdida y el Error Absoluto durante el Entrenamiento y Validación

Ambas métricas disminuyen rápidamente en las primeras épocas, lo que indica que el modelo está aprendiendo a ajustarse a los datos.

A partir de las primeras 20 épocas, las pérdidas se estabilizan y mantienen valores bajos, lo que muestra que el modelo ha convergido.

Figura 30

Precisión de humedad diaria



Nota. Precisión y Pérdida del Modelo en Función del Tiempo de Entrenamiento

Promedio de temperatura diaria

La rápida disminución (pasa de 0.0029 a 0.0010) en loss muestra que el modelo está aprendiendo eficientemente.

La estabilidad (se mantiene cerca de 0.0009) de val_loss en valores bajos indica que el modelo generaliza bien y no está sobreajustando.

Figura 31

Métricas temperatura diaria

```
TARGET DAILY_AVG_TEMP:

Procesando daily_avg_temp en tarde:
Epoch 1/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 348s 26ms/step - loss: 0.0029 - mae: 0.0341 - val_loss: 9.5012e-04 - val_mae: 0.0138
Epoch 2/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 369s 28ms/step - loss: 0.0012 - mae: 0.0198 - val_loss: 0.0010 - val_mae: 0.0186
Epoch 3/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 358s 27ms/step - loss: 0.0012 - mae: 0.0186 - val_loss: 9.0854e-04 - val_mae: 0.0149
Epoch 4/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 355s 27ms/step - loss: 0.0012 - mae: 0.0183 - val_loss: 9.1344e-04 - val_mae: 0.0157
Epoch 5/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 336s 25ms/step - loss: 0.0011 - mae: 0.0179 - val_loss: 8.9395e-04 - val_mae: 0.0146
Epoch 6/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 292s 22ms/step - loss: 0.0011 - mae: 0.0176 - val_loss: 9.4765e-04 - val_mae: 0.0163
Epoch 7/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 279s 21ms/step - loss: 0.0011 - mae: 0.0175 - val_loss: 9.0678e-04 - val_mae: 0.0148
Epoch 8/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 288s 21ms/step - loss: 0.0011 - mae: 0.0173 - val_loss: 9.8122e-04 - val_mae: 0.0176
Epoch 9/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 243s 18ms/step - loss: 0.0011 - mae: 0.0172 - val_loss: 9.0443e-04 - val_mae: 0.0139
Epoch 10/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 246s 18ms/step - loss: 0.0011 - mae: 0.0172 - val_loss: 9.0719e-04 - val_mae: 0.0147
Epoch 11/100
...
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 258s 19ms/step - loss: 0.0010 - mae: 0.0158 - val_loss: 8.7706e-04 - val_mae: 0.0134
Epoch 49/100
13383/13383 ━━━━━━━━━━━━━━━━━━━━ 257s 19ms/step - loss: 0.0010 - mae: 0.0156 - val_loss: 8.7244e-04 - val_mae: 0.0117
1673/1673 ━━━━━━━━━━━━━━━━━━━━ 15s 9ms/step
```

Nota. Evolución de la Pérdida y el Error Absoluto durante el Entrenamiento y Validación

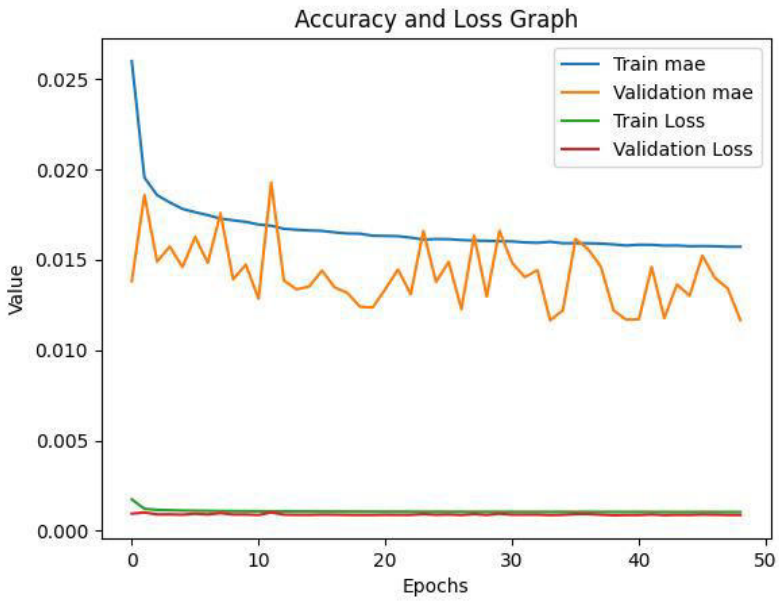
La diferencia entre Train MAE y Validation MAE es pequeña, lo que indica que el modelo está generalizando correctamente.

Las oscilaciones en Validation MAE son normales y podrían deberse a la variabilidad en los datos de validación propia de los datos climaticos.

Los valores de Loss son bajos y similares entre entrenamiento y validación, lo que demuestra que el modelo no está sobreajustando.

Figura 32

Precisión de temperatura diaria



Nota. Precisión y Pérdida del Modelo en Función del Tiempo de Entrenamiento

Velocidad del Viento promedio diario

Los valores de pérdida son bajos y similares entre entrenamiento y validación, lo que indica un buen ajuste.

El MAE de validación es consistentemente más bajo que el de entrenamiento, lo que podría significar que el modelo puede estar generalizando ligeramente mejor en los datos de validación.

Figura 33

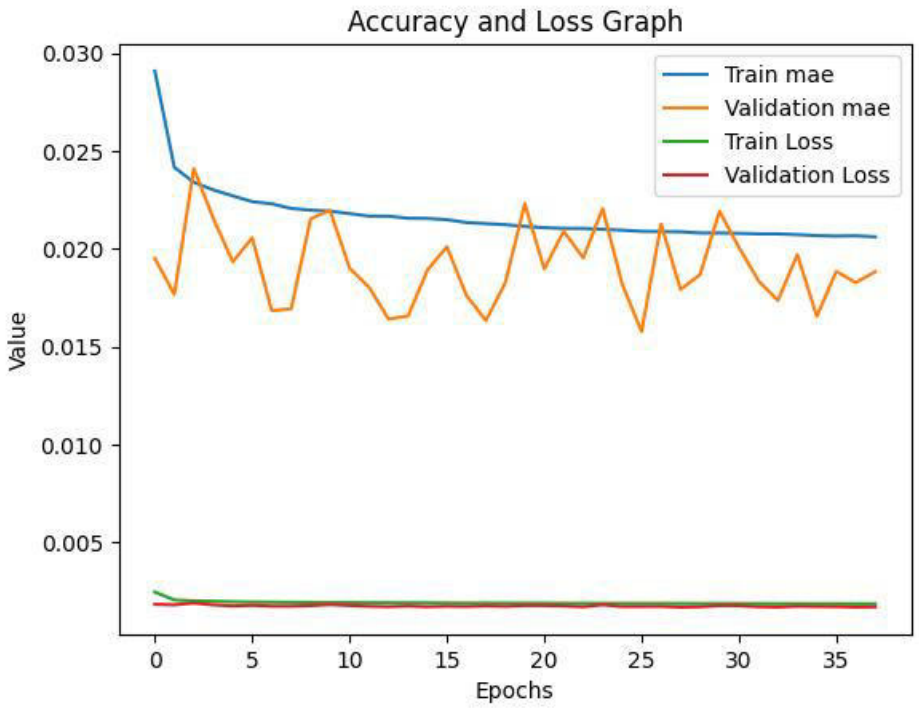
Métricas Velocidad del viento promedio diaria

Nota. Evolución de la Pérdida y el Error Absoluto durante el Entrenamiento y Validación

La Validation MAE tiene mayor variabilidad al inicio, comienza en 0.030 hasta llegar a 0.020, lo que es normal cuando el modelo ajusta sus pesos. Se observa una clara convergencia del Train MAE y el Validation MAE, de igual forma el los en Train y Validation lo que indica un buen ajuste.

Figura 34

Precisión de viento promedio diario



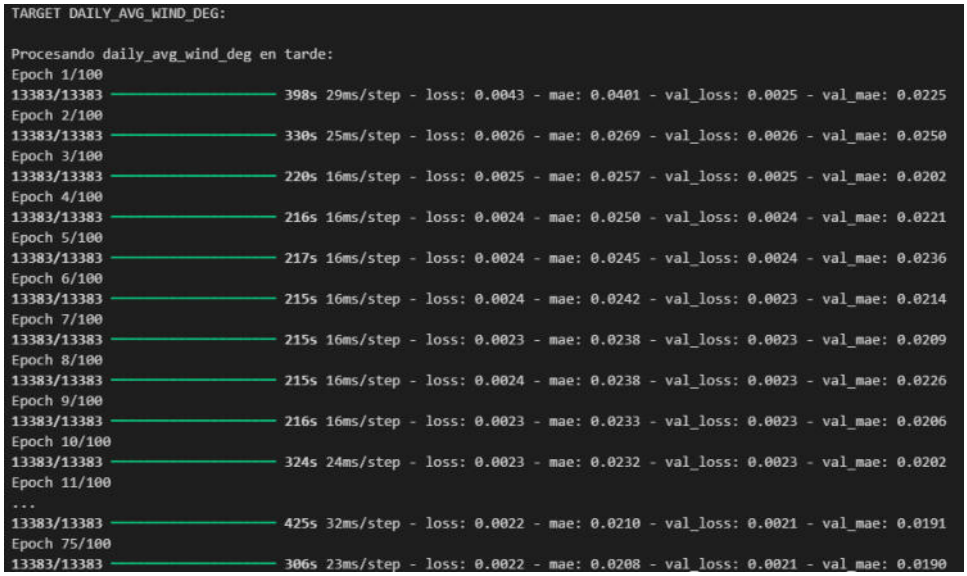
Nota. Precisión y Pérdida del Modelo en Función del Tiempo de Entrenamiento

Grados del viento promedio diario

La pérdida se reduce y estabiliza en la época 10 (cerca de 0.0022) lo que indica un correcto aprendizaje. Algo similar ocurre con el MAE, que al comparar Train contra Validation se observa una variación muy pequeña que indica buena generalización.

Figura 35

Métricas de grado del viento promedio

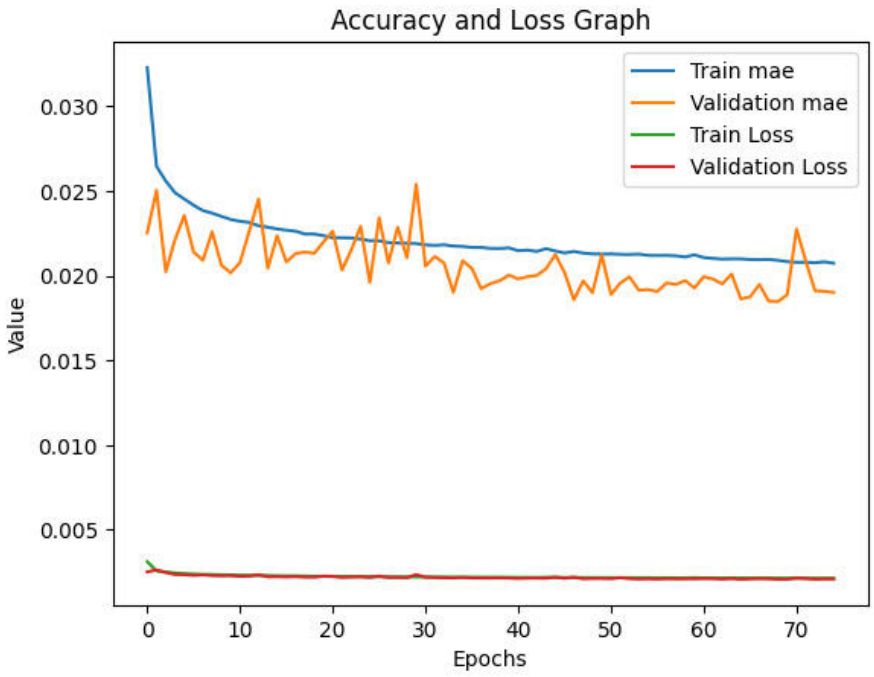


Nota. Evolución de la Pérdida y el Error Absoluto durante el Entrenamiento y Validación

La validación tiene mayor variabilidad que el entrenamiento, lo cual es de esperarse debido a que los datos climáticos son más diversos o menos regulares en validación. En cuanto a 'loss' se observa una excelente convergencia al igual que en las otras variables.

Figura 36

Precisión de grado del viento promedio diario



Nota. Precisión y Pérdida del Modelo en Función del Tiempo de Entrenamiento

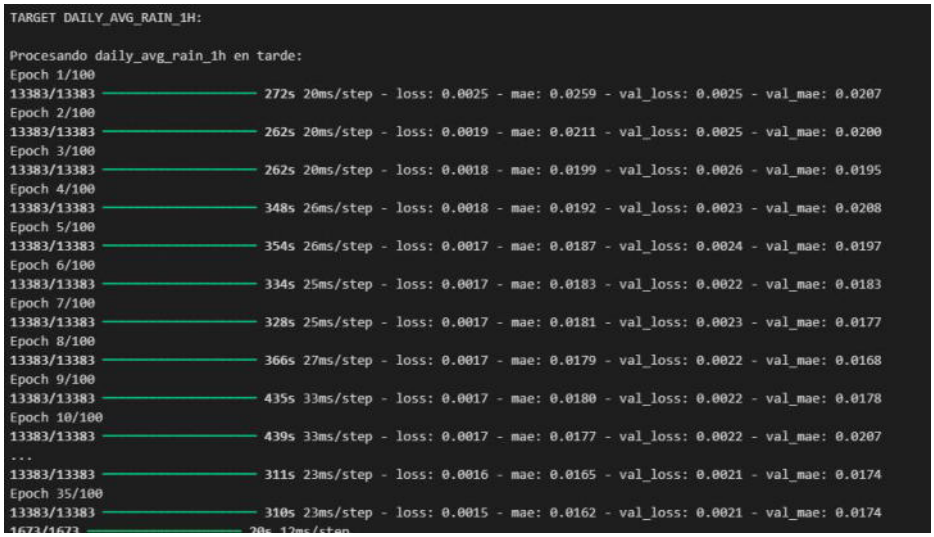
Lluvia de una hora promedio diario

Al comparar la perdida en entrenamiento y validación se observa que se mantiene bastante cercana, con una diferencia de apenas 0.0006, lo que da muestra de un modelo bien ajustado.

