Facultad Técnica Superior de Ingeniería (Comillas ICAI)

PREDICCIÓN DE VARIABLES MACROECONÓMICAS EN ESPAÑA

Autor: Carlos Gahete Morillo

Director: Atilano Fernández-Pacheco Sánchez-Migallón

71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO

Digitally signed by 71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO
Date: 2023.06.23 23:12:29 +02'00'

Madrid | Junio 2023

Agradecimientos

Agradezco a mis familiares por haberme apoyado siempre en mis estudios en ingeniería y BA, mis amigos por ser un apoyo siempre en los momentos complicados, a mis profesores, tanto por su paciencia y sus enseñanzas durante estos 5 años y a Atilano mi director de TFG por su paciencia, comprensión y ayuda en este proyecto.

Resumen

Este trabajo proporciona una visión general de diversos temas relacionados con el Machine Learning y sus aplicaciones en múltiples ámbitos. Inicia con una documentación sobre la evolución histórica del ML y su estado actual. Se exploran los avances en algoritmos y modelos de ML. Se examina la importancia de la explicabilidad y la interpretabilidad en los modelos de aprendizaje profundo, así como los desarrollos recientes en este campo. Además, se aborda el concepto de aprendizaje federado y su aplicación en el contexto del aprendizaje profundo.

Se desarrolla un modelo de predicción del IPC mediante la comparación de las técnicas de regresión lineal, random forest y K-Nearest Neighbors (KNN).

Además, se desarrolla una aplicación web interactiva utilizando Python y Dash para visualizar y utilizar el modelo de predicción. Se profundiza en las principales políticas monetarias y fiscales, y se proponen una serie de medidas aplicables en caso de que la predicción generada por el modelo indique un valor del IPC superior o inferior al deseado.

Palabras clave

Aprendizaje Automático, Aprendizaje Profundo, Explicabilidad, Aprendizaje Federado, Procesamiento de Lenguaje Natural, Transferencia de Aprendizaje, Predicción Financiera, Framework Dash, Visualización de Datos, Aplicaciones Web Interactivas

Abstract

This work provides a comprehensive overview of various topics related to Machine Learning and its applications across multiple domains. It begins with documenting the historical evolution of ML and its current state. Advances in ML algorithms and models are explored. The importance of explainability and interpretability in deep learning models, along with recent developments in this field, are examined. Additionally, the concept of federated learning and its application in the context of deep learning is addressed.

A prediction model for the Consumer Price Index (CPI) is developed by comparing the techniques of linear regression, random forest, and K-Nearest Neighbors (KNN). Furthermore, an interactive web application is developed using Python and Dash to visualize and utilize the prediction model. The work delves into the primary monetary and fiscal policies, proposing a series of applicable measures in case the model's prediction indicates a CPI value higher or lower than the desired target.

Key words

Machine Learning, Deep Learning, Explainability, Federated Learning, Natural Language Processing (NLP), Transfer Learning, Financial Prediction, Dash Framework, Data Visualization, Interactive Web Applications

Tabla de contenido

1. Introducción 1

1.1. Propósito General 1

1.2. Objetivos del trabajo 1

1.3. Metodología 2

1.4. Desarrollo 3

2. Estado del arte 3

2.1. Machine Learning 3

2.1.1. Regresión lineal 6

2.1.2. Algoritmo k-NN 7

2.1.3. Redes neuronales 9

2.1.4. Random forest 11

2.2. Python 12

2.2.1. Scikit-learn 14

2.2.2. NumPy 14

2.3. Web 14

2.3.1. Dash 14

3. Evolución del Machine Learning 15

3.1. Orígenes del Machine Learning 16

3.2. Avances en algoritmos y modelos de ML 16

3.3. Aplicaciones destacadas del ML a lo largo de la historia 17

3.3.1. Reconocimiento de voz 17

3.3.2. Traducción automática. 18

3.3.3. ML en finanzas. 19

3.3.4. Avances en la medicina y salud gracias a ML. 19

3.4. Tendencias y desarrollos recientes en el ML, Deep Learning 21

3.4.1. Explicabilidad e interpretabilidad. 22

3.4.2. Federated Learning. 23

4. Modelo de ML para la predicción de variables macroeconómicas 23

4.1. Descripción del problema de predicción. 23

4.2. Obtención y procesado de los datos. 24

4.3. Elección del modelo de ML. 28

4.3.1. Elección del modelo para este caso de estudio. 29

4.3.1.1. Regresión lineal 29

4.3.1.2. Random Forest 30

4.3.1.3. KNN 33

4.4. Análisis y evaluación de los resultados obtenidos. 35

5. Aplicación web con Dash. 38

5.1. Introducción. 38

5.2. Desarrollo de la aplicación. 40

6. Aplicaciones de políticas económicas basada en la predicción de movimientos. 42

6.1. Marco teórico de las políticas aplicables. 42

6.1.1. Políticas monetarias. 42

6.1.1.1. Políticas monetarias expansivas. 42

6.1.1.2. Políticas monetarias restrictivas. 43

6.1.2. Políticas fiscales. 44

6.1.2.1. Políticas fiscales expansivas. 44

6.1.2.2. Políticas fiscales restrictivas. 45

6.2. Implementación de las políticas basándonos en la predicción del IPC. 46

6.2.1. IPC predicho mayor del deseado 46

6.2.2. IPC predicho menor del deseado 47

6.3. Importancia de aplicar de estas medidas a tiempo. 49

7. Concordancia de este trabajo con los Objetivos de Desarrollo Sostenible. 49

8. Bibliografía 50

9. Anexo 1: Código modelo Dash. 53

10. Anexo 2: Código Python. 56

Índice de ilustraciones

Ilustración 1: b2301 24
Ilustración 2: si_1_1 25
Ilustración 3: be23a 26
Ilustración 4: be1091 26
Ilustración 5: tablaModelo 28
Ilustración 6: Resultados train modelo de regresión 29
Ilustración 7: Resultados test modelo de regresión 30
Ilustración 8: Tabla de correlaciones 31
Ilustración 9: Resultados train Random Forest 32
Ilustración 10: Resultados test Random Forest 33
Ilustración 11: Gráfico del codo KNN 34
Ilustración 12: Resultados train KNN 35
Ilustración 13: Resultados test KNN 35
Ilustración 14: Gráfico MSE Train y Test ambos modelos 36
Ilustración 15: Gráfico resultados train de todos los modelos 37
Ilustración 16: Gráfico resultados test todos los modelos 37
Ilustración 17: Pantalla nada más acceder al enlace DASH 39
Ilustración 18: Pantalla visible al bajar al máximo sin introducir nada DASH 39
Ilustración 19: Pantalla visible al introducir los datos necesarios para realizar la predicción DASH 40

Listado de abreviaturas

IA – Inteligencia Artificial

ML – Machine Learning

RF – Random Forest

MSE – Error medio al cuadrado

KNN – K nearest neighbors

1. Introducción

1.1. Propósito General

Las variables macroeconómicas aportan mucha información a la hora tomar unas políticas u otras. Muchas veces estas políticas son poco optimas o tienen cambios muy bruscos debido a que la gente responsable de aplicarlas no es capaz de adelantarse a los cambios en la economía que estas variables macroeconómicas reflejan. Si fuese posible de alguna manera prever con la suficiente confianza los cambios que estas variables y por tanto la economía va a sufrir se podrían optimizar mucho estas políticas y conseguir mejores resultados en menos tiempo y con menor coste. Por tanto, el propósito de este trabajo pasa por construir un modelo que sea capaz de predecir variables macroeconómicas con la suficiente confianza como para que la información que arroje sea útil, y que este modelo sea accesible mediante una web para que terceros puedan hacer uso de esta información.

1.2. Objetivos del trabajo

El objetivo fundamental del trabajo explicar cómo ha evolucionado el Machine Learning a lo largo de la historia, cuáles han sido sus utilidades más destacadas, intentar aplicar estos conocimientos para desarrollar un modelo de predicción de variables macroeconómicas y la creación de una aplicación web para que terceros puedan beneficiarse de este trabajo.

Se abren diferentes objetivos específicos para abordar este trabajo.:

1.Comprender la evolución del Machine Learning y sus utilidades a lo largo de la historia.

2.Comprender como funcionan los diferentes modelos de machine learning.

3.Comprender que variables macroeconómicas es posible predecir a futuro y valorar que modelo de machine learning es el que mejor se ajusta a nuestro problema teniendo en cuenta la confianza de los resultados y la complejidad del modelo y la creación de una pequeña aplicación web en la que terceros puedan hacer uso del modelo.

1.3. Metodología

Se va a usar una metodología en cascada mezclando con ciertas ceremonias agiles.

Modelo en cascada: se completa una etapa antes de comenzar la siguiente.

Procesos ágiles: la planificación es incremental, se realizan varias actividades en paralelo y es más fácil modificar el proceso para reflejar las necesidades cambiantes de los clientes.

La ceremonia ágil que vamos a usar es sprint.

Sprint:

La división de los features que se escogen en el backlog y que se van a realizar en un determinado periodo de tiempo se denomina sprint.

Las historias de usuario que componen en un sprint se denomina sprint backlog.

El sprint durara entre 2 o 3 semanas.

Al finalizar cada sprint se presentarán los resultados obtenidos. Durante la duración del sprint no se introducirá ninguna tarea.

Debe existir una reunión al principio del sprint para organizarse.

Diariamente se realizará una daily para detectar bloqueo y planificarse la jornada de trabajo.

Al finalizar el sprint se hará el sprint review.

Consta de 5 fases:

Análisis y diseño: mes de septiembre

Desarrollo: desde octubre hasta abril

Pruebas: desde abril hasta junio

Subida a producción: junio

Documentación (durante todo el proyecto)

1.4. Desarrollo

El trabajo se puede dividir en 3 desarrollos principales:

Para empezar, el primer desarrollo tratara de documentar la evolución del Machine Learning a lo largo de la historia y de su estado actualmente.

El segundo desarrollo, trata de crear un modelo de machine learning haciendo uso de Python que permita predecir variables macro y la creación de una web en la que terceros puedan hacer uso del modelo.

El ultimo desarrollo, se basará en aplicación de ciertas políticas que podrían ser aplicadas en caso de conseguir un modelo que nos permita anticiparnos a estos movimientos.

2. Estado del arte

2.1. Machine Learning

El aprendizaje automático es una disciplina que forma parte de la inteligencia artificial y se centra en el desarrollo de algoritmos y modelos que permiten a las máquinas aprender a partir de los datos y mejorar su rendimiento de manera automática. En lugar de programar explícitamente todas las reglas y condiciones, el aprendizaje automático permite que los sistemas analicen los datos y encuentren patrones y relaciones por sí mismos.

En el proceso de aprendizaje automático, los datos desempeñan un papel fundamental. Estos datos se dividen en dos tipos principales: los atributos de entrada y el objetivo o target. Los atributos de entrada son las características o variables que se utilizan para realizar predicciones o clasificaciones, mientras que el objetivo es el resultado deseado que se quiere predecir. Por ejemplo, si deseamos predecir si un correo electrónico es spam o no, los atributos de entrada podrían ser el contenido del correo, el remitente, etc., y el objetivo sería la clasificación de spam o no spam.

Existen diferentes enfoques en el aprendizaje automático, y dos de los más comunes son el aprendizaje no supervisado y el aprendizaje supervisado.

El aprendizaje no supervisado implica el procesamiento de los datos de entrada sin tener conocimiento previo del etiquetado o la salida esperada. En este enfoque, el

algoritmo busca patrones, estructuras o relaciones interesantes en los datos por sí mismo. Puede agrupar los datos en diferentes categorías o encontrar características comunes entre ellos sin recibir instrucciones específicas sobre qué buscar. Esto es útil para descubrir información oculta, realizar segmentación de datos o resumir conjuntos de datos complejos.

Por otro lado, en el aprendizaje supervisado, los algoritmos tienen conocimiento tanto de los datos de entrada como de los resultados esperados. Se divide en dos fases principales: entrenamiento y prueba. Durante la fase de entrenamiento, el modelo se expone a un conjunto de datos de entrenamiento que contiene tanto los atributos de entrada como los objetivos correspondientes. El modelo ajusta sus parámetros y realiza predicciones sobre los datos de entrenamiento. Luego, se compara el resultado predicho con el objetivo conocido, y el modelo actualiza sus parámetros en función del error cometido. Este proceso se repite varias veces, en ciclos o épocas, hasta que el modelo logra minimizar el error y mejorar su rendimiento.

Una vez entrenado, el modelo se somete a la fase de prueba, donde se evalúa su capacidad para hacer predicciones precisas sobre nuevos datos de entrada que no ha visto antes. Se pueden utilizar diversas métricas y técnicas de evaluación para medir el rendimiento del modelo, como la matriz de confusión en problemas de clasificación, que muestra la cantidad de aciertos y errores en las predicciones en cada categoría.

En resumen, el aprendizaje automático utiliza métodos algorítmicos para modelar y descubrir patrones en los datos. El aprendizaje no supervisado permite al sistema aprender y relacionar los datos por sí mismo, mientras que el aprendizaje supervisado utiliza datos de entrada y salida conocidos para entrenar y mejorar el rendimiento del modelo. Ambos enfoques son útiles en diferentes escenarios y aplicaciones, y el rendimiento del modelo se puede evaluar utilizando diversas técnicas y métricas para medir su eficacia.

Podemos entrar también en el detalle de las diferencias entre algoritmos de regresión y algoritmos de clasificación.

Algoritmos de regresión:

Los algoritmos de regresión se emplean cuando el objetivo es predecir un valor numérico continúo basado en un conjunto de variables de entrada. El propósito principal de la regresión es modelar la relación entre las variables independientes y la variable dependiente continua. Algunos ejemplos comunes de algoritmos de regresión incluyen la regresión lineal, la regresión logística y la regresión polinómica.

Características clave de los algoritmos de regresión:

Objetivo: Predecir un valor numérico continuo.

Salida: El resultado es un número continuo dentro de un rango determinado.

Tipos de variables: Las variables de entrada pueden ser tanto continuas como categóricas.

Métricas de evaluación: Se utilizan métricas como el error cuadrático medio (MSE), el coeficiente de determinación (R^2) y el error absoluto medio (MAE) para evaluar el rendimiento del modelo.

- Ejemplos de aplicaciones: Predicción de precios de viviendas, pronóstico de ventas, estimación de ingresos.

Algoritmos de clasificación:

Los algoritmos de clasificación se emplean cuando el objetivo es asignar una etiqueta o clase a una instancia o conjunto de datos. La clasificación se utiliza para dividir los datos en categorías predefinidas en función de las características de entrada. Los algoritmos de clasificación son fundamentales en muchas aplicaciones de aprendizaje automático y se utilizan ampliamente en campos como la detección de spam, el diagnóstico médico y la clasificación de imágenes.

Algunas características clave de los algoritmos de clasificación son:

Objetivo: Asignar una etiqueta o clase a los datos de entrada.

Salida: El resultado es una clase o etiqueta que representa la categoría a la que pertenece la instancia.

Tipos de variables: Las variables de entrada pueden ser continuas o categóricas, pero la variable objetivo es categórica.

Métricas de evaluación: Se utilizan métricas como la precisión, el puntaje F1, la matriz de confusión y la curva ROC para evaluar el rendimiento del modelo.

Ejemplos de aplicaciones: Detección de spam, diagnóstico médico, clasificación de imágenes.

En resumen, los algoritmos de regresión y los algoritmos de clasificación son dos enfoques fundamentales en el campo del aprendizaje automático. Mientras que los algoritmos de regresión se utilizan para predecir valores numéricos continuos, los algoritmos de clasificación se emplean para asignar etiquetas o clases a los datos de entrada. Estas diferencias se reflejan en los objetivos, salidas, tipos de variables, métricas de evaluación y aplicaciones prácticas asociadas con cada tipo de algoritmo. La elección adecuada entre algoritmos de regresión y algoritmos de clasificación depende de la naturaleza del problema y los datos disponibles.

(IAAR, n.d.)

2.1.1. Regresión lineal

La regresión lineal es una técnica que se utiliza para modelar la relación entre una variable dependiente continua y una o más variables independientes. Su objetivo es encontrar la mejor línea recta que se ajuste a los datos disponibles, de tal manera que pueda predecir los valores de la variable dependiente para nuevos conjuntos de valores de las variables independientes.

El modelo de regresión lineal se basa en la suposición de que existe una relación lineal aproximada entre las variables predictoras y la variable de respuesta. Esta relación se puede expresar mediante una ecuación lineal de la forma:

$$Y = \beta0 + \beta1*X1 + \beta2*X2 + ... + \beta n*Xn + \varepsilon$$

Donde:

- Y es la variable dependiente que queremos predecir.

- $\beta0$ es el intercepto o término constante.

- $\beta1$, $\beta2$, ..., βn son los coeficientes de regresión que representan la influencia de las variables independientes X1, X2, ..., Xn en la variable dependiente Y.

- X1, X2, ..., Xn son las variables independientes.

- ε es el término de error, que representa las diferencias entre los valores predichos y los valores reales de Y que no se pueden explicar por las variables independientes.

El objetivo de la regresión lineal es encontrar los valores óptimos de los coeficientes de regresión (β0, β1, ..., βn) que minimicen la suma de los cuadrados de los residuos (SSR), es decir, las diferencias entre los valores observados de Y y los valores predichos por el modelo.

Para estimar los coeficientes de regresión, se utilizan diversos métodos, como el método de mínimos cuadrados ordinarios (MCO). Este método encuentra los coeficientes que minimizan la suma de los cuadrados de las diferencias entre los valores observados y los valores predichos.

Una vez estimados los coeficientes, se puede utilizar el modelo de regresión lineal para realizar predicciones sobre nuevos conjuntos de valores de las variables independientes. Esto permite estimar el valor esperado de la variable dependiente para diferentes combinaciones de las variables predictoras.

