



MEMORIA DEL PROYECTO FINANCIAL KEEPCODERS

Transformando las finanzas con IA

Proyecto realizado por:

Carles Matoses Miret

Sara Díaz-P. Martín

Bootcamp en Big Data, IA y ML – KeepCoding

19 de septiembre de 2024

Este documento tiene como fin informar al profesorado de KeepCoding sobre el trabajo técnico que hemos realizado en la fase de procesamiento y modelado del proyecto final de Bootcamp.

Este proyecto ha sido todo un reto para nosotros, y creemos que hemos alcanzado la meta que nos propusimos en esta primera versión. Nuestro objetivo era poder automatizar el proceso de extracción, transformación y carga de datos (ETL), seguido del análisis y modelado basado en estos datos, utilizando herramientas como TensorFlow, scikit-learn y otras librerías populares en el ámbito del análisis de datos y Machine Learning, con el fin de **mejorar el análisis de los mercados financieros**.

Empezaremos explicando la arquitectura seguida para la obtención de datos y continuaremos la memoria con el procesamiento y modelado.

Arquitectura

1. Creación de una cuenta de Google para la colaboración en el proyecto:

El primer paso en nuestro proyecto fue la creación de una cuenta de Google. Esta cuenta nos sirvió para dos propósitos clave: almacenar y compartir archivos mediante Google Drive y configurar un repositorio colaborativo en GitHub. Además, esta cuenta de Google fue esencial para acceder a Google Cloud Platform (GCP), un servicio crucial para el procesamiento y almacenamiento de datos durante el proyecto.

2. Creación de un repositorio en GitHub:

Como parte de nuestra estrategia de colaboración, configuramos una cuenta en GitHub para utilizarla como repositorio común. Este repositorio centralizado permitió la gestión del código fuente de manera organizada y controlada, habilitando el uso de git como sistema de control de versiones.

3. Creación de una cuenta en Google Cloud Platform (GCP):

El siguiente paso fue la creación de una cuenta en GCP, que se convirtió en la plataforma principal para nuestras tareas de procesamiento y almacenamiento de datos. Dentro de GCP, creamos un proyecto específico para centralizar todos los recursos y servicios relacionados con nuestro trabajo. Además, configuramos una service account dentro de GCP. Esta service account es una identidad asociada con privilegios específicos que permitía a nuestros scripts y aplicaciones acceder de manera segura a los servicios de Google, especialmente a BigQuery, sin necesidad de una intervención manual constante. Fue un paso clave para automatizar procesos y asegurar que todo funcionara de manera fluida.

4. Arquitectura de BigQuery: Datasets en medallas (Bronze, Plata, Oro):

Una vez creada la cuenta en GCP y la service account, procedimos a trabajar en la parte de la arquitectura de datos dentro de BigQuery. Seguimos una estrategia de almacenamiento basada en una arquitectura de medallas (Bronze, Silver, Gold), un enfoque que categoriza los datos en diferentes fases según su nivel de procesamiento:

- Dataset Bronce (Bronze): El primer dataset que configuramos fue el de bronce, que contiene los datos en su forma más cruda. Aquí almacenamos los datos tal como los recibimos de diversas APIs. Estos datos no se han sometido a ningún tipo de procesamiento o transformación, lo que nos asegura tener una fuente confiable y completa de los datos originales.
- Dataset Plata (Silver): En esta fase intermedia, los datos se transforman y enriquecen. Creamos el dataset Silver, donde aplicamos las transformaciones necesarias a los datos crudos de Bronce para limpiar, filtrar, y estructurar la información. Este proceso incluyó operaciones como la normalización de datos, la eliminación de duplicados y el manejo de valores faltantes. El objetivo aquí es tener datos más consistentes y utilizables.
- Dataset Oro (Gold): Finalmente, creamos el dataset de oro, que contiene los datos listos para ser usados en modelos de machine learning y reportes. En esta etapa, los datos han sido completamente procesados, refinados y están en su forma óptima. Estos datos finales son los que se utilizaron para entrenar los modelos predictivos de machine learning y, posteriormente, para generar visualizaciones y reportes en Power BI.

5. Interacción con BigQuery y carga incremental:

Para simplificar la interacción con BigQuery, creamos una clase que nos permitía realizar operaciones comunes de manera eficiente. Esta clase facilitó la lectura de datos de las tablas almacenadas en los distintos datasets, así como la creación de nuevas tablas y la carga de datos en BigQuery. Gracias a esta abstracción, se redujo significativamente la complejidad del código, lo que permitió un manejo más claro y eficiente de las tareas ETL.

Dado el gran volumen de datos históricos que manejamos, implementamos funciones específicas que permitían realizar cargas incrementales en BigQuery. Estas funciones verificaban cuáles eran los últimos datos cargados en las tablas de cada ticker, consultando únicamente los días que no estaban almacenados previamente. Este enfoque optimizó considerablemente el proceso, ya que evitaba la duplicación de datos y reducía el tiempo de consulta a las APIs y de carga de datos en BigQuery, mejorando la eficiencia de nuestras ETLs.

Fuentes de datos utilizadas

Durante el desarrollo del proyecto, recopilamos información financiera y macroeconómica clave a través de diversas APIs externas, que sirvieron como base para nuestras fases de análisis y modelado. Las principales APIs empleadas fueron yfinance, yahoo_fin, y la Fred API, las cuales nos proporcionaron datos variados desde el comportamiento histórico de las acciones hasta indicadores macroeconómicos relevantes.

1. Yfinance y Yahoo_fin APIs:

Para la obtención de datos financieros, utilizamos conjuntamente las APIs de yfinance y yahoo_fin, que nos proporcionaron acceso a una gran cantidad de información financiera a partir de Yahoo Finance. Estas APIs resultaron ser complementarias, permitiéndonos extraer los siguientes datos:

- Precios y valores de acciones: Ambas APIs nos proporcionaron datos históricos y actuales de precios de acciones. Entre la información obtenida se incluían los precios de apertura, cierre, máximos, mínimos y el volumen de transacciones para múltiples acciones de distintas empresas.
- Información fundamental de empresas: También obtuvimos información fundamental sobre las empresas, como la capitalización de mercado o los dividendos.
- Precios históricos de divisas: Las APIs de yfinance y yahoo_fin nos permitieron acceder a precios históricos de diversas divisas, proporcionando datos clave para el análisis del mercado de divisas y las fluctuaciones en los tipos de cambio. Esta información fue útil para estudiar la correlación entre las divisas y el mercado accionario.

