Máster en Ingeniería del Software: Cloud, Datos y Gestión TI
Machine Learning Engineering (MLE)

Researching different stock market shares and making price predictions through the application of ML techniques

José Antonio Zamudio Amaya
Carlos Núñez Arenas

1. Análisis del dominio

Introducción

Las instituciones financieras de todo el mundo negocian a diario miles de millones de dólares. Empresas de inversión, fondos de cobertura e incluso particulares han estado utilizando modelos financieros para comprender mejor el comportamiento del mercado y realizar inversiones y operaciones rentables.

La cantidad de información disponible sobre el comportamiento del mercado financiero y las empresas es abrumadora y requiere un procesamiento y análisis exhaustivo para comprender de manera clara y precisa cómo funcionan. Las instituciones financieras, empresas de inversión y particulares utilizan modelos financieros para ayudarlos a tomar decisiones informadas sobre cómo invertir y operar en el mercado.

En este proyecto, crearemos un predictor del precio de las acciones de distintas macro-empresas, tomando como entrada los datos de negociación diaria de un determinado intervalo de fechas y para generar estimaciones proyectadas en determinadas fechas de consulta. Para esto, usaremos datos como el precio de apertura (Open), el precio más alto al que se negociaron las acciones (High), cuántas acciones se negociaron (Volume) y el precio de cierre ajustado por divisiones de acciones y dividendos (Adjusted Close), para cada día registrado en bolsa.

El éxito de este proyecto dependerá de la calidad de los datos utilizados y de la eficacia de nuestro modelo de aprendizaje automático. Es importante tener en cuenta que los mercados financieros son altamente volátiles y pueden verse afectados por una amplia variedad de factores, por lo que nuestras estimaciones deben ser interpretadas como una guía y no como una predicción precisa.

Datasets a utilizar

En nuestro caso, usaremos datasets sacados del popular sitio web Yahoo Finance. En concreto trabajaremos con los datos del mercado financiero de las empresas: Apple, Microsoft, Google, Amazon y Meta.

Líneas de trabajo y retos

Las líneas de trabajo y los retos en el desarrollo de un predictor de precios de acciones son los siguientes:

Adquisición de datos: Es fundamental tener acceso a datos precisos y actualizados sobre el comportamiento del mercado y las empresas. Se requiere investigar fuentes confiables de datos y asegurarse de que se cuenta con una cantidad suficiente de datos históricos para entrenar nuestro modelo de aprendizaje automático.

1. Procesamiento de datos: Una vez obtenidos los datos, es necesario preprocesarlos para eliminar cualquier ruido o datos irrelevantes y prepararlos para su uso en el modelo.

2. Entrenamiento y evaluación: Crearemos distintos modelos de aprendizaje para predecir los valores del mercado bursátil. También será necesario entrenarlos utilizando los datos históricos y evaluar su precisión en la generación de predicciones. Es posible que sea necesario realizar ajustes en los modelo y volver a entrenarlo hasta obtener resultados satisfactorios.

3. Selección del modelo: De los distintos modelos entrenados, nos quedaremos con aquél que se ajuste mejor a nuestro dominio mediante distintas técnicas comparativas. Es importante investigar y seleccionar el modelo que mejor se ajuste a nuestras necesidades y a los datos que tenemos disponibles.

4. Integración e implementación: Finalmente, es necesario integrar el modelo entrenado en una aplicación o sistema que permita a los usuarios generar predicciones de precios de acciones de manera eficiente y fácil de usar.

Entre los retos que se presentan en este proyecto, destacan la complejidad de los mercados financieros y la cantidad de factores que pueden afectar los precios de las acciones. Además, la volatilidad del mercado puede generar una gran cantidad de ruido en los datos, lo que puede afectar la precisión de las predicciones. Por lo tanto, es fundamental tener en cuenta estos factores y buscar soluciones para minimizar su impacto en el modelo.

Las líneas de trabajo y los retos en el desarrollo de un predictor de precios de acciones incluyen la adquisición y procesamiento de datos, la selección del modelo, el entrenamiento y evaluación, y la integración e implementación. Sin embargo, es importante estar consciente de los retos asociados con la complejidad y volatilidad de los mercados financieros y buscar soluciones para minimizar su impacto en el modelo.

Análisis de herramientas

El análisis de herramientas es un aspecto clave en el desarrollo de un predictor de precios de acciones. En este proyecto, se pueden utilizar diversas herramientas de programación para ayudar en la adquisición, procesamiento y análisis de los datos, así como en la selección y entrenamiento del modelo.

Python es un lenguaje de programación de alto nivel que se utiliza ampliamente en el desarrollo de aplicaciones y proyectos en el ámbito científico y financiero. Algunas de las bibliotecas más relevantes en el contexto de este proyecto son:

1. Pandas: Es una biblioteca de análisis de datos que proporciona estructuras de datos y herramientas para manipular y procesar grandes cantidades de datos.

2. Numpy: Es una biblioteca de computación numérica que proporciona funciones y herramientas para trabajar con arrays multidimensionales y realizar cálculos complejos.

3. Matplotlib: Es una biblioteca de gráficos que permite visualizar los datos de manera sencilla y efectiva.

4. TensorFlow: Es una biblioteca de aprendizaje automático desarrollada por Google que se utiliza ampliamente en la investigación y desarrollo de modelos de aprendizaje automático.

5. scikit-learn (sklearn): Es una biblioteca de aprendizaje automático de código abierto que proporciona herramientas y algoritmos para seleccionar y entrenar modelos de aprendizaje automático.

Estas herramientas son esenciales para el desarrollo de un predictor de precios de acciones, ya que permiten adquirir y procesar los datos de manera eficiente, visualizar los resultados y entrenar y evaluar modelos de aprendizaje automático. Al utilizar estas herramientas, se puede acelerar el proceso de desarrollo y mejorar la precisión de las predicciones.

Búsqueda bibliográfica

La búsqueda bibliográfica se puede realizar en diversas fuentes, como bases de datos científicas, revistas especializadas, libros, conferencias y otros medios de comunicación científica. Algunas de las fuentes más relevantes incluyen:

1. Bases de datos científicas: Como ScienceDirect, PubMed o IEEE Xplore, que contienen miles de artículos y publicaciones científicas en diversos campos, incluyendo el análisis financiero y el aprendizaje automático.

2. Revistas especializadas: Como Journal of Financial Markets, Journal of Banking and Finance o Journal of Business Research, que publican investigaciones y artículos en el ámbito financiero y el análisis de datos.

3. Libros: Como Financial Modeling, Machine Learning for Financial Engineering o Handbook of Financial Time Series, que proporcionan una visión general de las técnicas y modelos utilizados en el análisis financiero y el aprendizaje automático.

4. Conferencias y simposios: Como NeurIPS, IJCAI o KDD, que son foros en los que se presentan investigaciones y desarrollos en el ámbito de la inteligencia artificial y el aprendizaje automático.

La búsqueda bibliográfica es esencial para conocer los avances y tendencias en el ámbito del predictor de precios de acciones, así como para identificar las fortalezas y debilidades de los modelos existentes.

En nuestro proyecto, nos hemos ayudado de diversa documentación científica y tutoriales para realizar un pequeño proyecto base como prueba de concepto y poder comprobar como podemos usar algoritmos entrenados para ayudarnos a la hora de actuar en el mercado financiero

Casos de éxito

El análisis del precio de las acciones es una aplicación importante del aprendizaje automático en el ámbito financiero, y existen diversos casos de éxito que muestran su efectividad en la toma de decisiones de inversión.

El primer caso de éxito que encontramos se encuentra en el ámbito de la predicción de tendencias, un modelo de aprendizaje automático puede ser entrenado para predecir la tendencia futura de los precios de las acciones, lo que puede ser muy útil

para los inversores a la hora de tomar decisiones de compra y venta. En el video de youtube Python Stock Analysis - Balance Sheet Trend Analysis encontramos un ejemplo de muchos de análisis de tendencias en Python que existen por internet.

Otro ejemplo de caso de éxito que hemos encontrado podemos verlo en el sitio web de Kagge, donde un usuario realiza un predictor con un modelo basado en LSTM para predecir el valor de las acciones de la compañía de coches eléctricos Tesla. Este es un ejemplo de muchos donde los usuarios comparten cómo analizar datasets de valores del mercado financiero.

Por último, también encontramos un ejemplo de éxito de dudosa veracidad, pero que sigue siendo ejemplo de cómo modelos de aprendizaje automático son capaces de influir en la compra venta de acciones. Estos modelos pueden ser utilizados para desarrollar alarmas de trading, que permiten a los inversores recibir alertas en tiempo real sobre cambios importantes en el mercado o en las acciones en las que invierten. En muchos casos estas alertas son intentos deliberados de influir en el mercado de las acciones para fines lucrativos de unas pocas personas.

Estos son solo algunos de los muchos casos de éxito del uso del aprendizaje automático en el análisis del precio de las acciones. Estos casos demuestran que los modelos de aprendizaje automático pueden ser muy útiles para mejorar la eficiencia y la precisión de las decisiones de inversión, y para optimizar el rendimiento de los portfolios de inversión.

2. Conceptos

En esta sección se explicarán los conceptos necesarios para familiarizarse con el contexto del trabajo, necesarios para entender el desarrollo que se redactará posteriormente en la sección tercera del documento.

Volatilidad

La volatilidad es una medida estadística de la fluctuación de los precios de un activo en un período de tiempo determinado. La volatilidad se utiliza a menudo como un indicador de la incertidumbre y el riesgo en los mercados financieros. Un activo con una volatilidad elevada significa que sus precios fluctúan ampliamente, lo que puede ser un riesgo para los inversores, mientras que un activo con una volatilidad baja significa que sus precios varían menos, lo que es una señal de estabilidad.