Es importante evaluar la calidad del ajuste del modelo de regresión lineal. Para ello, se utilizan diversas métricas, como el coeficiente de determinación (R^2), que indica la proporción de la variabilidad total de la variable dependiente que es explicada por el modelo. Valores más cercanos a 1 indican un mejor ajuste del modelo.

En conclusión, la regresión lineal es una técnica estadística fundamental para modelar la relación entre variables y realizar predicciones. Permite estimar el valor esperado de una variable dependiente en función de una o más variables independientes. Además, ofrece herramientas para evaluar la calidad del ajuste del modelo y determinar la influencia relativa de las variables predictoras en la variable dependiente.

(Gonzalez, 2018)

2.1.2. Algoritmo k-NN

El algoritmo k-Nearest Neighbors (k-NN) es un algoritmo de aprendizaje supervisado utilizado en el campo del aprendizaje automático para clasificar

instancias o predecir valores basándose en la similitud con instancias cercanas en el espacio de características. A diferencia de otros algoritmos, como la regresión lineal o los árboles de decisión, el k-NN no genera un modelo explícito, sino que se basa en la información de los datos de entrenamiento para realizar predicciones.

El funcionamiento básico del algoritmo k-NN es el siguiente:

1. Preprocesamiento de los datos: Antes de utilizar el algoritmo k-NN, es necesario realizar un preprocesamiento de los datos de entrenamiento. Esto puede incluir la normalización de los atributos, la eliminación de valores atípicos o la selección de características relevantes.

2. Cálculo de la distancia: El primer paso en el algoritmo k-NN es calcular la distancia entre la instancia que se quiere clasificar o predecir y todas las instancias de entrenamiento. La distancia más comúnmente utilizada es la distancia euclidiana, pero también se pueden utilizar otras métricas de distancia, como la distancia de Manhattan o la distancia de Minkowski.

3. Selección de los k vecinos más cercanos: Una vez calculadas las distancias, se seleccionan los k vecinos más cercanos a la instancia de interés. El valor de k es un parámetro del algoritmo y debe elegirse de antemano. Cuanto mayor sea el valor de k, más suave será la frontera de decisión, pero también puede haber una mayor interferencia de instancias de otras clases.

4. Votación y clasificación: Una vez seleccionados los k vecinos más cercanos, se utiliza su información para realizar la clasificación. En el caso de la clasificación, se realiza una votación para determinar la clase más frecuente entre los vecinos. La instancia se clasifica como perteneciente a la clase con mayor número de votos. En el caso de la regresión, se realiza un promedio de los valores objetivo de los vecinos para predecir el valor numérico.

Es importante tener en cuenta que el algoritmo k-NN no tiene una fase de entrenamiento explícita. En su lugar, todo el conjunto de datos de entrenamiento se utiliza directamente en el proceso de clasificación o predicción. Esto puede hacer que el tiempo de clasificación o predicción sea más lento en comparación con otros algoritmos que generan modelos.

El algoritmo k-NN tiene algunas consideraciones importantes a tener en cuenta:

Elección de k: El valor de k puede tener un impacto significativo en el rendimiento del algoritmo. Valores demasiado pequeños pueden hacer que el modelo sea sensible a ruido o valores atípicos, mientras que valores demasiado grandes pueden suavizar demasiado las fronteras de decisión.

Normalización de atributos: Debido a que el algoritmo k-NN se basa en la distancia entre instancias, es importante normalizar los atributos para que tengan un rango y una escala similares. De lo contrario, los atributos con rangos más grandes pueden dominar la contribución a la distancia.

- Sensibilidad a la dimensionalidad: El rendimiento del algoritmo k-NN puede verse afectado negativamente por la maldición de la dimensionalidad. A medida que aumenta la dimensionalidad de los datos, el espacio de características se vuelve más disperso y la noción de cercanía se vuelve menos significativa.

En resumen, el algoritmo k-Nearest Neighbors es un método simple pero poderoso para la clasificación y predicción basado en la similitud de las instancias vecinas. Es ampliamente utilizado en diferentes dominios y su rendimiento depende de la elección adecuada de k, el preprocesamiento de datos y la consideración de la dimensionalidad.

(IBM, ¿Qué es el algoritmo de k vecinos más cercanos?, n.d.)

2.1.3. Redes neuronales

Las redes neuronales, son modelos computacionales inspirados en el funcionamiento del cerebro humano. Estas redes están compuestas por un conjunto de unidades de procesamiento llamadas neuronas, que están interconectadas y organizadas en capas.

Cada neurona en una red neuronal procesa información y realiza una operación matemática para producir una salida. Estas salidas se transmiten a otras neuronas a través de conexiones ponderadas, que representan la fuerza o importancia de la conexión entre las neuronas. La información fluye a través de la red neuronal desde

la capa de entrada, donde se introducen los datos, hacia la capa de salida, donde se obtiene el resultado final.

Una característica fundamental de las redes neuronales es su capacidad para aprender y adaptarse a partir de ejemplos o datos de entrenamiento. Esto se logra mediante un proceso llamado entrenamiento, en el cual la red ajusta los valores de las conexiones ponderadas para minimizar la diferencia entre las salidas predichas y las salidas deseadas.

Existen varios tipos de redes neuronales, siendo las más comunes las redes neuronales feedforward y las redes neuronales recurrentes. Las redes feedforward se caracterizan por tener una dirección única del flujo de información, es decir, la información se propaga desde la capa de entrada hasta la capa de salida sin retroalimentación. Estas redes son ampliamente utilizadas en tareas de clasificación y regresión.

Por otro lado, las redes neuronales recurrentes (RNN) contienen conexiones retroalimentadas que permiten que la información fluya en bucles a través de la red. Esto las hace adecuadas para tareas que involucran secuencias de datos, como el procesamiento de lenguaje natural y el reconocimiento de voz.

El entrenamiento de una red neuronal implica alimentar los datos de entrenamiento a la red y ajustar los pesos de las conexiones mediante un algoritmo de optimización, como el descenso del gradiente. Este proceso busca minimizar una función de pérdida que mide la discrepancia entre las salidas predichas y las salidas deseadas.

Una vez entrenada, la red neuronal puede realizar predicciones o clasificar nuevas entradas que no se utilizaron durante el entrenamiento. Este proceso se conoce como inferencia y aprovecha los patrones aprendidos durante el entrenamiento para generar resultados.

Las redes neuronales han demostrado ser muy eficaces en una amplia gama de aplicaciones, como reconocimiento de imágenes, procesamiento de texto, conducción autónoma, diagnóstico médico y muchas otras áreas donde el modelado de patrones complejos es crucial.

Las redes neuronales son modelos computacionales inspirados en el cerebro humano que permiten el aprendizaje y la adaptación a partir de datos de

entrenamiento. Son capaces de resolver problemas complejos y han demostrado un gran éxito en diversas aplicaciones. Su flexibilidad y capacidad para modelar relaciones no lineales las convierten en una herramienta poderosa en el campo del aprendizaje automático y la inteligencia artificial.

(IBM, ¿Qué son las redes neuronales?, n.d.)

2.1.4. Random forest

Random Forest es un algoritmo de aprendizaje automático que se basa en la combinación de múltiples árboles de decisión para realizar tareas de clasificación y regresión. Es considerado uno de los algoritmos más populares y poderosos en el campo del aprendizaje automático debido a su capacidad para manejar conjuntos de datos grandes y complejos, y su capacidad para evitar el sobreajuste.

El funcionamiento del algoritmo Random Forest se puede resumir de la siguiente manera:

1. Construcción de los árboles de decisión: Random Forest crea un conjunto de árboles de decisión independientes entre sí. Cada árbol se construye utilizando un subconjunto aleatorio de los datos de entrenamiento y un subconjunto aleatorio de las características (variables predictoras). Estos subconjuntos se generan mediante un proceso conocido como remuestreo con reemplazo.

2. Entrenamiento de los árboles de decisión: Cada árbol se entrena utilizando el subconjunto de datos y características correspondientes. Durante el entrenamiento, cada árbol realiza divisiones en los datos de acuerdo con una regla que maximiza la homogeneidad dentro de los nodos resultantes. Este proceso se repite recursivamente hasta que se alcanza un criterio de parada, como un número mínimo de instancias en un nodo o una profundidad máxima del árbol.

3. Toma de decisiones por votación: Una vez que se han construido todos los árboles, Random Forest realiza la clasificación (o predicción en el caso de regresión) tomando una votación entre los árboles. Cada árbol emite una clasificación o predicción, y la clase o valor con mayor cantidad de votos se selecciona como el resultado final.

La principal fortaleza de Random Forest radica en su capacidad para reducir el sobreajuste. Al utilizar múltiples árboles construidos con diferentes subconjuntos de datos y características, se reduce la dependencia de cualquier árbol individual y se promueve una mayor diversidad en el conjunto de clasificadores. Esto ayuda a mejorar la generalización y a reducir el impacto de datos ruidosos o atípicos.

Otras ventajas de Random Forest incluyen su capacidad para manejar tanto variables categóricas como numéricas, su eficiencia en el manejo de grandes conjuntos de datos y su capacidad para proporcionar estimaciones de la importancia de las características, lo que permite evaluar la relevancia de cada variable en el proceso de toma de decisiones.

En resumen, Random Forest es un algoritmo de aprendizaje automático que combina múltiples árboles de decisión para realizar tareas de clasificación y regresión. Su capacidad para reducir el sobreajuste, su flexibilidad en el manejo de diferentes tipos de variables y su capacidad para proporcionar estimaciones de importancia de características lo convierten en una herramienta poderosa en el campo del aprendizaje automático.

(IBM, What is random forest?, n.d.)

2.2. Python

Python es un lenguaje de programación de alto nivel, interpretado.

Es ampliamente utilizado en diversos campos, como desarrollo web, ciencia de datos, inteligencia artificial, automatización de tareas, análisis de datos y más. Python se destaca por su legibilidad y su enfoque en la simplicidad y la eficiencia del código.

Algunas características y ventajas clave de Python son:

Sintaxis clara y legible, Python se destaca por su sintaxis clara y legible. Su diseño se enfoca en el uso de un código limpio y conciso, lo que facilita la comprensión y el mantenimiento del código a lo largo del tiempo. El uso de indentación (espacios en blanco) para delimitar bloques de código también fomenta la escritura estructurada y legible.

Amplia gama de bibliotecas y marcos de trabajo, Python cuenta con una amplia gama de bibliotecas y marcos de trabajo que cubren una amplia variedad de

aplicaciones. Por ejemplo, Django y Flask son populares marcos de trabajo para el desarrollo web, mientras que NumPy, Pandas y Matplotlib son bibliotecas esenciales para el procesamiento y análisis de datos. Además, existen bibliotecas especializadas para tareas como aprendizaje automático (scikit-learn, TensorFlow), procesamiento de imágenes (OpenCV), procesamiento de texto (NLTK) y mucho más.

Multiplataforma, es compatible con múltiples plataformas, lo que significa que puede ejecutarse en diferentes sistemas operativos, como Windows, macOS y Linux. Esto facilita el desarrollo de aplicaciones que pueden ejecutarse en diferentes entornos sin tener que realizar cambios significativos en el código fuente.

Facilidad de integración, se puede integrar fácilmente con otros lenguajes de programación, lo que permite aprovechar las fortalezas de diferentes tecnologías. Por ejemplo, es posible utilizar extensiones de C/C++ para mejorar el rendimiento de partes críticas del código, o interactuar con bibliotecas escritas en otros lenguajes a través de interfaces y herramientas de interoperabilidad.

Gran comunidad y soporte, cuenta con una comunidad activa y una amplia base de usuarios. Esto significa que hay abundante documentación, tutoriales, recursos en línea y foros de discusión donde los desarrolladores pueden encontrar respuestas a sus preguntas y recibir apoyo en su aprendizaje y desarrollo de proyectos.

Versatilidad y flexibilidad, es un lenguaje versátil y flexible, lo que significa que se puede utilizar en una amplia variedad de aplicaciones y contextos. Puede ser utilizado tanto para proyectos pequeños como para aplicaciones de gran escala. Además, Python admite varios paradigmas de programación, incluyendo programación orientada a objetos, programación imperativa y programación funcional, lo que permite adaptarse a diferentes estilos de programación y necesidades del proyecto.

Fácil prototipado y desarrollo rápido, La sintaxis simple y clara, junto con su amplia gama de bibliotecas y herramientas, lo convierten en una excelente opción para el prototipado rápido y el desarrollo ágil. Python permite desarrollar y probar ideas rápidamente, lo que es especialmente útil en entornos de desarrollo iterativos donde es necesario iterar y ajustar constantemente el código.

En conclusión, Python es un lenguaje de programación versátil, fácil de aprender y ampliamente utilizado en diversos campos. Su sintaxis clara, su amplia

gama de bibliotecas y marcos de trabajo, su multiplataformidad y su comunidad activa lo convierten en una excelente opción para desarrollar una amplia variedad de aplicaciones, desde pequeños scripts hasta proyectos de ciencia de datos y desarrollo web de gran escala.

2.2.1. Scikit-learn

La librería Scikit-learn (también conocida como sklearn) es una de las bibliotecas más populares y ampliamente utilizadas en el ámbito del aprendizaje automático (machine learning) en el lenguaje de programación Python. Scikit-learn proporciona un conjunto completo de herramientas y algoritmos para realizar tareas de aprendizaje supervisado y no supervisado, así como para evaluación de modelos y preprocesamiento de datos.

2.2.2. NumPy

NumPy (Numerical Python) es una biblioteca fundamental en el ecosistema de Python para el procesamiento numérico y científico de datos. Proporciona una estructura de datos eficiente para trabajar con matrices multidimensionales, junto con una amplia colección de funciones matemáticas de alto rendimiento para realizar operaciones numéricas complejas.

2.3. Web

2.3.1. Dash

Dash es un framework de desarrollo de aplicaciones web en Python que permite crear interfaces interactivas y personalizadas de forma sencilla. Basado en Flask, Plotly.js y React.js, Dash proporciona una solución flexible y eficiente para construir aplicaciones web enfocadas en la visualización de datos, análisis y creación de paneles interactivos.

Con Dash, es posible crear aplicaciones web con una amplia variedad de componentes interactivos, como gráficos, tablas, controles deslizantes y botones, que responden a las interacciones del usuario. Estos componentes pueden ser personalizados en términos de apariencia y comportamiento para adaptarse a las necesidades específicas del proyecto. Además, es posible agregar lógica de

programación para realizar cálculos, consultas a bases de datos u otras operaciones complejas.

El proceso de desarrollo de una aplicación web con Dash implica importar las bibliotecas requeridas, configurar la estructura de la aplicación mediante componentes HTML y Dash, definir la lógica de la aplicación mediante la creación de funciones de callback y finalmente, ejecutar la aplicación en un servidor local para visualizarla en un navegador web.

Dash ofrece una amplia gama de componentes predefinidos y personalizables que permiten crear interfaces de usuario atractivas y funcionales. Entre las opciones disponibles se encuentran gráficos interactivos, filtros dinámicos, tablas ordenables, mapas interactivos y muchos otros elementos que contribuyen a una experiencia de usuario enriquecedora.

Además de su flexibilidad, Dash es altamente personalizable y extensible. Permite utilizar hojas de estilo CSS para aplicar estilos personalizados a los componentes y adaptar la apariencia de la aplicación según los requerimientos específicos. También es posible integrar bibliotecas de Python, como Pandas y NumPy, para realizar análisis de datos complejos y mostrar los resultados en tiempo real.

En resumen, Dash se posiciona como una herramienta poderosa para el desarrollo de aplicaciones web interactivas en Python. Su combinación de bibliotecas y enfoque en la simplicidad y personalización brindan la capacidad de crear visualizaciones y paneles interactivos de manera eficiente y efectiva, lo que lo convierte en una opción atractiva para proyectos que requieran una interfaz web dinámica y atractiva.

(Kilcommins, 2021)

3. Evolución del Machine Learning

En este apartado trataremos de comprender como ha evolucionado el ámbito del ML a lo largo de la historia.

3.1. Orígenes del Machine Learning

El campo del Machine Learning se ha desarrollado a lo largo de décadas, con contribuciones clave de destacados investigadores en inteligencia artificial y teoría de la computación. En la década de 1950, Alan Turing, uno de los pioneros de la ciencia de la computación, presentó su famoso artículo "Computing Machinery and Intelligence". En este trabajo, Turing propuso la idea de las "máquinas de Turing" como modelos matemáticos capaces de simular el pensamiento humano.

Posteriormente, Arthur Samuel llevó el aprendizaje automático un paso más allá al desarrollar un programa que aprendía a jugar al juego de damas. Su trabajo, publicado en 1959 bajo el título "Some Studies in Machine Learning Using the Game of Checkers", sentó las bases para el enfoque del aprendizaje automático mediante la adaptación de parámetros a partir de datos.

En la misma época, Frank Rosenblatt presentó el Perceptrón, un modelo de red neuronal artificial, en su artículo "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". El Perceptrón se inspiró en el funcionamiento del cerebro humano y fue uno de los primeros intentos de simular el aprendizaje mediante conexiones ponderadas.

Además de estos trabajos influyentes, el libro "Machine Learning" de Tom M. Mitchell, publicado en 1997, es una referencia clásica en el campo. Mitchell proporciona una visión integral de los conceptos, algoritmos y aplicaciones del aprendizaje automático, y su trabajo ha sido una guía invaluable para comprender los fundamentos teóricos y prácticos de este campo en constante evolución.

3.2. Avances en algoritmos y modelos de ML

A lo largo de los años, el campo del Machine Learning ha experimentado avances notables en cuanto a algoritmos y modelos. Estos avances han permitido el desarrollo de sistemas más sofisticados y eficientes en diversas aplicaciones.

Uno de los logros más destacados ha sido el desarrollo de las redes neuronales artificiales y el enfoque del aprendizaje profundo. Este enfoque, inspirado en el funcionamiento del cerebro humano, ha tenido un impacto significativo en el campo del Machine Learning. Diversos investigadores, como Geoffrey Hinton, Yann LeCun y Yoshua Bengio, han realizado contribuciones importantes al desarrollo de

arquitecturas de redes neuronales profundas, como las redes convolucionales y recurrentes.