2. Fred API:

Además de los datos financieros obtenidos de Yahoo Finance, complementamos nuestro análisis con datos macroeconómicos a través de la Fred API (Federal Reserve Economic Data), proporcionada por el Banco de la Reserva Federal de St. Louis. Esta API nos permitió acceder a indicadores económicos clave de Estados Unidos, tales como:

- Datos de desempleo: Recogimos datos históricos y actuales sobre las tasas de desempleo.
- Producto Interno Bruto (PIB): Obtuvimos datos sobre el crecimiento del PIB, que fue utilizado para analizar el estado general de la economía de EE. UU. y su impacto en los movimientos del mercado.
- Otros indicadores macroeconómicos: Además, la Fred API nos proporcionó acceso a indicadores como la inflación, tipos de interés, y la deuda pública, entre otros.

Repositorios DataWarehouse

Bronze

- bronze_currency_data.ipynb: Se obtienen datos históricos de distintas divisas
- bronze_macro_data.ipynb: Se obtienen datos históricos de datos macroeconómicos de estados unidos
- bronze_ticker_data.ipynb: Se obtienen datos históricos de acciones de estados unidos
- bronze_ticker_info.ipynb: Se obtiene información concreta de las distintas empresas que están en las distintas bolsas de EEUU.

Silver

- silver_currency_data.ipynb: Se pivota el campo currency para que luego pueda unirse al dataframe que se utilizará para los modelos de machines learning y IA
- silver_macro_data.ipynb: Se pivota el campo que los datos macroeconómicos para que luego pueda unirse al dataframe que se utilizará para los modelos de machines learning y IA
- silver_indicators.ipynb: Se calculan los indicadores técnicos y otras variables para que luego unirlo al dataframe que se utilizará para los modelos de machines learning y IA

- silver_ticker_info.ipynb: Se reduce el número de tickers a los que tengan más de 9 años de histórico, además de filtrar los campos que creemos más relevantes

Gold

- gold_dim_ticker_info_sp500.ipynb: No se aplican más transformaciones, solamente se filtran los valores del sp500. Se crea para tener en Gold las tablas necesarias para Power BI y los modelos.
- gold_main_sp500.ipynb: Tabla final que reúne la información de las 4 tablas de Silver. Es el DataFrame inicial que se utiliza para los modelos de machine learning y IA

Clustering – Se usará más adelante en modelado

- ml_clustering_sp500: Modelo de Clustering que hace un K-Means de las series temporales de las acciones. Nos ha permitido hacer clusters con empresas que presentan una gráfica similar desde 2021.

Modelado: Iniciando el proyecto

1. Importación de librerías:

Comenzamos importando diversas librerías que son fundamentales para el manejo de datos, modelado y visualización. Estas incluyen librerías de procesamiento de datos como pandas y numpy, herramientas para el análisis técnico financiero como talib, procesamiento de lenguaje natural (NLP) con transformers y nltk, y librerías como scikit-learn y tensorflow para la construcción de modelos de Machine Learning y Deep Learning. También se incorporan librerías para la visualización de datos como matplotlib y seaborn.

2. Carga de variables de entorno:

A continuación, cargamos un archivo .env con variables de entorno necesarias para configurar la conexión con servicios externos, como BigQuery. Utilizamos la librería dotenv para gestionar esto. Aquí cargamos nuestro archivo “gold_main.csv” por agilizar el acceso a datos si queremos más rapidez computacional.

3. Conexión a BigQuery:

Seguidamente, configuramos la conexión a Google BigQuery, usando las librerías google.cloud.bigquery y google.oauth2.service_account para autenticar y acceder a los datos en la nube. Definimos algunos parámetros importantes, como el proyecto, dataset y tabla que queremos consultar.

Preprocesamiento básico y selección de acciones

Para empezar con nuestro dataframe (df) inicial en bruto, pasamos a los siguientes pasos esenciales:

1. Carga de datos:

Primero, se define una función encargada de cargar los datos desde un archivo CSV previamente generado, que contiene los resultados de la consulta ejecutada en BigQuery. Esta función devuelve un dataframe que será utilizado para el preprocesamiento.

2. Eliminación de duplicados:

Otra función clave se encarga de limpiar el dataframe, eliminando duplicados en las filas para asegurar que solo se mantengan entradas únicas a través de la fecha. Esto es crucial cuando se trabaja con grandes volúmenes de datos provenientes de distintas fuentes, donde es común que existan duplicidades.

3. Alineación de fechas:

En algunos casos, los datos provenientes de distintas fuentes pueden no estar completamente alineados temporalmente. Para resolver esto, se incluye una función que organiza los datos por fechas, alineando correctamente los registros.

Este preprocesamiento básico garantiza que los datos estén limpios y estructurados antes de pasar a las siguientes fases del análisis, asegurando que los datos tengan la calidad necesaria para que los modelos trabajen de manera efectiva. Los siguientes pasos son:

1. Selección de tickers:

Una de las funciones críticas en el archivo es la que limita la cantidad de tickers (símbolos bursátiles) a procesar. Esto se realiza por motivos de eficiencia computacional, ya que trabajar con una gran cantidad de tickers puede requerir una cantidad significativa de memoria y tiempo de procesamiento. Procesar muchos tickers de manera simultánea puede hacer que los tiempos de ejecución se alarguen considerablemente, además de aumentar el uso de recursos (CPU, RAM). Esta función garantiza que el análisis sea manejable en términos de tiempo y recursos, mientras sigue permitiendo obtener resultados útiles. En este caso, hemos limitado el número de tickers a 4.

2. Inputs para selección del sector y del cluster:

El archivo incluye un sistema de inputs donde el usuario puede seleccionar el sector y el cluster que desea analizar. A partir de esta selección, se eligen al azar cuatro acciones del sector y cluster indicados. Esto permite que el análisis sea más enfocado y personalizado según los intereses del usuario.