Bandas de Bollinger

Las Bandas de Bollinger son un indicador utilizado en el análisis técnico para evaluar la volatilidad de un activo. Estas bandas están compuestas por tres líneas: la línea central, que es una media móvil simple de los precios de cierre de un activo, y dos bandas externas que se encuentran a una determinada distancia estadística por encima y por debajo de la línea central. Las bandas externas se utilizan para identificar los niveles de sobrecompra y sobreventa del activo, y pueden ser utilizadas para determinar las señales de compra y venta en el mercado.

Daily Return

El Daily Return es el rendimiento diario de un activo, y se calcula como el cambio en el precio del activo dividido por su precio anterior. El Daily Return es una medida importante en la evaluación del desempeño de un activo, y es esencial para la gestión de carteras y el análisis del rendimiento de los inversores.

3. Desarrollo

A continuación se describe paso a paso, el proceso de trabajo de la prueba de concepto explicada en la introducción del documento, es decir, del análisis de los datos financieros de las empresas: Apple, Microsoft, Google, Amazon y Meta, además del desarrollo de un modelo predictor que calcule los valores de cierre de un día dado.

Importación de librerías y carga de datos

Este es un paso trivial en el cual instalamos y cargamos las librerías necesarias para el desarrollo del proyecto.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import sklearn as sk
from xgboost import XGBRegressor

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from sklearn.metrics import mean_squared_error

%matplotlib inline
```

```
AAPL = pd.read_csv("./datasets/AAPL (2018-2022).csv")
MSFT = pd.read_csv("./datasets/MSFT (2018-2022).csv")
GOOG = pd.read_csv("./datasets/GOOG (2018-2022).csv")
AMZN = pd.read_csv("./datasets/AMZN (2018-2022).csv")
META = pd.read_csv("./datasets/META (2018-2022).csv")

AAPL.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-01-02	42.540001	43.075001	42.314999	43.064999	40.950497	102223600
1	2018-01-03	43.132500	43.637501	42.990002	43.057499	40.943352	118071600
2	2018-01-04	43.134998	43.367500	43.020000	43.257500	41.133541	89738400
3	2018-01-05	43.360001	43.842499	43.262501	43.750000	41.601864	94640000
4	2018-01-08	43.587502	43.902500	43.482498	43.587502	41.447346	82271200

Preprocesado

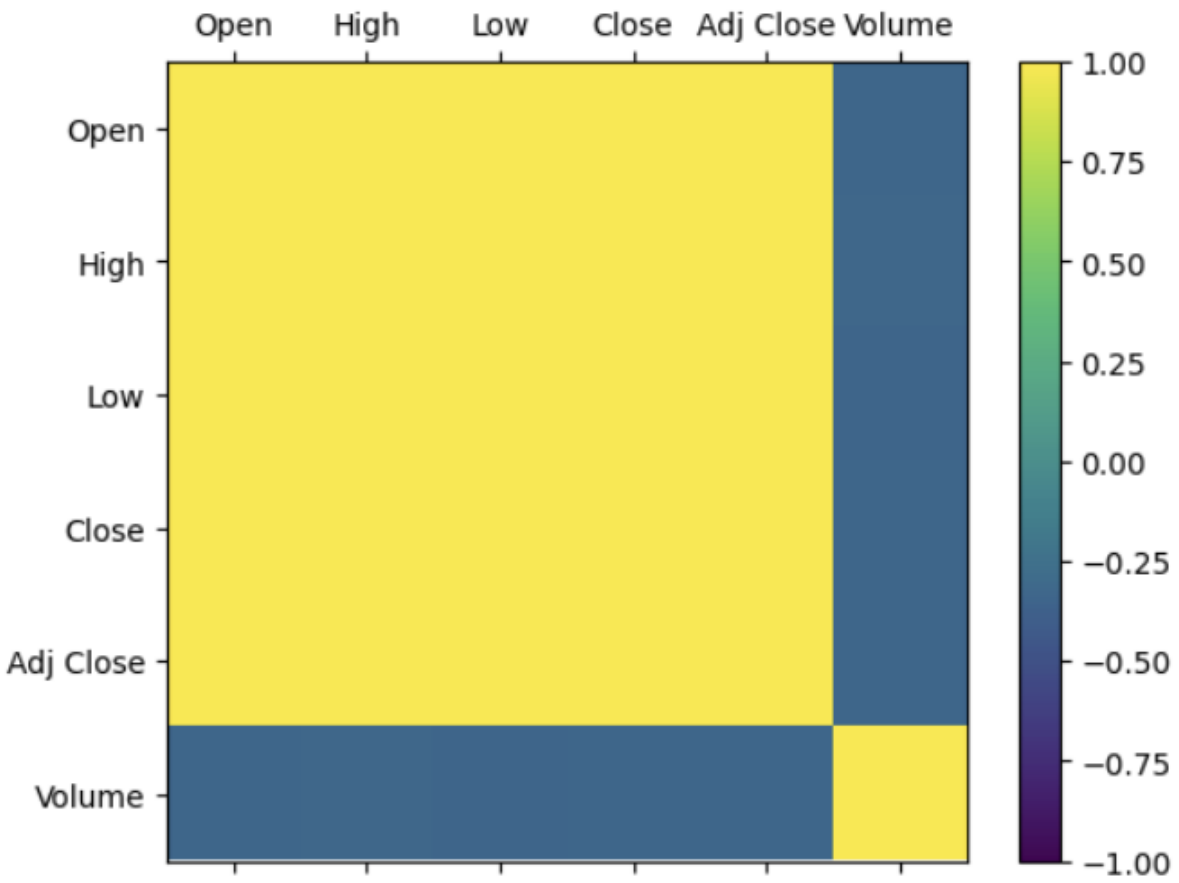
Desarrollaremos una función para formatear los datos para poder manejarlos mejor posteriormente. Para ello, hemos definido una función para utilizarla con los 5 datasets, ya que tienen la misma estructura, número de filas y columnas y clases. Cambiamos las clases de las columnas para tratar los datos correctamente, es decir, le daremos un tipo a cada atributo.

```
def preprocessing(original_df):
    df = original_df.copy()
    df['Date'] = pd.to_datetime(df['Date'])
    df.set_index('Date', inplace=True)
    df['Open'] = df['Open'].astype('float')
    df['High'] = df['High'].astype('float')
    df['Low'] = df['Low'].astype('float')
    df['Close'] = df['Close'].astype('float')
    df['Adj Close'] = df['Adj Close'].astype('float')
    df['Volume'] = df['Volume'].astype('int')
    return df

PRE_AAPL = preprocessing(AAPL)
PRE_MSFT = preprocessing(MSFT)
PRE_GOOG = preprocessing(GOOG)
PRE_AMZN = preprocessing(AMZN)
PRE_META = preprocessing(META)
```

Correlación de atributos

Calculamos la correlación entre atributos de cada dataset para ver si podemos obtener algo de información adicional antes de tratar los datos.
Para nuestra sorpresa, los 5 datasets nos dan aproximadamente los mismos resultados.

```
def correlation_analysis(original_df):
    df = original_df.copy()
    corr = df.corr()
    corr.style.background_gradient(cmap='coolwarm')
    # plot correlation matrix
    fig = plt.figure()
    ax = fig.add_subplot(111)
    cax = ax.matshow(corr, vmin=-1, vmax=1, interpolation='none')
    fig.colorbar(cax)
    ticks = np.arange(0,6,1)
    ax.set_xticks(ticks)
    ax.set_yticks(ticks)
    ax.set_xticklabels(df.columns)
    ax.set_yticklabels(df.columns)
    plt.show()

correlation_analysis(PRE_AAPL)
correlation_analysis(PRE_MSFT)
correlation_analysis(PRE_GOOG)
correlation_analysis(PRE_AMZN)
correlation_analysis(PRE_META)
```

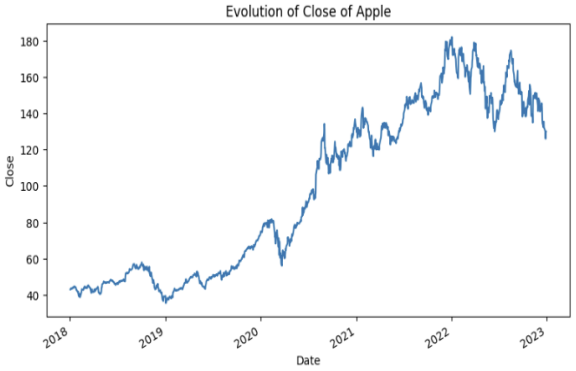


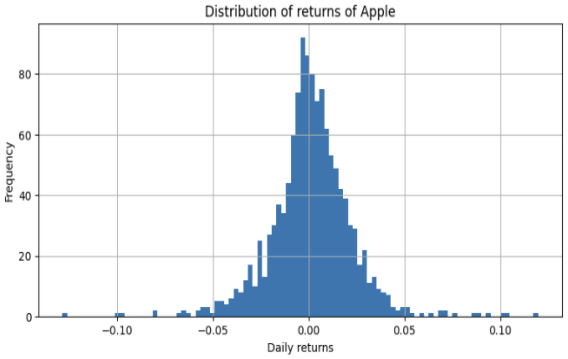
Teniendo en cuenta que el análisis de correlación se refiere al precio de las acciones, se puede concluir que el precio de apertura (Open), el precio máximo (High), el precio mínimo (Low), el precio de cierre (Close) y el precio de cierre ajustado (Adj Close) están altamente relacionados entre sí. Esto sugiere que el precio de las acciones varían de manera similar en relación a estas variables. Sin embargo, el volumen negociado de las acciones está inversamente relacionado con el resto de las variables. Esto podría indicar que cuando el volumen de negociación es alto, el precio de las acciones de las compañías estudiadas tienden a disminuir, por ley de oferta y demanda.