El MAE de igual forma decrece en entrenamiento y validación hasta tener una diferencia de 0.0012

Figura 37

Métricas lluvia de una hora promedio



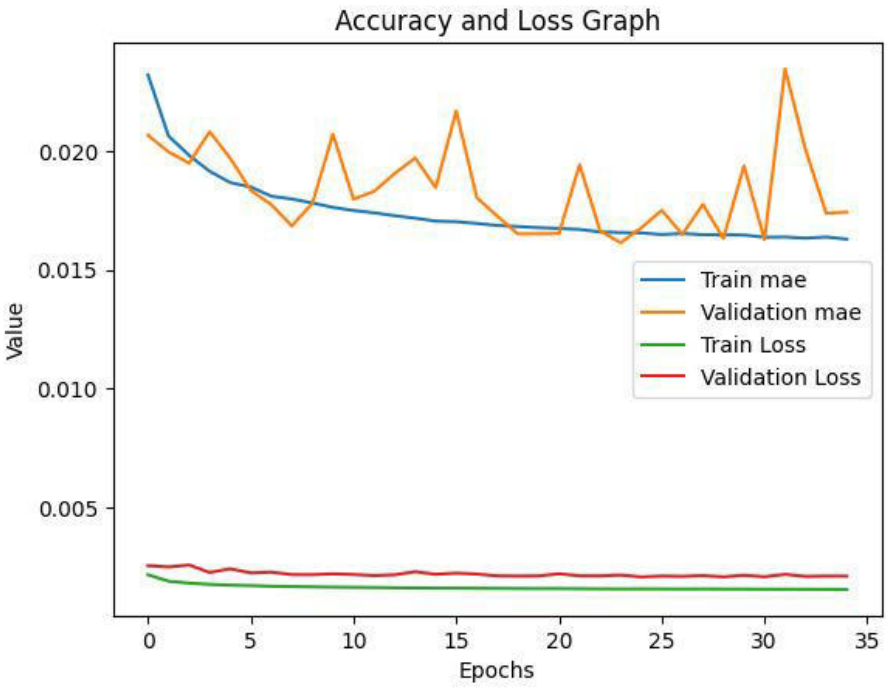
Nota. Evolución de la Pérdida y el Error Absoluto durante el Entrenamiento y Validación

Aunque hay una variación más clara entre entrenamiento y validación en el MAE en comparación a las demás variables, ambas terminan convergiendo en valores similares y no muy distantes, cabe recalcar que la lluvia es una de las características más variantes en cuanto clima, lo que podría justificar este comportamiento.

Similarmente se observa cierta discrepancia en la pérdida, aunque no es muy significativa, igualmente se presume que debido a la variabilidad de los datos de validación.

Figura 38

Precisión de lluvia 1h promedio diaria



Nota. Precisión y Pérdida del Modelo en Función del Tiempo de Entrenamiento

Predicción

Los resultados de las predicciones se presentan de manera organizada, desglosados por variable meteorológica, período de tiempo (mañana, tarde, noche). La figura ### muestra los valores futuros predichos por el modelo. Posteriormente, el modelo se guarda en formato joblib para facilitar su integración en una interfaz web interactiva que permitirá visualizar y explorar los resultados de manera más dinámica.

Figura 39

Resultados de modelo predictor

Variable	Temp			Pressure			Humidity			Wind_Speed		
Periodo	Mañana	Tarde	Noche	Mañana	Tarde	Noche	Mañana	Tarde	Noche	Mañana	Tarde	Noche
0	[15.775396, 15.721596, 15.642554]	[17.80726, 17.944807, 17.869024]	[20.934267, 20.777472, 20.669502]	[1017.2639, 1017.14484, 1017.09985]	[1017.333, 1017.53986, 1017.5451]	[1014.04736, 1014.0664, 1013.9147]	[95.37705, 95.00179, 95.43323]	[86.37904, 85.59914, 84.97289]	[70.194595, 70.707596, 71.350914]	[0.8208951, 0.841504, 0.8476216]	[1.0667198, 1.0065272, 1.0133864]	[1.2916691, 1.3452287, 1.444568]
1	[16.037304, 15.927567, 15.809573]	[17.97162, 18.079601, 17.983128]	[20.859234, 20.658997, 20.465996]	[1016.8303, 1016.8018, 1016.8349]	[1017.2113, 1017.4441, 1017.50806]	[1013.757, 1013.82434, 1013.751]	[94.72019, 94.53165, 95.05744]	[85.71139, 85.04434, 84.537186]	[71.02149, 71.7494, 72.615]	[0.94291115, 0.9160417, 0.902923]	[1.1658028, 1.0717278, 1.0533483]	[1.2184664, 1.2746639, 1.3608726]
2	[16.096586, 15.980115, 15.852659]	[18.064157, 18.162888, 18.054682]	[20.568865, 20.362732, 20.13592]	[1016.79987, 1016.7637, 1016.83405]	[1017.2391, 1017.47974, 1017.58185]	[1013.9561, 1013.9964, 1013.96783]	[94.510475, 94.402016, 94.94158]	[85.42521, 84.842224, 84.462524]	[72.99171, 73.47691, 74.39707]	[0.93511033, 0.9143387, 0.9010166]	[1.1518822, 1.0546399, 1.0224887]	[1.1356915, 1.1897275, 1.2548355]
3	[16.192406, 16.058956, 15.916393]	[18.058327, 18.158936, 18.047983]	[20.72428, 20.447323, 20.197388]	[1016.75446, 1016.75604, 1016.89154]	[1017.3979, 1017.62634, 1017.7522]	[1013.7741, 1013.86053, 1013.87146]	[93.91181, 93.93429, 94.56299]	[84.46977, 84.19151, 84.03349]	[71.9291, 73.00876, 74.11403]	[1.1370775, 1.065923, 1.0398991]	[1.4017168, 1.2163262, 1.1640294]	[1.1371043, 1.1721389, 1.2180674]
4	[16.110403, 15.990497, 15.854459]	[18.220978, 18.304522, 18.175684]	[20.629776, 20.358479, 20.165106]	[1016.8872, 1016.8744, 1017.0261]	[1017.3403, 1017.6067, 1017.7457]	[1013.62756, 1013.7432, 1013.7754]	[94.036125, 94.09402, 94.747925]	[84.154945, 84.07889, 84.0624]	[73.361664, 74.22319, 74.84883]	[1.0804931, 1.0420176, 1.0258234]	[1.3443433, 1.17958, 1.1304697]	[1.123336, 1.1500558, 1.1927954]
...	...	...	...	...	...	...	...	...	...	...	...	...
342	[15.105197, 15.142978, 15.166121]	[17.509438, 17.688713, 17.620848]	[19.346292, 19.290442, 19.296535]	[1017.19696, 1017.09674, 1016.98346]	[1017.5767, 1017.76245, 1017.79877]	[1015.0491, 1015.1015, 1014.9622]	[96.0231, 95.549065, 96.00305]	[86.10324, 85.10908, 84.69053]	[76.78283, 76.772865, 77.347145]	[0.5779824, 0.6648055, 0.7170442]	[0.7903912, 0.8031741, 0.8411587]	[1.7561516, 1.606476, 1.6165761]
343	[15.508284, 15.504722, 15.498284]	[17.897482, 18.021584, 17.914423]	[19.221497, 19.131372, 19.118517]	[1016.8481, 1016.7539, 1016.65454]	[1017.2964, 1017.53296, 1017.57385]	[1014.7466, 1014.8525, 1014.69824]	[93.94328, 93.785576, 94.34121]	[82.912254, 82.45919, 82.06871]	[76.58748, 76.67519, 77.14721]	[0.90819657, 0.8861791, 0.8892342]	[1.0386151, 0.96488273, 1.0041064]	[1.7625133, 1.640663, 1.7012883]
344	[15.63117, 15.62112, 15.6028595]	[18.32319, 18.39346, 18.243408]	[19.748394, 19.564255, 19.500881]	[1016.66974, 1016.52795, 1016.4313]	[1017.08997, 1017.37274, 1017.43823]	[1014.24915, 1014.3983, 1014.2481]	[93.08727, 93.08198, 93.69072]	[79.895065, 79.894936, 79.65641]	[72.76194, 73.72494, 74.41879]	[0.9164863, 0.89114577, 0.87537086]	[1.3268635, 1.1981168, 1.2220235]	[1.5710006, 1.5474793, 1.7072964]
345	[15.359821, 15.43929, 15.450923]	[18.273466, 18.37909, 18.255266]	[20.28199, 20.04604, 19.947985]	[1017.1955, 1016.8708, 1016.78516]	[1017.4879, 1017.6563, 1017.701]	[1013.8919, 1014.0487, 1013.89233]	[89.702774, 90.15749, 90.9883]	[74.95605, 75.639015, 75.810745]	[70.90081, 72.106476, 72.66484]	[1.512526, 1.3245852, 1.2543824]	[1.7066984, 1.5384793, 1.5394837]	[1.4646171, 1.5320967, 1.7530274]
346	[15.141248, 15.272745, 15.3270645]	[18.211037, 18.35929, 18.258997]	[19.985188, 19.810154, 19.645763]	[1017.381, 1016.95984, 1016.80273]	[1017.6241, 1017.74066, 1017.7543]	[1014.2604, 1014.3077, 1014.13165]	[89.84354, 90.31518, 91.209015]	[73.180374, 74.12306, 74.71367]	[72.18555, 72.608765, 73.18868]	[1.0889393, 1.0537703, 1.0263975]	[1.4921507, 1.4153737, 1.4041646]	[1.7800547, 1.8105483, 1.9373199]

Wind_Deg			Rain			Clouds		
Mañana	Tarde	Noche	Mañana	Tarde	Noche	Mañana	Tarde	Noche
[152.26282, 154.04356, 153.16238]	[134.20575, 125.63692, 133.00966]	[184.70934, 190.15002, 177.35187]	[0.20512548, 0.17768894, 0.17211197]	[0.13425867, 0.1262939, 0.11814485]	[1.0345843, 1.0401722, 1.0561166]	[98.68058, 95.71601, 96.18958]	[87.96519, 86.99519, 88.78519]	[87.07447, 85.38306, 86.64735]
[142.61223, 148.22803, 149.51561]	[127.758606, 122.07165, 131.6252]	[169.5915, 179.55202, 174.58745]	[0.18475778, 0.16803943, 0.16889788]	[0.12500384, 0.121507965, 0.121935695]	[1.0857737, 1.0768672, 1.0983597]	[98.36501, 95.61279, 96.00244]	[87.21192, 86.215225, 88.130974]	[87.34671, 85.80344, 87.12904]
[141.67593, 147.48965, 148.7293]	[127.37695, 122.60016, 132.49261]	[177.6595, 190.22401, 192.05061]	[0.17580566, 0.1681571, 0.17800562]	[0.13356024, 0.13545682, 0.14241828]	[1.169385, 1.1082957, 1.1216098]	[97.301315, 94.870735, 95.2372]	[86.88497, 85.818115, 87.86608]	[86.95346, 85.31521, 87.05399]
[133.44023, 140.31004, 142.40082]	[124.50027, 120.08228, 129.6896]	[166.88916, 180.33531, 181.56502]	[0.16176733, 0.16267753, 0.17805567]	[0.1318823, 0.14398786, 0.1469549]	[1.0650704, 1.0317214, 1.0512216]	[95.838165, 93.59632, 93.87189]	[82.9102, 82.32387, 84.90572]	[86.492096, 85.55031, 87.44191]
[134.9057, 141.26071, 142.87962]	[121.86023, 118.38436, 128.55313]	[169.13043, 181.94594, 182.36015]	[0.17012519, 0.17341174, 0.19113521]	[0.13425505, 0.15194294, 0.14555722]	[1.0020599, 0.9673886, 0.9814913]	[95.41649, 93.161026, 93.455475]	[84.74945, 83.9731, 86.49566]	[89.072044, 87.89029, 88.768585]
...	...	...	...	...	...	...	...	...
[183.74307, 176.79361, 169.46916]	[179.21678, 159.13327, 154.32031]	[272.9683, 282.9951, 274.0834]	[0.20520827, 0.14946464, 0.11148652]	[0.21231124, 0.19310707, 0.14928819]	[0.9360249, 0.91724485, 0.91532]	[100.31049, 96.7206, 96.88207]	[91.736015, 89.65335, 91.691605]	[96.8287, 95.14156, 94.60181]
[154.08557, 154.28572, 153.13177]	[139.928, 128.87833, 133.15121]	[276.56177, 288.05704, 278.9812]	[0.15024999, 0.11360103, 0.09253877]	[0.1560887, 0.13822266, 0.07464139]	[0.904824, 0.905998, 0.9015819]	[96.34075, 93.88573, 94.2072]	[88.618675, 87.20994, 88.73307]	[99.19253, 96.994064, 95.43535]
[146.9028, 150.20549, 151.11281]	[119.76903, 114.12809, 124.419136]	[239.17276, 249.94135, 234.84914]	[0.12800336, 0.10900949, 0.1083428]	[0.0758903, 0.059295375, 0.00677729]	[0.8043065, 0.8596305, 0.87216854]	[94.99732, 92.80324, 93.20963]	[84.49011, 83.74074, 85.132126]	[96.07243, 93.99505, 92.40642]
[128.46358, 132.5989, 134.60635]	[113.451004, 107.92071, 118.14037]	[212.64207, 220.89151, 203.4049]	[0.08486532, 0.08714709, 0.10321682]	[-0.0028916073, -0.006472332, -0.022736663]	[0.79257333, 0.86293644, 0.87876683]	[86.531845, 85.65088, 86.42901]	[74.720505, 74.84457, 77.07608]	[92.835396, 90.22436, 88.26538]
[139.89076, 140.20941, 139.31541]	[135.1856, 122.77282, 128.24486]	[239.35191, 246.38112, 234.52548]	[0.1201588, 0.12005947, 0.13838398]	[-0.0057922592, -0.003870676, 0.023163913]	[0.91773057, 0.9099637, 0.913136]	[86.67232, 85.2986, 86.170975]	[76.139275, 75.71621, 78.40197]	[89.02947, 85.99136, 85.32469]

Nota. Resultados de cada variable de predicción.