3. Selección manual de tickers:

En caso de que el usuario prefiera analizar acciones específicas en lugar de elegir las al azar, puede definir los cuatro tickers directamente en el archivo "variables.env", rellenando la variable `tickers_random` en forma de lista. Esto brinda flexibilidad al proceso, permitiendo tanto la selección aleatoria como el análisis de tickers concretos según la preferencia del usuario.

4. **Filtrado por volumen:**

Otra función importante es la que filtra los tickers basándose en el volumen medio de las transacciones del sector. El objetivo de esta función es eliminar aquellos tickers cuyo volumen sea inferior al promedio del sector, bajo la premisa de que tickers con bajo volumen pueden no ser representativos o tener datos poco consistentes para el análisis.

Análisis exploratorio y Preprocesamiento

Una vez que tenemos las acciones que queremos predecir y nuestro dataframe filtrado por ellas, pasamos a complementar con nuevas variables independientes y a preprocesar nuestro dataframe para su limpieza:

1. **Cálculo de indicadores técnicos:**

Es una de las partes más fundamentales del procesamiento, la creación de indicadores técnicos con la ayuda de la librería TA-Lib. Entre los indicadores más utilizados se incluyen las medias móviles, RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), entre otros. Patrones de vela que nos ayudan a las señales de compra e indicadores de volatilidad y volumen.

2. **Variables dependientes:**

Generamos las variables a predecir en cada problema: clasificación o regresión. En regresión nuestra variable dependiente será "Log_Return" (que son los retornos diarios en formato logarítmico) mientras que en clasificación nuestra variable dependiente será "Target" que refleja si la acción ha subido o caído respecto al día anterior. Estos indicadores se calculan para cada ticker y se añaden al dataframe, proporcionando información clave para el análisis financiero y la posterior modelización.

3. **Clustering con K-means:**

A través de la volatilidad general del dataframe, generamos un régimen de mercado que nos indica si está en etapa alcista, bajista o lateral, lo que nos aporta una variable con información muy interesante para el análisis.

4. **Generación de noticias sintéticas:**

Generamos noticias sintéticas utilizando nltk, una herramienta potente para el procesamiento de lenguaje natural. Estas noticias se crean para simular eventos o situaciones relacionadas con los activos financieros (tickers), de manera que se pueda tener más información textual sobre los mismos.

5. **Análisis de sentimiento con FinBERT:**

Una vez generadas las noticias sintéticas, se utiliza FinBERT (algoritmo de Hugging Face) para realizar un análisis de sentimiento sobre ellas. Este modelo especializado en el ámbito financiero clasifica las noticias en categorías como "positiva", "negativa" o "neutral", basándose en su contenido. Con esto, conseguimos implementar la técnica de "Transfer Learning" a nuestro problema.

6. Matriz de correlación:

Representamos una matriz de correlación para identificar relaciones entre las diferentes variables. Esta matriz muestra qué tan relacionadas están las características entre sí, lo cual es útil para identificar posibles redundancias o patrones que puedan interferir en el desempeño del modelo. Las correlaciones altas entre las variables pueden indicar multicolinealidad, un problema que afecta la estabilidad de ciertos modelos. Identificamos las correlaciones por encima o por debajo del umbral de 0.8 (Kennedy, 2008; Pal and Soriya, 2012) para tratarlas en el preprocesamiento.

División de la muestra, escalado y validación walk-forward

El siguiente paso será la división de la muestra en Train y en Test, evitando la fuga de datos en todo momento:

1. División en conjuntos de entrenamiento (Train) y prueba (Test):

El primer paso en este proceso es dividir los datos en dos conjuntos: uno para entrenar los modelos y otro para evaluarlos. Se hace con la intención de que los modelos puedan aprender patrones en los datos del conjunto de entrenamiento y luego se evalúe su capacidad para generalizar a datos nuevos, en este caso, el conjunto de prueba:

- **Entrenamos con 3 acciones (Train):** El archivo está configurado para que los modelos se entrenen utilizando los datos de tres acciones (tickers). Esto permite que los modelos aprendan de estos activos específicos.
- **Predicción con la 4ª acción (Test):** La evaluación se realiza utilizando los datos del cuarto ticker. Este enfoque garantiza que el modelo esté expuesto a nuevos datos no vistos durante el entrenamiento, lo que es fundamental para validar su capacidad predictiva. Esto nos puede ayudar a la generalización ya que se valida la capacidad del modelo frente a datos no vistos, simulando así un escenario realista donde se desea predecir el comportamiento de una acción desconocida.

2. Escalado de variables:

En esta etapa, se manejan aspectos como la normalización y escalado de las características. El escalado se realiza para que todas las variables estén en la misma escala y se evite que algunas con valores mayores dominen el modelo.

- **Separación de variables objetivo y características:** En este paso se dividen las características (inputs) y las variables objetivo (output) en ambos conjuntos de datos (Train y Test), para que el modelo pueda aprender a mapear las entradas con las salidas deseadas, según el problema específico de regresión o clasificación.
- **Escalado con MinMaxScaler:** Se usa para normalizar las variables entre un rango específico, normalmente entre 0 y 1. Este paso es importante en modelos basados en algoritmos como redes neuronales o regresión, donde las diferencias de escala pueden influir en los resultados. **Aquí tenemos que diferenciar entre fit_transform y transform:**

- **fit_transform:** Este método se usa en el conjunto de entrenamiento. Lo que hace es calcular los parámetros (como los valores mínimos y máximos) basados solo en los datos del conjunto de entrenamiento, y luego aplica la transformación correspondiente. Solo se debe aplicar el fit_transform sobre el conjunto de entrenamiento para evitar la fuga de datos. Si lo aplicamos en el conjunto de test, se estaría utilizando información de los datos de Test para transformar los datos de Train, lo que influiría negativamente en la evaluación del modelo. Este paso es crucial para asegurar que el modelo no se vea afectado por los datos del futuro (test), que no estarían disponibles al momento de entrenar.
- **transform:** Una vez que hemos calculado el escalado con fit_transform en el conjunto de entrenamiento, usamos transform en el conjunto de prueba. Este método aplica la misma transformación (usando los parámetros obtenidos en Train) a los datos de Test sin recalcular los valores mínimos y máximos. Usar solo transform en el conjunto de Test asegura que el modelo utilice los mismos parámetros que se calcularon en Train, simulando un escenario real en el que se trabaja con datos nuevos sin haber visto las respuestas.