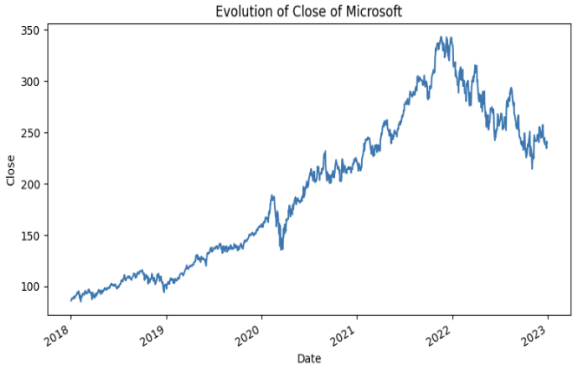
Visualización de la evolución y Daily Return de las compañías

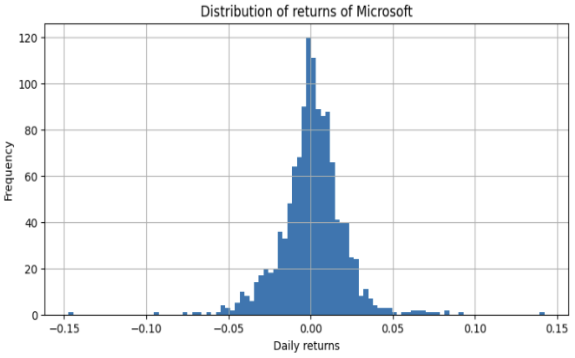
Visualizamos la evolución del valor de cierre de mercado diario de cada compañía, y además el rango de retorno en base a su frecuencia. Esto nos dará una idea aproximada del estado histórico y actual de la valoración en el mercado de cada compañía. Al ser 5 de las empresas más grandes del mundo, estos valores son muy altos.

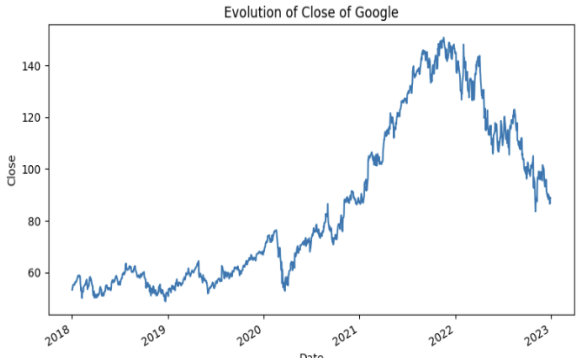
```python
def compute_stats(original_df):
    df = original_df.copy()
    df['close_daily_change'] = df['Close'].pct_change()
    df['30d_ma'] = df['Close'].rolling(window=30).mean()
    df['30d_std'] = df['Close'].rolling(window=30).std()
    df['upper_band'] = df['30d_ma'] + 2 * df['30d_std']
    df['lower_band'] = df['30d_ma'] - 2 * df['30d_std']
    df['rolling_mean'] = df['Close'].rolling(window=30).mean()
    df['rolling_std'] = df['Close'].rolling(window=30).std()
    df['daily_range'] = df['High'] - df['Low']
    df['P/E'] = df['Close'] / df['Adj Close']
    df['30d_volatility'] = df['close_daily_change'].rolling(window=30).std()
    df['30d_avg_volume'] = df['Volume'].rolling(window=30).mean()
    df['30d_avg_daily_range'] = df['daily_range'].rolling(window=30).mean()
    df['30d_avg_close_daily_change'] = df['close_daily_change'].rolling(window=30).mean()
    df['30d_avg_daily_volatility'] = df['30d_volatility'].rolling(window=30).mean()
    df['30d_avg_P/E'] = df['P/E'].rolling(window=30).mean()
    return df

COMP_AAPL = compute_stats(PRE_AAPL)
COMP_MSFT = compute_stats(PRE_MSFT)
COMP_GOOG = compute_stats(PRE_GOOG)
COMP_AMZN = compute_stats(PRE_AMZN)
COMP_META = compute_stats(PRE_META)
```

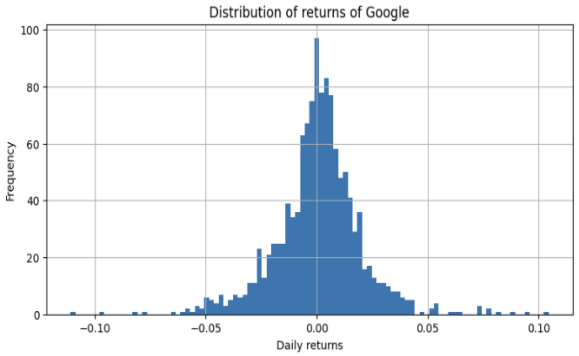


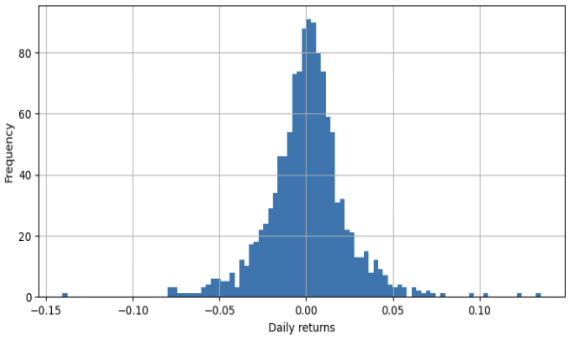


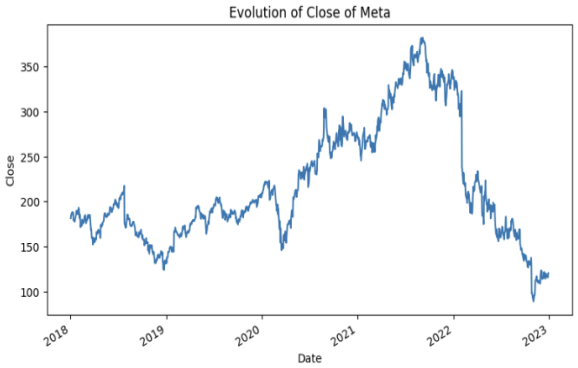


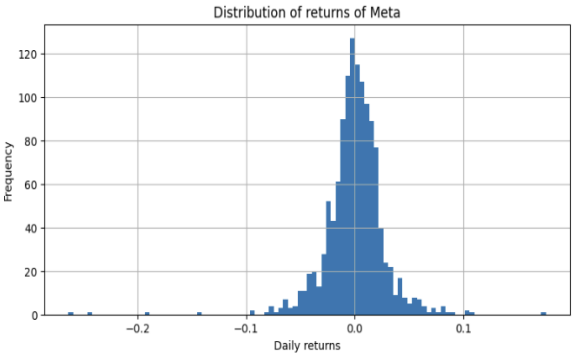








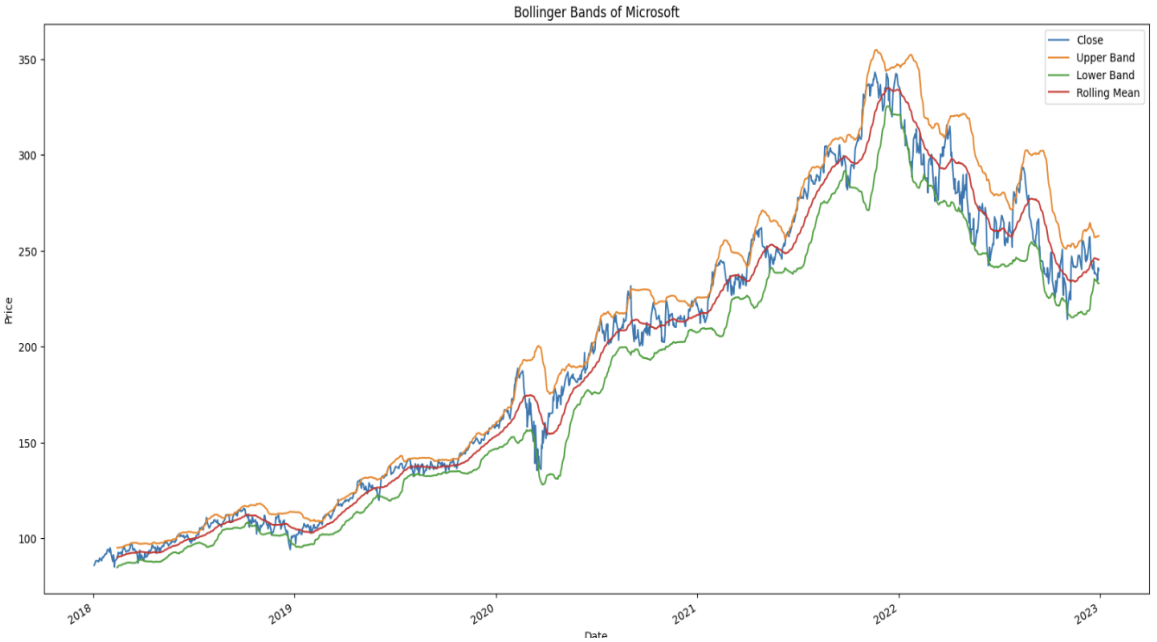




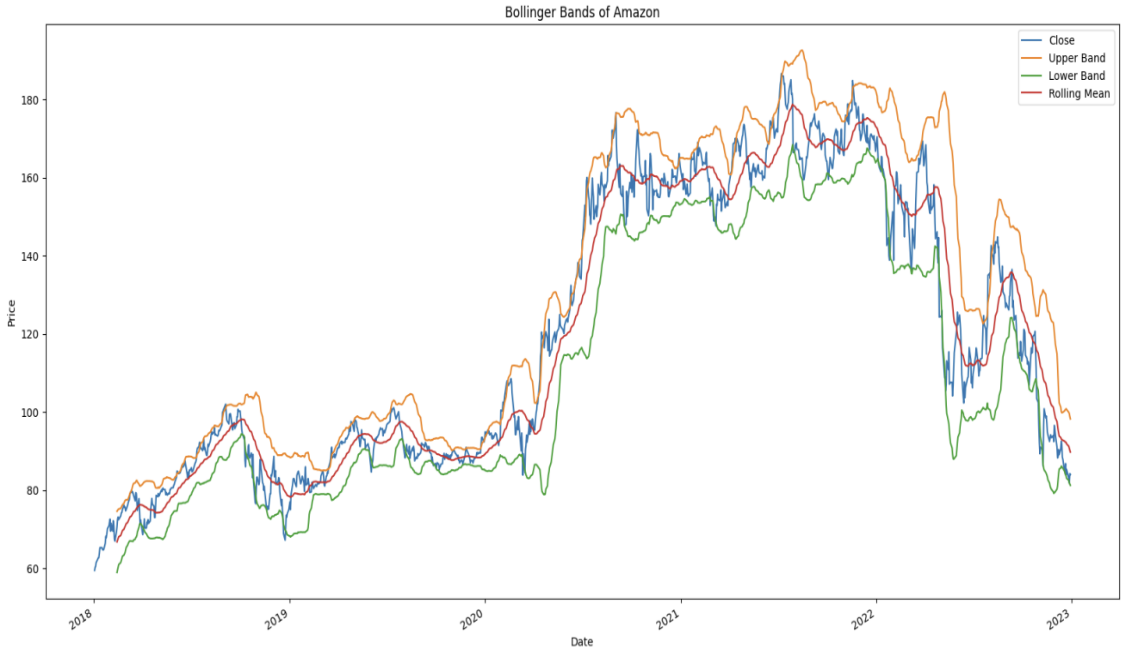
Visualización de las Bandas de Bollinger