Resultados Clasificador

Esquema conceptual

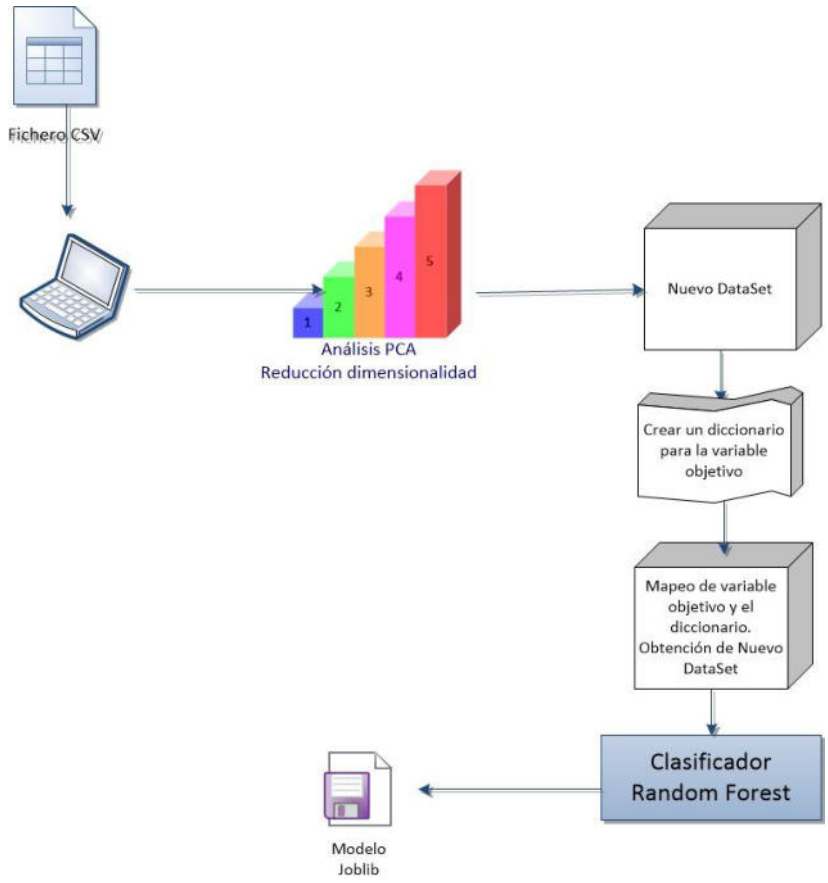
En la figura 40 se detalla el esquema del modelo clasificador para la predicción climática, los pasos describen a continuación:

- Ingreso de datos crudos: Se procesa la información climática histórica con múltiples características recopiladas en Ecuador.
- Reducción de dimensionalidad: Se aplica Análisis de Componentes Principales (PCA) para reducir los datos a 7 características clave, manteniendo la mayor cantidad de variabilidad posible.
- Clasificación: Utilizando un modelo Random Forest, el sistema clasifica los datos en 14 categorías climáticas diferentes (ej., soleado, lluvioso, tormentoso, etc.).
- Diccionario de categorías: Se crea un diccionario que asocia cada variable del modelo con su correspondiente categoría climática, facilitando la interpretación de los resultados.
- Almacenamiento del modelo: El modelo entrenado se guarda utilizando Joblib, permitiendo su reutilización para predicciones futuras.

Este flujo asegura un análisis eficiente y una clasificación precisa del clima, con una fácil interpretación de los resultados gracias al diccionario de categorías.

Figura 40

Esquema conceptual del modelo para clasificar el clima usando Random Forest.



Nota. Proceso de funcionamiento del clasificador

Diagrama de funcionalidad

En la figura 41 se presenta análisis funcional describe el diseño y la implementación de un modelo de inteligencia artificial para la clasificación del clima basado en datos históricos extensos del Ecuador. Utilizando un enfoque estructurado y metodológico, este modelo se apoya en técnicas avanzadas de reducción de dimensionalidad y clasificación para proporcionar predicciones precisas sobre las condiciones climáticas.

- Reducción de dimensionalidad: Se aplica el Análisis de Componentes Principales (PCA) para reducir las características relevantes a 8, lo que permite simplificar el modelo manteniendo la calidad de la información.
- Preprocesamiento de datos: Se manejan valores faltantes y se realiza la transformación de la variable objetivo ("Weather Main") en categorías numéricas

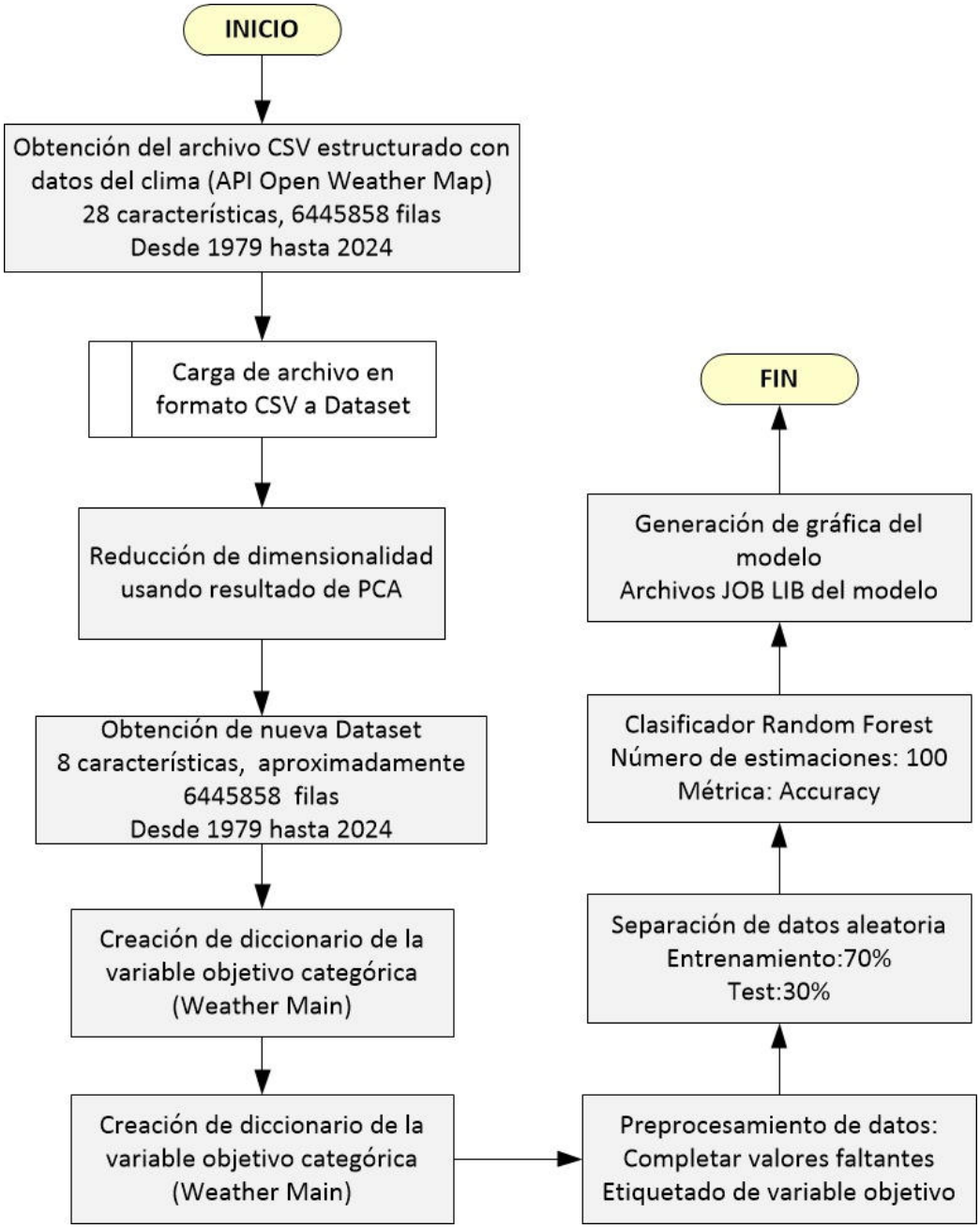
mediante un diccionario. Este paso asegura que los datos sean adecuados para el modelo de aprendizaje automático.

- Separación de datos: El dataset procesado se divide en conjuntos de entrenamiento (70%) y prueba (30%) mediante un enfoque aleatorio, preservando la representatividad de las clases objetivo.
- Clasificación mediante Random Forest: Se utiliza un clasificador Random Forest con 100 estimadores, evaluando su desempeño a través de la métrica de exactitud (Accuracy). Este enfoque robusto permite capturar patrones complejos en los datos y realizar clasificaciones confiables.
- Exportación y visualización: Finalmente, el modelo entrenado y los escaladores son exportados en formato JOBLIB para facilitar su integración en aplicaciones web, y se genera una representación gráfica de la arquitectura del modelo.

Este flujo de trabajo proporciona un marco integral para abordar la clasificación climática, optimizando el procesamiento y el análisis de grandes volúmenes de datos. El modelo es adaptable y escalable, lo que lo convierte en una herramienta efectiva para aplicaciones en la predicción y monitoreo del clima.

Figura 41

Esquema conceptual del modelo para clasificar el clima usando Random Forest.



Nota. Proceso del clasificador

Métricas del entrenamiento

En la tabla 21 se detalla las características del algoritmo de clasificación del clima.

Tabla 21

Métricas de entrenamiento

Clima principal	Descripción	Clase	Precisión	Recall	F1-Score	Soporte
Clouds	Nublado	**0**	0.97	0.99	0.98	1156226
Rain	Lluvia	**1**	0.99	0.99	0.99	691874
Clear	Despejado	**2**	0.97	1.00	0.99	29663
Haze	Calina	**3**	0.58	0.26	0.36	27335
Fog	Niebla	**4**	0.52	0.44	0.48	17141
Drizzle	Llovizna	**5**	0.04	0.02	0.03	4899
Mist	Neblina	**6**	0.29	0.18	0.22	4435
Thunderstorm	Tormenta eléctrica	**7**	0.00	0.00	0.00	1637
Smoke	Humo	**8**	0.01	0.00	0.00	353
Dust	Polvo	**9**	0.03	0.01	0.01	167
Ash	Ceniza	**10**	0.00	0.00	0.00	18
Tornado	Tornado	**11**	0.00	0.00	0.00	3
Squall	Chubasco Violento	**12**	0.00	0.00	0.00	6
Snow	Nieve	**13**	0.00	0.00	0.00	1

Nota. Resultados de las métricas del entrenamiento para cada clima.

Interpretación General

Valores bajos: Indican un buen rendimiento del modelo para predecir esa variable.

El modelo tiene un buen rendimiento general, pero se debe prestar atención a las clases con menor rendimiento y soporte para mejorar la clasificación.

Análisis por Variable

- El modelo tiene una precisión y recall altos para las clases 0, 1 y 2, lo que indica un buen rendimiento en la clasificación de estas clases.
- Las clases 3, 4, 5, 6 tienen un rendimiento más bajo, especialmente en precisión y recall.