Todo esto es importante para evitar la fuga de datos:

Uno de los aspectos más importantes en este flujo de trabajo es prevenir la **fuga de datos**, un problema que ocurre cuando la información del conjunto de prueba "se filtra" de alguna manera en el conjunto de entrenamiento, lo que puede llevar a un rendimiento inflado de los modelos.

- **Procesar primero en Train y luego en Test:** El procesamiento de los datos, como el escalado o la selección de características, **debe hacerse primero en el conjunto de entrenamiento**. Esto asegura que el modelo aprenda únicamente de los datos disponibles en Train y no esté expuesto a la información del conjunto de test. Posteriormente, el mismo procesamiento que se aplicó en Train se replica en test, usando los parámetros calculados con Train. Si se aplicara el procesamiento directamente sobre Test junto con Train, se correría el riesgo de fuga de datos, lo que invalidaría la evaluación del modelo.
- **Por ejemplo**, si se escalasen los datos con valores que incluyen tanto Train como test, el modelo estaría expuesto a información de Test que, en un escenario real, no estaría disponible al hacer predicciones.

3. Validación Walk-Forward en series temporales financieras:

Cuando trabajamos con **series temporales financieras**, como el precio de las acciones, la validación de los modelos es un desafío mayor que en otros tipos de datos debido a la naturaleza secuencial del tiempo. Para abordar este reto, usamos una técnica llamada **validación walk-forward**. La validación walk-forward es una metodología que respeta el orden temporal de los datos. En lugar de realizar una simple división en Train y

test, como se haría con datos no secuenciales, se crean múltiples divisiones o **splits** del conjunto de datos en periodos de tiempo consecutivos. Esto simula un escenario en el que se entrena un modelo con los datos pasados y se evalúa con datos futuros.

- En el código, la validación walk-forward se implementa dividiendo los datos financieros en múltiples **splits temporales**. Cada split contiene un conjunto de entrenamiento (pasado) y un conjunto de prueba (futuro), respetando siempre la secuencia temporal de los datos.
- **Por ejemplo**, si tenemos datos históricos de 10 años, podríamos dividirlos en splits anuales. En el primer split, entrenamos el modelo con los datos de los primeros 5 años y lo probamos con el año 6. En el segundo split, entrenamos con los primeros 6 años y probamos con el año 7, y así sucesivamente.

4. Estructura combinada de los datos:

Una vez que se aplica la técnica de validación walk-forward, los datos se dividen en varios intervalos temporales (splits). Después de realizar el entrenamiento en los primeros splits y la evaluación en los splits futuros, se combinan los datos procesados de manera que puedan ser utilizados eficientemente por el modelo.

¿Qué hemos hecho en el proyecto para garantizar esta consistencia?

- **Almacenamiento de datos por splits:** Cada uno de los bloques de entrenamiento y prueba (Train y test) se estructura de manera separada. Esto garantiza que el modelo pueda ser entrenado en datos históricos y luego evaluado en datos futuros de una manera secuencial, manteniendo la consistencia temporal de los datos financieros.
- **Recombinar los datos:** Tras realizar los múltiples pasos de validación walk-forward, los datos son reestructurados en un formato único que el modelo pueda consumir para su entrenamiento final y validación. Esto asegura que, independientemente de cómo se hayan dividido los splits, los datos puedan ser reunidos y pasados al modelo de forma coherente, sin perder su estructura temporal.

Modelado

Una vez que tenemos nuestros datos limpios, preprocesados, escalados y con la aplicación de la validación walk-forward, llegamos al paso del modelado. En el código, hemos implementado el entrenamiento de modelos tanto para **clasificación** como para **regresión**, con el objetivo de abordar diferentes tipos de problemas en el análisis de datos financieros. Dependiendo del objetivo final (clasificación de señales de compra/venta o predicción de precios futuros), se seleccionan distintos modelos de **Machine Learning** y **Deep Learning**. A continuación, se resume cuáles hemos utilizado y por qué:

1. Modelos de clasificación:

Los modelos de clasificación se utilizan para predecir **clases discretas**, como señales de compra o venta en función de las características de los datos. El enfoque está en aprender patrones que determinen cuándo realizar una acción en el mercado basado en indicadores técnicos, sentimientos financieros y otros factores.

- **Random Forest Classifier:**

- Es un modelo basado en la combinación de varios árboles de decisión, que ofrece robustez frente al sobreajuste y suele funcionar bien en problemas donde se requiere identificar patrones complejos a partir de variables como indicadores técnicos.
- Permite la interpretación de la importancia de las características, lo que es útil para identificar qué variables influyen más en la clasificación (por ejemplo, indicadores como MACD, RSI).
- Maneja bien tanto características numéricas como categóricas, lo que lo hace adecuado para la mezcla de variables técnicas y fundamentales.

- **Redes Neuronales para clasificación (LSTM y GRU):**

- Las redes neuronales, especialmente en problemas de clasificación financiera, pueden capturar relaciones no lineales complejas entre las variables, lo que es crucial en series temporales con patrones de comportamiento no evidentes.
- Se utiliza una red con varias capas ocultas y activaciones no lineales para capturar las interacciones entre los diferentes indicadores técnicos y el sentimiento del mercado.
- En este caso, la red está diseñada para clasificar eventos en clases discretas (señales de compra/venta).

- **Conv1D (Convolución 1D):**

- El modelo **Conv1D** es una arquitectura de redes neuronales convolucionales adaptada para trabajar con datos secuenciales. Aunque las CNN son más conocidas por su éxito en el procesamiento de imágenes, las convoluciones 1D pueden ser muy útiles en problemas de series temporales como las finanzas, ya que pueden identificar patrones locales en la secuencia de datos.
- **Captura de patrones locales:** Las convoluciones 1D son excelentes para detectar patrones locales o características en los datos secuenciales, como subidas o bajadas rápidas de precios en una ventana temporal.

→ Métricas utilizadas en clasificación:

- **Precisión (accuracy):** Evalúa el porcentaje de predicciones correctas.
- **F1-Score:** Es una métrica robusta que equilibra precisión y sensibilidad (importante en datos financieros donde los falsos positivos y negativos tienen un impacto significativo).