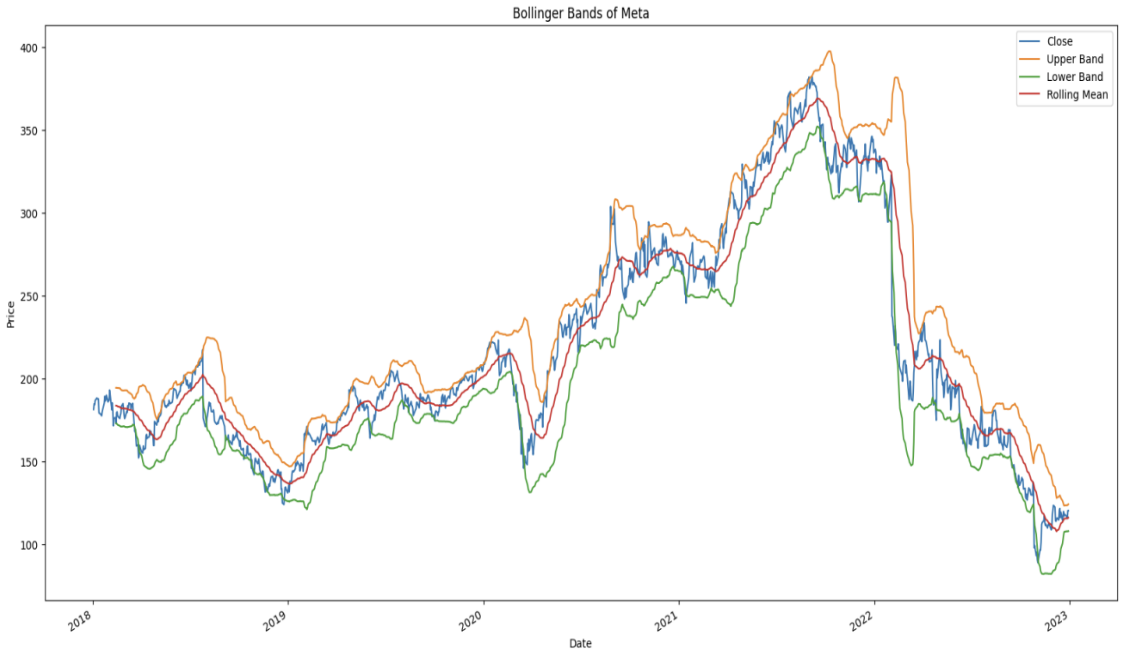
Ahora vamos a realizar un análisis según bandas de Bollinger. Como ya se ha explicado anteriormente, las bandas de Bollinger son un indicador técnico utilizado en análisis financiero para evaluar la volatilidad de un activo. Con estas gráficas, podemos identificar patrones de comportamiento de los precios y evaluar la volatilidad del mercado, lo que nos puede servir para tomar decisiones informadas sobre la compra y venta de acciones a futuro.

```python
def show_bollinger_bands(original_df, company):
    df = original_df.copy()
    df['Close'].plot(figsize=(20,10), label='Close')
    df['upper_band'].plot(label='Upper Band')
    df['lower_band'].plot(label='Lower Band')
    df['rolling_mean'].plot(label='Rolling Mean')
    plt.legend(loc='best')
    plt.title("Bollinger Bands of " + company)
    plt.ylabel('Price')
    plt.xlabel('Date')
    plt.show()

show_bollinger_bands(COMP_AAPL, 'Apple')
show_bollinger_bands(COMP_MSFT, 'Microsoft')
show_bollinger_bands(COMP_GOOG, 'Google')
show_bollinger_bands(COMP_AMZN, 'Amazon')
show_bollinger_bands(COMP_META, 'Meta')
```


Bollinger Bands of Microsoft
Close
Upper Band
Lower Band
Rolling Mean
Price
350
300
250
200
150
100
2018
2019
2020
2021
2022
2023
Date


Bollinger Bands of Google
Close
Upper Band
Lower Band
Rolling Mean
Price
140
120
100
80
60
2018
2019
2020
2021
2022
2023
Date





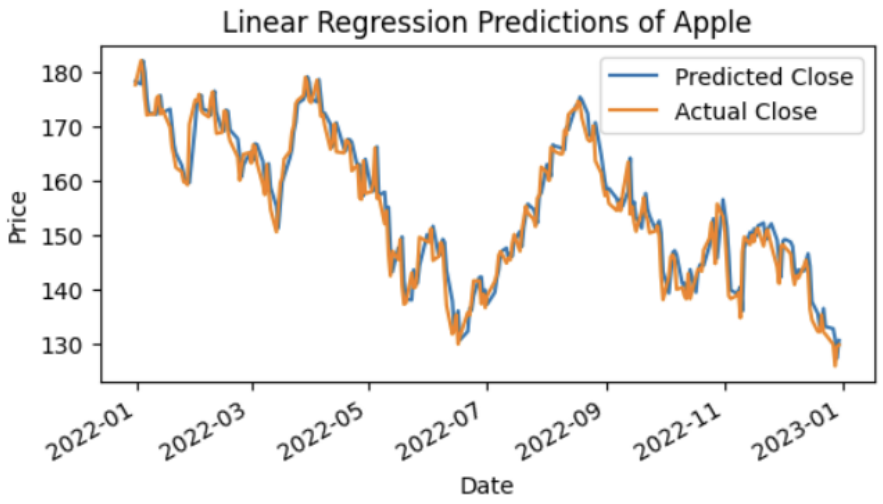
Bollinger observó que si observamos la volatilidad reciente de la acción, si es muy volátil, podríamos descartar el movimiento por encima y por debajo de la media. Pero si no es muy volátil, podemos prestarle atención. La media móvil puede darnos una idea de los verdaderos precios subyacentes de una acción. Si hay una desviación significativa por debajo o por encima de la media móvil, puede darnos una idea sobre una posible oportunidad de compra o venta, respectivamente. El reto sigue siendo saber cuándo esta desviación es lo suficientemente significativa como para prestarle atención.

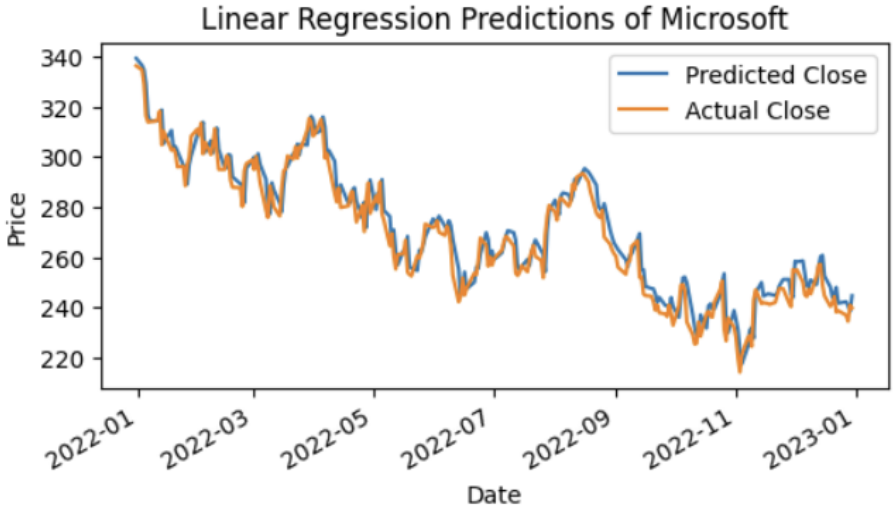
Entrenamiento de modelos

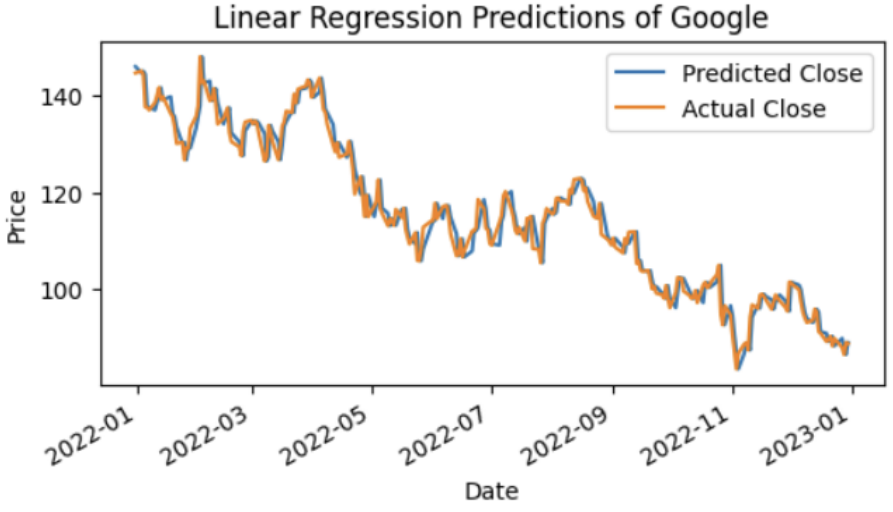
Vamos a entrenar varios modelos de predicción para intentar predecir el valor de cierre de un stock en el futuro, para posteriormente evaluar los modelos y quedarnos con el que mejor se ajuste.