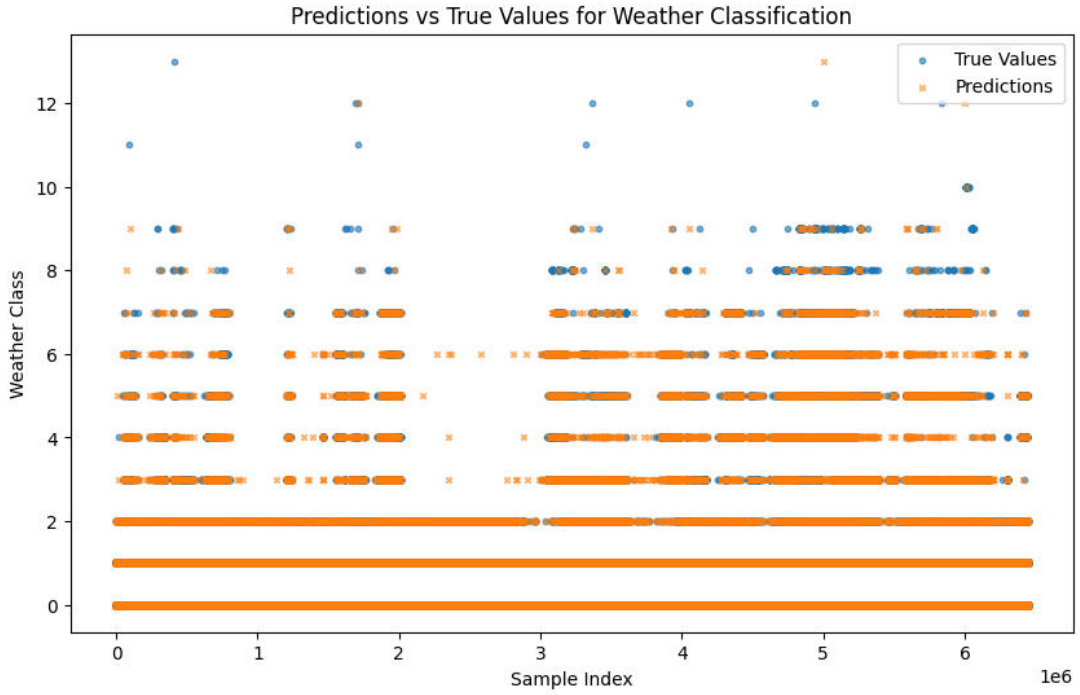
- Las clases 7, 8, 9, 10, 11, 12 y 13 tienen muy pocas muestras, lo que dificulta la evaluación del modelo para estas clases. Puede ser necesario recopilar más datos o utilizar técnicas de aumento de datos.
- El accuracy global es alta (0.97), lo que indica un buen rendimiento general del modelo.
- Weighted Avg es alto, influenciado por el buen rendimiento en las clases con mayor soporte.

Clasificación

El modelo demuestra un rendimiento satisfactorio. La figura 42 ilustra la comparación entre los valores reales y las predicciones del modelo, donde estas últimas se representan en color tomate.

Figura 42

Resultados de clasificaciones



Nota. Resultados de la clasificación del clima.

Prueba

Utilizando variables extraídas de los datos de OpenWeather, se construyó un conjunto de datos para la evaluación del modelo clasificador.

Figura 43

Datos de prueba

```
# Crear un nuevo DataFrame con los valores de las variables para la predicción
new_data = pd.DataFrame({
    'temp': [20.31],
    'pressure': [1019],
    'humidity': [56],
    'wind_speed': [2.05],
    'wind_deg': [104],
    'rain_1h': [0],
    'clouds_all': [1]
})

new_data
```

	temp	pressure	humidity	wind_speed	wind_deg	rain_1h	clouds_all
0	20.31	1019	56	2.05	104	0	1

Nota. Creación de dataset para prueba de clasificación

La predicción del modelo resultó en la clase número 2, la cual se asocia con el tipo de clima 'despejado', corroborando la información presente en los datos originales sin procesar.

Figura 44

Resultados de Clasificación

```
# Hacer la predicción usando el modelo 'model'
prediction = model.predict(new_data)  # Reemplazar 'rf_model' con 'model'

# Obtener la clase predicha (asumiendo que 'class_mapping' está definido)
predicted_class = class_mapping[prediction[0]]

# Imprimir la clase predicha
print(f"Clase predicha: {predicted_class}")
```

```
Clase predicha: 2
```

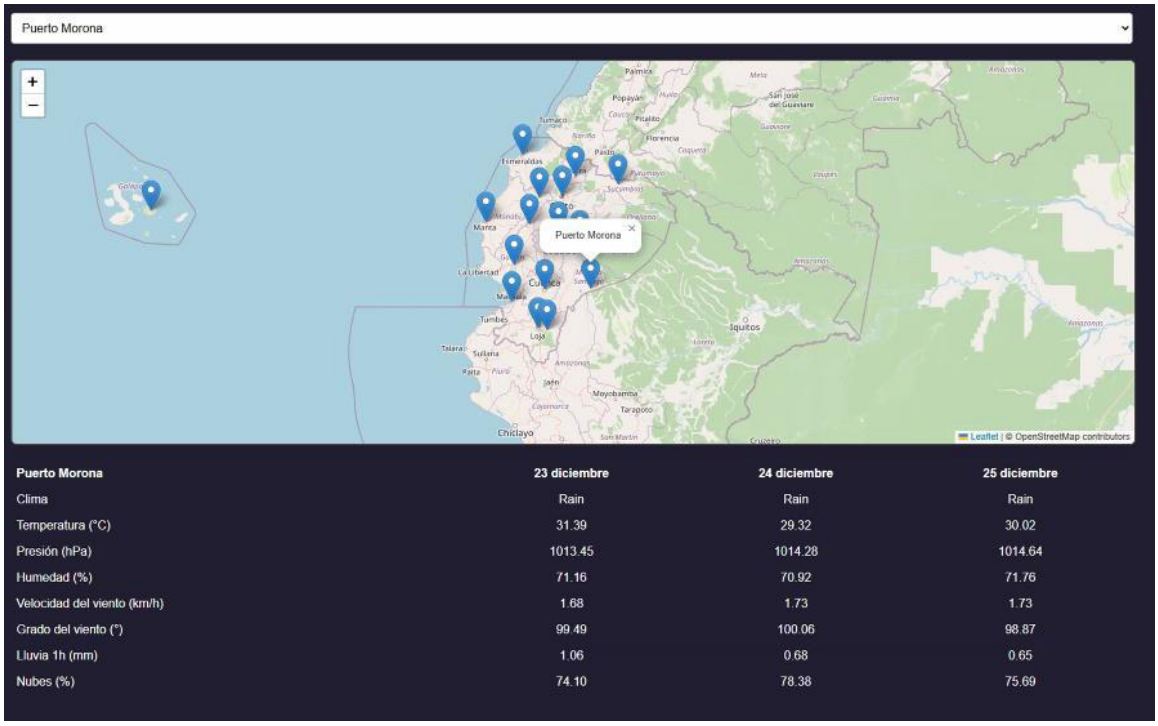
Nota. Resultados de la clasificación para nuevos datos.

Visualización Interactiva

Como un componente adicional a este proyecto, se ha desarrollado una interfaz web interactiva que permite a los usuarios acceder y visualizar las predicciones del modelo de clasificación del clima de manera más intuitiva y atractiva. Esta interfaz, diseñada como un valor añadido, busca enriquecer la experiencia del usuario al proporcionar una plataforma amigable para explorar los resultados del modelo.

Figura 45

Interfaz interactiva.



Puerto Morona	23 diciembre	24 diciembre	25 diciembre
Clima	Rain	Rain	Rain
Temperatura (°C)	31.39	29.32	30.02
Presión (hPa)	1013.45	1014.28	1014.64
Humedad (%)	71.16	70.92	71.76
Velocidad del viento (km/h)	1.68	1.73	1.73
Grado del viento (°)	99.49	100.06	98.87
Lluvia 1h (mm)	1.06	0.68	0.65
Nubes (%)	74.10	78.38	75.69

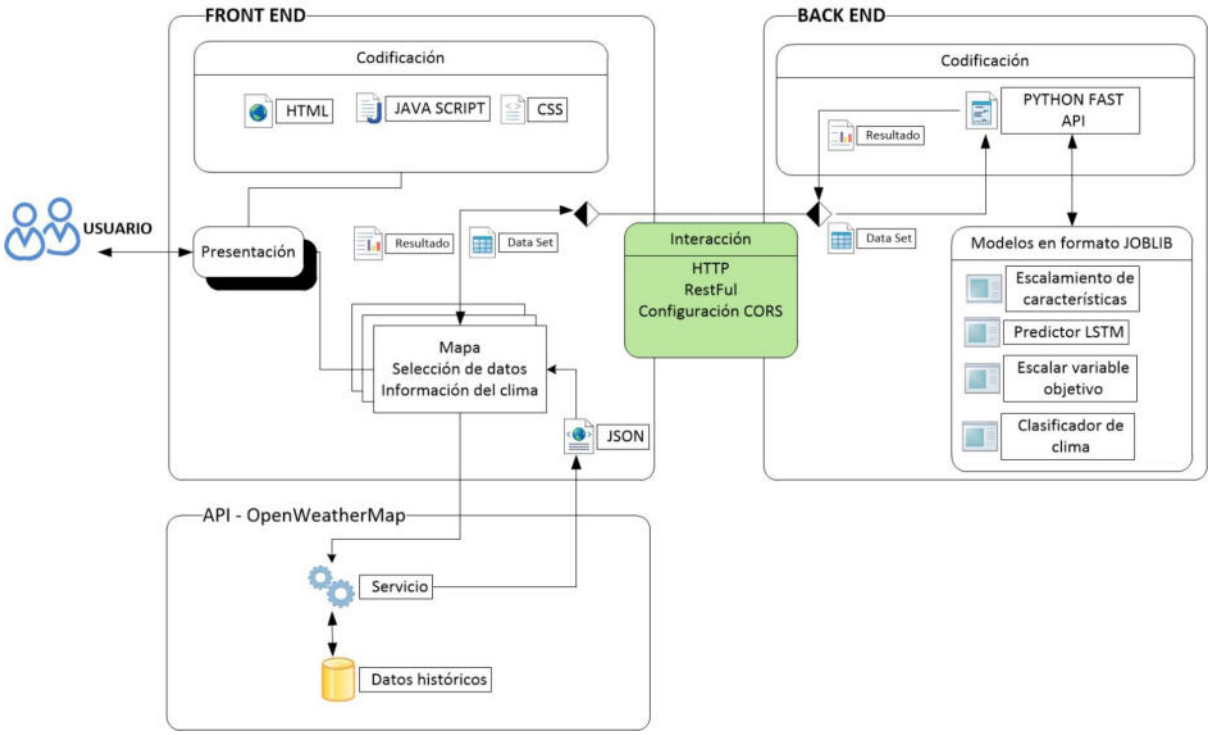
Nota. Interfaz interactiva para visualización de datos de predicción y clasificación

Esquema conceptual

La figura 46 detalla el esquema conceptual del sistema está diseñado como una aplicación web que realiza la tarea de predecir condiciones climáticas. Se utiliza una arquitectura cliente-servidor.

Figura 46

Esquema conceptual de la interfaz de usuario



Nota. Esquema conceptual del front end y back end y su funcionamiento

Backend (lado servidor)

El backend está implementado en Python utilizando el framework FastAPI, escogido por su ligereza y eficiencia para construir APIs RESTful. Se encarga de manejar las solicitudes del cliente, procesar datos y generar predicciones climáticas. El soporte con la librería Pydantic para la validación de los datos en los endpoint fue otro motivo de peso para seleccionarla.

El lado servidor utiliza modelos previamente entrenados en formatos “. joblib” para predecir siete variables climáticas: temperature, pressure, humidity, wind_speed, wind_deg, rain_1h y clouds_all. También incluye un clasificador que usa estas variables climáticas y mapea las condiciones climáticas a categorías predefinidas.

Cuenta con configuraciones de CORS para permitir la comunicación segura entre el frontend y el backend.

El endpoint principal es /predict y se encarga de procesar los datos de entrada que luego serán utilizados por los modelos mencionados para obtener las predicciones y devuelve los resultados al frontend.

El backend recibe estos datos formateados desde el frontend, de esta manera, el frontend actúa como el intermediario que recopila la información necesaria del usuario y la API, mientras que el backend lleva a cabo el análisis avanzado y devuelve los resultados al frontend para mostrarlos al usuario.

Frontend (lado cliente)

El frontend está desarrollado con HTML, CSS y JavaScript garantizando al usuario una experiencia fluida y amigable. Este cuenta con un mapa interactivo que permite seleccionar 16 ciudades del Ecuador y muestra predicciones climáticas detalladas basadas en los resultados recibidos desde el backend.

El mapa interactivo hace uso de la librería Leaflet.js y permite interactuar al usuario a con las diferentes ciudades: Ibarra, Loja, Zamora, Lago Agrío, Puyo, Puerto Morona, Santa Cruz Island, Quevedo, Manta, Santo Domingo, Cuenca, Ambato, Quito, Machala, Guayaquil y Esmeraldas.

Cuenta con una tabla dinámica que muestra las predicciones climáticas para los próximos tres días a partir de la ciudad seleccionada.

La lógica en el lado cliente es responsable de interactuar directamente con la API de OpenWeatherMap a partir de la ciudad seleccionada por el usuario en la interfaz, así se obtienen los datos meteorológicos de los últimos 16 días mediante una solicitud a dicha API para luego ser formateados adecuadamente antes de ser enviados al backend.

Frontend y el Backend

El frontend y el backend están conectados a través de solicitudes HTTP. La arquitectura utiliza el patrón RESTful, donde el frontend envía datos al backend, que procesa las solicitudes y devuelve las predicciones en formato JSON. Esto permite una separación entre el negocio y la presentación.

CAPITULO IV

Conclusiones

El proyecto presentado describe un sistema de predicción climática para Ecuador que aprovecha técnicas avanzadas de aprendizaje automático. A partir de la información proporcionada, podemos extraer las siguientes conclusiones:

Modelo Híbrido Robusto la combinación de LSTM y Random Forest Classifier demuestra ser una estrategia efectiva para abordar la complejidad de los datos climáticos. LSTM captura las dependencias temporales, mientras que Random Forest proporciona una clasificación precisa de los diferentes tipos de clima. Separar los procesos en un modelo de predicción y un modelo de clasificación fue crucial para mejorar la precisión y la eficacia de nuestro sistema de predicción del clima. Esta estrategia permitió una mayor especialización, flexibilidad y control sobre cada tarea individual, lo que llevó a resultados más robustos y confiables.

