Texto

Descripción generada automáticamente

**MEMORIA DEL PROYECTO FINANCIAL KEEPCODERS**

**PROCESAMIENTO Y MODELADO**

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

Este proyecto ha sido todo un reto para nosotros, y la meta que queríamos conseguir en esta primera versión, creemos que la hemos logrado. Esta meta consistía en 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, para mejorar el análisis de los mercados financieros.

**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.

1. **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 .csv por agilizar el acceso a datos si queremos más rapidez computacional.

1. **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 elimina columnas innecesarias y devuelve un DataFrame que puede ser utilizado para el preprocesamiento.

1. **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.

1. **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.

1. **Inputs para selección de sector y cluster**:

E 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.

1. **Selección manual de tickers**:

En caso de que el usuario prefiera analizar acciones específicas en lugar de elegirlas 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.

1. **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.

1. **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) mientas 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.

1. **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 qus nos aporta una variable con información muy interesante para el análisis.

1. **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.

1. **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.

1. **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 cruzada**

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**: 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.

1. **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.

* **Por qué es importante 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.

1. **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.

1. **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 walk-forward validation, 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.
  + 1. **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.
  + 1. **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 backtesnting?** 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 logarítimico 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, 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 dradown (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 breakeven (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.
  1. **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.

**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.

1. **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.

1. **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.

1. **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).

1. **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 creeos 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.

**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.