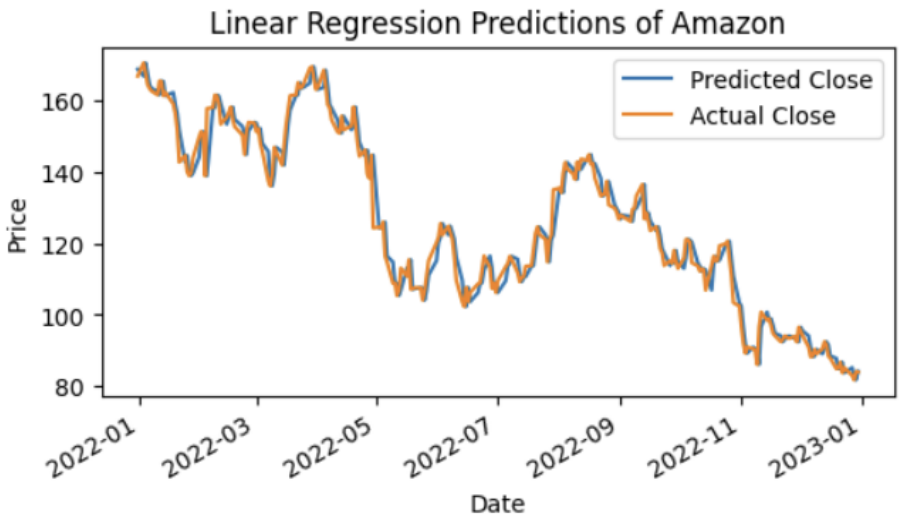
Regresión Lineal

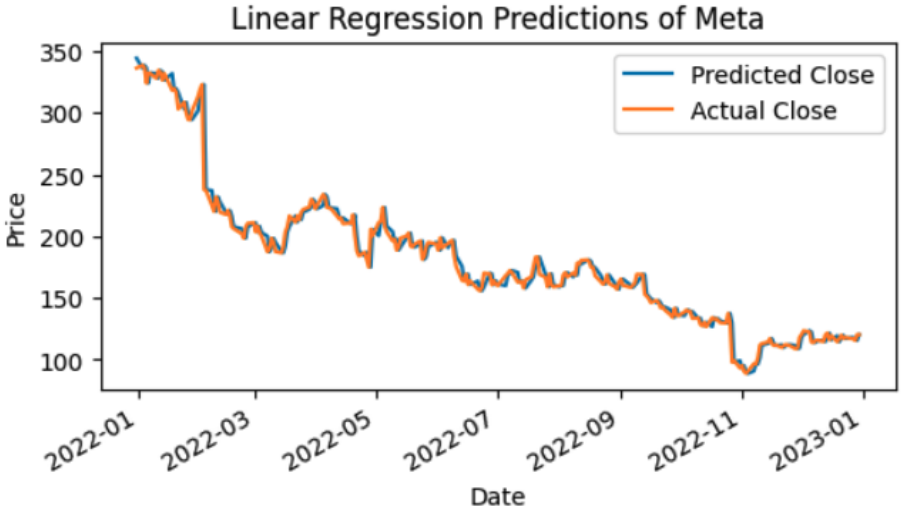
```
def train_with_linear_regression(original_df, company):
    df = original_df.copy()
    train, test = train_test_split(df, test_size=0.2, shuffle=False)
    X_train = train.drop(['Close'], axis=1)
    y_train = train['Close']
    X_test = test.drop(['Close'], axis=1)
    y_test = test['Close']

    lr = LinearRegression()
    lr.fit(X_train, y_train)
    lr_pred = lr.predict(X_test)
    lr_pred = pd.DataFrame(lr_pred, index=y_test.index, columns=['Close'])
    lr_pred['Close'] = lr_pred['Close'].shift(1)
    lr_pred['Close'].iloc[0] = y_train.iloc[-1]
    lr_pred['Close'].plot(figsize=(6,3), label='Predicted Close')
    y_test.plot(label='Actual Close')
    plt.legend(loc='best')
    plt.title("Linear Regression Predictions of " + company)
    plt.ylabel('Price')
    plt.xlabel('Date')
    plt.show()
    lf_rmse = np.sqrt(mean_squared_error(y_test, lr_pred))
    print("Linear Regression RMSE of " + company + ": " + str(lf_rmse))
    return [lr, lf_rmse]

LR_AAPL = train_with_linear_regression(PRE_AAPL, 'Apple')
LR_MSFT = train_with_linear_regression(PRE_MSFT, 'Microsoft')
LR_GOOG = train_with_linear_regression(PRE_GOOG, 'Google')
LR_AMZN = train_with_linear_regression(PRE_AMZN, 'Amazon')
LR_META = train_with_linear_regression(PRE_META, 'Meta')
```


Linear Regression Predictions of Apple
Predicted Close
Actual Close
Price
180
170
160
150
140
130
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date


Linear Regression Predictions of Microsoft
Predicted Close
Actual Close
Price
340
320
300
280
260
240
220
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

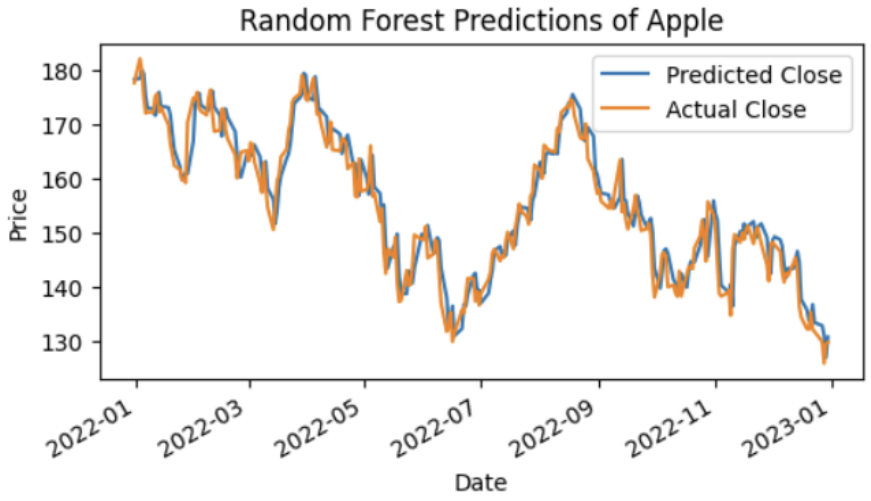

Linear Regression Predictions of Google
Predicted Close
Actual Close
Price
140
120
100
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

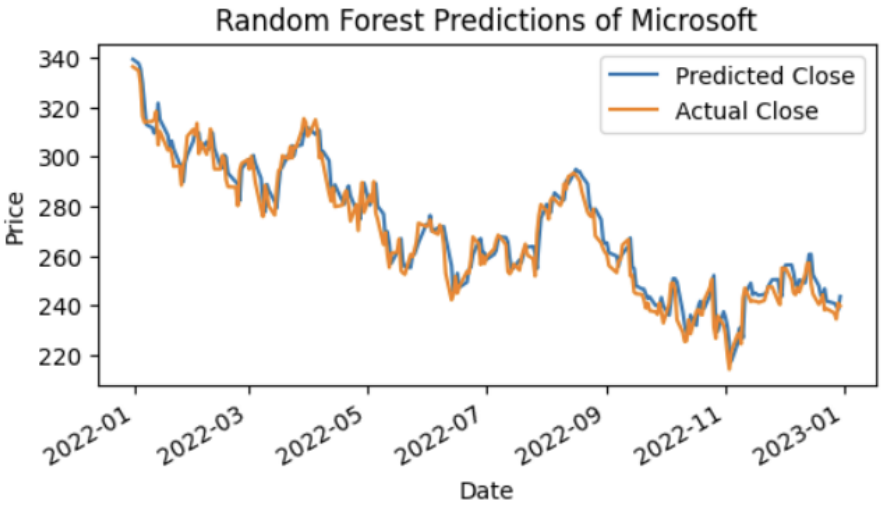

Linear Regression Predictions of Amazon
Predicted Close
Actual Close
Price
160
140
120
100
80
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

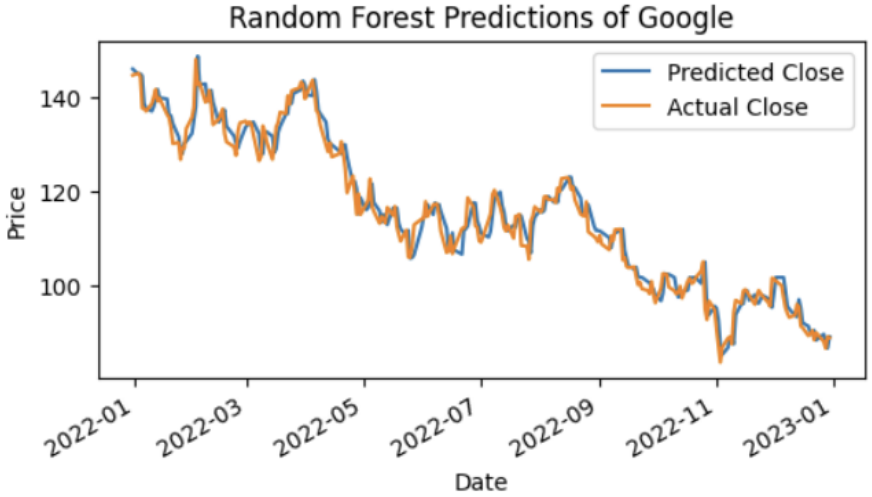

Linear Regression Predictions of Meta
Predicted Close
Actual Close
Price
350
300
250
200
150
100
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