Al eliminar el ruido introducido por las fluctuaciones horarias menos relevantes, el modelo se enfocó en los patrones climáticos a largo plazo y realizar predicciones más precisas. Esta selección estratégica de datos permitió al modelo aprender de manera más eficiente y generalizar mejor a nuevas situaciones.

La creación de ventanas para el análisis de predicción con el modelo LSTM, generó un mejor aprovechamiento de la data, ya que las ventanas están diseñadas para procesar relaciones temporales, lo que resulta en predicciones más robustas y precisas.

El modelo de clasificación exhibe un sólido desempeño general, pero es crucial abordar las clases con menor rendimiento y soporte para optimizar la precisión de la clasificación. La disminución en el rendimiento para estas clases podría atribuirse a la baja frecuencia o incluso ausencia de estos tipos climáticos específicos en el contexto ecuatoriano.

La arquitectura del sistema se basa en una aplicación web que actúa como interfaz central. Esta aplicación realiza consultas a la API externa OpenWeather para obtener datos históricos de los últimos 16 días. Los datos procesados son enviados a un modelo de predicción que genera pronósticos para las próximas 72 horas. Posteriormente, estos pronósticos son ingresados a un modelo de clasificación que determina el tipo de clima. El backend del sistema está desarrollado en Python, mientras que el frontend utiliza JavaScript como resultado se obtiene una estructura modular.

La estructura modular del sistema, con la interfaz web, facilita su integración en diferentes aplicaciones y permite su escalamiento para cubrir más regiones o aumentar la frecuencia de las predicciones que permite la escalabilidad y flexibilidad del proyecto.

El proyecto tiene una precisión y fiabilidad por la evaluación de múltiples configuraciones un total de 25 para la selección de los modelos óptimos garantizan un alto nivel de precisión en las predicciones.

Recomendaciones

Extender el alcance de las predicciones climáticas más allá de tres días, explorando métodos avanzados de modelado para proyecciones a mediano y largo plazo.

Incorporar más métricas de evaluación, como la precisión específica por clase y análisis de errores, para entender mejor el rendimiento de los modelos y orientar su mejora.

Investigar la incorporación de información geográfica, como la topografía, la vegetación, entre otros, para incrementar la exactitud de las proyecciones a escala local.

Establecer un sistema de actualización automática de los modelos para asegurar que continúen siendo exactos frente a variaciones climáticas y la accesibilidad a nuevos datos. Además, contrastar las proyecciones del modelo con datos recogidos en tiempo real para valorar su rendimiento y hacer modificaciones si se requiere.

BIBLIOGRAFIA

ALTER. (1 de 1 de 2024). *Breve explicación del proceso KDD*. Obtenido de Certificacion y equipos alter technology: https://certificacionyequipos.altertechnology.com/breve-explicacion-del-proceso-kdd/

Amazon Web Services. (1 de 1 de 2024). *¿Qué es Python?* Obtenido de AWS: https://aws.amazon.com/es/what-is/python/

BID. (06 de 03 de 2024). *iadb.org*. Obtenido de https://www.iadb.org/es/noticias/cambio-climatico-en-america-latina-y-el-caribe

Bogdan, B., & Ustrnul, Z. (23 de 1 de 2022). Machine learning in weather prediction and climate analyses applications and perspectives. *Atmosphere*, 1. doi:https://doi.org/10.3390/atmos13020180

CEPAL. (1 de 1 de 2023). *Acerca de Cambio Climático*. Obtenido de CEPAL: https://www.cepal.org/es/temas/cambio-climatico/acerca-cambio-climatico

CertiDevs. (s.f.). *Tutorial Pandas, identificación de valores faltantes*. Obtenido de CertiDevs: https://certidevs.com/tutorial-pandas-identificacion-de-valores-faltantes

Chivatá, J. D. (2024). ¿Cómo se entrenan las redes neuronales? *EIG / UIDE*, 1.

Code Labs Academy. (1 de 1 de 2024). *El papel de la regularización l1 y l2 en la prevención del sobreajuste y la mejora de la generalización de modelos*. Obtenido de Code Labs Academy: https://codelabsacademy.com/es/blog/the-role-of-l1-and-l2-regularization-in-preventing-overfitting-and-enhancing-model-generalization

Dans, E. (14 de 11 de 2023). *Prediciendo el tiempo mediante machine learning*. Obtenido de Ed: https://www.enriquedans.com/2023/11/prediciendo-el-tiempo-mediante-machine-learning.html

Datahack Consulting. (30 de 11 de 2020). *Análisis en Big Data: métodos de minería de datos y herramientas*. Obtenido de Datahack: https://www.datahack.es/analisis-big-data/

Devadiga, S. (s.f.). *Your Source for Level1 and Atmospheric Data*. Obtenido de https://ladsweb.modaps.eosdis.nasa.gov/

Deyimar, A. (10 de 1 de 2023). *¿Qué es JSON?* Obtenido de Hostinger Tutoriales: https://www.hostinger.es/tutoriales/que-es-json#:~:text=Se%20trata%20de%20un%20formato,rendimiento%20de%20tu%20sitio%20web.

Domino Data Lab Inc. (1 de 1 de 2024). *¿Qué es Scikit-learn?* Obtenido de Domino: https://domino.ai/data-science-dictionary/sklearn

EducaOpen. (1 de 1 de 2023). *¿Qué son las series temporales y para qué se usan en machine learning?* Obtenido de educaOpen: https://www.educaopen.com/digital-lab/blog/inteligencia-artificial/algoritmos-de-aprendizaje-profundo

Ehijo, E. V. (28 de 8 de 2024). *Inteligencia Artificial y predicción climática: una alianza con gran potencial para hacer frente al cambio climático*. Obtenido de Universidad Internacional de Valencia: https://www.universidadviu.com/ec/actualidad/noticias/inteligencia-artificial-y-prediccion-climatica-una-alianza-con-gran-potencial-para-hacer-frente-al-cambio-climatico

Federica , Z., Simeoni, C., Torresan, S., Aslan, S., Critto, A., & Marcomini, A. (27 de 07 de 2021). *Exploring machine learning potential for climate change risk assessment*. doi:https://doi.org/10.1016/j.earscirev.2021.103752

Fernández, R. (1 de 09 de 2021). *Series Temporales Avanzadas: Aplicación de Redes Neuronales para el Pronóstico de Series de Tiempo.* Obtenido de Universidad de Granada: https://masteres.ugr.es/estadistica-aplicada/sites/master/moea/public/inline-files/TFM_FernAndez%20SalgueroRicardo%20Alonzo.pdf

Gobierno de México. (18 de 05 de 2018). *https://www.gob.mx*. Recuperado el 8 de 9 de 2024, de Gobierno de México: https://www.gob.mx/inecc/acciones-y-programas/que-es-el-clima

Gullo, F. (2020). From Patterns in Data to Knowledge Discovery. *ScienceDirect*, 4.

Hassan , A., Aitazaz , F., Travis, E., Arnold, S., Qamar, Z., Farhat, A., & Melanie, B. (05 de 06 de 2023). Modelado neuronal artificial para prácticas de gestión del agua en agricultura de precisión. *Agricultura de precisión*, 169. Obtenido de https://www.sciencedirect.com/science/article/abs/pii/B9780443189531000052

Haya, P. (29 de 11 de 2021). *La metodología crisp dm en ciencia de datos*. Obtenido de Instituto de Ingeniería del Conocimiento: https://www.iic.uam.es/innovacion/metodologia-crisp-dm-ciencia-de-datos/

Heras, J. (1 de 8 de 2024). *Hiperparámetros de una red neuronal*. Obtenido de Inteligencia Artificial: https://blog.javierheras.website/machine-learning/

IBM. (23 de 11 de 2021). *¿Qué es el aprendizaje no supervisado?* Obtenido de IBM: https://www.ibm.com/think/topics/unsupervised-learning

INAMHI. (14 de 10 de 2024). *Boletín 07.* Obtenido de Boletines: https://servicios.inamhi.gob.ec/clima/

Instituto Nacional de Ecología y Cambio Climático . (s.f.). *¿Qué es el clima?* Obtenido de Gobierno de México: https://www.gob.mx/inecc/acciones-y-programas/que-es-el-clima

Julio, Q., & Armando, S. (2022). Aprendizaje Automático como herramienta para el manejo integrado de recursos naturales en un contexto de cambio climático. *Revista Cubana de Transformación Digital*, 1. Obtenido de https://portal.amelica.org/ameli/journal/389/3893437002/html/

Kili Technology. (2 de 1 de 2023). *Conjuntos de entrenamiento, validación y prueba: cómo dividir los datos de aprendizaje automático*. Obtenido de Kili: https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data

Lema, D., Natalia, P., & Toapanta, E. (21 de Agosto de 2023). *Patrones de comportamiento de temperatura en el Ecuador en modelos de circulación atmosférica mediante Clustering.* Manta, Ecuador: Revista Científico-Académica Multidisciplinaria. Obtenido de https://polodelconocimiento.com/ojs/index.php/es/article/view/5962

Markus, R., Gustau, C., Stevens, B., & Martin, J. (14 de February de 2019). Deep learning and process understanding. *Perspective, 566*, 2. doi:https://doi.org/10.1038/s41586-019-0912-1

Marzieh Fathi, M. H. (28 de 06 de 2021). Big Data Analytics in Weather Forecasting: A Systematic Review. *SpringerLink*, 1. doi:https://doi.org/10.1007/s11831-021-09616-4

MathWorks. (1 de 1 de 2024). *Aprendizaje supervisado*. Obtenido de MathWorks: https://la.mathworks.com/discovery/supervised-learning.html#:~:text=El%20aprendizaje%20supervisado%20se%20emplea,en%20mantenimiento%20predictivo%2C%20para%20estimar

Mauricio, J. A. (2007). Introducción al análisis de series temporales. En J. A. Mauricio, *Series temporales* (pág. 1). Madrid.

Meneses, B., Brescia, F., & Deyvis, Q. (2023). Machine Learning for predicting climate change in the environment: Review. *Serie de conferencias*, 1. Obtenido de https://www.researchgate.net/publication/377856446_Machine_Learning_para_la_prediccion_de_cambios_climaticos_en_el_medio_ambiente_Revision

MG, S., Betancourt, C., Gong, B., & Kleinert, F. (2021). Can deep learning beatnumerical weather prediction? *Royal Society Publishing*, 8. doi:https://doi.org/10.1098/rsta.2020.0097

Morocho, E. (2024). *Análisis exploratorio de datos de vinos Piedmont.* Quito: EIG / UIDE.

OpenWeather. (1 de 1 de 2024). *API Meteorológica*. Obtenido de OpenWeather: https://docs.openweather.co.uk/api

OpenWeather. (1 de 1 de 2024). Mapas del tiempo 2.0. *Mapas meteorológicos históricos, actuales y de previsión. 15 capas de mapas meteorológicos* ., pág. 1.

Oracle. (24 de 11 de 2020). *¿Qué es una base de datos?* Obtenido de Oracle México: https://www.oracle.com/mx/database/what-is-database/

Organización Meteorológica Mundial. (1 de 1 de 2024). *Clima*. Obtenido de wmo: https://wmo.int/topics/climate

Rogel Alexander, H. A. (31 de 07 de 2024). Aplicación de redes neuronales artificiales para la estimación de precipitaciones: caso de estudio de la cuenca del río pastaza, ecuador. *Finibus*, 2. Obtenido de https://publicacionescd.uleam.edu.ec/index.php/finibus/article/view/819/1370

Sánchez, A. (14 de 05 de 2022). *Aprende con Alf*. Obtenido de La librería Pandas: https://aprendeconalf.es/docencia/python/manual/pandas/

Sanchez, D. (19 de 1 de 2023). *Métodos de pronóstico de Series Temporales.* Obtenido de Machine Learning Pills: https://mlpills.dev/series-temporales/pronostico-de-series-temporales/

SaturnCloud. (6 de 7 de 2023). *¿Qué es la varianza explicada PCA de Sklearn y la diferencia de razón de varianza explicada?* Obtenido de https://saturncloud.io/: https://saturncloud.io/blog/what-is-sklearn-pca-explained-variance-and-explained-variance-ratio-difference/#explained-variance-in-sklearn-pca

Smolic, H. (10 de 05 de 2024). *Grapite Note*. Obtenido de understanding-target-variables-in-machine-learning: https://graphite-note.com/understanding-target-variables-in-machine-learning/

Solanky, S. (17 de 08 de 2022). *joblib - Parallel Processing/Computing in Python*. Obtenido de Coderz column: https://coderzcolumn.com/tutorials/python/joblib-parallel-processing-in-python

TensorFlow. (1 de 1 de 2024). *Introducción a TensorFlow*. Obtenido de TensorFlow: https://www.tensorflow.org/learn?hl=es-419

Thattamparambil, N. (17 de 02 de 2020). *Cómo elegir la metodología de investigación más adecuada para su estudio.* Obtenido de Editage Insights: https://www.editage.com/insights/how-to-choose-the-research-methodology-best-suited-for-your-study?refer=scroll-to-1-article&refer-type=article

United Nations. (01 de 08 de 2021). *https://www.un.org*. Obtenido de UN: https://www.un.org/es/climatechange/what-is-climate-change

Vasquez, K. O. (1 de 1 de 2022). *Guia de Estudio N°4 Aspectos Climáticos del Aire.* Obtenido de Studocu: https://www.studocu.com/latam/document/universidad-politecnica-de-ingenieria/fisica/04-temperatura-del-aire/85091920

Visual Studio Code. (12 de 11 de 2024). *Code faster with AI*. Obtenido de Código de Visual Studio: https://code.visualstudio.com/docs/languages/overview

Waskom, M. (1 de 1 de 2024). *Visualización de datos estadísticos*. Obtenido de Seaborn: https://seaborn.pydata.org/tutorial/introduction.html

Williamson, T. (2 de 02 de 2024). *¿Qué son las características dependientes e independientes en el aprendizaje automático?* Obtenido de Deep Checks: https://www.deepchecks.com/question/what-are-dependent-and-independent-features-in-machine-learning/

World Bank Group. (2024). *Ecuador - Country note on climate change aspects in agriculture.* Washington, D.C.: World Bank Group.