Además, se ha progresado en la aplicación de técnicas de aprendizaje no supervisado, que permiten a los modelos descubrir patrones y estructuras ocultas en los datos sin la necesidad de etiquetas de clase. En este sentido, se destacan los avances en las Redes Generativas Adversariales (GAN), una técnica desarrollada por Ian Goodfellow. Las GAN han demostrado una capacidad excepcional para generar muestras realistas de datos y han encontrado aplicaciones en áreas como la generación de imágenes y la síntesis de voz.

En el ámbito de la clasificación y el reconocimiento de patrones, se han realizado mejoras significativas en los algoritmos de aprendizaje supervisado. Por ejemplo, los métodos de ensamblaje, como el Bosque Aleatorio introducido por Leo Breiman, han demostrado ser eficaces para mejorar la precisión y la robustez de los modelos predictivos al combinar múltiples clasificadores más débiles.

El desarrollo de técnicas de aprendizaje por refuerzo ha abierto nuevas posibilidades en la interacción de los sistemas de Machine Learning con su entorno. Investigadores destacados en este campo, como Richard Sutton y Andrew Barto, han sentado bases teóricas sólidas y han propuesto algoritmos eficientes para enseñar a los agentes a tomar decisiones óptimas en entornos dinámicos y desconocidos.

3.3. Aplicaciones destacadas del ML a lo largo de la historia

3.3.1. Reconocimiento de voz

El reconocimiento por voz es una aplicación destacada del Machine Learning que ha experimentado avances significativos en los últimos años. Este campo se centra en desarrollar algoritmos y modelos capaces de interpretar y comprender el habla humana. A través del uso de técnicas de aprendizaje automático, se han logrado avances en la precisión y la eficiencia de los sistemas de reconocimiento de voz.

Investigadores como Geoffrey Hinton, uno de los pioneros en el campo del aprendizaje profundo, han realizado contribuciones clave en este ámbito. Sus investigaciones han permitido el desarrollo de arquitecturas de redes neuronales profundas, como las redes neuronales convolucionales y recurrentes, que han mejorado significativamente la capacidad de los sistemas de reconocimiento por voz para comprender el habla humana.

Según estudios científicos (Unesco, 2021) se ha demostrado que los modelos de reconocimiento de voz basados en Machine Learning han logrado tasas de precisión superiores al 90% en diversas tareas, como el reconocimiento de palabras y la transcripción de voz. Estos resultados respaldan la efectividad y el potencial de esta tecnología en aplicaciones prácticas.

Además, el reconocimiento por voz ha encontrado aplicaciones en diversos campos, como la asistencia virtual, los sistemas de navegación basados en voz y la transcripción automática. Estas aplicaciones mejoran la accesibilidad y la interacción humano-máquina en diferentes entornos, desde dispositivos móviles hasta sistemas de control de voz en el automóvil.

(Calderon, 2019)

3.3.2. Traducción automática.

La traducción automática es una aplicación fundamental del Machine Learning, ha experimentado avances significativos en las últimas décadas. Este campo se enfoca en desarrollar algoritmos y modelos capaces de traducir texto o voz de un idioma a otro de manera automática y precisa. A través del uso de técnicas de aprendizaje automático, se ha logrado mejorar la calidad y la fluidez de las traducciones automáticas.

Investigadores destacados en este campo, como Yoshua Bengio, Geoffrey Hinton y Yann LeCun, han realizado importantes contribuciones en el desarrollo de arquitecturas de redes neuronales profundas utilizadas en la traducción automática. Sus investigaciones han impulsado la adopción de modelos de traducción basados en redes neuronales, como los modelos de traducción neuronal, que han demostrado resultados prometedores en términos de precisión y naturalidad en las traducciones.

Se han encontrado aplicaciones en diversos campos, como el comercio electrónico, la comunicación internacional y la accesibilidad en línea. Estas aplicaciones mejoran la comunicación intercultural y facilitan el acceso a información en diferentes idiomas.

(Josende, 2022)

3.3.3. ML en finanzas.

La aplicación del Machine Learning en el ámbito de las Finanzas y la Banca ha demostrado ser altamente prometedora, ofreciendo beneficios significativos en términos de eficiencia, precisión y toma de decisiones informada. Diversos estudios científicos respaldan la efectividad de los modelos de Machine Learning en la predicción de variables financieras y el análisis de riesgos.

Investigadores como Andrew Ng, Pedro Domingos y Yaser S. Abu-Mostafa, han realizado contribuciones clave en el desarrollo de algoritmos y modelos de Machine Learning aplicados a las finanzas. Sus investigaciones han proporcionado fundamentos teóricos y prácticos para la aplicación efectiva de técnicas de aprendizaje automático en el sector financiero.

Los modelos de Machine Learning han demostrado una capacidad notable para predecir variables macroeconómicas, como el crecimiento del PIB (Barrios, Escobar, Leslie, Martin, & Peña, 2021). Estos modelos han superado en muchos casos a los enfoques tradicionales, permitiendo a las instituciones financieras tomar decisiones más informadas y mejorar la gestión de riesgos.

Además, el Machine Learning se ha utilizado en la detección de fraudes, donde ha logrado tasas de precisión significativamente superiores a los métodos convencionales. Estos modelos de detección de fraudes basados en Machine Learning pueden analizar grandes volúmenes de datos transaccionales en tiempo real, identificando patrones y anomalías que indican posibles actividades fraudulentas.

(Molina-Muñoz, 2021)

3.3.4. Avances en la medicina y salud gracias a ML.

La integración del Machine Learning en la Medicina y la Salud ha revolucionado la forma en que se diagnostican, tratan y gestionan las enfermedades. Numerosos estudios científicos respaldan la efectividad de los modelos de Machine Learning en una amplia gama de aplicaciones médicas, desde el diagnóstico y la predicción hasta la atención personalizada y la gestión de la salud.

Investigadores destacados, como Andrew Ng, Ziad Obermeyer e Isaac Kohane, han realizado contribuciones clave en el desarrollo y la aplicación de algoritmos de

Machine Learning en el campo de la salud. Sus investigaciones han allanado el camino para el uso exitoso de modelos de Machine Learning en diversas áreas, mejorando la precisión y la eficiencia de los procesos médicos.

Un caso de éxito destacado en la aplicación del Machine Learning en la Medicina es el diagnóstico de enfermedades mediante imágenes médicas, los algoritmos de Machine Learning pueden analizar grandes volúmenes de imágenes, como tomografías computarizadas y resonancias magnéticas, para identificar patrones y anomalías que podrían pasar desapercibidos para los médicos. Por ejemplo, en el uso de modelos de Machine Learning en la detección temprana de cáncer de mama a partir de mamografías, donde se ha logrado una mejora significativa en la precisión y la detección precoz de tumores malignos.

También en la medicina de precisión. A través del análisis de datos genéticos y clínicos, los modelos de Machine Learning pueden predecir la respuesta de un paciente a ciertos medicamentos o terapias. Esto permite una atención médica personalizada, donde los tratamientos se adaptan a las características individuales de cada paciente, maximizando la efectividad y reduciendo los efectos secundarios. Por ejemplo, en el campo de la oncología, se han utilizado modelos de Machine Learning para predecir qué pacientes con cáncer de pulmón se beneficiarán más de la inmunoterapia, lo que ayuda a los médicos a seleccionar el tratamiento óptimo para cada paciente.

Además de los casos de éxito en el diagnóstico y el tratamiento, el Machine Learning también ha demostrado su utilidad en la gestión de la salud y la atención médica. Los algoritmos de Machine Learning pueden analizar grandes conjuntos de datos de registros médicos electrónicos para identificar patrones de enfermedades, predecir la carga de trabajo hospitalaria y optimizar la asignación de recursos. Esto mejora la eficiencia operativa, reduce los costos y mejora la calidad de la atención al paciente.

Estos avances están transformando la práctica médica, mejorando la atención al paciente y abriendo nuevas oportunidades para la investigación médica y científica.

(Hinestroza Ramírez, 2018)

3.4. Tendencias y desarrollos recientes en el ML, Deep Learning

El Deep Learning es una rama del Machine Learning que se basa en redes neuronales artificiales de múltiples capas para aprender y extraer representaciones complejas de los datos. A diferencia de otros enfoques de Machine Learning, que dependen en gran medida de la ingeniería de características manuales, el Deep Learning permite que los modelos aprendan automáticamente las características y patrones relevantes a partir de los datos.

El concepto central del Deep Learning es la utilización de redes neuronales profundas, que son estructuras compuestas por múltiples capas de nodos interconectados. Cada capa de la red neuronal procesa y transforma la información recibida antes de pasarla a la siguiente capa. Las capas iniciales suelen encargarse de extraer características básicas, como bordes y formas simples, mientras que las capas posteriores se encargan de aprender representaciones más abstractas y complejas. Esto permite que el modelo aprenda gradualmente características de mayor nivel de abstracción a medida que se profundiza en la red.

El aprendizaje en el Deep Learning se realiza a través de algoritmos de optimización, como el descenso de gradiente, que ajustan los pesos y las conexiones de las neuronas para minimizar la diferencia entre las predicciones del modelo y los valores reales. Esta optimización se realiza utilizando grandes cantidades de datos de entrenamiento y se basa en la retropropagación del error, donde los errores se propagan hacia atrás en la red para ajustar los parámetros de las capas anteriores.

Una de las ventajas clave del Deep Learning es su capacidad para manejar grandes volúmenes de datos y aprender características altamente discriminativas de manera automática. Esto ha llevado a avances significativos en áreas como el reconocimiento de imágenes, el procesamiento del lenguaje natural, la traducción automática, la detección de objetos, el diagnóstico médico y muchas otras aplicaciones.

Un caso de éxito destacado en el campo del Deep Learning es el reconocimiento de imágenes. Los modelos basados en redes neuronales profundas, como las redes convolucionales, han superado consistentemente a los enfoques tradicionales en desafíos de reconocimiento de imágenes, como el reconocimiento de objetos y la clasificación de imágenes.

(Goodfelow, Bengio, & Courville, 2016)

3.4.1. Explicabilidad e interpretabilidad.

La explicabilidad se refiere a la capacidad de proporcionar explicaciones claras y comprensibles sobre las decisiones tomadas por un modelo de Deep Learning. Esto implica poder explicar por qué el modelo tomo cierta decisión en particular o que características han influido en la predicción, es especialmente importante en aplicaciones críticas donde se requiere transparencia y responsabilidad, como la medicina y la toma de decisiones financieras.

La interpretabilidad se relaciona con la capacidad de interpretar y comprender cómo un modelo de Deep Learning llega a sus resultados. Esto implica analizar las representaciones internas y los procesos que ocurren en las capas de la red neuronal. Comprender la lógica y las transformaciones realizadas por el modelo puede ayudar a identificar patrones, características relevantes y posibles sesgos presentes en los datos.

Para abordar la explicabilidad y la interpretabilidad en el Deep Learning, se han propuesto varias técnicas y enfoques:

Visualización de activaciones, permite visualizar las activaciones de las diferentes capas de la red neuronal para comprender cómo los datos se transforman a medida que pasan por el modelo. Esto puede revelar patrones y características relevantes utilizadas por el modelo para tomar decisiones.

Mapas de atención, algunos modelos de Deep Learning, como los Transformers, utilizan mecanismos de atención para resaltar partes importantes de la entrada durante el procesamiento. Los mapas de atención pueden mostrar qué áreas de la entrada son más relevantes para las predicciones realizadas por el modelo.

Es importante destacar que la explicabilidad y la interpretabilidad del Deep Learning son áreas de investigación activas y desafiantes. Aunque se han logrado avances significativos, aún existen desafíos en el desarrollo de técnicas efectivas y generalizables que permitan una comprensión más profunda de los modelos de aprendizaje profundo. La transparencia y la confianza en estos modelos son fundamentales para su adopción responsable y ética en diversas aplicaciones.

(Hernández, n.d.)

3.4.2. Federated Learning.

Federated Learning es una técnica de entrenamiento de modelos de Deep Learning que permite aprovechar el poder de cómputo distribuido en entornos descentralizados. En lugar de recopilar todos los datos en un servidor centralizado para entrenar un modelo, el proceso de entrenamiento se lleva a cabo en los dispositivos locales de los participantes, respetando la privacidad de los datos.

En el contexto del Deep Learning, Federated Learning permite entrenar modelos de redes neuronales profundas utilizando datos distribuidos en diferentes dispositivos. Cada dispositivo, como un teléfono móvil o un dispositivo IoT, tiene su propia copia de los datos y ejecuta el entrenamiento localmente utilizando su capacidad de procesamiento.

El proceso se realiza en varias rondas. En cada ronda, los modelos iniciales se distribuyen a los dispositivos participantes. Luego, cada dispositivo realiza el entrenamiento local utilizando sus datos y actualiza el modelo con los resultados obtenidos. Sin embargo, en lugar de enviar los datos actualizados al servidor centralizado, solo se envían las actualizaciones del modelo, como los cambios en los pesos y los gradientes.

Estas actualizaciones del modelo se agregan en el servidor centralizado y se realiza un promedio ponderado para obtener una versión actualizada del modelo global. El proceso de agregación se realiza de forma segura, manteniendo la privacidad de los datos locales y protegiendo la confidencialidad de la información sensible.

(GOOGLE INC, 2020)

4. Modelo de ML para la predicción de variables macroeconómicas

4.1. Descripción del problema de predicción.

El objetivo del presente estudio es desarrollar un modelo de Machine Learning para predecir el Índice de Precios al Consumidor (IPC) en uno o varios trimestres

futuros. El IPC es una medida importante utilizada para medir la inflación y los cambios en el nivel general de precios de bienes y servicios en una economía.

El enfoque de Machine Learning seleccionado para este estudio se basará en algoritmos de aprendizaje supervisado, donde se utiliza un conjunto de datos históricos etiquetados con valores de IPC previos para entrenar el modelo. Se explorarán diversas técnicas y algoritmos, como regresión lineal, árboles de decisión, redes neuronales, entre otros, con el objetivo de identificar el enfoque más adecuado que brinde las mejores predicciones de IPC.

4.2. Obtención y procesado de los datos.

Los datos utilizados en este estudio han sido recopilados de fuentes fiables, específicamente de la web del Banco de España y esta a su vez de INE. Se ha llevado a cabo la descarga de varias tablas con la intención de obtener la información necesaria para el análisis.

be2301

	NOMBRE DE LA SERIE	DCIIPI2015IND.M	DCIIPI2015IND_D09B00.M	DCIIPI2015IND_D09C00.M	DCIIPI2015IND_D09D00.M	DCIIPI2015IND_D09ENR.M	DCIIPI2015IND_D(
0	NÚMERO SECUENCIAL	3272316	3272317	3272348	3272765	3272791	
1	ALIAS DE LA SERIE	BE_23_1.1	BE_23_1.2	BE_23_1.3	BE_23_1.4	BE_23_1.5	BI
2	DESCRIPCIÓN DE LA SERIE	Estadísticas generales. IPI. Índice general	Estadísticas generales. IPI. Por rama de activ...	Estadísticas generales. IPI. Por rama de activ...	Estadísticas generales. IPI. Por rama de activ...	Estadísticas generales. IPI. Por destino econó...	Estadísticas gener Por destino
3	DESCRIPCIÓN DE LAS UNIDADES	Base_2015=100	Base_2015=100	Base_2015=100	Base_2015=100	Base_2015=100	Base_2
4	FRECUENCIA	MENSUAL	MENSUAL	MENSUAL	MENSUAL	MENSUAL	M
...	...	...	...	...	...	...	
581	ENE 2023	101.166	66.774	101.349	104.101	101.456	
582	FEB 2023	103.407	83.990	106.071	94.185	92.554	
583	MAR 2023	118.207	92.367	122.794	100.175	99.415	
584	FUENTE	Instituto Nacional de Estadistica	Instituto Nacional de Estadistica	Instituto Nacional de Estadistica	Instituto Nacional de Estadistica	Instituto Nacional de Estadistica	Instituto Na Es
585	NOTAS	NaN	NaN	NaN	NaN	NaN	

586 rows × 11 columns

Ilustración 1: b2301

En primer lugar, se ha descargado la tabla 'be2301', la cual contiene múltiples variables relevantes. Sin embargo, se ha realizado una selección específica, centrándonos únicamente en el índice de producción industrial que se encuentra bajo

el nombre de la serie "DCIIPI2015IND.M". Esta variable se considera crucial para analizar las tendencias en el sector industrial.

si_1_1

	NOMBRE DE LA SERIE	DSPC102016VP30000_ES14A_TSC.T	D_1KH90101	D_1KH99500_D09	DCMICN2015INDVD_ETOT_TSC.M	D_1KN31000	DOEECIIDIND_D09C
0	NÚMERO SECUENCIAL	3778094	805995	1832510	3272949	1525635	
1	ALIAS DE LA SERIE	SI_1_1.1	SI_1_1.2	SI_1_1.3	SI_1_1.4	SI_1_1.5	
2	DESCRIPCIÓN DE LA SERIE	CNE. Base 2016. Consumo final. Hogares e ISFLS...	Estadísticas generales. Encuestas de opinión. ...	Estadísticas generales. Encuesta de comercio m...	Estadísticas generales. Índice del comercio al...	Estadísticas generales. ANFAC. Ventas de autom...	Estadísticas g
3	DESCRIPCIÓN DE LAS UNIDADES	Base_2016=100	Porcentaje neto	Porcentaje neto	Base_2015=100	Vehículos	
4	FRECUENCIA	TRIMESTRAL	MENSUAL	MENSUAL	MENSUAL	MENSUAL	
...	...	...	...	...	...	...	
740	ABR 2023	_	-14.1	8.9	_	74749	
741	MAY 2023	_	_	_	_	_	
742	JUN 2023	_	_	_	_	_	
743	FUENTE	Instituto Nacional de Estadistica	CEE	COMISION DE LA UE (EUROPEAN ECONOMY.SUPLEMENT B)	Instituto Nacional de Estadistica	ANFAC	Ministerio de Indust
744	NOTAS	NaN	Se obtiene como media aritmética de 4 series.A...	MEDIA ARITMETICA DE LOS SALDOS DE LAS OPINIONE...	NaN	NaN	