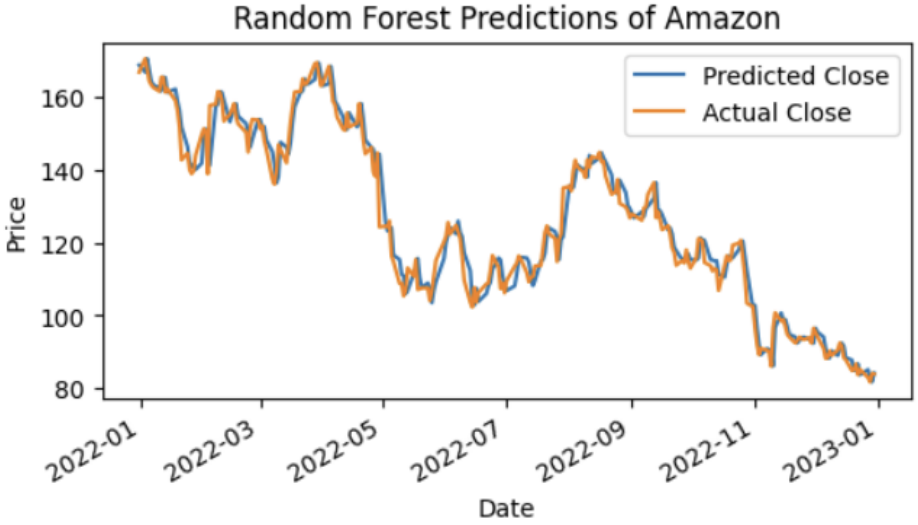
Random Forest

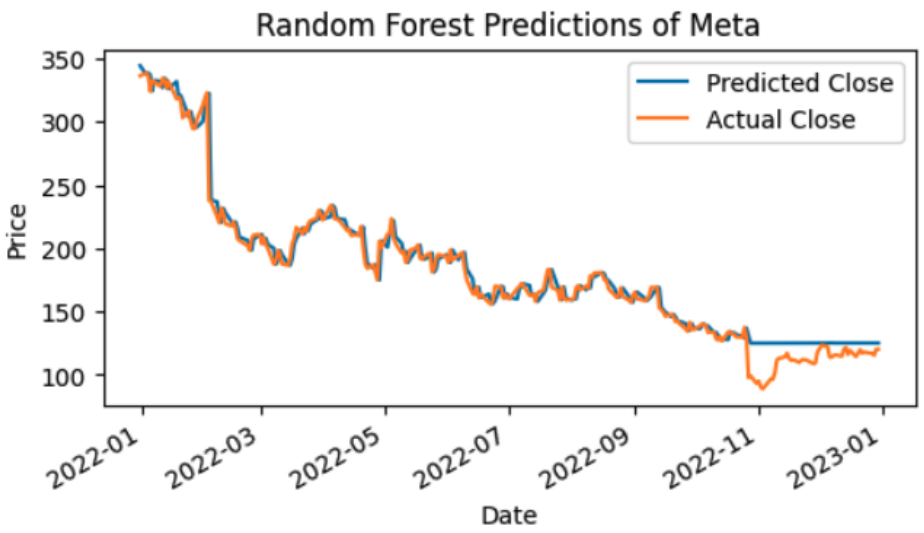
```python
def train_with_random_forest(original_df, company):
    df = original_df.copy()
    train, test = train_test_split(df, test_size=0.2, shuffle=False)
    X_train = train.drop(['Close'], axis=1)
    y_train = train['Close']
    X_test = test.drop(['Close'], axis=1)
    y_test = test['Close']

    rf = RandomForestRegressor()
    rf.fit(X_train, y_train)
    rf_pred = rf.predict(X_test)
    rf_pred = pd.DataFrame(rf_pred, index=y_test.index, columns=['Close'])
    rf_pred['Close'] = rf_pred['Close'].shift(1)
    rf_pred['Close'].iloc[0] = y_train.iloc[-1]
    rf_pred['Close'].plot(figsize=(6,3), label='Predicted Close')
    y_test.plot(label='Actual Close')
    plt.legend(loc='best')
    plt.title("Random Forest Predictions of " + company)
    plt.ylabel('Price')
    plt.xlabel('Date')
    plt.show()
    rf_rmse = np.sqrt(mean_squared_error(y_test, rf_pred))
    print("Random Forest RMSE of " + company + ": " + str(rf_rmse))
    return [rf, rf_rmse]

RF_AAPL = train_with_random_forest(PRE_AAPL, 'Apple')
RF_MSFT = train_with_random_forest(PRE_MSFT, 'Microsoft')
RF_GOOG = train_with_random_forest(PRE_GOOG, 'Google')
RF_AMZN = train_with_random_forest(PRE_AMZN, 'Amazon')
RF_META = train_with_random_forest(PRE_META, 'Meta')
```




Random Forest Predictions of Microsoft
Predicted Close
Actual Close
Price
340
320
300
280
260
240
220
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

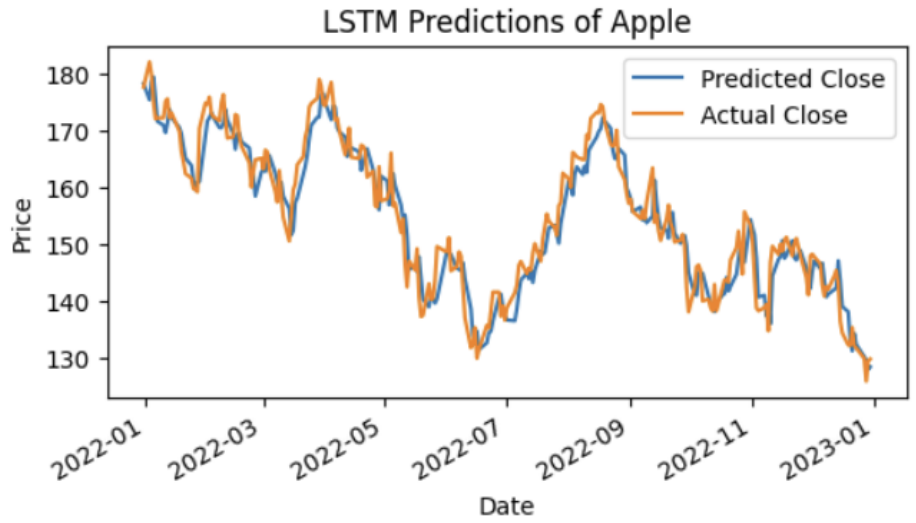

Random Forest Predictions of Google
Predicted Close
Actual Close
Price
140
120
100
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

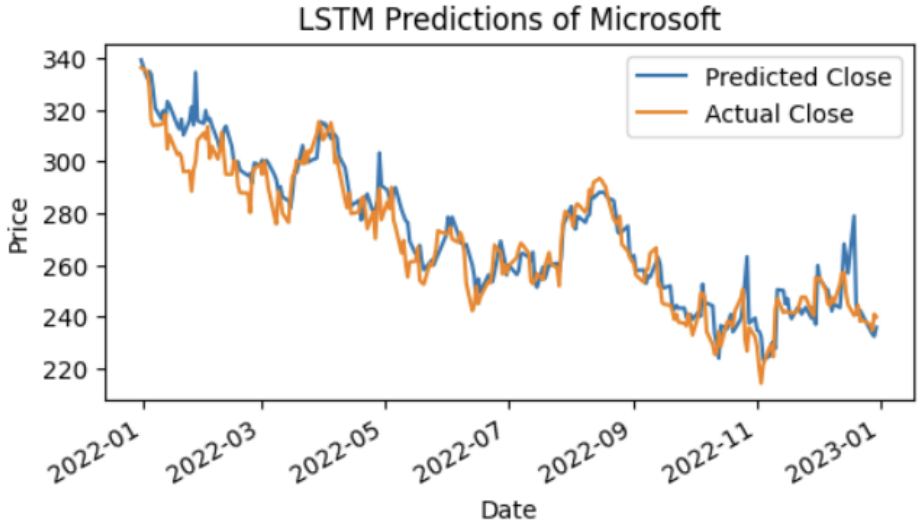

Random Forest Predictions of Amazon
Predicted Close
Actual Close
Price
160
140
120
100
80
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

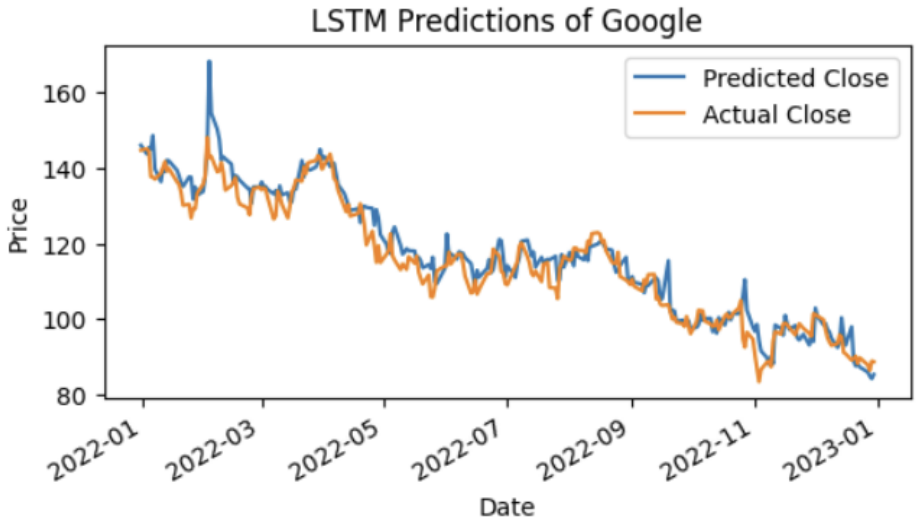

Random Forest Predictions of Meta
Predicted Close
Actual Close
Price
350
300
250
200
150
100
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