- **Matriz de Confusión:** Ayuda a entender cómo el modelo está clasificando las señales de compra/venta.

2. Modelos de regresión:

Los modelos de regresión se utilizan para predecir **valores continuos**, como el precio futuro de una acción o el retorno logarítmico como es nuestro caso. El objetivo es generar predicciones precisas sobre valores de precios a partir de indicadores técnicos y otras variables.

- **Random Forest Regressor:**
 - Es una versión del Random Forest adaptada a la regresión. Se selecciona debido a su capacidad para manejar grandes cantidades de características y capturar interacciones no lineales entre las variables. Además, es robusto frente al sobreajuste y maneja bien los datos con ruido, lo cual es común en los datos financieros.
 - Genera múltiples árboles de decisión y promedia sus predicciones, lo que ayuda a mitigar la volatilidad de los datos de mercado.
 - Al igual que en la clasificación, permite analizar la importancia de las características para entender qué variables influyen más en la predicción del precio.
- **Redes Neuronales para regresión (LSTM y GRU):**
 - Las redes neuronales tipo LSTM (Long Short-Term Memory) son ideales para problemas de **series temporales**, como la predicción de precios financieros, ya que pueden recordar patrones de largo plazo en los datos secuenciales.
 - Hemos utilizado una arquitectura recurrente (LSTM) que puede capturar la secuencia temporal de los datos y prever precios futuros en función de precios pasados y otros indicadores técnicos.
 - LSTM permite manejar la dependencia temporal entre los datos financieros, mejorando la capacidad de predecir movimientos de precios a lo largo del tiempo.
 - El modelo **GRU** es una variante de las redes neuronales recurrentes (RNN) diseñada para mejorar la capacidad de capturar dependencias a largo plazo en las series temporales, al igual que LSTM, pero con una estructura más sencilla y menos parámetros.
 - **Tiene menor complejidad que LSTM:** Al ser más simple en su arquitectura que LSTM, las GRU suelen entrenarse más rápido y requieren menos recursos computacionales, lo que es ideal para cuando se quiere balancear rendimiento y eficiencia.
 - Hemos implementado una red GRU para capturar las relaciones temporales en los datos, especialmente cuando se trabaja con secuencias largas de precios y otros indicadores.

→ Métricas utilizadas en regresión:

- **MSE (Mean Squared Error):** Se utiliza para medir el error cuadrático medio entre los valores predichos y los valores reales. Es útil para capturar la magnitud de las predicciones erróneas.
- **R2 Score:** Indica qué tan bien las predicciones del modelo se ajustan a los datos reales. Un R2 cercano a 1 indica un buen ajuste.

3. Modelos híbridos:

En el proyecto también hemos experimentado con modelos híbridos, que combinan diferentes arquitecturas (por ejemplo, **LSTM, GRU y Random Forest**) para aprovechar las ventajas de cada uno y mejorar el rendimiento del modelo general. Los modelos híbridos permiten que el modelo sea más flexible y pueda capturar una mayor variedad de patrones en los datos, lo cual es especialmente útil en los mercados financieros, donde hay fluctuaciones tanto a corto como a largo plazo. Esta combinación permite que el modelo sea más robusto al procesar secuencias de tiempo, ya que se obtienen ventajas tanto de los filtros convolucionales como de la capacidad de memoria de las redes recurrentes.

Backtesting

Cuando ya hemos entrenado todos nuestros modelos, pasamos a la última etapa, el backtesting. En finanzas, no te puedes quedar solo con los datos de precisión de los modelos para tomar decisiones financieras, sino que debes combinar los datos con técnicas de inversión para hacer el tándem casi perfecto.

Pero, empecemos por el principio, ¿qué es un backtesting?

El **backtesting** es el proceso de evaluar una estrategia de trading o inversión utilizando datos históricos para simular las decisiones de compra y venta que un modelo habría tomado en el pasado. Esto permite medir el rendimiento de la estrategia bajo condiciones de mercado reales, sin necesidad de ejecutar la estrategia en tiempo real. Ojo, rentabilidades pasadas no aseguran rendimientos futuros.

En nuestro proyecto, hemos implementado un **backtesting basado en las predicciones de los modelos**, donde la lógica es: compramos si la predicción es positiva y vendemos cuando hay una predicción negativa. En clasificación es fácil identificar 0 y 1, pero en regresión lo que hacemos es usar el signo del retorno logarítmico para convertir nuestro problema en un backtesting factible.

Nuestro benchmark es “buy and hold”, es decir, comprar la acción y mantenerla durante todo el tiempo. No tiene sentido hacer inversiones en un determinado activo si no batimos a su subyacente en el histórico. Por tanto, siempre compararemos nuestra rentabilidad con este dato.

Por tanto, el backtesting nos permite simular cómo habría actuado un modelo basado en datos históricos, aplicando sus predicciones a un entorno pasado, y evaluando el rendimiento de una estrategia de inversión o trading:

- **Medición del rendimiento:**

A través del backtesting, podemos calcular métricas de rendimiento como la **rentabilidad**, el **riesgo** (medido a través del drawdown), y el **ratio de Sharpe**, que es la rentabilidad obtenida en base al riesgo asumido. Estas métricas son clave para validar el desempeño del modelo más allá de las métricas de entrenamiento, como la precisión o el error cuadrático medio.

- **Por ejemplo**, podemos simular una estrategia de compra cuando el modelo predice un aumento en el precio o venta cuando predice una caída, pero si el drawdown (peor racha de pérdidas) de esta estrategia es de un 50%, la estrategia deja de ser viable porque necesitamos el 100% para volver a nuestro punto de break-even (capital inicial), y es muy difícil reponerse a eso.

- **Por todo esto, las métricas financieras son muy importantes en este entorno de inversión.**

Proceso de Backtesting:

1. Generación de señales de compra y venta:

Una vez que los modelos han sido entrenados y han generado predicciones, estas predicciones se transforman en **señales de compra o venta**. Estas señales se basan en si el modelo prevé una subida o bajada en el precio de una acción:

- **Señal de compra:** Si el modelo predice un aumento de precio.
- **Señal de venta:** Si el modelo predice una caída de precio.
- **Mantener posición:** Si el modelo sigue prediciendo aumento.