745 rows × 128 columns

Ilustración 2: si_1_1

Asimismo, se ha descargado la tabla 'si_1_1', la cual también presenta diversas variables. Sin embargo, para nuestro análisis, se han seleccionado únicamente dos variables de interés. La primera es la tasa de paro, identificada por el nombre de la serie "DEPC200540_TSPA.T", y la segunda es el IPC interanual, reconocido por el nombre de la serie "DIP2021LTVA_E00.M". Estas variables son relevantes para evaluar la situación del mercado laboral y los cambios en el nivel general de precios.

be23a

	NOMBRE DE LA SERIE	DSPC102016CP30000_ES1_VCNTRA.T	DSPC102016CP30000_ES14.T	DSPC102016CP30000_ES15.T	DSPC102016CP30000_ES13_VCNTRA.T	DSPC
0	NÚMERO SECUENCIAL	3777714	3777708	3777711	3777707	
1	ALIAS DE LA SERIE	BE_23_A.1	BE_23_A.2	BE_23_A.3	BE_23_A.4	
2	DESCRIPCIÓN DE LA SERIE	CNE. Base 2016. Gasto en consumo final. Precio...	CNE. Base 2016. Gasto en consumo final. Hogare...	CNE. Base 2016. Gasto en consumo final. ISFLSH...	CNE. Base 2016. Gasto en consumo final. AAPP. ...	CN
3	DESCRIPCIÓN DE LAS UNIDADES	Millones de Euros	Millones de Euros	Millones de Euros	Millones de Euros	
4	FRECUENCIA	TRIMESTRAL	TRIMESTRAL	TRIMESTRAL	TRIMESTRAL	
...	...	...	...	...	...	
114	JUN 2022	258193	185237	3430	69526	
115	SEP 2022	254941	188909	3169	62863	
116	DIC 2022	267914	186526	3914	77474	
117	FUENTE	Instituto Nacional de Estadistica	Instituto Nacional de Estadistica	Instituto Nacional de Estadistica	Instituto Nacional de Estadistica	
118	NOTAS	NaN	NaN	NaN	NaN	

119 rows × 23 columns

Ilustración 3: be23a

La tabla 'be23a' también ha sido descargada, y al igual que las anteriores, contiene múltiples variables. Sin embargo, para este estudio, se ha realizado una selección específica, enfocándonos exclusivamente en el Producto Interno Bruto (PIB) a precios corrientes. Esta variable, identificada por el nombre de la serie "DSPC102016CB1QB00_SS1.T", es fundamental para comprender la evolución económica en términos de valor monetario.

be1091

	NOMBRE DE LA SERIE	D_1NBAF472
0	ENE 1999	3.062
1	FEB 1999	3.030
2	MAR 1999	3.046
3	ABR 1999	2.756
4	MAY 1999	2.683
...	...	...
288	ENE 2023	3.337
289	FEB 2023	3.534
290	MAR 2023	3.647
291	ABR 2023	3.757
292	MAY 2023	3.862

293 rows × 2 columns

Ilustración 4: be1091

Por último, se ha extraído información de la tabla 'be1091', en la cual se ha seleccionado la variable del Euribor a 1 año. Esta variable, identificada por el nombre de la serie "D_1NBAF472", es relevante en el contexto financiero y se utiliza para analizar las tasas de interés.

En resumen, los datos utilizados en este estudio han sido obtenidos del Banco de España y se han descargado las tablas 'be2301', 'si_1_1', 'be23a' y 'be1091'. Se ha llevado a cabo una selección cuidadosa de variables específicas de cada tabla para enfocar el análisis en el índice de producción industrial, la tasa de paro, el IPC interanual, el PIB a precios corrientes y el Euribor a 1 año. Estas variables son fundamentales para comprender y evaluar las tendencias económicas y financieras relevantes para este estudio.

El proceso de la obtención y procesamiento de datos ha presentado varios desafíos, principalmente en la búsqueda de una fuente que contuviese todas las variables consideradas necesarias. Aunque esto pueda parecer trivial, ha resultado ser una tarea complicada. Sin embargo, gracias a este esfuerzo, el hecho de que todas las tablas provengan de la misma fuente ha facilitado en gran medida la tarea de combinar y unir los datos.

El proceso general de procesamiento de datos ha seguido el siguiente modelo:

1. Descarga de los datos en formato CSV desde la página web del Banco de España.

2. Lectura del archivo CSV utilizando la biblioteca pandas y la función read_csv(), especificando los parámetros delimiter= "," y encoding= "latin".

3. Creación de un dataframe por cada variable en Python, seleccionando únicamente las columnas necesarias, como el nombre de la serie y el nombre de la variable correspondiente.

En el caso específico del Euribor ("be1091"), el modelo de procesamiento de datos ha seguido un enfoque ligeramente diferente debido a problemas en el formato de guardado de los datos descargables en formato CSV. En su lugar, se ha llevado a cabo el siguiente proceso:

1. Descarga de la tabla en formato Excel.

2. Copia de las columnas relevantes ("NOMBRE DE LA SERIE" y "D_1NBAF472") a otro archivo de Excel.

3. Guardado del nuevo archivo de Excel en la carpeta de datos del proyecto.

4. Lectura del archivo de Excel utilizando la biblioteca pandas y la función read_excel().

Una vez que todos los datos han sido cargados, se ha realizado una unión entre las cinco tablas utilizando la clave "NOMBRE DE LA SERIE" como referencia, y dentro de esta, se encuentra el campo de fecha. Esto nos ha permitido manejar las diferencias en la periodicidad de los datos, ya sea trimestral o mensual. Al realizar la unión, se ha establecido el límite de la periodicidad y se han seleccionado únicamente los datos trimestrales ya que son los que coinciden para su posterior análisis.

```
tablaModelo
```

	fecha	tasaDeParo	pib	ipi	euribor	ipc
12	MAR 2002	11.55	177793	118.228	3.816	3.10
13	JUN 2002	11.15	189754	123.683	3.869	3.40
14	SEP 2002	11.49	183547	125.332	3.236	3.50
15	DIC 2002	11.61	198458	112.012	2.873	4.00
16	MAR 2003	11.99	190914	129.772	2.411	3.70
...	...	...	...	...	...	...
91	DIC 2021	13.33	329317	100.119	-0.502	6.50
92	MAR 2022	13.65	310639	112.236	-0.237	9.80
93	JUN 2022	12.48	332194	113.414	0.852	10.20
94	SEP 2022	12.67	328980	110.842	2.233	8.90
95	DIC 2022	12.87	355295	96.911	3.018	5.70

84 rows × 6 columns

Ilustración 5: tablaModelo

Este enfoque de procesamiento de datos nos ha permitido obtener conjuntos de datos coherentes y consistentes para llevar a cabo el análisis y la modelización requeridos en este proyecto.

4.3. Elección del modelo de ML.

En el proceso de selección del mejor modelo, es importante considerar diversos factores que influyen en su desempeño y aplicabilidad. El error del modelo, la capacidad del modelo para minimizar el error es fundamental. La complejidad del modelo es la capacidad de capturar patrones y relaciones en los datos. Un modelo demasiado simple puede ser insuficiente para capturar la complejidad subyacente, mientras que uno demasiado complejo puede llevar al sobreajuste. La eficiencia temporal del modelo es un factor importante a considerar. Se deben tener en cuenta los recursos computacionales necesarios para entrenar y utilizar el modelo. Algunos algoritmos, como las redes neuronales profundas, pueden requerir una gran capacidad de cómputo y memoria

Al evaluar estos factores y considerar las necesidades y restricciones del problema en cuestión, se puede seleccionar el modelo más adecuado que cumpla con los requisitos de error, complejidad, velocidad y recursos computacionales. La elección del modelo requiere un análisis comparativo y un enfoque iterativo, donde se prueban diferentes modelos y se ajustan sus parámetros para encontrar la mejor solución posible.

4.3.1. Elección del modelo para este caso de estudio.

4.3.1.1. Regresión lineal

En el presente modelo de regresión lineal, se emplearon las variables tasa de desempleo, Producto Interno Bruto (PIB), Euribor e Índice de Producción Industrial (IPI) con el objetivo de predecir el Índice de Precios al Consumidor (IPC). Si bien los resultados obtenidos no son sobresalientes, considerando la simplicidad del modelo de regresión lineal utilizado, pueden considerarse satisfactorios.

Análisis de los resultados en el conjunto de entrenamiento:

En el conjunto de entrenamiento, se obtuvo un Error Cuadrático Medio (MSE) de 3.74, lo cual puede considerarse como un resultado razonable. Al observar la gráfica generada, se puede apreciar una cercanía considerable entre los valores predichos y los valores reales.

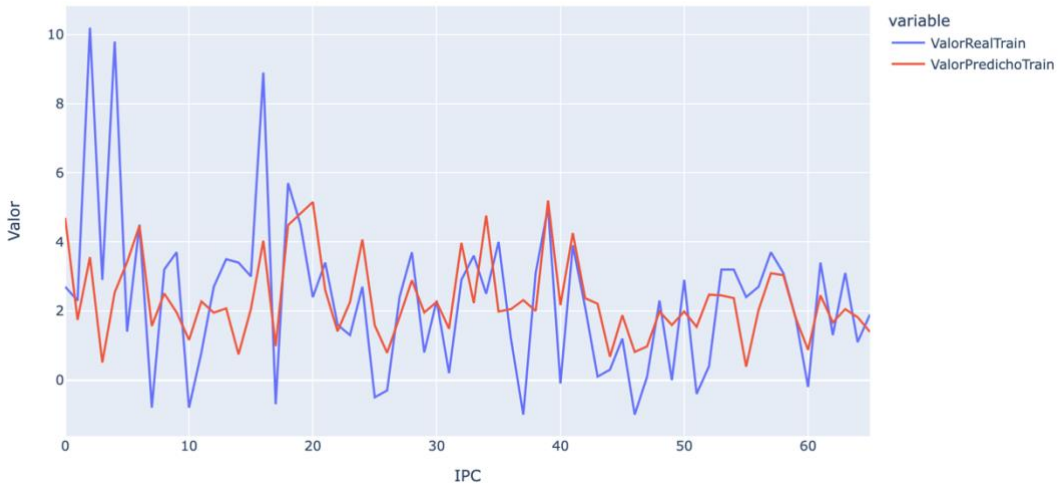

Ilustración 6: Resultados train modelo de regresión

Análisis de los resultados en el conjunto de prueba:

En el conjunto de prueba, se obtuvieron resultados aún mejores, lo cual puede resultar inusual y podría deberse al reducido tamaño del conjunto de datos. Se reportó un MSE de 2.25 , el cual sigue siendo un resultado razonable considerando el tamaño de la muestra. Asimismo, al examinar la gráfica correspondiente, se evidencia una notable aproximación entre los valores predichos y los valores reales.

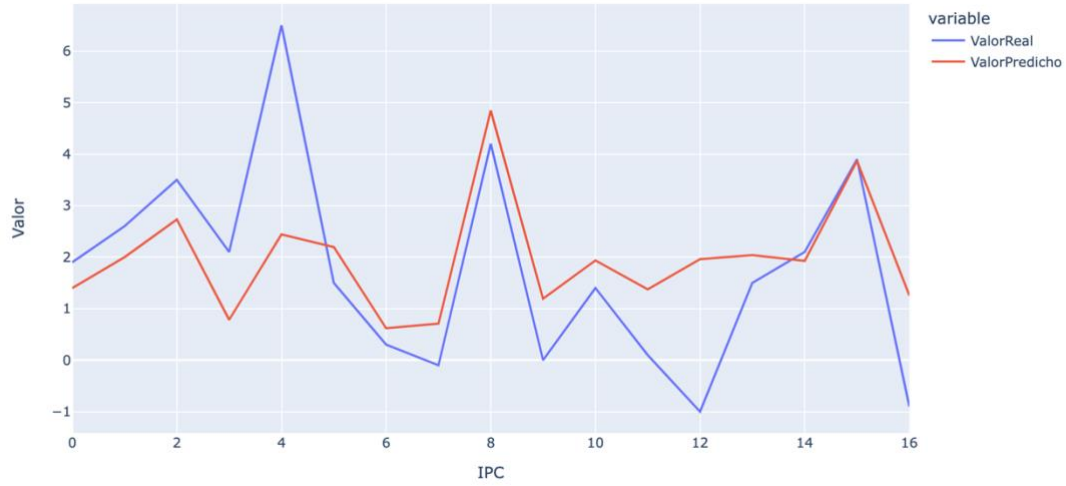

Ilustración 7: Resultados test modelo de regresión

Es importante tener en cuenta que estos resultados deben interpretarse con cautela, ya que se trata de un modelo de regresión lineal sencillo y se asumen ciertas limitaciones inherentes a esta técnica. Sin embargo, considerando las circunstancias y las características del modelo, los resultados obtenidos pueden considerarse satisfactorios.

4.3.1.2. Random Forest

Antes de proceder al análisis con el modelo Random Forest, es recomendable realizar una matriz de correlaciones entre las variables involucradas. Esta matriz permite identificar las relaciones lineales entre las variables y proporciona información útil para la selección y ponderación de variables en el modelo.

En el caso específico del problema de predicción del Índice de Precios al Consumidor (IPC), al realizar la matriz de correlaciones se observa que las variables más correlacionadas con el IPC son la tasa de desempleo, el Euribor y el Índice de Producción Industrial (IPI).

La tasa de desempleo muestra una correlación negativa con el IPC, lo que indica que un aumento en la tasa de desempleo tiende a estar asociado con una disminución en el IPC. Por otro lado, el Euribor muestra una correlación positiva con el IPC, lo que sugiere que un aumento en el Euribor se relaciona con un aumento en el IPC. Finalmente, el Índice de Producción Industrial (IPI) también muestra una correlación positiva con el IPC, lo que implica que un incremento en el IPI está relacionado con un aumento en el IPC.

Estas correlaciones identificadas entre las variables y el IPC proporcionan información valiosa para ajustar los pesos y la importancia relativa de cada variable en el modelo Random Forest. En función de estas correlaciones, se puede considerar la posibilidad de asignar mayor peso a las variables más correlacionadas con el IPC con el fin de mejorar la capacidad predictiva del modelo.

	fecha	tasaDeParo	pib	ipi	euribor	ipc
fecha	NaN	NaN	NaN	NaN	NaN	NaN
tasaDeParo	NaN	1.000000	0.152416	-0.748293	-0.573503	-0.463708
pib	NaN	0.152416	1.000000	-0.487340	-0.429188	0.066163
ipi	NaN	-0.748293	-0.487340	1.000000	0.680875	0.361131
euribor	NaN	-0.573503	-0.429188	0.680875	1.000000	0.405031
ipc	NaN	-0.463708	0.066163	0.361131	0.405031	1.000000

Ilustración 8: Tabla de correlaciones

En el presente análisis del modelo de Random Forest, se ha utilizado la biblioteca sklearn de Python, específicamente la función RandomForestRegressor(), para abordar el problema de predicción del Índice de Precios al Consumidor (IPC).

Se han vuelto a identificar limitaciones significativas en cuanto al tamaño de la muestra disponible para entrenamiento.

Empezando el análisis del modelo de Random Forest, se ha realizado el cálculo de la importancia de las variables utilizadas en el modelo, y se ha observado que esta importancia es consistente con las correlaciones previas de las variables con el IPC.

Sin embargo, se ha identificado una diferencia notable en el caso del Producto Interno Bruto (PIB), el cual muestra una correlación muy baja pero una importancia media en el modelo. Esta discrepancia sugiere la posibilidad de que exista una relación no lineal entre el PIB y el IPC. Aunque la correlación directa entre ambas variables es baja, el PIB todavía puede desempeñar un papel relevante en la predicción del IPC a través de una relación no lineal.

Es importante tener en cuenta que esta observación específica sobre el PIB resalta la necesidad de considerar la posibilidad de relaciones no lineales y la importancia de utilizar modelos como Random Forest, que pueden capturar estas relaciones más complejas.

A pesar de las limitaciones, los resultados obtenidos en el conjunto de prueba son muy prometedores, con un Error Cuadrático Medio (MSE) de 0.46. No obstante, es importante tener en cuenta que este resultado podría estar relacionado con la presencia de overfitting en el modelo. Lamentablemente, no es posible aumentar el tamaño del conjunto de entrenamiento.

Al examinar la gráfica entre el valor predicho en train y el valor real, se puede apreciar claramente cómo el modelo se ajusta muy bien a los datos de entrenamiento, lo que refuerza la sospecha de overfitting aunque no la confirma.