APENDICE A

Código Predictor

```
import warnings
from pathlib import Path
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.decomposition import PCA
from matplotlib import pyplot as plt
from sklearn.linear_model import LinearRegression
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Reshape, Softmax
from tensorflow.keras.callbacks import EarlyStopping
import joblib

pd.set_option('display.max_columns', None)
warnings.filterwarnings('ignore')

# Verificando en dodne se encuentra este archivo
print(Path.cwd())

path = Path('.').resolve().parent.parent
path

df = pd.read_csv("data_clima.csv")
df
```

```
df.columns

df['dt'] = pd.to_datetime(df['dt'], unit='s')

df

df['temp'] = df['temp'].astype('float64')

data_2 = df[(df['dt'].dt.year >= 1979) & (df['dt'].dt.year <= 2024)]
data_2

data_2 = data_2[['dt', 'city_name', 'temp', 'dew_point', 'feels_like', 'temp_min', 'temp_max',
   'pressure', 'humidity', 'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all']]

data_2['temp'].dtypes

data = data_2.copy()
#data = data[data['city_name'].isin(['Ibarra'])]
data

# Convert 'dt' column to datetime objects if not already done
data['dt'] = pd.to_datetime(data['dt'])
# Extract year, month, day, and hour
data['year'] = data['dt'].dt.year
data['month'] = data['dt'].dt.month
data['day'] = data['dt'].dt.day
data['hour'] = data['dt'].dt.hour
data
```

```
data_morn = data.copy()

data_aft = data.copy()

data_ngt = data.copy()


def calculate_averages(df, columns_to_average=[], hours_to_average=[]):
    # Filter data for the specified hours
    filtered_df = df[df['hour'].isin(hours_to_average)]
    # Group by year, month, day and calculate mean for each specified column
    daily_averages = filtered_df.groupby(['year', 'month',
'day'])[columns_to_average].mean().reset_index()
    # Rename columns to indicate daily averages
    daily_averages = daily_averages.rename(columns={col: f'daily_avg_{col}' for col in
columns_to_average})
    # Merge daily averages back into the original DataFrame
    df = pd.merge(df, daily_averages, on=['year', 'month', 'day'], how='left')
    return df


hours_morning_to_average = [6,8,10]

hours_aft_to_average = [11,13,15]

hours_ngt_to_average = [18,20,22]


# Group the DataFrame by date and get the first occurrence of each day

data_morn = calculate_averages(data_morn, columns_to_average=['temp', 'dew_point',
'feels_like', 'temp_min', 'temp_max', 'pressure', 'humidity',
                                                'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all'],
hours_to_average=hours_morning_to_average)

# Use pd.Grouper to specify the frequency

morning_data = data_morn.groupby([pd.Grouper(key='dt', freq='D'),
'city_name']).first().reset_index()

# Display the new DataFrame

morning_data.head(10)
```

```
# Group the DataFrame by date and get the first occurrence of each day
data_aft = calculate_averages(data_aft, columns_to_average=['temp', 'dew_point', 'feels_like', 'temp_min', 'temp_max', 'pressure', 'humidity',
                                                            'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all'], hours_to_average=hours_aft_to_average)
# Use pd.Grouper to specify the frequency
aft_data = data_aft.groupby([pd.Grouper(key='dt', freq='D'), 'city_name']).first().reset_index()
# Display the new DataFrame
aft_data.head(10)


# Group the DataFrame by date and get the first occurrence of each day
data_ngt = calculate_averages(data_ngt, columns_to_average=['temp', 'dew_point', 'feels_like', 'temp_min', 'temp_max', 'pressure', 'humidity',
                                                            'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all'], hours_to_average=hours_ngt_to_average)
# Use pd.Grouper to specify the frequency
ngt_data = data_ngt.groupby([pd.Grouper(key='dt', freq='D'), 'city_name']).first().reset_index()
# Display the new DataFrame
ngt_data.head(10)


morning_data.columns


aft_data.columns


ngt_data.columns


columns_drop = {'dt', 'city_name', 'temp', 'dew_point', 'feels_like', 'temp_min',
    'temp_max', 'pressure', 'humidity', 'wind_speed', 'wind_deg', 'rain_1h',
    'clouds_all', 'year', 'month', 'day', 'hour'}
```

```
columns = ['daily_avg_temp','daily_avg_pressure', 'daily_avg_humidity',
'daily_avg_wind_speed',
        'daily_avg_wind_deg', 'daily_avg_rain_1h', 'daily_avg_clouds_all']


def drop_columns(df, columns_drop, columns):
    df = df.drop(columns=columns_drop)
    df = df[columns].fillna(0)
    return df


morning_data = drop_columns(morning_data, columns_drop, columns)
morning_data.head()


aft_data = drop_columns(aft_data, columns_drop, columns)
aft_data.head()


ngt_data = drop_columns(ngt_data, columns_drop, columns)
ngt_data.head()


def create_sequences_classification(data, target, window_size, dias_futuro):
    sequences = []
    labels = []

    for i in range(len(data) - window_size - dias_futuro):
        # Agregar la ventana de datos como secuencia de entrada
        sequences.append(data[i:i + window_size])

        # Las etiquetas para los siguientes dias_futuro días
        labels.append(target[i + window_size + 1:i + window_size + dias_futuro + 1])

    return np.array(sequences), np.array(labels)
```

```
num_clases = 1
N = 1 # features
dias_historia = 16
dias_futuro = 3

def train_fit_predit(df, features, target, num_clases, N, dias_historia, dias_futuro):
  scaler = MinMaxScaler()
  sc_data = scaler.fit_transform(df[features])
  scaled = pd.DataFrame(sc_data, columns=features)

  sequences, labels = create_sequences_classification(
    scaled[features].values,  # Convertir a array solo las columnas relevantes
    scaled[target].values.flatten(),  # Convertir el target a array
    dias_historia,
    dias_futuro
  )

  train_size = int(0.8 * len(sequences))
  X_train, X_test = sequences[:train_size], sequences[train_size:]
  y_train, y_test = labels[:train_size], labels[train_size:]

  # Crear el modelo
  model = Sequential()
  model.add(LSTM(256, return_sequences=True, input_shape=(dias_historia, N)))
  model.add(Dropout(0.2))
  model.add(LSTM(128))
  model.add(Dropout(0.2))
  model.add(Dense(dias_futuro))

  model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

```
  # Configuración del Early Stopping
  early_stopping = EarlyStopping(
    monitor='val_loss',       # Métrica que se monitorea
    patience=10,              # Número de épocas sin mejora antes de detenerse
    restore_best_weights=True # Restaura los mejores pesos al finalizar
  )

  history = model.fit(X_train, y_train, epochs=100, batch_size=16, validation_data=(X_test,
y_test), callbacks=[early_stopping] )

  predictions = model.predict(X_test)

  return model, history, predictions

def accuracy_graf(history):
  plt.plot(history.history['mae'], label='Train mae')
  plt.plot(history.history['val_mae'], label='Validation mae')
  plt.plot(history.history['loss'], label='Train Loss')
  plt.plot(history.history['val_loss'], label='Validation Loss')
  plt.xlabel('Epochs')
  plt.ylabel('Value')
  plt.title('Accuracy and Loss Graph')
  plt.legend()
  plt.show()

def predictions_scaled(df, model, features, target):
  scaler_features = MinMaxScaler()
  scaler_target = MinMaxScaler()

  data_features_scaled = scaler_features.fit_transform(df[features])
  data_target_scaled = scaler_target.fit_transform(df[target])
```

```
  target_scaler_filename = f'{target[0]}_target_scaler.joblib'
  joblib.dump(scaler_target, target_scaler_filename)

  features_scaler_filename = f'{target[0]}_features_scaler.joblib'
  joblib.dump(scaler_features, features_scaler_filename)

  sequences, labels = create_sequences_classification(
    data_features_scaled,
    data_target_scaled.flatten(),
    dias_historia,
    dias_futuro
  )

  train_size = int(0.8 * len(sequences))
  X_train, X_test = sequences[:train_size], sequences[train_size:]
  y_train, y_test = labels[:train_size], labels[train_size:]

  predictions = model.predict(X_test)

  predictions_rescaled = scaler_target.inverse_transform(predictions)
  y_test_rescaled = scaler_target.inverse_transform(y_test)

  return predictions_rescaled, y_test_rescaled

def show_results(df, model, features, target):
  predictions_rescaled, y_test_rescaled = predictions_scaled(df, model, features, target)

  results = []
  for i in range(len(predictions_rescaled)):
```

```
    results.append({f'Prediction_{target[0]}': predictions_rescaled[i]})
    # print(f"Predicción: {predictions_rescaled[i]}, Real: {y_test_rescaled[i]}")

  df_results = pd.DataFrame(results)

  return df_results

"""### TEMPERATURA"""

features = ['daily_avg_temp','daily_avg_pressure', 'daily_avg_humidity',
'daily_avg_wind_speed',
        'daily_avg_wind_deg', 'daily_avg_rain_1h', 'daily_avg_clouds_all']

#features = ['daily_avg_temp']
target = ['daily_avg_temp']

df_final_results = pd.DataFrame()

targetOld = 'pepe'
for feature in features:
    target[0] = feature

    print(f'\nTARGET {feature.upper()}:')

    # mañana
    '''print(f'\nProcesando {feature} en mañana:')
    model, history, predictions = train_fit_predit(morning_data, features, target, num_clases,
N, dias_historia, dias_futuro)
    accuracy_graf(history)
    df_results_mrn = show_results(morning_data, model, features, target)'''
```

```
    # tarde
    print(f'\nProcesando {feature} en tarde:')
    model, history, predictions = train_fit_predit(aft_data, target, target, num_clases, N,
dias_historia, dias_futuro)
    accuracy_graf(history)
    df_results_aft = show_results(aft_data, model, target, target)
    # Define el nombre del archivo del modelo
    model_filename = f'{feature}.joblib'
    # Guarda el modelo en el entorno de Colab (temporalmente)
    joblib.dump(model, model_filename)


    # noche
    '''print(f'\nProcesando {feature} en noche:')
    model, history, predictions = train_fit_predit(ngt_data, features, target, num_clases, N,
dias_historia, dias_futuro)
    accuracy_graf(history)
    df_results_ngt = show_results(ngt_data, model, features, target)'''


    '''typeAxis = 0
    if target[0] == targetOld:
        df_final_results = pd.concat([df_final_results, df_results_mrn, df_results_aft,
df_results_ngt], axis=0)
    else:
        df_final_results = pd.concat([df_final_results, df_results_mrn, df_results_aft,
df_results_ngt], axis=1)'''


    #targetOld = target[0]


df_results_aft


model = joblib.load("daily_avg_temp.joblib")
feature_scaler = joblib.load("daily_avg_temp_features_scaler.joblib")
```

```
target_scaler = joblib.load("daily_avg_temp_target_scaler.joblib")

input_data = pd.DataFrame({
"daily_avg_temp": [17.0, 16.721596, 17.1, 16.205051, 17.18127, 16.1423, 17.14212, 16.24103, 17.192286, 16.0, 17.2, 16.14212, 17.24103, 16.192286, 17.0, 18.2]
})

input_data = target_scaler.transform(input_data)

# Convertir el DataFrame a un arreglo NumPy
#input_array = input_data.to_numpy()

# Cambiar la forma para que sea (1, 16, 7)
input_array_reshape = input_data.reshape(1, *input_data.shape)

# Predicción del modelo de regresión
regression_results = model.predict(input_array_reshape)

# Desescalar las predicciones
regression_results_scaled = target_scaler.inverse_transform(regression_results)

regression_results_scaled_list = regression_results_scaled.tolist()
print(regression_results_scaled_list)

model = joblib.load("daily_avg_pressure.joblib")
feature_scaler = joblib.load("daily_avg_pressure_features_scaler.joblib")
target_scaler = joblib.load("daily_avg_pressure_target_scaler.joblib")

input_data = pd.DataFrame({
   "daily_avg_pressure": [1017, 1017, 1017, 1020, 1017, 1020, 1017, 1019, 1020, 1018, 1019, 1018, 1017, 1019, 1017, 1020]
```

```
})

input_data = target_scaler.transform(input_data)

# Convertir el DataFrame a un arreglo NumPy
#input_array = input_data.to_numpy()

# Cambiar la forma para que sea (1, 16, 7)
input_array_reshape = input_data.reshape(1, *input_data.shape)

# Predicción del modelo de regresión
regression_results = model.predict(input_array_reshape)

# Desescalar las predicciones
regression_results_scaled = target_scaler.inverse_transform(regression_results)

regression_results_scaled_list = regression_results_scaled.tolist()
print(regression_results_scaled_list)