2. Cálculo de resultados:

Durante el backtesting, se calculan una serie de métricas clave para evaluar el rendimiento de la estrategia:

- **Rentabilidad:** Se mide el total de ganancias o pérdidas generadas por la estrategia en función de las decisiones de compra/venta, en porcentaje.
- **Drawdown:** El drawdown mide la mayor pérdida que experimenta la estrategia desde un pico hasta un mínimo. Esto es importante para evaluar el riesgo máximo al que está expuesto un inversor.
- **Ratio de Sharpe:** Sirve para evaluar el rendimiento ajustado al riesgo de la estrategia. Un ratio Sharpe más alto indica que la estrategia ofrece una mayor rentabilidad por unidad de riesgo.

El backtesting proporciona información valiosa sobre cómo habría funcionado la estrategia basada en las predicciones de los modelos:

- **Validación de predicciones:** Nos ayuda a validar si las predicciones de los modelos realmente habrían generado decisiones rentables en el mercado. Si el backtesting muestra una baja rentabilidad o un alto drawdown, podría significar que el modelo no es adecuado para predecir movimientos de mercado.
- **Optimización de estrategias:** Permite ajustar y optimizar la estrategia de inversión basada en los resultados obtenidos. Si una estrategia no es lo suficientemente rentable, se pueden ajustar parámetros del modelo o de la estrategia de trading para mejorar su rendimiento.

Resultados

En esta primera versión, los resultados obtenidos de los modelos no son particularmente relevantes. Aunque los resultados del backtesting pueden sugerir cierta rentabilidad, es posible que estos resultados se deban al azar, dada la naturaleza limitada de los datos. La mayor parte de las pruebas realizadas se han hecho con 4 acciones seleccionadas al azar debido a limitaciones computacionales, lo que dificulta que los resultados sean consistentes y generalizables:

1. Resultados en regresión:

Los resultados en los modelos de regresión han sido limitados, con algunos algoritmos mostrando valores negativos en la métrica R^2 , lo que indica que el modelo no captura adecuadamente la variabilidad de los datos. La mejor predicción obtenida ha sido un R^2 de 34%, lo que refleja una capacidad de predicción moderada en este contexto.

Ejemplo de resultado para regresión en la muestra de PFE:

```
MSE promedio en entrenamiento: 0.002822087920500033
R² promedio en entrenamiento: 0.3625056874483548
Ratio de Sharpe promedio: 0.041846954291497145
Max Drawdown promedio: 0.30702535962425487
MSE en la última acción: 0.0002845690445400637
R² en la última acción: 0.06924587780714453
```

2. Resultados en clasificación:

Los modelos de clasificación no superaron los resultados esperados por azar, con una precisión que ronda el 50%, similar a tirar una moneda. Aunque las clases no estaban desbalanceadas, la matriz de confusión indicó que los modelos tendían a predecir con mayor frecuencia el escenario alcista, lo que sugiere una preferencia del modelo por este resultado sin una base sólida.

Ejemplo de resultado para clasificación en la muestra de PFE:

```
¿Deseas hacer las predicciones de PFE? (si/no): si
Random Forest Entrenamiento - Average Accuracy (RF): 0.5163
Random Forest Entrenamiento - Average Precision (RF): 0.5409
Random Forest Entrenamiento - Average Recall (RF): 0.6881
Random Forest Entrenamiento - Average F1 Score (RF): 0.5682
Classification Report Test - PFE (Random Forest):
              precision    recall  f1-score   support

     0           0.54       0.46       0.50         628
     1           0.47       0.54       0.51         549

 accuracy          0.50          0.50          0.50        1177
 macro avg         0.50          0.50          0.50        1177
 weighted avg      0.51          0.50          0.50        1177

Confusion Matrix Test - PFE (Random Forest):
[[292 336]
 [250 299]]
```

3. Modelos preentrenados:

Se han entrenado todos modelos con 4 acciones representativas del SP500: AAPL, MSFT, JPM y PFE. Estos modelos se han guardado (extensión `_EXPO.h5` y `_EXPO.pkl`) para agilizar futuros entrenamientos si el usuario lo solicita a través de "input" en la consola. Además, esto permite mostrar la aplicación web de manera más rápida. Sin embargo, esta muestra, aunque representativa, es limitada, lo que dificulta la generalización de los resultados obtenidos. Por este motivo, no es posible presentar una única métrica concluyente para todo el proceso. Esta **primera versión del proyecto** requiere más desarrollo y ajustes continuos para alcanzar resultados más sólidos y consistentes.

Aplicación Flask

Por último, como valor añadido del proyecto, hemos realizado una **aplicación web en Flask**. Esta aplicación actúa como una interfaz interactiva y accesible para los usuarios, permitiendo ejecutar consultas y análisis financieros sin necesidad de interactuar directamente con el código.

Funcionalidades clave:

1. Interfaz de usuario sencilla:

La aplicación Flask ofrece una interfaz sencilla y accesible donde los usuarios pueden seleccionar sectores, clusters y acciones (tickers) para su análisis. Esto proporciona flexibilidad y personalización al proceso de análisis financiero, permitiendo que los usuarios adapten el sistema a sus necesidades específicas.

2. Consulta a BigQuery y carga de datos:

Flask permite que los usuarios ejecuten consultas directamente sobre la base de datos de Google BigQuery para obtener datos financieros. En caso de que ya existan datos locales en

un archivo CSV, el sistema prioriza la carga de estos archivos, acelerando el proceso y reduciendo la dependencia de la base de datos.

3. Selección de sector y cluster:

El usuario puede elegir en la interfaz el sector y cluster que prefiera para comenzar con la predicción de 4 acciones y el testeo de estrategias. Esto da personalización y flexibilidad a la aplicación, resultando muy útil para el usuario.

4. Ejecución de modelos predictivos:

A través de la API expuesta por Flask, el usuario puede cargar los modelos predictivos de Machine Learning y Deep Learning para realizar análisis y predicciones de movimientos en los mercados financieros. Esto incluye la capacidad de ejecutar modelos de clasificación (señales de compra/venta) y de regresión (predicción de precios futuros).