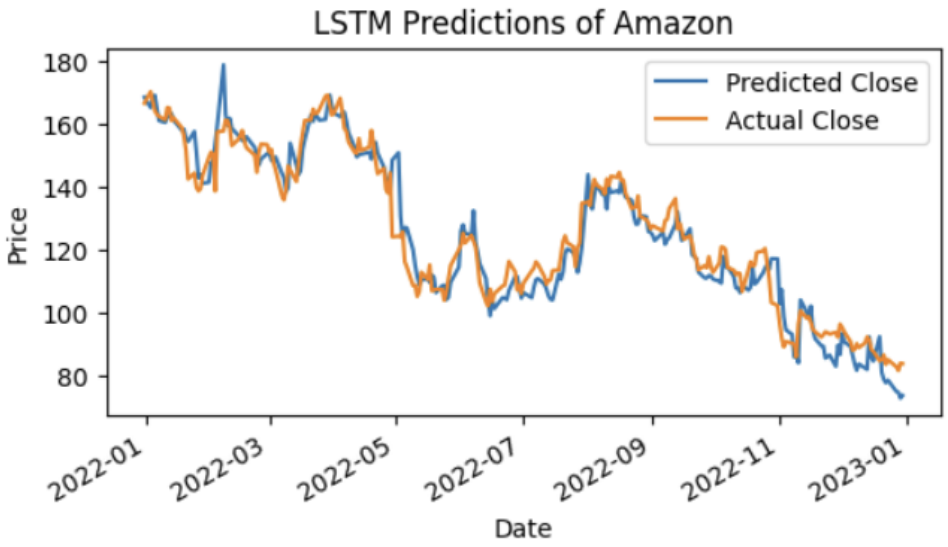
Long short-term memory (LSTM)

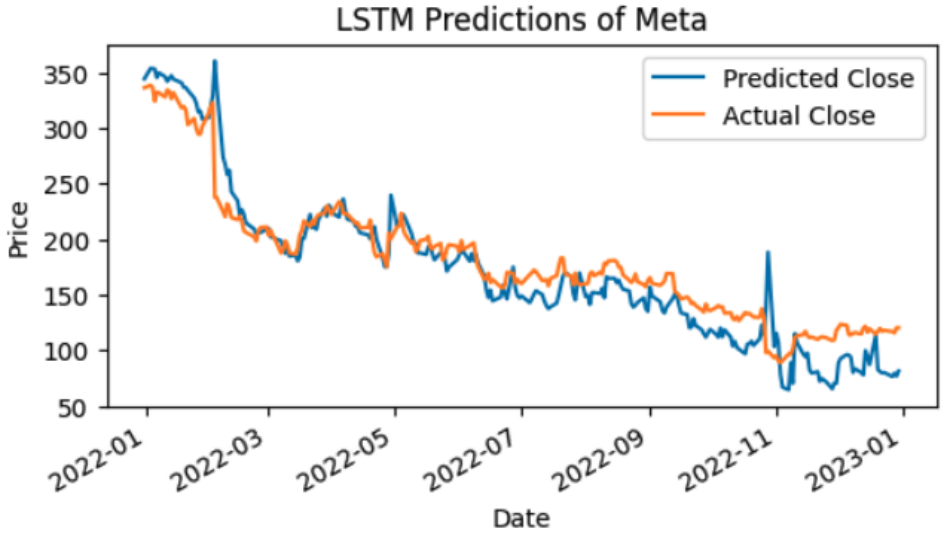
```python
def train_with_lstm(original_df, company):
    df = original_df.copy()
    train, test = train_test_split(df, test_size=0.2, shuffle=False)
    X_train = train.drop(['Close'], axis=1)
    y_train = train['Close']
    X_test = test.drop(['Close'], axis=1)
    y_test = test['Close']

    scaler = MinMaxScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    X_train = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
    X_test = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])

    lstm = Sequential()
    lstm.add(LSTM(50, activation='relu', input_shape=(1, X_train.shape[2])))
    lstm.add(Dense(1))
    lstm.compile(optimizer='adam', loss='mse')
    lstm.fit(X_train, y_train, epochs=100, verbose=0)

    lstm_pred = lstm.predict(X_test)
    lstm_pred = pd.DataFrame(lstm_pred, index=y_test.index, columns=['Close'])
    lstm_pred['Close'] = lstm_pred['Close'].shift(1)
    lstm_pred['Close'].iloc[0] = y_train.iloc[-1]
    lstm_pred['Close'].plot(figsize=(6,3), label='Predicted Close')
    y_test.plot(label='Actual Close')
    plt.legend(loc='best')
    plt.title("LSTM Predictions of " + company)
    plt.ylabel('Price')
    plt.xlabel('Date')
    plt.show()
    lstm_rmse = np.sqrt(mean_squared_error(y_test, lstm_pred))
    print("LSTM RMSE of " + company + ": " + str(lstm_rmse))
    return [lstm, lstm_rmse]
```


LSTM Predictions of Apple
Predicted Close
Actual Close
Price
180
170
160
150
140
130
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date


LSTM Predictions of Microsoft
Predicted Close
Actual Close
Price
340
320
300
280
260
240
220
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

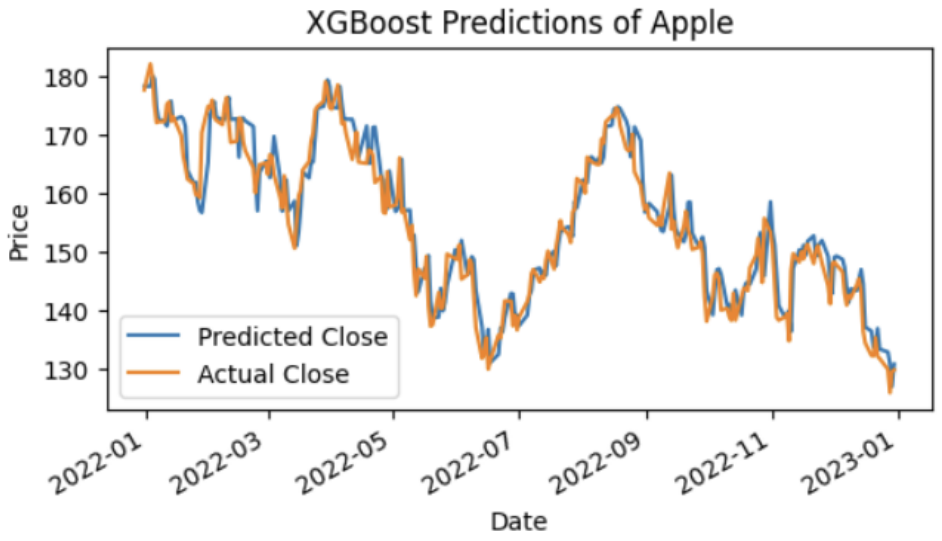

LSTM Predictions of Google
Predicted Close
Actual Close
Price
160
140
120
100
80
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

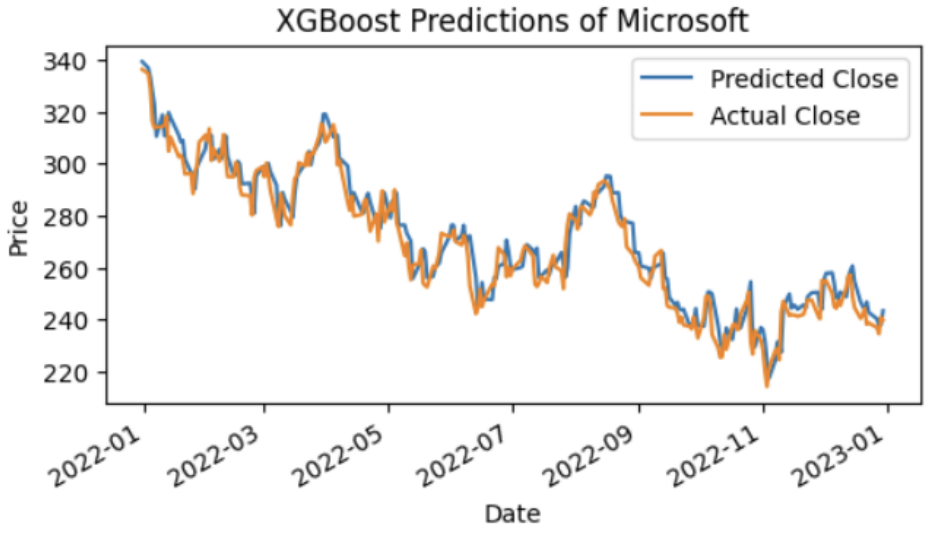

LSTM Predictions of Amazon
Predicted Close
Actual Close
Price
180
160
140
120
100
80
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

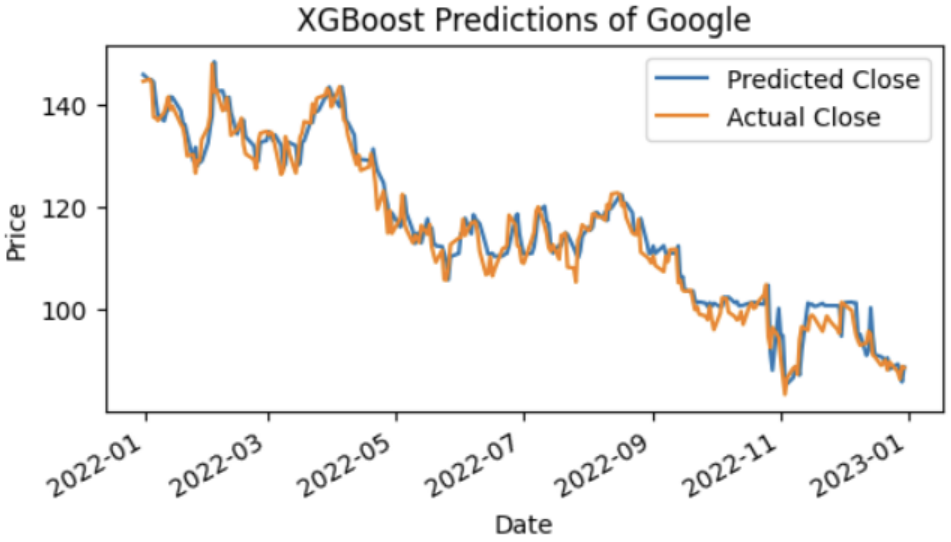

LSTM Predictions of Meta
Predicted Close
Actual Close
Price
350
300
250
200
150
100
50
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