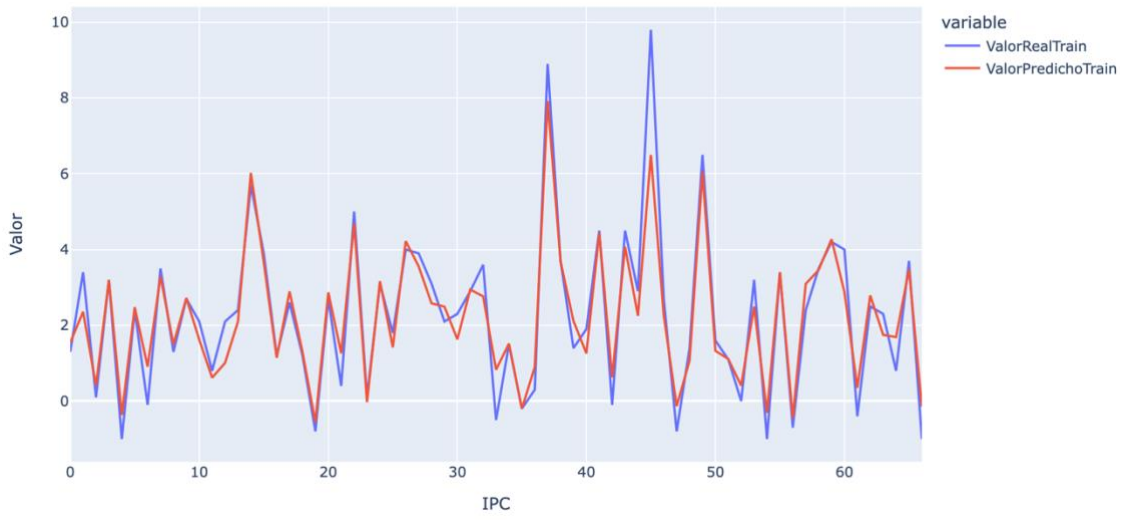

Ilustración 9: Resultados train Random Forest

En el conjunto de prueba, aunque los resultados son ligeramente inferiores con un MSE de 1.6, siguen siendo buenos. En la gráfica correspondiente a este conjunto, se puede observar cómo el modelo continúa adaptándose de manera adecuada a los cambios en los datos, lo que sugiere que el overfitting mencionado anteriormente puede no ser un problema real en este caso particular.

A pesar de las limitaciones en el tamaño de la muestra y las posibles preocupaciones sobre el overfitting, los resultados del modelo de Random Forest son alentadores en términos de su capacidad para predecir el IPC.

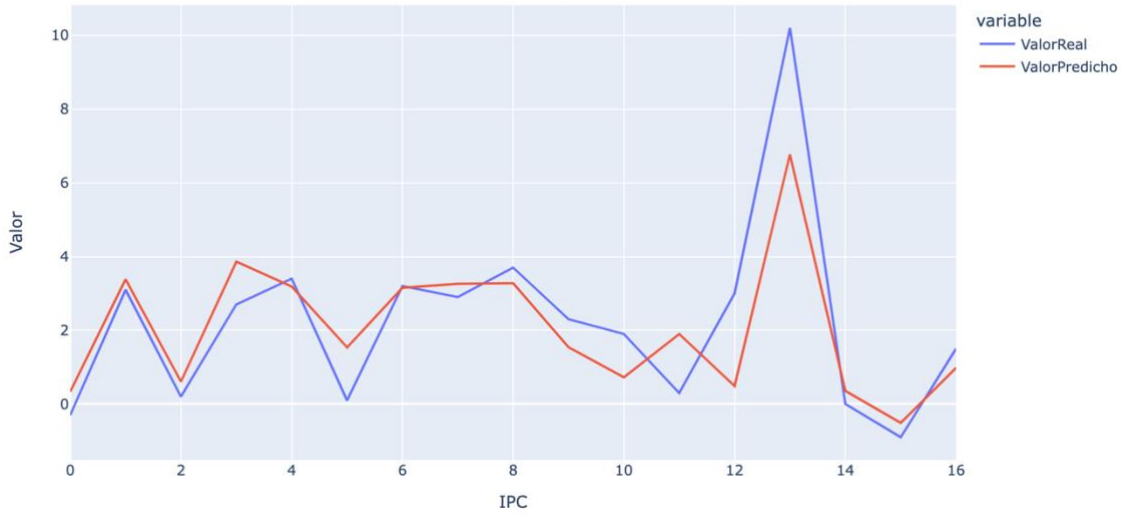

Ilustración 10: Resultados test Random Forest

4.3.1.3. KNN

Lo primero que haremos es iterar mediante un bucle con diferentes valores de k (número de clusters) y asi obtener los diferentes valores del MSE para construir la gráfica del codo y determinar el número óptimo de clusters para nuestro modelo.

Tras realizar varias pruebas y analizar diferentes combinaciones de datos de entrenamiento y test, se ha observado que el valor de k=7 es el que más veces ha sido identificado como la mejor opción. Sin embargo, se debe tener en cuenta que debido al tamaño reducido de nuestro conjunto de datos, los resultados pueden variar considerablemente. No obstante, para garantizar la durabilidad del modelo a largo plazo, se ha tomado la decisión de seleccionar k=7 como un número óptimo de clusters.

Es importante destacar que, al aumentar el tamaño del conjunto de datos en el futuro, se podrían realizar ajustes adicionales para encontrar un valor de k más preciso y adaptado a la nueva información disponible, aunque se ha intentado acercar lo máximo posible a lo que se considera que será el numero correcto de clusters más adelante.

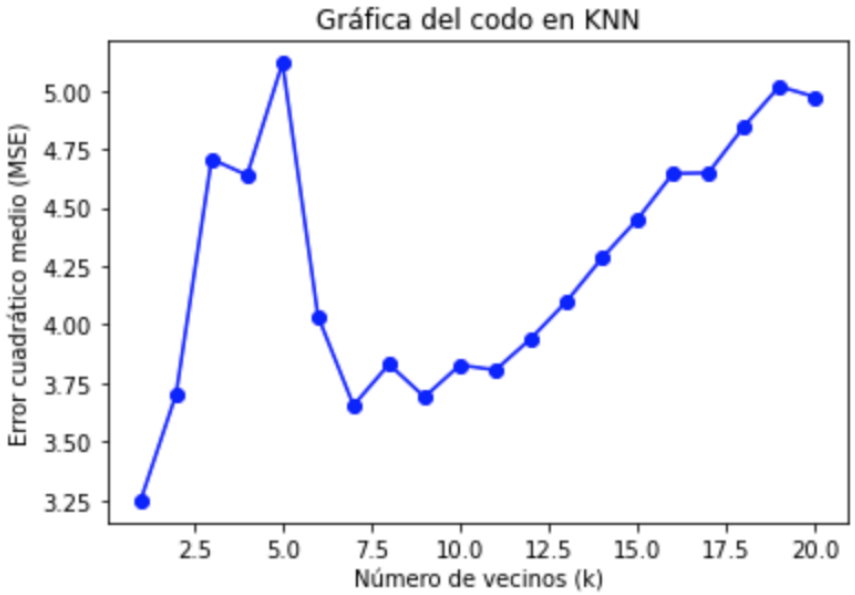

Ilustración 11: Gráfico del codo KNN

En los conjuntos de entrenamiento y prueba, se han obtenido resultados poco satisfactorios en términos de métrica de error cuadrático medio (MSE). En el conjunto de entrenamiento, se ha obtenido un MSE de 3.8, mientras que en el conjunto de prueba se ha obtenido un MSE de 3.7.

A continuación, se presentan las gráficas correspondientes a los conjuntos de train y test:

Train:

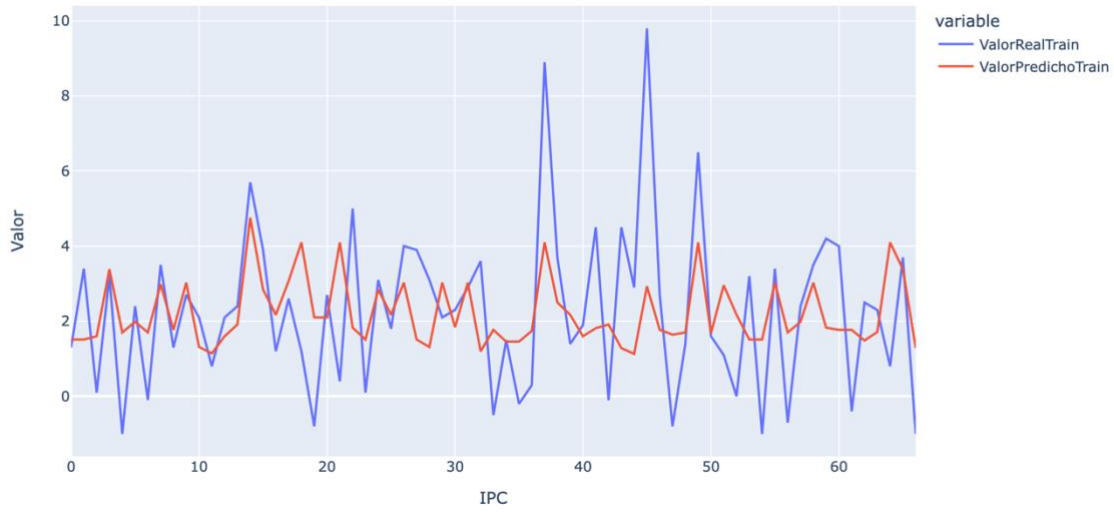

Ilustración 12: Resultados train KNN

Test:

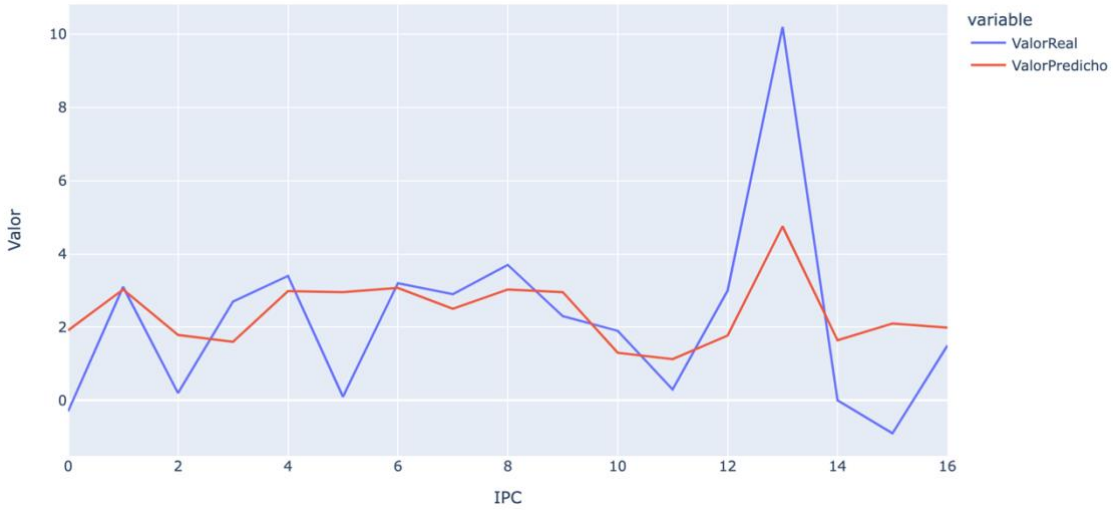

Ilustración 13: Resultados test KNN

Los resultados, tienen un error elevado, son demasiado generales y aportan poca información.

4.4. Análisis y evaluación de los resultados obtenidos.

Tras realizar un análisis comparativo de los errores cometidos por los modelos de regresión lineal simple, Random Forest y KNN, se ha llegado a la conclusión de que el modelo más adecuado a elegir es Random Forest. Si bien los tres modelos son

relativamente sencillos, el análisis de los errores muestra que Random Forest presenta un desempeño superior en términos de precisión.

A continuación, se presentan unas gráficas comparativas del error en los distintos modelos, con el objetivo de proporcionar una visualización más clara de los resultados obtenidos.

La primera gráfica presenta una comparativa de la suma de los errores cuadráticos medios (MSE) en los conjuntos de prueba y entrenamiento para cada uno de los modelos evaluados.

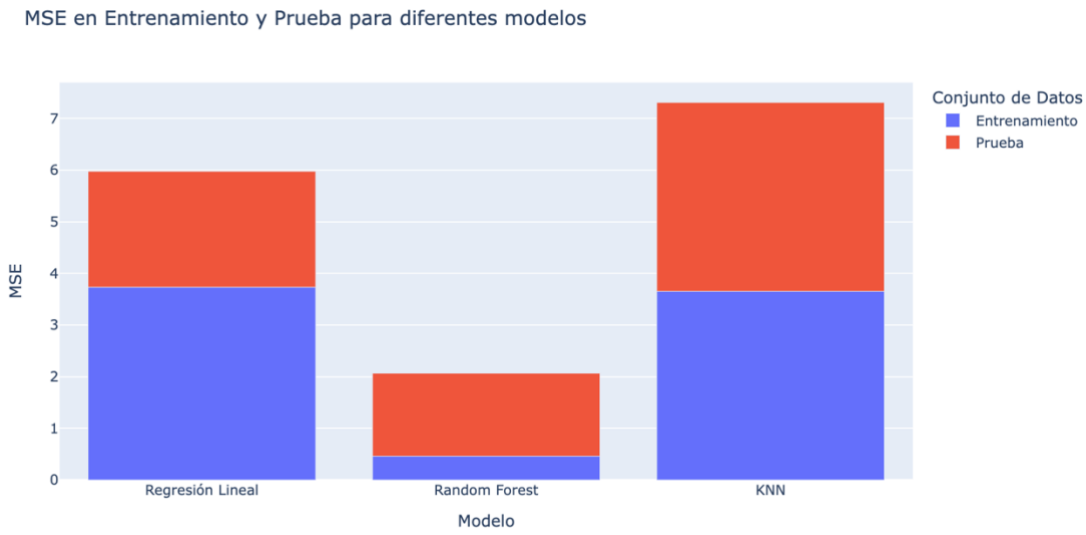

Ilustración 14: Gráfico MSE Train y Test ambos modelos

La segunda gráfica muestra las diferencias en el MSE entre los conjuntos de prueba para cada modelo.

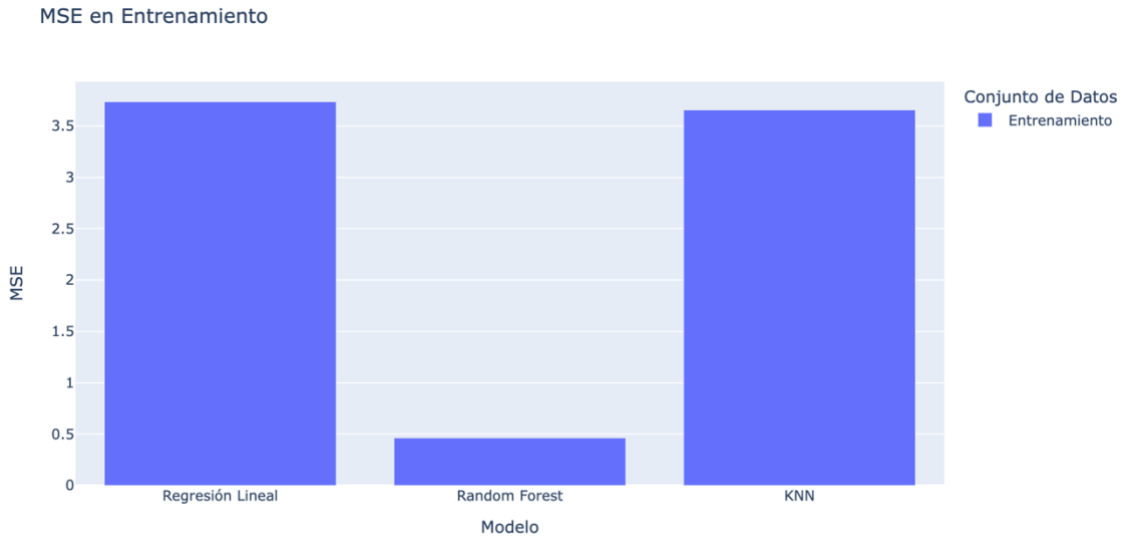

Ilustración 15: Gráfico resultados train de todos los modelos

Por último, la tercera gráfica es de especial relevancia, ya que muestra el MSE en el conjunto de prueba, el cual representa el escenario real de uso del modelo y donde no ha sido utilizado en el proceso de entrenamiento.

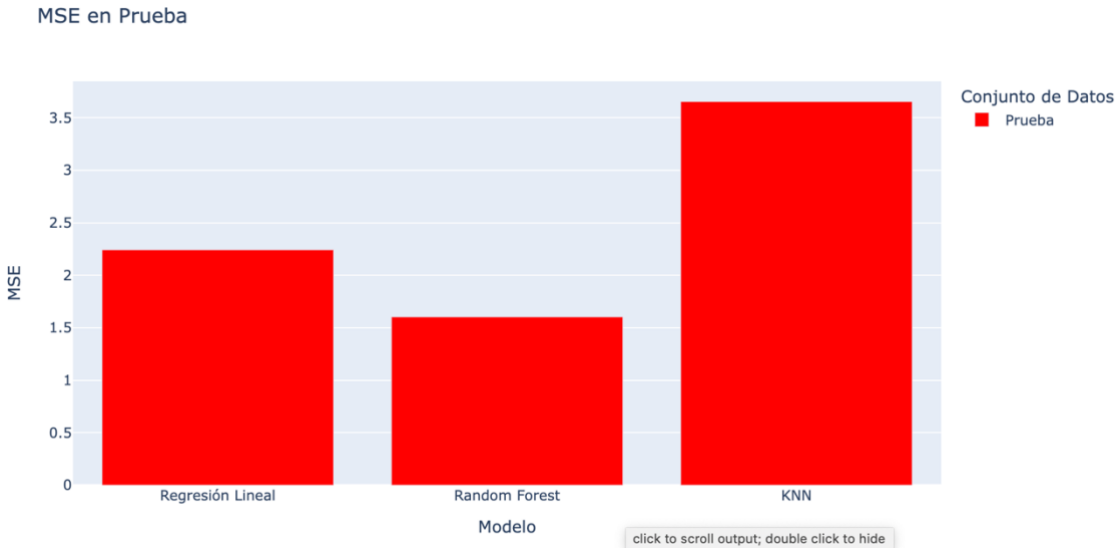

Ilustración 16: Gráfico resultados test todos los modelos

Estas gráficas permiten tener una mejor comprensión de las diferencias en el desempeño de los modelos en términos de precisión y proporcionan información

clave para tomar una decisión informada sobre qué modelo seleccionar. Ese modelo es Random Forest.

5. Aplicación web con Dash.

5.1. Introducción.

Se ha desarrollado una aplicación web utilizando la biblioteca Dash, la cual permite crear interfaces interactivas en Python. En esta aplicación, se ha importado el modelo de Random Forest previamente seleccionado, así como los datos necesarios para realizar las predicciones.