5. Resultados y Backtesting:

La aplicación Flask también permite mostrar el backtesting basado en las predicciones del modelo, enseñando cómo habrían funcionado esas predicciones en el pasado. El sistema presenta los resultados al usuario, facilitando la evaluación de la rentabilidad y el rendimiento de las estrategias de trading.

→ Valor añadido:

La implementación de Flask como parte del proyecto creemos que otorga un valor añadido significativo, al convertir un sistema complejo en una herramienta accesible para usuarios no técnicos. Mediante esta interfaz web se democratiza el uso del análisis financiero avanzado, permitiendo que las personas puedan acceder a modelos predictivos y análisis de mercado sin conocimientos avanzados en programación o Machine Learning.

[Enlace a las capturas de pantalla de la aplicación.](#)

Conclusiones

El análisis y modelaje implementados en el código ofrecen un primer enfoque para la predicción de movimientos financieros y la toma de decisiones de inversión mediante el uso de modelos de **Machine Learning** y **Deep Learning** aplicados a series temporales financieras. A lo largo del proceso, hemos abordado una amplia gama de técnicas, desde el preprocesamiento de los datos y la selección de tickers, hasta el entrenamiento de diversos modelos predictivos y la implementación de backtesting para evaluar el rendimiento de las estrategias. A continuación, se presentan algunas de las conclusiones clave:

1. **Preprocesamiento de datos eficiente:** El código incluye un preprocesamiento exhaustivo de los datos, con pasos como la **selección de tickers**, nuevas variables generadas, limpieza, filtrado por **sector, cluster y volumen**, la no correlación y la **normalización** de las características. Este preprocesamiento asegura que los datos utilizados en los modelos sean consistentes y representativos del comportamiento de los mercados.
2. **Flexibilidad en la selección de tickers:** El sistema permite al usuario seleccionar aleatoria o manualmente cuatro tickers basados en sector y cluster, lo que facilita la personalización del análisis según los intereses específicos del usuario. Esta flexibilidad es esencial para adaptarse a diferentes contextos y estrategias de inversión.
3. **Escalado correcto para prevenir la fuga de datos:** Creemos que el proceso de **escalado de los datos** se ha manejado adecuadamente aplicando `fit_transform` solo en el conjunto de entrenamiento y luego replicando el mismo escalado en el conjunto de Test con `transform`. Esto evita la fuga de información y garantiza que el rendimiento de los modelos sea evaluado correctamente.
4. **Validación Walk-Forward:** Implementar la validación walk-forward ha sido clave para garantizar que los modelos puedan generalizar de manera adecuada a datos futuros, respetando la secuencia temporal de los datos. Esto ha permitido una evaluación más realista del rendimiento del modelo en comparación con métodos tradicionales de validación cruzada, ya que este enfoque garantiza que los modelos se evalúan en datos futuros no vistos, asegurando que las predicciones no se vean afectadas por el "lookahead bias" (fuga de información).
5. **Enfoque multimodelo:** El uso de una combinación de modelos, incluidos **Random Forest, LSTM, GRU, Conv1D** y **modelos híbridos**, nos ha permitido abordar diferentes aspectos de los datos financieros, tanto en términos de patrones a corto plazo (capturados por Conv1D) como de relaciones a largo plazo (capturadas por LSTM y GRU).
6. **Backtesting realista:** La integración del backtesting con las predicciones del modelo ha proporcionado una evaluación práctica de cómo habrían funcionado las decisiones de trading en condiciones históricas de mercado, calculando métricas como rentabilidad, drawdown y el ratio de Sharpe. Esto permite validar el uso de los modelos no solo en términos de métricas estadísticas, sino también en escenarios de inversión real.

Limitaciones y futuras líneas de mejora

A pesar de los logros del enfoque utilizado, el proyecto presenta algunas limitaciones que deben tenerse en cuenta:

1. **Limitaciones computacionales:** La restricción de trabajar solo con cuatro tickers es una simplificación que se adoptó debido a las limitaciones computacionales. En aplicaciones del mundo real, se podrían analizar muchas más acciones de manera simultánea, pero esto requeriría mayor capacidad de procesamiento y almacenamiento.
2. **Datos históricos limitados:** Aunque el backtesting se basa en datos históricos, una limitación inherente es que los datos pasados no siempre son indicativos de los futuros. Cambios estructurales en los mercados, como políticas económicas, crisis financieras o eventos inesperados, podrían alterar los patrones detectados por los modelos.
3. **Simulación perfecta sin costes:** En el backtesting, no hemos considerado explícitamente aspectos importantes del trading real, como los **costos de transacción** (comisiones) y la **liquidez del mercado**. Estos factores podrían afectar el rendimiento de la estrategia en un entorno real y deberían incluirse en futuros desarrollos.

Para mejorar el enfoque desarrollado en este proyecto y ampliar su aplicabilidad en el ámbito financiero, se proponen las siguientes líneas de mejoras:

1. **Realización de Fine Tuning:** Aunque hemos utilizado varios modelos avanzados, en futuras mejoras se podría implementar **fine tuning** para ajustar estos modelos específicamente a nuestro problema financiero. Esta técnica permitiría optimizar el rendimiento de los modelos al congelar o ajustar capas selectivas, adaptándolos mejor a un mayor número de tickers y aumentando su capacidad de generalización en diferentes sectores y mercados. Esto permitiría trabajar de manera más eficiente con volúmenes más grandes de datos y mejorar las predicciones.
2. **Optimización de hiperparámetros automática:** Implementar técnicas de optimización automática de hiperparámetros, como **Grid Search** o **Bayesian Optimization**, podría mejorar el rendimiento de los modelos ajustando de manera más eficiente los parámetros clave en lugar de hacerlo manualmente.
3. **Consideración de factores de riesgo:** Incluir la evaluación del **riesgo ajustado** en las predicciones de los modelos, utilizando métricas como **Value at Risk (VaR)**, para ofrecer una visión más completa del rendimiento de las estrategias de trading.