# Crea una lista con los nombres de las columnas originales
columnas_originales = df_final_results.columns.tolist()

# Define los nombres de los niveles del índice jerárquico
niveles = ['Variable', 'Periodo']

# Crea las listas de etiquetas para cada nivel
etiquetas_variable = ['Temp', 'Temp', 'Temp',
                      'Pressure', 'Pressure', 'Pressure',
                      'Humidity', 'Humidity','Humidity',
                      'Wind_Speed', 'Wind_Speed', 'Wind_Speed',
```

```
            'Wind_Deg', 'Wind_Deg', 'Wind_Deg',
            'Rain', 'Rain', 'Rain',
            'Clouds', 'Clouds', 'Clouds']  # Repetir para cada periodo

etiquetas_periodo = ['Mañana', 'Tarde', 'Noche',
            'Mañana', 'Tarde', 'Noche',
            'Mañana', 'Tarde', 'Noche',
            'Mañana', 'Tarde', 'Noche',
            'Mañana', 'Tarde', 'Noche',
            'Mañana', 'Tarde', 'Noche',
            'Mañana', 'Tarde', 'Noche']  # Ajustar según los periodos

# Crea el índice jerárquico
indice_jerarquico = pd.MultiIndex.from_arrays([etiquetas_variable, etiquetas_periodo],
names=niveles)

# Asigna el índice jerárquico al DataFrame df_final_results
df_final_results.columns = indice_jerarquico

# Muestra el DataFrame con el índice jerárquico
df_final_results

# Apila las columnas del DataFrame
df_stacked = df_final_results.stack(level=1)  # level=1 para apilar el segundo nivel del índice

# Reorganiza el índice
df_stacked = df_stacked.reorder_levels([1, 0]).sort_index()

# Ordena el índice por periodo (Mañana, Tarde, Noche)
periodos_ordenados = ['Mañana', 'Tarde', 'Noche']  # Define el orden deseado
```

```
df_stacked = df_stacked.sort_values(by=['Periodo'], key=lambda x: x.map({p: i for i, p in enumerate(periodos_ordenados)}))


# Muestra el DataFrame apilado y reorganizado
df_stacked


df_stacked.to_excel(path/'e:\Escritorio\Folders\MASTER\TITULACION\CODIGOS\VERSIONES PREVIAS\XXI\Outputsdf_stacked.xlsx', index=False)


df_stacked.columns


df_stacked = df_stacked.explode(['Temp', 'Pressure', 'Humidity', 'Wind_Speed', 'Wind_Deg', 'Rain',
    'Clouds'])
df_stacked


df_stacked.to_excel(path/'e:\Escritorio\Folders\MASTER\TITULACION\CODIGOS\VERSIONES PREVIAS\XXI\Outputs/ predicciones.xlsx', index=False)
```

APENDICE B

Código Clasificador

```
import warnings
from pathlib import Path
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
#from imblearn.over_sampling import SMOTE
from sklearn.utils import resample
df = pd.read_csv(path/"C:\Tesis\data_clima.csv", delimiter=',')
df['dt'] = pd.to_datetime(df['dt'], unit='s')
df = df[(df['dt'].dt.year >= 1979) & (df['dt'].dt.year <= 2024)]
df['temp'] = df['temp'].astype('float64')
df['weather_main'].value_counts()
params = ['temp', 'pressure', 'humidity', 'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all',
'weather_main']
df = df.drop(columns=['dt', 'dt_iso', 'timezone', 'city_name', 'lat', 'lon', 'visibility', 'dew_point',
'feels_like', 'temp_min', 'temp_max',
                'sea_level', 'grnd_level','wind_gust','rain_3h', 'snow_1h', 'snow_3h', 'weather_id',
'weather_description', 'weather_icon'])
weather_main_to_name = {
  'Clouds': 0,
  'Rain': 1,
  'Clear' : 2,
  'Haze': 3,
  'Fog': 4,
  'Drizzle': 5,
  'Mist': 6,
```

```
 'Thunderstorm': 7,
 'Smoke': 8,
 'Dust': 9,
 'Ash': 10,
 'Tornado': 11,
 'Squall': 12,
 'Snow': 13
}

df['weather_main'] = df['weather_main'].map(weather_main_to_name)
df['weather_main'].value_counts()
df = df[['temp', 'pressure', 'humidity', 'wind_speed', 'wind_deg', 'rain_1h', 'clouds_all',
'weather_main']].fillna(0.0)
X = df.drop('weather_main', axis=1)
y = df['weather_main']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=25)
from sklearn.utils import check_X_y
X_train, y_train = check_X_y(X_train, y_train, dtype='numeric')
model = RandomForestClassifier(n_estimators=100, random_state=25)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
accuracy, report
class_mapping = dict(enumerate(df['weather_main'].astype('category').cat.categories))
y_test_decoded = y_test.map(class_mapping)
y_pred_decoded = pd.Series(y_pred).map(class_mapping)

# Visualize predictions vs. true values
plt.figure(figsize=(10, 6))
```

```
plt.scatter(y_test_decoded.index, y_test_decoded, label='True Values', alpha=0.6, marker='o', s=10)

plt.scatter(y_test_decoded.index, y_pred_decoded, label='Predictions', alpha=0.6, marker='x', s=10)

plt.title('Predictions vs True Values for Weather Classification')

plt.xlabel('Sample Index')

plt.ylabel('Weather Class')

plt.legend()

plt.show()

class_mapping

import joblib

# Define el nombre del archivo del modelo

model_filename = 'T1classificador.joblib'


# Guarda el modelo en el entorno de Colab (temporalmente)

joblib.dump(model, model_filename)
```

APENDICE C

Código Web

Backend

```
from fastapi import FastAPI
from pydantic import BaseModel
import joblib
import pandas as pd
from fastapi.responses import JSONResponse
from fastapi.middleware.cors import CORSMiddleware

# Crear instancia de la aplicación FastAPI
app = FastAPI()

# Configuración del middleware CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:8001", "https://weather.2x3.app"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Cargar modelos joblib
# classification_model = joblib.load("./mini_classificador.joblib")
classification_model = joblib.load("./ClasificadorGeneral.joblib")

models = {
    "temp": {
        "model": joblib.load("./daily_avg_temp.joblib"),
        "features_scaler": joblib.load("./daily_avg_temp_features_scaler.joblib"),
```

```
    "target_scaler": joblib.load("./daily_avg_temp_target_scaler.joblib")
},
"pressure": {
    "model": joblib.load("./daily_avg_pressure.joblib"),
    "features_scaler": joblib.load("./daily_avg_pressure_features_scaler.joblib"),
    "target_scaler": joblib.load("./daily_avg_pressure_target_scaler.joblib")
},
"humidity": {
    "model": joblib.load("./daily_avg_humidity.joblib"),
    "features_scaler": joblib.load("./daily_avg_humidity_features_scaler.joblib"),
    "target_scaler": joblib.load("./daily_avg_humidity_target_scaler.joblib")
},
"wind_speed": {
    "model": joblib.load("./daily_avg_wind_speed.joblib"),
    "features_scaler": joblib.load("./daily_avg_wind_speed_features_scaler.joblib"),
    "target_scaler": joblib.load("./daily_avg_wind_speed_target_scaler.joblib")
},
"wind_deg": {
    "model": joblib.load("./daily_avg_wind_deg.joblib"),
    "features_scaler": joblib.load("./daily_avg_wind_deg_features_scaler.joblib"),
    "target_scaler": joblib.load("./daily_avg_wind_deg_target_scaler.joblib")
},
"rain_1h": {
    "model": joblib.load("./daily_avg_rain_1h.joblib"),
    "features_scaler": joblib.load("./daily_avg_rain_1h_features_scaler.joblib"),
    "target_scaler": joblib.load("./daily_avg_rain_1h_target_scaler.joblib")
},
"clouds_all": {
    "model": joblib.load("./daily_avg_clouds_all.joblib"),
    "features_scaler": joblib.load("./daily_avg_clouds_all_features_scaler.joblib"),
```

```
        "target_scaler": joblib.load("./daily_avg_clouds_all_target_scaler.joblib")
    }
}

# Función genérica para realizar escalado, predicción y desescalado
def predict_feature(input_data, model, features_scaler, target_scaler):
    # Escalar las características de entrada
    input_scaled = features_scaler.transform(input_data)

    # Remodelar los datos para que coincidan con el modelo
    input_reshaped = input_scaled.reshape(1, *input_scaled.shape)

    # Realizar la predicción
    predictions = model.predict(input_reshaped)

    # Desescalar las predicciones
    predictions_scaled = target_scaler.inverse_transform(predictions)

    # Convertir a lista para facilitar su manejo
    return predictions_scaled.tolist()

# Mapeo de valores de clasificación a descripciones de clima
climate_mapping = {
    0: 'Clouds',
    1: 'Rain',
    2: 'Clear',
    3: 'Haze',
    4: 'Fog',
    5: 'Drizzle',
    6: 'Mist',
```

```
    7: 'Thunderstorm',
    8: 'Smoke',
    9: 'Dust',
    10: 'Ash',
    11: 'Tornado',
    12: 'Squall',
    13: 'Snow'
}

# Esquema de entrada
class InputData(BaseModel):
    temp: list[float]
    pressure: list[float]
    humidity: list[float]
    wind_speed: list[float]
    wind_deg: list[float]
    rain_1h: list[float]
    clouds_all: list[float]

# Endpoint para obtener las predicciones climáticas sin parámetros
@app.post("/predict")
async def predict(data: InputData):
    try:
        # Datos de entrada para cada característica
        inputs = {
            "temp": pd.DataFrame({"daily_avg_temp": data.temp}),
            "pressure": pd.DataFrame({"daily_avg_pressure": data.pressure}),
            "humidity": pd.DataFrame({"daily_avg_humidity": data.humidity}),
            "wind_speed": pd.DataFrame({"daily_avg_wind_speed": data.wind_speed}),
            "wind_deg": pd.DataFrame({"daily_avg_wind_deg": data.wind_deg}),
```

```
    "rain_1h": pd.DataFrame({"daily_avg_rain_1h": data.rain_1h}),
    "clouds_all": pd.DataFrame({"daily_avg_clouds_all": data.clouds_all})
}

# Obtener las predicciones para cada característica
predictions = {}
for feature, model_data in models.items():
    predictions[feature] = predict_feature(
        inputs.get(feature, pd.DataFrame()),  # Si no hay datos, pasa un DataFrame vacío
        model_data["model"],
        model_data["features_scaler"],
        model_data["target_scaler"]
    )

# Inicializamos tres diccionarios para cada día
day_1 = {}
day_2 = {}
day_3 = {}

# Asignamos las características a cada diccionario correspondiente
for key, values in predictions.items():
    day_1[key] = [values[0][0]]
    day_2[key] = [values[0][1]]
    day_3[key] = [values[0][2]]

# Convertimos los valores de day_1, day_2, day_3 a enteros regulares
for key in day_1:
    day_1[key] = [float(val) for val in day_1[key]]
    day_2[key] = [float(val) for val in day_2[key]]
    day_3[key] = [float(val) for val in day_3[key]]
```

```
    day1_data = pd.DataFrame(day_1)
    day1_prediction = classification_model.predict(day1_data)
    day1_predicted_class = int(day1_prediction[0])
    day_1["wether_id"] = [climate_mapping[day1_predicted_class]]

    day2_data = pd.DataFrame(day_2)
    day2_prediction = classification_model.predict(day2_data)
    day2_predicted_class = int(day2_prediction[0])
    day_2["wether_id"] = [climate_mapping[day2_predicted_class]]

    day3_data = pd.DataFrame(day_3)
    day3_prediction = classification_model.predict(day3_data)
    day3_predicted_class = int(day3_prediction[0])
    day_3["wether_id"] = [climate_mapping[day3_predicted_class]]

    predictions_dict = {
        "day_1": day_1,
        "day_2": day_2,
        "day_3": day_3
    }

    # Devuelve los resultados con las cabeceras CORS
    response = JSONResponse(content=predictions_dict)
    response.headers["Access-Control-Allow-Origin"] = "*"
    response.headers["Access-Control-Allow-Methods"] = "POST, OPTIONS"
    response.headers["Access-Control-Allow-Headers"] = "*"
    return response

except Exception as e:
```

```
        print(f"Error al hacer la predicción: {str(e)}")
        return {"error": str(e)}