La aplicación web consta de varios elementos visuales. En primer lugar, se muestra un banner con el título "TFG CARLOS GAHETE MORILLO", que identifica claramente el propósito de la aplicación. A continuación, se presenta otro título que indica que se trata de un modelo de Random Forest para la predicción del IPC.

Además, se incluyen dos gráficas que representan los resultados del modelo para los conjuntos de datos de prueba y entrenamiento. Estas gráficas permiten visualizar el desempeño del modelo en diferentes escenarios para que el usuario sea consciente de los riesgos.

En la siguiente sección de la aplicación, se brinda al usuario la posibilidad de introducir valores mediante cuatro cuadros de texto. Estos campos corresponden a las variables IPC, PIB, tasa de desempleo y Euribor, los cuales son relevantes para la predicción del IPC.

Al hacer clic en el botón "Predecir IPC", se activa una función de callback que utiliza el modelo de Random Forest para realizar una predicción del IPC basada en los valores ingresados por el usuario. El resultado de esta predicción se muestra en pantalla.

Pantalla nada más acceder al enlace:

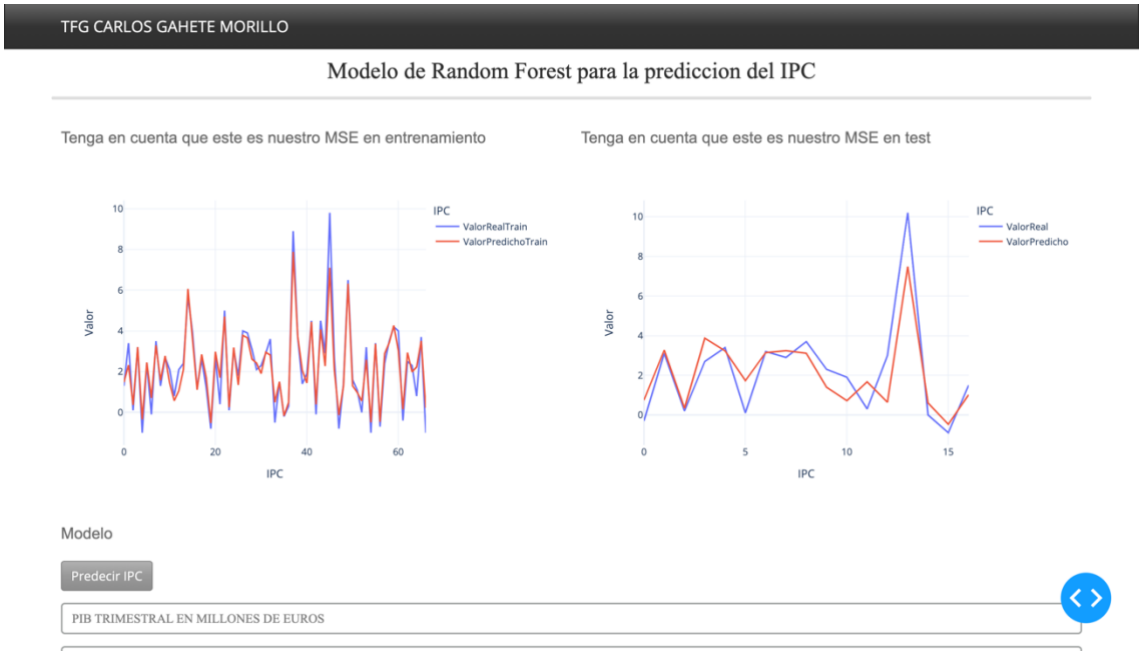

Ilustración 17: Pantalla nada más acceder al enlace DASH

Pantalla visible al bajar al máximo sin introducir nada:

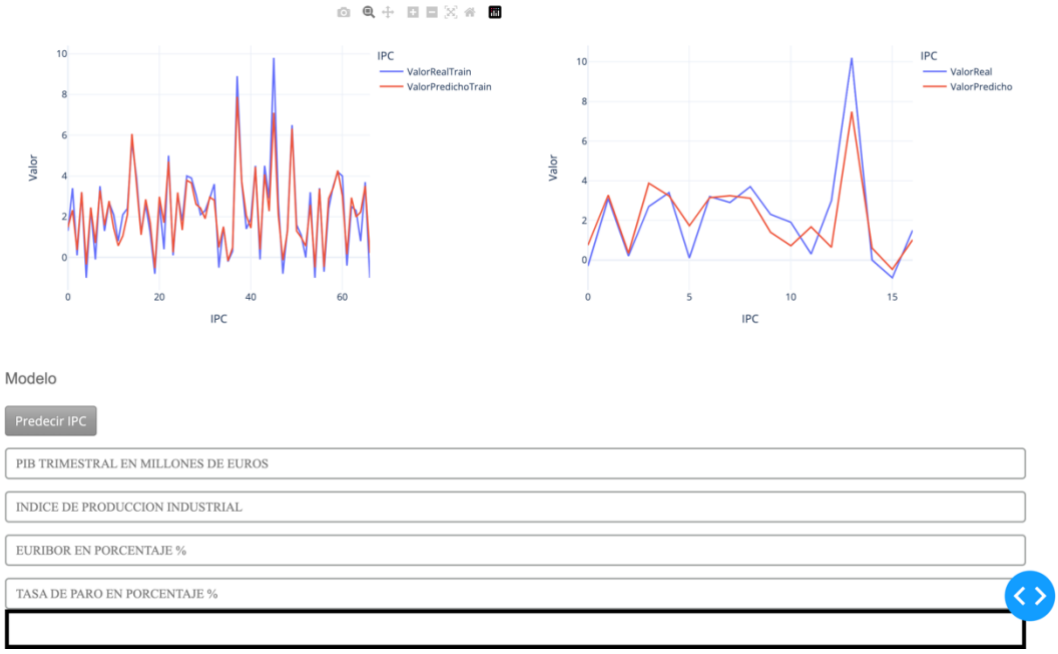

Ilustración 18: Pantalla visible al bajar al máximo sin introducir nada DASH

Pantalla visible al introducir los datos necesarios para realizar la predicción:

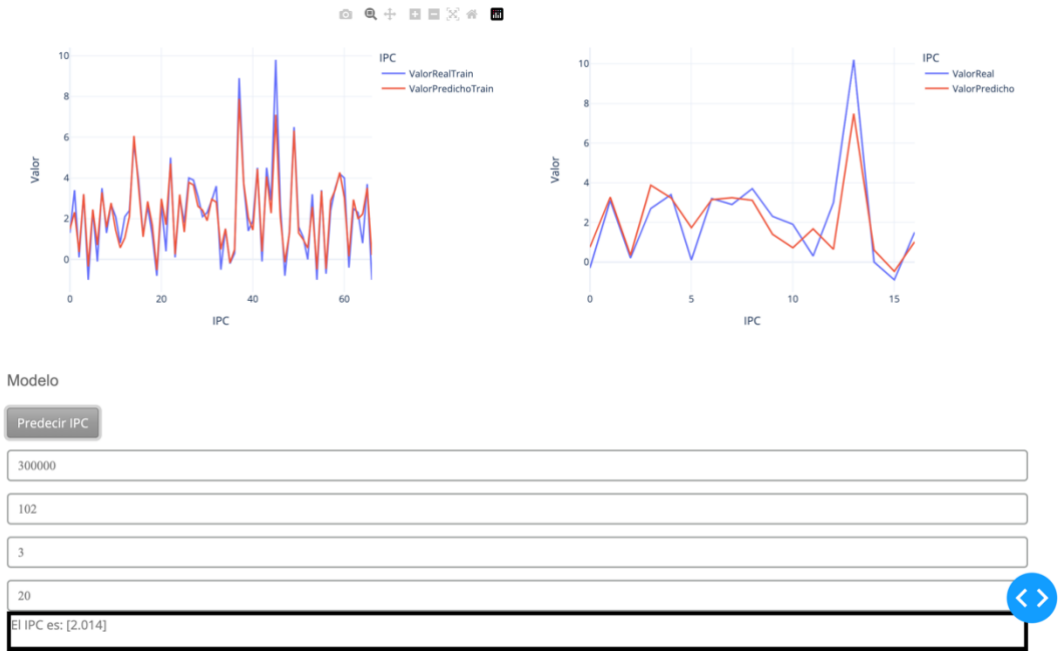

Ilustración 19: Pantalla visible al introducir los datos necesarios para realizar la predicción DASH

5.2. Desarrollo de la aplicación.

El desarrollo de la aplicación web se ha llevado a cabo utilizando la biblioteca Dash de Python. Dash es una herramienta que permite crear interfaces interactivas de manera sencilla y eficiente. En este caso, se ha aprovechado el análisis previo realizado en Python, donde se generaron los gráficos y se entrenó el modelo de Random Forest.

Para utilizar el modelo de Random Forest en la aplicación Dash, se ha exportado y guardado mediante la librería pickle. Esto facilita la posterior carga del modelo en el entorno Dash. Además del modelo, se han exportado también los datos necesarios para su funcionamiento.

Una vez cargados el modelo y los datos en la aplicación Dash, se ha procedido a configurarla de acuerdo con los requisitos establecidos. Esto implica definir la apariencia visual, establecer los componentes interactivos y definir las funcionalidades deseadas.

La exportación y carga del modelo de Random Forest y los datos en la aplicación Dash brinda la capacidad de realizar predicciones interactivas y personalizadas. La aplicación permite al usuario introducir valores específicos en los

campos correspondientes, lo que desencadena un proceso de predicción utilizando el modelo de Random Forest. Esta funcionalidad se logra mediante la implementación de un callback, que se encarga de procesar los datos ingresados por el usuario y generar la predicción correspondiente.

En conclusión, la aplicación web desarrollada en Dash combina la potencia del modelo de Random Forest con una interfaz intuitiva y fácil de usar. Permite a los usuarios obtener predicciones personalizadas y explorar el comportamiento del modelo a través de gráficos interactivos.

En el contexto del uso de Dash en la aplicación desarrollada, es importante destacar algunas peculiaridades relacionadas con la forma de introducir los datos al modelo y la función de los callbacks

Al recopilar la información ingresada en los cuadros de texto (utilizando "dbc.Input") para pasarla al modelo, se debe tener en cuenta que el modelo de Random Forest fue entrenado utilizando los nombres de las columnas como referencia. Por lo tanto, al proporcionar los datos desde los cuadros de texto, es necesario asegurarse de introducirlos de la misma manera en que se encuentran en las columnas originales del conjunto de datos utilizado para entrenar el modelo. Esta consideración puede parecer trivial, pero difiere del comportamiento en un notebook de Python, donde no sería necesario pasar los datos con los nombres de las columnas, aunque se mostraría una advertencia. En Dash, es necesario proporcionar los datos con los nombres correspondientes para evitar errores.

Los callbacks desempeñan un papel fundamental en la interactividad de una aplicación Dash. Sin embargo, en el caso particular de esta aplicación, que cuenta con cuatro cuadros de texto para ingresar los datos, cada vez que se realiza una modificación en uno de ellos, se activa el callback correspondiente y se llama al modelo para realizar una predicción. Esta dinámica puede generar posibles errores o una experiencia poco agradable para el usuario, ya que cada modificación activa una nueva llamada al modelo. Para evitar esta situación, se ha incorporado un botón de "predecir IPC". Aunque no sería estrictamente necesario añadir este botón, su presencia permite controlar el flujo de ejecución mediante un contador interno. Al presionar el botón, se incrementa el contador y se permite ingresar a una condición (if) que muestra por pantalla el IPC predicho.

Estas consideraciones adicionales en el uso de Dash, tanto en la forma de introducir los datos al modelo como en la función de los callbacks, son aspectos importantes a tener en cuenta para garantizar el correcto funcionamiento de la aplicación y brindar una experiencia fluida al usuario.

6. Aplicaciones de políticas económicas basada en la predicción de movimientos.

6.1. Marco teórico de las políticas aplicables.

Para este análisis, a pesar de contar con una amplia gama de políticas aplicables, nos centraremos en las políticas económicas y fiscales. Estas políticas desempeñan un papel fundamental en la gestión y el control de la economía.

Las políticas económicas se refieren a las medidas adoptadas por el gobierno y las autoridades monetarias para influir en los distintos aspectos de la economía.

Esto incluye la política monetaria, que implica el control de la oferta monetaria, las tasas de interés y otras herramientas para regular la actividad económica y mantener la estabilidad financiera.

Por otro lado, las políticas fiscales se centran en la gestión de los ingresos y gastos del gobierno. Esto implica decisiones relacionadas con la recaudación de impuestos, la asignación de recursos públicos y la regulación económica. Estas políticas tienen como objetivo estimular el crecimiento económico, fomentar la equidad y mantener la estabilidad macroeconómica.

(Europeo, Las políticas monetaria y fiscal en una unión monetaria, 2021)

6.1.1. Políticas monetarias.

Dentro de las políticas monetarias, se encuentran dos grandes enfoques: expansiva y restrictiva.

6.1.1.1. Políticas monetarias expansivas.

Las políticas monetarias expansivas son herramientas empleadas por los bancos centrales durante períodos de recesión y depresión económica, con el propósito de fomentar el crecimiento económico y la creación de empleo. Estas políticas se basan en el aumento de la oferta monetaria a través de diferentes instrumentos disponibles.

Una de las medidas clave en la política expansiva es la reducción de las tasas de interés en las operaciones de mercado abierto. Al disminuir los tipos de interés, se promueve un acceso más asequible a la financiación para las entidades financieras. Como resultado, los bancos comerciales pueden transferir esta reducción de tasas a los clientes, lo que se traduce en una disminución de los costos de financiamiento para individuos y empresas.

Este enfoque de reducción de tasas de interés tiene un impacto en cascada en la economía, ya que al hacer que el crédito sea más asequible, se estimula la inversión, el consumo y la demanda agregada. Esto, a su vez, puede impulsar el crecimiento económico y generar empleo.

Es importante destacar que las políticas monetarias expansivas se implementan con cautela y consideración, ya que también pueden tener efectos secundarios, como presiones inflacionarias. Por lo tanto, los bancos centrales deben evaluar cuidadosamente las condiciones económicas y los riesgos asociados antes de aplicar estas medidas.

(Europeo, Las políticas monetaria y fiscal en una unión monetaria, 2021)

6.1.1.2. Políticas monetarias restrictivas.

Las políticas monetarias restrictivas son utilizadas cuando existe un exceso de liquidez en la economía y se busca controlar la inflación.

El principal instrumento empleado por los bancos centrales en este tipo de políticas es el aumento de las tasas de interés. Al encarecer la financiación para los bancos comerciales, se desincentiva la obtención de créditos de los bancos centrales, lo que reduce el capital disponible para prestar. Esta situación también se traslada a

los clientes de las entidades de crédito, quienes experimentarán tasas de interés más altas en sus operaciones de financiamiento

Otras medidas incluyen la reducción de facilidades, como incrementar el tipo marginal de depósito y disminuir el de crédito. Esta acción busca desincentivar las facilidades marginales de crédito, al tiempo que se fomenta el uso de las facilidades de depósito. De esta manera, se busca que las entidades comerciales presten menos y depositen más dinero, lo que reduce la masa monetaria en circulación. También se puede considerar el aumento del coeficiente de caja, lo que obliga a las entidades de crédito a aumentar sus reservas mínimas. Esto reduce su capital disponible y disminuye la cantidad de dinero en circulación.

(Europeo, Las políticas monetaria y fiscal en una unión monetaria, 2021)

6.1.2. Políticas fiscales.

Dentro de las políticas fiscales, se encuentran dos grandes enfoques: expansiva y restrictiva.

6.1.2.1. Políticas fiscales expansivas.

La política fiscal expansiva se basa en dos grandes enfoques: el aumento de las partidas de gasto público en el presupuesto de un país o territorio y la reducción de los impuestos. Estas medidas suelen ser implementadas con el objetivo de estimular la economía, impulsar el crecimiento y generar empleo.

En primer lugar, el aumento del gasto público implica destinar más recursos a áreas como infraestructuras, educación, salud, investigación y desarrollo, entre otros. Estas inversiones buscan impulsar la demanda agregada, ya que generan empleo directo e indirecto, fomentan el consumo y promueven el desarrollo económico a largo plazo.

Por otro lado, la reducción de impuestos tiene como finalidad estimular la actividad económica al proporcionar un alivio fiscal a los ciudadanos y las empresas.

Mediante la disminución de la carga tributaria, se busca incentivar la inversión, aumentar la capacidad adquisitiva de las personas y mejorar la competitividad de las empresas. Esto puede traducirse en un incremento en la demanda de bienes y servicios, así como en la generación de empleo y el fomento de la inversión privada.

Estas medidas suelen ser implementadas de manera simultánea para maximizar su impacto en la economía. Sin embargo, en algunos casos, se pueden aplicar de forma independiente dependiendo de las circunstancias y los objetivos específicos de la política fiscal.

(Europeo, Las políticas monetaria y fiscal en una unión monetaria, 2021)

6.1.2.2. Políticas fiscales restrictivas.

Mediante la implementación de políticas fiscales contractivas, los Estados buscan generar un equilibrio en sus finanzas públicas al enfocarse en aumentar las recaudaciones de impuestos en comparación con el gasto público. Esto implica tomar medidas orientadas a reducir el déficit presupuestario y controlar la deuda pública.

En primer lugar, la reducción del gasto público es una estrategia utilizada para disminuir el nivel de gasto del gobierno en distintas áreas, como subsidios, inversiones y programas sociales. Esta reducción puede implicar recortes en el gasto en infraestructuras, educación, salud y otros sectores, con el objetivo de ajustar las finanzas públicas y lograr un mayor equilibrio entre ingresos y gastos.