ANEXOS DE IMÁGENES DE FLASK

Página de inicio:




Selecciona el tipo de problema:

Clasificación

Regresión

Financial KeepCoders - Transformando las finanzas con IA
Bootcamp Big Data, IA y ML - KeepCoding
Carles Matoses Miret & Sara Díaz-P. Martín

Página de sector:



Selecciona el sector:

Financiero

Salud


Tecnología

Industria

Consumo

Financial KeepCoders - Transformando las finanzas con IA
Bootcamp Big Data, IA y ML - KeepCoding
Carles Matoses Miret & Sara Díaz-P. Martín

Página de cluster:



Selecciona el cluster:

Cluster 0

Cluster 1

Cluster 2


Cluster 3

Cluster 4

Cluster 5

Financial KeepCoders - Transformando las finanzas con IA
Bootcamp Big Data, IA y ML - KeepCoding
Carles Matoses Miret & Sara Díaz-P. Martín

Warning si no se encuentran 4 acciones:



Selecciona el sector:

No se encontraron 4 tickers. Por favor, seleccione de nuevo sector y cluster:

Financiero

Salud


Tecnología

Industria

Consumo

Financial KeepCoders - Transformando las finanzas con IA
Bootcamp Big Data, IA y ML - KeepCoding
Carles Matoses Miret & Sara Díaz-P. Martín

Página de acciones seleccionadas:
Clasificación:



Acciones Seleccionadas

Estos son los tickers que se han seleccionado:

- GOOGL
- META
- AAPL
- AMD


Ver Backtesting Basado en Predicciones

Financial KeepCoders - Transformando las finanzas con IA

Bootcamp Big Data, IA y ML - KeepCoding

Carles Matoses Miret & Sara Díaz-P. Martín

Regresión:



Acciones Seleccionadas

Estos son los tickers que se han seleccionado:

- MU
- AAPL
- META
- AMD

Ver Backtesting Basado en Predicciones

Financial KeepCoders - Transformando las finanzas con IA

Bootcamp Big Data, IA y ML - KeepCoding

Carles Matoses Miret & Sara Díaz-P. Martín

Página de resultados de Backtesting para clasificación:

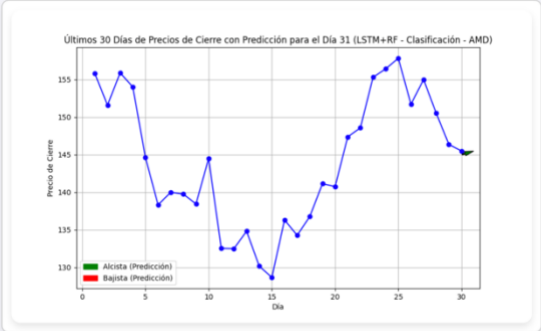


Resultados del Backtesting para AMD

Resultados del Modelo:

- El mejor modelo es LSTM+RF - Clasificación - AMD con un retorno acumulado de 199.54%
- Buy and Hold Cumulative Return: 212.61%

Gráficas de Resultados:



Probar de nuevo

Página de resultados de Backtesting para regresión:

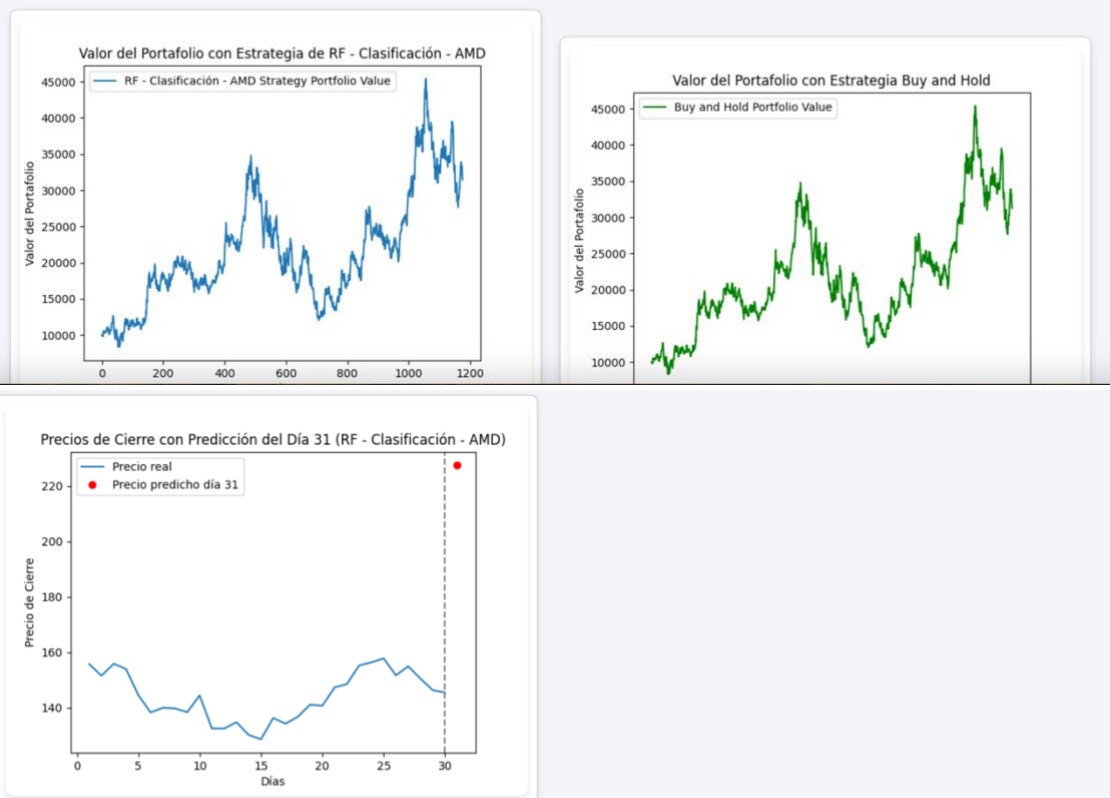


Resultados del Backtesting para AMD

Resultados del Modelo:

- El mejor modelo es RF - Clasificación - AMD con un retorno acumulado de 214.48%
- Buy and Hold Cumulative Return: 212.61%
- Precio de cierre del día 30 (real): 145.49
- Precio de cierre del día 31 (predicho): 227.46

Gráficas de Resultados:



Probar de nuevo

Financial KeepCoders - Transformando las finanzas con IA

Bootcamp Big Data, IA y ML - KeepCoding

Carles Matoses Miret & Sara Díaz-P. Martín