# Endpoint de prueba backend funcionando
@app.get("/online")
def read_root():
    return {"message": "Backend para predicción de clima funcionando"}
```

Frontend

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Predicción de Clima en Ecuador</title>
    <link rel="stylesheet" href="styles.css" />
    <!-- Integrar librería para mapas interactivos -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.9.4/leaflet.js"></script>
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.9.4/leaflet.css"
    />
  </head>
  <body>
    <div id="loading-overlay">
      <div id="loading-spinner">
        <div class="spinner"></div>
        <span style="color: black; font-size: 16px"
          >Cargando datos climáticos...</span
        >
```

```
  </div>
</div>

<div class="container">
  <div class="map-section">
    <!-- Lista desplegable -->
    <select id="province-select">
      <option value="">Selecciona una ciudad</option>
      <option value="ibarra">Ibarra</option>
      <option value="loja">Loja</option>
      <option value="zamora">Zamora</option>
      <option value="lagoAgrio">Lago Agrío</option>
      <option value="puyo">Puyo</option>
      <option value="puertoMorona">Puerto Morona</option>
      <option value="santaCruzIsland">Santa Cruz Island</option>
      <option value="quevedo">Quevedo</option>
      <option value="manta">Manta</option>
      <option value="santoDomingo">Santo Domingo</option>
      <option value="cuenca">Cuenca</option>
      <option value="ambato">Ambato</option>
      <option value="quito">Quito</option>
      <option value="machala">Machala</option>
      <option value="guayaquil">Guayaquil</option>
      <option value="esmeraldas">Esmeraldas</option>
    </select>

    <!-- Mapa -->
    <div id="map"></div>
  </div>
```

```
    <!-- Sección de resultados -->
    <div id="results">
      <ul id="climate-data">
        <li>Selecciona una ciudad para ver los datos del clima.</li>
      </ul>
    </div>
  </div>

  <script src="script.js"></script>
 </body>
</html>

/* General Styles */
body {
 font-family: Arial, sans-serif;
 margin: 0;
 padding: 0;
 box-sizing: border-box;
 background-color: #1e1e2f;
 color: #fff;
}

h1,
h2 {
 text-align: center;
}

.container {
 width: 80%;
 margin: auto;
```

```
  padding: 20px;
}

#map {
  height: 500px;
  width: 100%;
  margin: 20px 0;
  border-radius: 8px;
  border: 2px solid #444;
}

#province-select {
  display: block;
  margin: 0 auto 20px auto;
  padding: 10px;
  font-size: 1rem;
  border-radius: 5px;
  width: 100%;
}

#results {
  text-align: center;
  margin-top: 20px;
}

#climate-data {
  list-style: none;
  padding: 0;
  font-size: 1rem;
}
```

```
/* Fondo oscuro bloqueante detrás del spinner */
#loading-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.5);
  z-index: 999;
  display: none;
}

/* Estilos del spinner */
#loading-spinner {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background: #ffffff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  text-align: center;
  z-index: 1000;

  display: flex;
  align-items: center;
  gap: 10px;
}
```

```
.spinner {
  border: 4px solid rgba(0, 0, 0, 0.1);
  width: 36px;
  height: 36px;
  border-radius: 50%;
  border-left-color: #09f;

  animation: spin 1s ease infinite;
}

@keyframes spin {
  0% {
    transform: rotate(0deg);
  }

  100% {
    transform: rotate(360deg);
  }
}

let isUpdatingSelect = false;

// Inicializar el mapa usando Leaflet.js
const map = L.map("map").setView([-1.8312, -78.1834], 6); // Coordenadas del Ecuador

// Cargar mapa base desde OpenStreetMap
L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
  attribution: "© OpenStreetMap contributors",
}).addTo(map);
```

```
const provinces = {
  ibarra: { coords: [0.3471, -78.1324], name: "Ibarra" },
  loja: { coords: [-3.9952, -79.2022], name: "Loja" },
  zamora: { coords: [-4.0621, -78.9486], name: "Zamora" },
  lagoAgrio: { coords: [0.1128, -76.9119], name: "Lago Agrío" },
  puyo: { coords: [-1.4929, -77.9998], name: "Puyo" },
  puertoMorona: { coords: [-2.8827, -77.6878], name: "Puerto Morona" },
  santaCruzIsland: { coords: [-0.6394, -90.3372], name: "Santa Cruz Island" },
  quevedo: { coords: [-1.0225, -79.4604], name: "Quevedo" },
  manta: { coords: [-0.9677, -80.7089], name: "Manta" },
  santoDomingo: { coords: [-0.2538, -79.1763], name: "Santo Domingo" },
  cuenca: { coords: [-2.9001, -79.0059], name: "Cuenca" },
  ambato: { coords: [-1.2543, -78.6229], name: "Ambato" },
  quito: { coords: [-0.2233, -78.5141], name: "Quito" },
  machala: { coords: [-3.2581, -79.9554], name: "Machala" },
  guayaquil: { coords: [-2.1891, -79.8899], name: "Guayaquil" },
  esmeraldas: { coords: [0.9706, -79.653], name: "Esmeraldas" },
};

// Devuelve el nombre del mes
function getMonthName(monthIndex) {
  const months = [
    "Enero",
    "Febrero",
    "Marzo",
    "Abril",
    "Mayo",
    "Junio",
    "Julio",
```

```
    "Agosto",
    "Septiembre",
    "Octubre",
    "Noviembre",
    "Diciembre",
  ];
  return months[monthIndex];
}

// Calcula el día ajustado para manejar límites del mes
function getAdjustedDate(day, month, year) {
  const daysInMonth = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];

  // Ajustar para años bisiestos
  if ((year % 4 === 0 && year % 100 !== 0) || year % 400 === 0) {
    daysInMonth[1] = 29;
  }

  // Ajustar día y mes si excede los límites
  if (day > daysInMonth[month]) {
    day = 1;
    month = (month + 1) % 12;
  }

  return { day, month };
}

// Mapeo de íconos para el clima en el frontend
const climateMapping = {
  Clouds: { icon: "☁️" },
```

```
  Rain: { icon: "🌧️" },
  Clear: { icon: "☀️" },
  Haze: { icon: "🌫️" },
  Fog: { icon: "🌁" },
  Drizzle: { icon: "🌦️" },
  Mist: { icon: "🌫️" },
  Thunderstorm: { icon: "⛈️" },
  Smoke: { icon: "💨" },
  Dust: { icon: "🌪️" },
  Ash: { icon: "🌋" },
  Tornado: { icon: "🌪️" },
  Squall: { icon: "💨" },
  Snow: { icon: "❄️" },
};

// Función auxiliar para obtener el ícono y el nombre del clima
function getClimateDescription(wetherId) {
  const climate = climateMapping[wetherId] || { icon: " ❓ " };
  return `${climate.icon} ${wetherId}`;
}

// Genera la tabla HTML para mostrar los datos climáticos
function generateClimateTable(provinceName, day, month, dataPred) {
  const monthName = getMonthName(month);
  return `
    <table style="width:100%" border="0" cellspacing="0" cellpadding="8">
      <tr>
        <th style="width:40%; text-align:left;">${provinceName}</th>
        <th style="width:20%">${day + 1} ${monthName}</th>
```

```
  <th style="width:20%">${day + 2} ${monthName}</th>
  <th style="width:20%">${day + 3} ${monthName}</th>
</tr>
<tr><td style="text-align:left;">Clima</td>
  <td>${getClimateDescription(dataPred["day_1"].wether_id[0])}</td>
  <td>${getClimateDescription(dataPred["day_2"].wether_id[0])}</td>
  <td>${getClimateDescription(dataPred["day_3"].wether_id[0])}</td>
</tr>
<tr><td style="text-align:left;">Temperatura (°C)</td>
  <td>${dataPred["day_1"].temp[0].toFixed(2)}</td>
  <td>${dataPred["day_2"].temp[0].toFixed(2)}</td>
  <td>${dataPred["day_3"].temp[0].toFixed(2)}</td>
</tr>
<tr><td style="text-align:left;">Presión (hPa)</td>
  <td>${dataPred["day_1"].pressure[0].toFixed(2)}</td>
  <td>${dataPred["day_2"].pressure[0].toFixed(2)}</td>
  <td>${dataPred["day_3"].pressure[0].toFixed(2)}</td>
</tr>
<tr><td style="text-align:left;">Humedad (%)</td>
  <td>${dataPred["day_1"].humidity[0].toFixed(2)}</td>
  <td>${dataPred["day_2"].humidity[0].toFixed(2)}</td>
  <td>${dataPred["day_3"].humidity[0].toFixed(2)}</td>
</tr>
<tr><td style="text-align:left;">Velocidad del viento (km/h)</td>
  <td>${dataPred["day_1"].wind_speed[0].toFixed(2)}</td>
  <td>${dataPred["day_2"].wind_speed[0].toFixed(2)}</td>
  <td>${dataPred["day_3"].wind_speed[0].toFixed(2)}</td>
</tr>
<tr><td style="text-align:left;">Grado del viento (°)</td>
  <td>${dataPred["day_1"].wind_deg[0].toFixed(2)}</td>
```

```
      <td>${dataPred["day_2"].wind_deg[0].toFixed(2)}</td>
      <td>${dataPred["day_3"].wind_deg[0].toFixed(2)}</td>
    </tr>
    <tr><td style="text-align:left;">Lluvia 1h (mm)</td>
      <td>${dataPred["day_1"].rain_1h[0].toFixed(2)}</td>
      <td>${dataPred["day_2"].rain_1h[0].toFixed(2)}</td>
      <td>${dataPred["day_3"].rain_1h[0].toFixed(2)}</td>
    </tr>
    <tr><td style="text-align:left;">Nubes (%)</td>
      <td>${dataPred["day_1"].clouds_all[0].toFixed(2)}</td>
      <td>${dataPred["day_2"].clouds_all[0].toFixed(2)}</td>
      <td>${dataPred["day_3"].clouds_all[0].toFixed(2)}</td>
    </tr>
  </table>
 `;
}

// Función para mostrar datos climáticos simulados
async function showClimateData(provinceName, provinceKey) {
 const loadingSpinner = document.getElementById("loading-overlay");
 loadingSpinner.style.display = "block";

 try {
  const dataOW = await fetchData(provinceKey);
  if (!dataOW) throw new Error("No se pudieron obtener datos climáticos.");

  const dataPred = await fetchPrediction(dataOW);
  if (!dataPred)
   throw new Error("No se pudieron obtener datos de predicción.");
```

```
    const date = new Date();
    const { day, month } = getAdjustedDate(
      date.getDate(),
      date.getMonth(),
      date.getFullYear()
    );
    const tableHTML = generateClimateTable(provinceName, day, month, dataPred);

    // Actualiza el DOM con la tabla generada
    document.getElementById("climate-data").innerHTML = tableHTML;

    // Realiza la actualización adicional si es necesario
    updateSelectAndClimate(provinceKey);
  } catch (error) {
    console.error("Error al procesar los datos climáticos:", error);
  } finally {
    // Ocultar el indicador de carga
    loadingSpinner.style.display = "none";
  }
}

function convertFahrenheitToCelsius(temp) {
  return ((temp - 32) * 5) / 9;
}

function formatData(data) {
  const tempData = [];
  const pressureData = [];
  const humidityData = [];
  const wind_speedData = [];
```

```
const wind_degData = [];
const rain_1hData = [];
const clouds_allData = [];

for (let i = 0; i < data.list.length; i++) {
  tempValue = data.list[i].temp.eve
    ? convertFahrenheitToCelsius(data.list[i].temp.eve)
    : 0;
  pressureValue = data.list[i].pressure ? data.list[i].pressure : 0;
  humidityValue = data.list[i].humidity ? data.list[i].humidity : 0;
  wind_speedValue = data.list[i].speed ? data.list[i].speed : 0;
  wind_degValue = data.list[i].deg ? data.list[i].deg : 0;
  rain_1hValue = data.list[i].rain ? data.list[i].rain / 24 : 0;
  clouds_allValue = data.list[i].clouds ? data.list[i].clouds : 0;

  tempData.push(tempValue);
  pressureData.push(pressureValue);
  humidityData.push(humidityValue);
  wind_speedData.push(wind_speedValue);
  wind_degData.push(wind_degValue);
  rain_1hData.push(rain_1hValue);
  clouds_allData.push(clouds_allValue);
}

const formattedData = {
  temp: tempData,
  pressure: pressureData,
  humidity: humidityData,
  wind_speed: wind_speedData,
  wind_deg: wind_degData,
```

```
    rain_1h: rain_1hData,
    clouds_all: clouds_allData,
  };

  return formattedData;
}

// Consulta API OpenWeatherMap
async function fetchData(provinceKey) {
  const lat = provinces[provinceKey].coords[0];
  const lon = provinces[provinceKey].coords[1];
  const apiKey = "cee523f3167b7acd21d1dd24d7353bce";
  const url =
`https://api.openweathermap.org/data/2.5/forecast/daily?lat=${lat}&lon=${lon}&cnt=16&ap
pid=${apiKey}`;

  try {
    const response = await fetch(url);
    const data = await response.json();
    // console.log(data);
    const formattedData = formatData(data);
    return formattedData;
  } catch (error) {
    console.error("Error al obtener datos de OpenWeatherMap:", error);
    return null;
  }
}

// Consulta Backend de Prediction
async function fetchPrediction(dataOW) {
  try {
```

```
    // const response = await fetch("http://localhost:8000/predict", {
    const response = await fetch("https://weather.2x3.app/api/predict", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(dataOW),
    });

    if (!response.ok) {
      throw new Error(`Error en la respuesta del servidor: ${response.status}`);
    }

    const data = await response.json();
    //console.log(data);
    return data;
  } catch (error) {
    console.error("Error al obtener datos de Prediction:", error);
    return null;
  }
}

async function updateSelectAndClimate(provinceKey) {
  const selectElement = document.getElementById("province-select");

  isUpdatingSelect = true;
  selectElement.value = provinceKey;
  isUpdatingSelect = false;
}
```

```
// Añadir eventos al mapa
for (const [key, value] of Object.entries(provinces)) {
  L.marker(value.coords)
    .addTo(map)
    .bindPopup(value.name)
    .on("click", () => showClimateData(value.name, key));
}

// Manejo del selector desplegable
document
  .getElementById("province-select")
  .addEventListener("change", (event) => {
    if (isUpdatingSelect) return;

    const selectedProvince = event.target.value;
    if (selectedProvince && provinces[selectedProvince]) {
      const { coords, name } = provinces[selectedProvince];
      map.flyTo(coords, 8);
      showClimateData(name, selectedProvince);
    }
  });
```

APENDICE D

Repositorio laboratorio y código fuente generado para que el TFM

Enlace OneDrive – UIDE: https://mailinternacionaledu-my.sharepoint.com/:f:/g/personal/pvizcaino_uide_edu_ec/Epa8SwJU_6VPt-Vgsvf0OxMBbv0lk6Kp7-jjRvAmvdOrfA?e=W0lkxr