Por otro lado, la subida de impuestos es otra medida característica de las políticas fiscales contractivas. Consiste en incrementar la carga tributaria a los ciudadanos y las empresas, lo que implica un aumento en los impuestos sobre la renta, el consumo, la propiedad, entre otros. Esta estrategia busca aumentar las recaudaciones fiscales y reducir los déficits presupuestarios.

Es importante destacar que estas medidas pueden ser aplicadas de manera conjunta o separada, dependiendo de las circunstancias y los objetivos específicos de cada país. En algunos casos, la combinación de la reducción del gasto público y el aumento de impuestos es utilizada para lograr un mayor impacto en la reducción del déficit presupuestario y controlar la deuda pública.

La implementación de políticas fiscales contractivas puede ser necesaria en situaciones en las que el país enfrenta desequilibrios fiscales y un alto nivel de endeudamiento público. Estas medidas tienen como finalidad principal restablecer la sostenibilidad fiscal y mejorar la confianza de los mercados financieros.

(Europeo, Las políticas monetaria y fiscal en una unión monetaria, 2021)

6.2. Implementación de las políticas basándonos en la predicción del IPC.

6.2.1. IPC predicho mayor del deseado

Para reducir la inflación desde el Banco Central Europeo se podría utilizar los siguientes instrumentos de política monetaria restrictiva:

1. El BCE puede aumentar los tipos de interés oficiales. Al elevar los tipos de interés, se encarece el coste del crédito, lo que desalienta el gasto y la inversión de consumidores y empresas. Esto ayuda a controlar la demanda agregada y, por lo tanto, a reducir la presión inflacionaria.

2. Modificación de las facilidades permanentes: El BCE puede ajustar las facilidades permanentes, como la facilidad de depósito y la facilidad de crédito. Por ejemplo, puede aumentar el tipo de interés de la facilidad de depósito, lo que incentiva a los bancos a mantener fondos en el BCE y desincentiva el flujo de dinero hacia la economía real. Asimismo, puede aumentar el tipo de interés de la facilidad de crédito para encarecer el financiamiento de los bancos comerciales y reducir su capacidad de otorgar préstamos.

(Europeo, ¿Qué es el tipo de interés de la facilidad marginal de crédito?, 2018; Europeo, ¿Qué es el tipo de interés de la facilidad marginal de crédito?, 2018) (Banco Central Europeo, 2022)

3. Cambio en los requerimientos de reservas mínimas: El BCE puede modificar los requerimientos de reservas mínimas que los bancos deben mantener. Al aumentar estos requisitos, se reduce la disponibilidad de dinero en el sistema bancario, lo que

limita la capacidad de los bancos para otorgar préstamos y controla la expansión del crédito.

(Banco De España, n.d.)

Para reducir la inflación también se pueden aplicar políticas fiscales restrictivas.

1. Aumento de impuestos: El gobierno puede incrementar los impuestos, especialmente aquellos que gravan el consumo y la importación. Al aumentar los impuestos, se reduce la demanda agregada y se desincentiva el gasto, lo que contribuye a controlar la inflación. Este es el enfoque tradicional de la teoría fiscal, aunque ciertos economistas señalan que no es una medida efectiva.

(ROTELLAR, 2022)

2. Reducción del gasto público: El gobierno puede tomar medidas para reducir el gasto público en áreas no prioritarias. Esto implica recortar el presupuesto en proyectos de infraestructura, programas sociales o cualquier otro tipo de gasto gubernamental. La reducción del gasto público ayuda a controlar la demanda y evita la presión inflacionaria.

(Fondo Monetario Internacional, 2023)

6.2.2. IPC predicho menor del deseado

Para incrementar la inflación desde el Banco Central Europeo se podría utilizar los siguientes instrumentos de política monetaria expansiva:

1. Ajuste de los tipos de interés oficiales: El BCE puede reducir los tipos de interés oficiales en sus operaciones de mercado abierto. Al reducir los tipos de interés, baja el coste del crédito, lo que incentiva el gasto y la inversión de consumidores y empresas. Esto ayuda a incrementar la demanda agregada y, por lo tanto, a incrementar el IPC.

2. Modificación de las facilidades permanentes: El BCE puede ajustar las facilidades permanentes, como la facilidad de depósito y la facilidad de crédito. Por ejemplo, puede reducir el tipo de interés de la facilidad de depósito, lo que desincentiva a los bancos a mantener fondos en el BCE y promueve el flujo de dinero

hacia la economía real. Asimismo, puede disminuir el tipo de interés de la facilidad de crédito para abaratar el financiamiento de los bancos comerciales e incrementar su capacidad de otorgar préstamos.

(Banco Central Europeo, 2022)

3. Cambio en los requerimientos de reservas mínimas: El BCE puede modificar los requerimientos de reservas mínimas que los bancos deben mantener. Al reducir estos requisitos, se aumenta la disponibilidad de dinero en el sistema bancario, lo que aumenta la capacidad de los bancos para otorgar préstamos e incentiva la expansión del crédito.

(Banco Central Europeo, 2022)

4. Programa de compras de activos: El BCE puede llevar a cabo un programa de compras de activos, adquiriendo bonos y otros activos en el mercado estimulando la actividad económica.

(Europeo, ¿Qué es el programa de compras de activos del BCE?, 2022)

Para incrementar la inflación, es decir, cuando la economía está experimentando una situación de deflación o una inflación demasiado baja también se pueden aplicar políticas fiscales expansivas.

1. Reducción de impuestos: El gobierno puede reducir los impuestos sobre el consumo, la renta o las empresas. Al disminuir la carga fiscal, se busca aumentar la disponibilidad de ingresos para los consumidores y las empresas, lo que estimula el gasto y la inversión, impulsando la demanda agregada y potencialmente generando presiones inflacionarias. Como hemos comentado anteriormente depende de la situación específica de la economía.

2. Aumento del gasto público: El gobierno puede aumentar el gasto público en proyectos de infraestructura, programas de inversión, educación o salud. Estos programas de gasto generan empleo, aumentan los ingresos de los trabajadores y fomentan el consumo, lo que contribuye a estimular la demanda agregada y puede impulsar la inflación.

(Fondo Monetario Internacional, 2023)

6.3. Importancia de aplicar de estas medidas a tiempo.

Existen políticas específicas destinadas a estabilizar la economía, conocidas como estabilizadores automáticos. Un ejemplo de ello es el Impuesto sobre la Renta de las Personas Físicas (IRPF) en España, donde a medida que los ingresos de una persona aumentan, también lo hace la recaudación debido a su carácter progresivo. Además, el tramo impositivo se incrementa, lo que provoca que el aumento no sea lineal. Otro ejemplo es la prestación por desempleo. Cuando la economía funciona bien y hay un bajo nivel de desempleo, el gasto público disminuye, ya que hay menos personas que requieren esta ayuda, y los ingresos por concepto de IRPF aumentan. Sin embargo, en situaciones de deterioro económico, con un incremento en el desempleo, el gasto público se eleva debido al aumento en la cantidad de personas que requieren este ayuda, mientras que los ingresos por IRPF disminuyen, ya que hay menos personas trabajando y generando ingresos.

Además de los estabilizadores automáticos existentes, mediante modelos de forecasting, no solo para predecir el Índice de Precios al Consumidor (IPC), sino también para otros indicadores macroeconómicos que escapan de este análisis, podrían desarrollarse otros "estabilizadores automáticos" variables capaces de mantener la economía en equilibrio en condiciones normales, anticipándose a los cambios esperados en dichos indicadores. Adelantarse a estos cambios permitiría que se implementen políticas menos agresivas al mantener estos valores dentro de rangos estándar. Estos nuevos "estabilizadores automáticos" tendrían limitaciones y no evitarían situaciones como la ocurrida durante la pandemia de COVID-19, que aunque hubiesen estado implementados obligatoriamente hubieran conducido a cambios bruscos en las políticas económicas, ya que estas situaciones no muestran ninguna relación con las variables que podrían ser introducidas en un modelo, lo que las hace imposibles de predecir.

(Svensson & Woodford, 2003)

(BBVA, 2018)

7. Concordancia de este trabajo con los Objetivos de Desarrollo Sostenible.

Este trabajo concuerda con los objetivos de desarrollo sostenible, en especial con los objetivos: 8(Trabajo decente y crecimiento económico), 9 (Industria, innovación e infraestructura), 11 (Ciudades y comunidades sostenibles), 12(Producción y consumos responsables).

La capacidad de nuestro modelo para predecir cambios en el IPC permite la adopción anticipada de medidas económicas de manera más eficiente, lo que favorece una correcta asignación de los recursos económicos para contrarrestar dichos cambios. Esta anticipación reduce tanto el gasto económico como el tiempo requerido, lo que nos permite destinar los recursos restantes a otras necesidades más prioritarias. Como resultado, se promueve el desarrollo de ciudades y comunidades más sostenibles.

Además, la optimización de estas medidas económicas favorece el crecimiento económico. El hecho de que nuestro modelo esté disponible para cualquier persona, incluso aquellos sin conocimientos técnicos, a través de una aplicación web, representa un avance tecnológico y una modernización que se alinea con el objetivo de promover la industria, la innovación y la infraestructura.

En resumen, el uso de nuestro modelo de predicción del IPC mediante una aplicación web interactiva ofrece diversas ventajas en términos de eficiencia en la toma de decisiones económicas, asignación de recursos. Además, su accesibilidad y facilidad de uso contribuyen a fomentar la adopción de tecnologías modernas y avanzadas, en línea con los objetivos de desarrollo sostenible relacionados con la industria, la innovación y la infraestructura.

8. Bibliografía

Banco Central Europeo. (21 de Octubre de 2022). ¿Qué es el tipo de interés de la facilidad de depósito? Recuperado el Junio de 2023, de Banco Central Europeo: https://www.ecb.europa.eu/ecb/educational/explainers/tell-me/html/what-is-the-deposit-facility-rate.es.html

Banco De España. (s.f.). ¿Qué son los requerimientos de reservas mínimas? Recuperado el Junio de 2023, de Banco De España: https://www.bde.es/wbe/es/areas-actuacion/politica-monetaria/politica-monetaria-area-euro/tipos-interes-bce/que-son-los-requerimientos-de-reservas-minimas.html

Barrios, J. J., Escobar, J., Leslie, J., Martin, L., & Peña, W. (Octubre de 2021). Recuperado el 20 de Abril de 2023, de Banco interamericano del desarrollo: https://publications.iadb.org/es/nowcasting-para-predecir-actividad-economica-en-tiempo-real-los-casos-de-belice-y-el-salvador

BBVA. (7 de Junio de 2018). ¿Qué son los estabilizadores automáticos de la economía? Recuperado el Junio de 2023, de BBVA: https://www.bbva.com/es/economia-todos-los-estabilizadores-automaticos-la-economia/

Calderon, J. L. (Enero de 2019). Facultad de infromática de Barcelona. Recuperado el Junio de 2023, de https://upcommons.upc.edu/bitstream/handle/2117/131756/137630.pdf

Europeo, B. C. (13 de Septimebre de 2018). ¿Qué es el tipo de interés de la facilidad marginal de crédito? Recuperado el Junio de 2023, de Banco Central Europeo: https://www.ecb.europa.eu/ecb/educational/explainers/tell-me/html/marginal_lending_facility_rate.es.html

Europeo, B. C. (Septiembre de 2021). Las políticas monetaria y fiscal en una unión monetaria. Recuperado el Junio de 2023, de Banco Central Europeo: https://www.ecb.europa.eu/pub/pdf/scpops/ecb.op273~fae24ce432.es.pdf?43454399aaedea571985f4825dfb74e6

Europeo, B. C. (25 de Noviembre de 2022). ¿Qué es el programa de compras de activos del BCE? Recuperado el Junio de 2023, de Banco Central Europeo: https://www.ecb.europa.eu/ecb/educational/explainers/tell-me-more/html/asset-purchase.es.html

Fondo Monetario Internacional. (Abril de 2023). Fondo Monetario Internacional. Recuperado el Junio de 2023, de Chapter 2: Inflation and Disinflation: What Role for Fiscal Policy?: https://www.imf.org/en/Publications/FM/Issues/2023/04/03/fiscal-monitor-april-2023?cid=bl-com-spring2023flagships-FMOEA2023001

Gonzalez, L. (30 de Noviembre de 2018). Regresión Lineal – Teoría. Recuperado el Mayo de 2023, de AprendeIA: https://aprendeia.com/algoritmo-regresion-lineal-simple-machine-learning/

Goodfelow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

GOOGLE INC. (2020). GENERATIVE MODELS FOR EFFECTIVE ML ON PRIVATE, DECENTRALIZED DATASETS. Recuperado el Junio de 2023, de ICLR 2020: https://openreview.net/pdf?id=SJgaRA4FPH

Hernández, A. H. (s.f.). Universidad de la Laguna. Obtenido de https://riull.ull.es/xmlui/bitstream/handle/915/21778/Visulazacion%20e%20interpretacion%20de%20redes%20neuronales%20convolucionales%20mediante%20dropout%20espacial..pdf?sequence=1

Hinestroza Ramírez, D. (2018). Universidad Libre. Recuperado el Junio de 2023, de https://repository.unilibre.edu.co/handle/10901/17289

IAAR, C. d. (s.f.). GitHub. Obtenido de Introducción al Machine Learning : https://iaarbook.github.io/ML/

IBM. (s.f.). ¿Qué es el algoritmo de k vecinos más cercanos? Recuperado el Mayo de 2023, de IBM: https://www.ibm.com/es-es/topics/knn

IBM. (s.f.). ¿Qué son las redes neuronales? Recuperado el Mayo de 2023, de IBM: https://www.ibm.com/es-es/topics/neural-networks?mhsrc=ibmsearch_a&mhq=redes%20neuronales

IBM. (s.f.). What is random forest? Recuperado el Mayo de 2023, de IBM: https://www.ibm.com/topics/random-forest?mhsrc=ibmsearch_a&mhq=random%20forest

Josende, A. P. (Mayo de 2022). Universidad Politécnica de Cataluña Facultad de informática de Barcelona. Recuperado el Junio de 2023, de Machine learning basado en modelos transformer aplicado a la traducción automática: https://upcommons.upc.edu/bitstream/handle/2117/362445/163911.pdf?sequence=1

Kilcommins, S. (8 de Marzo de 2021). Medium. Recuperado el Mayo de 2023, de Plotly Dash — Everything You Need To Know: https://medium.datadriveninvestor.com/plotly-dash-everything-you-need-to-know-bc09a5e45395

Molina-Muñoz, J. (2021). Asociación Colombiana de Facultades de Administración. Recuperado el Junio de 2023, de ANÁLISIS BIBLIOMETRICO DEL USO DE MACHINE LEARNING EN FINANZAS A TRAVÉS DE UN MODELO K-

MEANS: https://pure.urosario.edu.co/ws/portalfiles/portal/47168777/104_Texto_del_art_culo_201_1_10_20211209.pdf

ROTELLAR, J. M. (3 de Octubre de 2022). Los impuestos bajos ayudan a controlar la inflación. Recuperado el Junio de 2023, de Libre Mercado: https://www.libremercado.com/2022-10-03/jose-maria-rotellar-los-impuestos-bajos-ayudan-a-controlar-la-inflacion-6938460/

Svensson, L. E., & Woodford, M. (Mayo de 2003). IMPLEMENTING OPTIMAL POLICY THROUGH INFLATION-FORECAST TARGETING. Recuperado el Junio de 2023, de NBER: https://www.nber.org/system/files/working_papers/w9747/w9747.pdf

Unesco. (2021). Inteligencia artificial y educación: guía para las personas a cargo de formular políticas. Recuperado el 5 de Abril de 2023, de https://unesdoc.unesco.org/ark:/48223/pf0000379376

9. Anexo 1: Código modelo Dash.

```
import dash_bootstrap_components as dbc
import dash_core_components as dcc
from dash.dependencies import Input, Output
import pandas as pd
import plotly.express as px
import pickle




# Lectura de datos
df_predicciones = pd.read_csv("data/df_predicciones.csv")
df_prediccionesTrain = pd.read_csv("data/df_prediccionesTrain.csv")

modelo = pickle.load(open("data/modeloRF", 'rb'))



cont_team=1


# Definicion de funciones
```

```
#Graficas

#FIG1
fig1 = px.line(df_prediccionesTrain, x=range(67), y=['ValorRealTrain',
'ValorPredichoTrain'],
               labels={'variable': 'IPC'})
fig1.update_layout(xaxis_title='IPC',
yaxis_title='Valor',template="plotly_white")



# FIG2
fig2 = px.line(df_predicciones, x=range(17), y=['ValorReal', 'ValorPredicho'],
               labels={'variable': 'IPC'})
fig2.update_layout(xaxis_title='IPC',
yaxis_title='Valor',template="plotly_white")




# Actualizar el diseño del subplot



# Build of the app
app = dash.Dash(__name__,
                external_stylesheets=[dbc.themes.SPACELAB],
                meta_tags=[{"name": "viewport", "content": "width=device-
width"}],
                suppress_callback_exceptions=True)

# Definir el layout de la página inicial
app.layout = html.Div([
    html.Div([
        dbc.NavbarSimple(
            brand="TFG CARLOS GAHETE MORILLO",
            color="dark",
            dark=True,
        ),
    ]),
    dbc.Container([
        dbc.Row([
            html.Center(html.H3("Modelo de Random Forest para la prediccion
del IPC",style={'margin-top':'10px','margin-bottom':'15px',
                                                                    'font-
family': "Times New Roman", 'font-size':'29px'})),
            html.Hr(style={'border': '2px solid gray','border-radius':
'10px'})
            ]),
        dbc.Row([
            dbc.Col([
                html.H5("Tenga en cuenta que este es nuestro MSE en
entrenamiento",style={'margin-top':'20px','font-family': 'Arial',
'color':'#6C6C6C'}),
                dcc.Graph(figure=fig1)
            ]),
            dbc.Col([
                html.H5("Tenga en cuenta que este es nuestro MSE en
test",style={'margin-top':'20px','font-family': 'Arial', 'color':'#6C6C6C'}),
```

```
                dcc.Graph(figure=fig2)
            ])
        ]),
        dbc.Row([
            dbc.Col([
                html.H5("Modelo",style={'margin-top':'20px','font-family':
'Arial', 'color':'#6C6C6C'}),
                dbc.Button("Predecir IPC",id="team_button", color="secondary",
n_clicks=0,
                       style={'margin-top':'15px'}),
                dbc.Input(id="pib", placeholder="PIB TRIMESTRAL EN MILLONES DE
EUROS",
                      type="number",min=0, max=1000000000,
step=1,autocomplete="off",style={'margin-top':'15px','border-width':'2px',

'border-color':'#a0a3a2','border-radius':'5px',

'font-family': "Times New Roman"}),
                dbc.Input(id="ipi", placeholder="INDICE DE PRODUCCION
INDUSTRIAL",
                      type="number",min=0, max=1000000000,
step=1,autocomplete="off",style={'margin-top':'15px','border-width':'2px',

'border-color':'#a0a3a2','border-radius':'5px',

'font-family': "Times New Roman"}),
                dbc.Input(id="euribor", placeholder="EURIBOR EN PORCENTAJE %",
                      type="number",min=-10, max=1000000000,
step=1,autocomplete="off",style={'margin-top':'15px','border-width':'2px',

'border-color':'#a0a3a2','border-radius':'5px',

'font-family': "Times New Roman"}),
                dbc.Input(id="desempleo", placeholder="TASA DE PARO EN
PORCENTAJE %",
                      type="number",min=0, max=1000000000,
step=1,autocomplete="off",style={'margin-top':'15px','border-width':'2px',

'border-color':'#a0a3a2','border-radius':'5px',

'font-family': "Times New Roman"}),

                html.Div(
                    id='ipc',
                    children=[
                        html.H2('Número', style={'font-size': '150px'}),  #
Cambia el tamaño de fuente aquí
                    ],
                    style={'height': '50px', 'border': '5px solid black'}
                )
            ])

        ])
    ])
])


# Definir callbacks
@app.callback(Output('ipc','children'),
```

```
              [Input('team_button','n_clicks'),
              Input('pib','value'),
              Input('ipi','value'),
              Input('euribor','value'),
              Input('desempleo','value')]
)
def createTeamBudget(click,pib,ipi,euribor,desempleo):
    global cont_team
    if click == cont_team:
        cont_team=cont_team+1
        columnas = ['tasaDeParo', 'pib', 'ipi','euribor']
        X = pd.DataFrame([[desempleo,pib,ipi,euribor]], columns=columnas)
        ipc=modelo.predict(X)

        return f'El IPC es: {ipc}'
    else:
        return []



# Run de la aplicación
if __name__ == '__main__':
    app.run_server(debug=True)
```

10. Anexo 2: Código Python.

```
# In[29]:
import pandas as pd
import numpy as np
import datetime as dt
from pandas_datareader import eurostat
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
be2301 = pd.read_csv('be2301.csv', delimiter=',', encoding='latin1')#ipi
si_1_1 = pd.read_csv('si_1_1.csv', delimiter=',', encoding='latin1')
#tasaDeParo,ipc_interanual
be23a = pd.read_csv('be23a.csv', delimiter=',', encoding='latin1')
#pibPreciosCorrientes
be1091=pd.read_excel('euribor1año.xls')#Euribor a 1 año



be2301


# In[30]:


si_1_1


# In[31]:
```

```
be23a


# In[32]:


be1091


# In[3]:


tasaDeParo = si_1_1[['NOMBRE DE LA SERIE','DEPC200540_TSPA.T']]
tasaDeParo


# In[4]:


ipc_interanual = si_1_1[['NOMBRE DE LA SERIE','DIP2021LTVA_E00.M']]
ipc_interanual


# In[5]:


pibPreciosCorrientes = be23a[['NOMBRE DE LA SERIE','DSPC102016CB1QB00_SS1.T']]
pibPreciosCorrientes


# In[6]:


ipi = be2301[['NOMBRE DE LA SERIE','DCIIPI2015IND.M']]
ipi


# In[7]:


euribor=be1091
euribor


# In[ ]:







# In[8]:


import pandas as pd

def extraer_texto_antes_coma(texto):
    if isinstance(texto, str):
        indice_coma = texto.find(',')
        if indice_coma != -1:
```

```
            return texto[:indice_coma].strip()
        else:
            return texto.strip()
    else:
        return texto

# Aplicar la función a la columna 'Columna'
euribor['NOMBRE DE LA SERIE'] = euribor['NOMBRE DE LA
SERIE'].apply(extraer_texto_antes_coma)

# Mostrar el DataFrame resultante
print(euribor)


# In[9]:


be2301


# In[33]:


df_merged = pd.merge(tasaDeParo, pibPreciosCorrientes, on='NOMBRE DE LA
SERIE')
df_merged2 = pd.merge(df_merged, ipi, on='NOMBRE DE LA SERIE')
df_merged3 = pd.merge(df_merged2, euribor, on='NOMBRE DE LA SERIE')
tabla = pd.merge(df_merged3, ipc_interanual, on='NOMBRE DE LA SERIE')

tabla



# In[34]:


tabla.rename(columns={'DEPC200540_TSPA.T': 'tasaDeParo'}, inplace=True)
tabla.rename(columns={'DSPC102016CB1QB00_SS1.T': 'pib'}, inplace=True)
tabla.rename(columns={'DCIIPI2015IND.M': 'ipi'}, inplace=True)
tabla.rename(columns={'DIP2021LTVA_E00.M': 'ipc'}, inplace=True)
tabla.rename(columns={'D_1NBAF472': 'euribor'}, inplace=True)
tabla.rename(columns={'NOMBRE DE LA SERIE': 'fecha'}, inplace=True)

tabla


# In[87]:


tablaIntermedia = tabla.drop(index=tabla.index[:12]) #hasta marzo de 2002 no
hay datos del paro
tablaModelo = tablaIntermedia.dropna() #quitamos Nas
tablaModelo




from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(tablaModelo[['tasaDeParo',
'pib', 'ipi','euribor']], tablaModelo['ipc'], test_size=0.2, random_state=42)

# Crear una instancia del modelo de regresión lineal
modelo = LinearRegression()

# Ajustar el modelo utilizando los datos de entrenamiento
modelo.fit(X_train, y_train)

# Obtener las predicciones del conjunto de entrenamiento y de prueba
predicciones_train = modelo.predict(X_train)
predicciones_test = modelo.predict(X_test)

# Evaluar la calidad del modelo en el conjunto de entrenamiento
score_train = modelo.score(X_train, y_train)
print(score_train)
# Evaluar la calidad del modelo en el conjunto de prueba
score_test = modelo.score(X_test, y_test)
print(score_test)


# In[98]:


import pandas as pd

# Crear un DataFrame con los valores predichos y los valores reales
df_prediccionesTrain = pd.DataFrame({'ValorRealTrain': y_train.astype(float),
'ValorPredichoTrain': predicciones_train})

# Mostrar el DataFrame
print(df_prediccionesTrain)


# In[100]:


import plotly.express as px

# Graficar las dos líneas
fig = px.line(df_prediccionesTrain, x=range(67), y=['ValorRealTrain',
'ValorPredichoTrain'])
fig.update_layout(xaxis_title='IPC', yaxis_title='Valor')


# Mostrar la gráfica
fig.show()


# In[101]:


import pandas as pd

# Crear un DataFrame con los valores predichos y los valores reales
```

```
df_predicciones = pd.DataFrame({'ValorReal': y_test.astype(float),
'ValorPredicho': predicciones_test})

# Mostrar el DataFrame
print(df_predicciones)




df_predicciones.dtypes



import plotly.express as px

# Graficar las dos líneas
fig = px.line(df_predicciones, x=range(17), y=['ValorReal', 'ValorPredicho'])
fig.update_layout(xaxis_title='IPC', yaxis_title='Valor')


# Mostrar la gráfica
fig.show()


# In[94]:


mse_train_regresion = (mean_squared_error(y_train, predicciones_train))
mse_test_regresion = (mean_squared_error(y_test, predicciones_test))
print(mse_train_regresion)
print(mse_test_regresion)


# In[27]:


##REGRESION LINEAL
import pandas as pd
from sklearn.linear_model import LinearRegression
# Convierte la columna de fecha al formato adecuado
#tablaModelo['fecha'] = pd.to_datetime(tablaModelo['fecha'])

# Ordena los datos por fecha ascendente
tablaModelo = tablaModelo.sort_values('fecha')

# Divide los datos en características (X) y variable objetivo (y)
X = tablaModelo[['tasaDeParo', 'pib', 'ipi','euribor']]
y = tablaModelo['ipc']

# Divide los datos en entrenamiento y prueba (sin incluir los últimos 20
valores)
X_train = X[:-20]
y_train = y[:-20]
X_test = X[-20:]
y_test = tablaModelo['ipc'][-20:]
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
# In[28]:


# Crear un DataFrame con los valores predichos y reales
tabla_resultados = pd.DataFrame({'Valor Real': y_test, 'Valor Predicho':
y_pred})

# Imprimir la tabla de resultados
print(tabla_resultados)


# In[89]:


import pandas as pd

# Carga los datos en un DataFrame

# Convertir columnas a tipo numérico
tablaModelo["fecha"] = pd.to_numeric(tablaModelo["fecha"], errors="coerce")
tablaModelo["tasaDeParo"] = pd.to_numeric(tablaModelo["tasaDeParo"],
errors="coerce")
tablaModelo["pib"] = pd.to_numeric(tablaModelo["pib"], errors="coerce")
tablaModelo["ipi"] = pd.to_numeric(tablaModelo["ipi"], errors="coerce")
tablaModelo["ipc"] = pd.to_numeric(tablaModelo["ipc"], errors="coerce")
tablaModelo["euribor"] = pd.to_numeric(tablaModelo["euribor"],
errors="coerce")

#tablaModelo["Valor Anterior"] = pd.to_numeric(tablaModelo["Valor Anterior"],
errors="coerce")

# Ver los tipos de datos actualizados
print(tablaModelo.dtypes)


# In[93]:


# Calcula la matriz de correlación

correlation_matrix = tablaModelo.corr()

# Imprime la matriz de correlación
correlation_matrix



# In[146]:


#print(tablaModelo.dtypes)
tablaModelo


# In[156]:


###RANDOM FOREST
```

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import locale

# Establecer el locale en español
locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')


X_train, X_test, y_train, y_test = train_test_split(tablaModelo[['tasaDeParo',
'pib', 'ipi','euribor']], tablaModelo['ipc'], test_size=0.2, random_state=42)


# Entrenamiento del modelo
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Evaluación del modelo
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

msetrainRF = mean_squared_error(y_train, y_pred_train)
msetestRF = mean_squared_error(y_test, y_pred_test)

print("Error cuadrático medio en train (MSE):", msetrainRF)
print("Error cuadrático medio test (MSE):", msetestRF)

# Crear un DataFrame con los resultados predichos y reales
df_resultados = pd.DataFrame({"Real": y_test, "Predicho": y_pred_test})
print(df_resultados)



# In[173]:


datos_entrada = np.array([[10, 3, 4, 2]])
a=model.predict(datos_entrada)
a


# In[172]:


import pandas as pd

# Crear un DataFrame con los valores predichos y los valores reales
df_prediccionesTrain = pd.DataFrame({'ValorRealTrain': y_train.astype(float),
'ValorPredichoTrain': y_pred_train})
df_prediccionesTrain
# Crear un DataFrame con los valores predichos y los valores reales
df_predicciones = pd.DataFrame({'ValorReal': y_test.astype(float),
'ValorPredicho': y_pred_test})


# In[158]:


import plotly.express as px
```

```
# Graficar las dos líneas
fig = px.line(df_prediccionesTrain, x=range(67), y=['ValorRealTrain',
'ValorPredichoTrain'])
fig.update_layout(xaxis_title='IPC', yaxis_title='Valor')


# Mostrar la gráfica
fig.show()


# In[159]:


import plotly.express as px

# Graficar las dos líneas
fig = px.line(df_predicciones, x=range(17), y=['ValorReal', 'ValorPredicho'])
fig.update_layout(xaxis_title='IPC', yaxis_title='Valor')


# Mostrar la gráfica
fig.show()


# In[162]:


import pickle

pickle.dump(model,open("modeloRF", 'wb'))
df_predicciones.to_csv("df_predicciones.csv", index=False)


# In[163]:


df_prediccionesTrain.to_csv("df_prediccionesTrain.csv", index=False)


# In[160]:


importance = model.feature_importances_

# Crea un DataFrame con las características y su importancia
feature_importance = pd.DataFrame({"Característica": X.columns, "Importancia":
importance})
feature_importance = feature_importance.sort_values("Importancia",
ascending=False)

# Imprime la tabla de importancia de características
print(feature_importance)




###MODELO KNN
```

```
# In[140]:


import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(tablaModelo[['tasaDeParo',
'pib', 'ipi','euribor']], tablaModelo['ipc'], test_size=0.2, random_state=42)

# Lista para almacenar los valores de MSE
mse_values = []

# Rango de valores de k
k_values = range(1, 21)

# Iterar sobre diferentes valores de k
for k in k_values:
    # Crear y ajustar el modelo KNN
    model = KNeighborsRegressor(n_neighbors=k)
    model.fit(X_train, y_train)

    # Realizar predicciones en el conjunto de prueba
    y_pred = model.predict(X_test)

    # Calcular el MSE y agregarlo a la lista
    mse = mean_squared_error(y_test, y_pred)
    mse_values.append(mse)

# Graficar la gráfica del codo
plt.plot(k_values, mse_values, 'bo-')
plt.xlabel('Número de vecinos (k)')
plt.ylabel('Error cuadrático medio (MSE)')
plt.title('Gráfica del codo en KNN')
plt.show()


# In[142]:


import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

X_train, X_test, y_train, y_test = train_test_split(tablaModelo[['tasaDeParo',
'pib', 'ipi','euribor']], tablaModelo['ipc'], test_size=0.2, random_state=42)


# Construir el modelo KNN
model = KNeighborsRegressor(n_neighbors=7)
model.fit(X_train, y_train)

# Evaluar el modelo
predictionsTain = model.predict(X_train)
predictions = model.predict(X_test)
mseTestKnn = mean_squared_error(y_test, predictions)
mseTrainKnn = mean_squared_error(y_train, predictionsTain)
```

```
r2 = r2_score(y_test, predictions)

print("Predicciones:", predictions)
print("Error cuadrático medio train  (MSE):", mseTrainKnn)
print("Error cuadrático medio test(MSE):", mseTestKnn)

print("Coeficiente de determinación (R2):", r2)


# In[143]:


import pandas as pd

# Crear un DataFrame con los valores predichos y los valores reales
df_prediccionesTrain = pd.DataFrame({'ValorRealTrain': y_train.astype(float),
'ValorPredichoTrain': predictionsTain})
# Crear un DataFrame con los valores predichos y los valores reales
df_predicciones = pd.DataFrame({'ValorReal': y_test.astype(float),
'ValorPredicho': predictions})


# In[166]:


import plotly.express as px

# Graficar las dos líneas
fig = px.line(df_prediccionesTrain, x=range(67), y=['ValorRealTrain',
'ValorPredichoTrain'],
              p)
fig.update_layout(xaxis_title='IPC', yaxis_title='Valor')


# Mostrar la gráfica
fig.show()


# In[165]:


import plotly.express as px

# Graficar las dos líneas
fig = px.line(df_predicciones, x=range(17), y=['ValorReal', 'ValorPredicho'],
              labels={'variable': 'IPC'})
fig.update_layout(xaxis_title='IPC', yaxis_title='Valor')


# Mostrar la gráfica
fig.show()


# In[147]:


import plotly.express as px
import pandas as pd
```

```
# Crear un DataFrame con los valores de MSE en entrenamiento y prueba
data = pd.DataFrame({
    'Modelo': ['Regresión Lineal', 'Random Forest','KNN' ],
    'Entrenamiento': [mse_train_regresion, msetrainRF, mseTestKnn],
    'Prueba': [mse_test_regresion, msetestRF, mseTestKnn]
})

# Crear la gráfica de barras
fig = px.bar(data, x='Modelo', y=['Entrenamiento', 'Prueba'],
             title='MSE en Entrenamiento y Prueba para diferentes modelos',
             labels={'value': 'MSE', 'variable': 'Conjunto de Datos'})

# Mostrar la gráfica
fig.show()


# In[148]:


import plotly.express as px
import pandas as pd

# Crear un DataFrame con los valores de MSE en entrenamiento y prueba
data = pd.DataFrame({
    'Modelo': ['Regresión Lineal', 'Random Forest','KNN' ],
    'Entrenamiento': [mse_train_regresion, msetrainRF, mseTestKnn],
    'Prueba': [mse_test_regresion, msetestRF, mseTestKnn]
})

# Crear la gráfica de barras
fig = px.bar(data, x='Modelo', y=['Entrenamiento'],
             title='MSE en Entrenamiento ',
             labels={'value': 'MSE', 'variable': 'Conjunto de Datos'})

# Mostrar la gráfica
fig.show()



import plotly.express as px
import pandas as pd

# Crear un DataFrame con los valores de MSE en entrenamiento y prueba
data = pd.DataFrame({
    'Modelo': ['Regresión Lineal', 'Random Forest', 'KNN'],
    'Entrenamiento': [mse_train_regresion, msetrainRF, mseTestKnn],
    'Prueba': [mse_test_regresion, msetestRF, mseTestKnn]
})

# Crear la gráfica de barras con barras de color rojo
fig = px.bar(data, x='Modelo', y=['Prueba'],
             title='MSE en Prueba',
             labels={'value': 'MSE', 'variable': 'Conjunto de Datos'},
             color_discrete_sequence=['red'])  # Establecer color rojo para
las barras

# Mostrar la gráfica
fig.show()
```

```
mse_train_regresion
mse_test_regresion
msetrainRF
msetestRF
mseTestKnn
mseTrainKnn
```