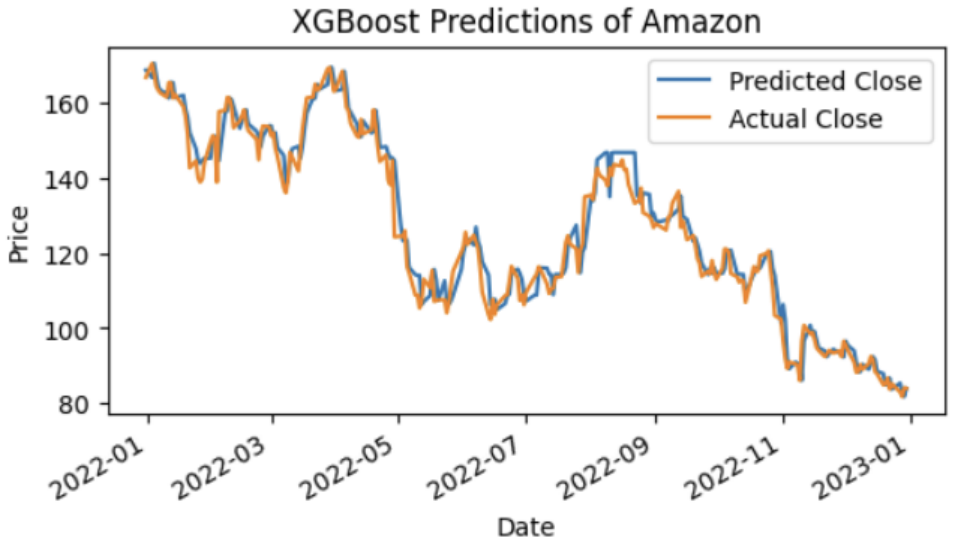
XGBoost

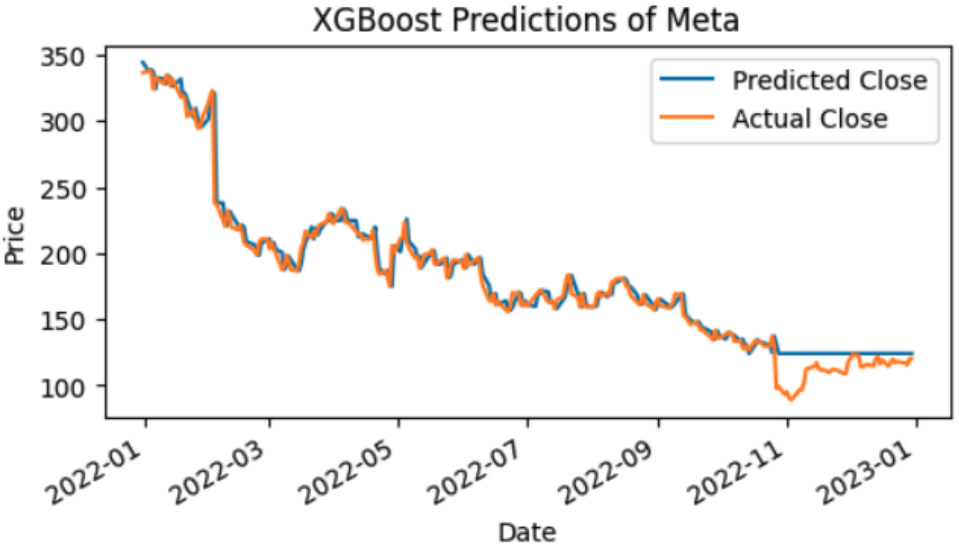
```python
def train_with_xgboost(original_df, company):
    df = original_df.copy()
    train, test = train_test_split(df, test_size=0.2, shuffle=False)
    X_train = train.drop(['Close'], axis=1)
    y_train = train['Close']
    X_test = test.drop(['Close'], axis=1)
    y_test = test['Close']

    xgb = XGBRegressor()
    xgb.fit(X_train, y_train)
    xgb_pred = xgb.predict(X_test)
    xgb_pred = pd.DataFrame(xgb_pred, index=y_test.index, columns=['Close'])
    xgb_pred['Close'] = xgb_pred['Close'].shift(1)
    xgb_pred['Close'].iloc[0] = y_train.iloc[-1]
    xgb_pred['Close'].plot(figsize=(6,3), label='Predicted Close')
    y_test.plot(label='Actual Close')
    plt.legend(loc='best')
    plt.title("XGBoost Predictions of " + company)
    plt.ylabel('Price')
    plt.xlabel('Date')
    plt.show()
    xgb_rmse = np.sqrt(mean_squared_error(y_test, xgb_pred))
    print("XGBoost RMSE of " + company + ": " + str(xgb_rmse))
    return [xgb, xgb_rmse]

XGB_AAPL = train_with_xgboost(PRE_AAPL, 'Apple')
XGB_MSFT = train_with_xgboost(PRE_MSFT, 'Microsoft')
XGB_GOOG = train_with_xgboost(PRE_GOOG, 'Google')
XGB_AMZN = train_with_xgboost(PRE_AMZN, 'Amazon')
XGB_META = train_with_xgboost(PRE_META, 'Meta')
```


XGBoost Predictions of Apple
Price
Date
Predicted Close
Actual Close
180
170
160
150
140
130
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01


XGBoost Predictions of Microsoft
Price
Date
Predicted Close
Actual Close
340
320
300
280
260
240
220
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01


XGBoost Predictions of Google
Price
Date
Predicted Close
Actual Close
140
120
100
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01


XGBoost Predictions of Amazon
Predicted Close
Actual Close
Price
160
140
120
100
80
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date


XGBoost Predictions of Meta
Predicted Close
Actual Close
Price
350
300
250
200
150
100
2022-01
2022-03
2022-05
2022-07
2022-09
2022-11
2023-01
Date

Evaluación de modelos

Con 4 modelos entrenados y evaluados, podemos ver que los resultados difieren bastante entre sí. Vamos a quedarnos con el mejor modelo para cada compañía.

```
BEST_AAPL = min(LR_AAPL[1], RF_AAPL[1], LSTM_AAPL[1], XGB_AAPL[1])
if BEST_AAPL == LR_AAPL[1]:
    print("Best model for Apple is Linear Regression")
    BEST_AAPL = LR_AAPL[0]
elif BEST_AAPL == RF_AAPL[1]:
    print("Best model for Apple is Random Forest")
    BEST_AAPL = RF_AAPL[0]
elif BEST_AAPL == LSTM_AAPL[1]:
    print("Best model for Apple is LSTM")
    BEST_AAPL = LSTM_AAPL[0]
else:
    print("Best model for Apple is XGBoost")
    BEST_AAPL = XGB_AAPL[0]

BEST_MSFT = min(LR_MSFT[1], RF_MSFT[1], LSTM_MSFT[1], XGB_MSFT[1])
if BEST_MSFT == LR_MSFT[1]:
    print("Best model for Microsoft is Linear Regression")
    BEST_MSFT = LR_MSFT[0]
elif BEST_MSFT == RF_MSFT[1]:
    print("Best model for Microsoft is Random Forest")
    BEST_MSFT = RF_MSFT[0]
elif BEST_MSFT == LSTM_MSFT[1]:
    print("Best model for Microsoft is LSTM")
    BEST_MSFT = LSTM_MSFT[0]
else:
    print("Best model for Microsoft is XGBoost")
    BEST_MSFT = XGB_MSFT[0]

BEST_GOOG = min(LR_GOOG[1], RF_GOOG[1], LSTM_GOOG[1], XGB_GOOG[1])
if BEST_GOOG == LR_GOOG[1]:
    print("Best model for Google is Linear Regression")
    BEST_GOOG = LR_GOOG[0]
elif BEST_GOOG == RF_GOOG[1]:
    print("Best model for Google is Random Forest")
    BEST_GOOG = RF_GOOG[0]
elif BEST_GOOG == LSTM_GOOG[1]:
```

```
elif BEST_GOOG == LSTM_GOOG[1]:
    print("Best model for Google is LSTM")
    BEST_GOOG = LSTM_GOOG[0]
else:
    print("Best model for Google is XGBoost")
    BEST_GOOG = XGB_GOOG[0]


BEST_AMZN = min(LR_AMZN[1], RF_AMZN[1], LSTM_AMZN[1], XGB_AMZN[1])
if BEST_AMZN == LR_AMZN[1]:
    print("Best model for Amazon is Linear Regression")
    BEST_AMZN = LR_AMZN[0]
elif BEST_AMZN == RF_AMZN[0]:
    print("Best model for Amazon is Random Forest")
    BEST_AMZN = RF_AMZN[0]
elif BEST_AMZN == LSTM_AMZN[0]:
    print("Best model for Amazon is LSTM")
    BEST_AMZN = LSTM_AMZN[0]
else:
    print("Best model for Amazon is XGBoost")
    BEST_AMZN = XGB_AMZN[0]

BEST_META = min(LR_META[1], RF_META[1], LSTM_META[1], XGB_META[1])
if BEST_META == LR_META[1]:
    print("Best model for Meta is Linear Regression")
    BEST_META = LR_META[0]
elif BEST_META == RF_META[1]:
    print("Best model for Meta is Random Forest")
    BEST_META = RF_META[0]
elif BEST_META == LSTM_META[1]:
    print("Best model for Meta is LSTM")
    BEST_META = LSTM_META[0]
else:
    print("Best model for Meta is XGBoost")
    BEST_META = XGB_META[0]
```

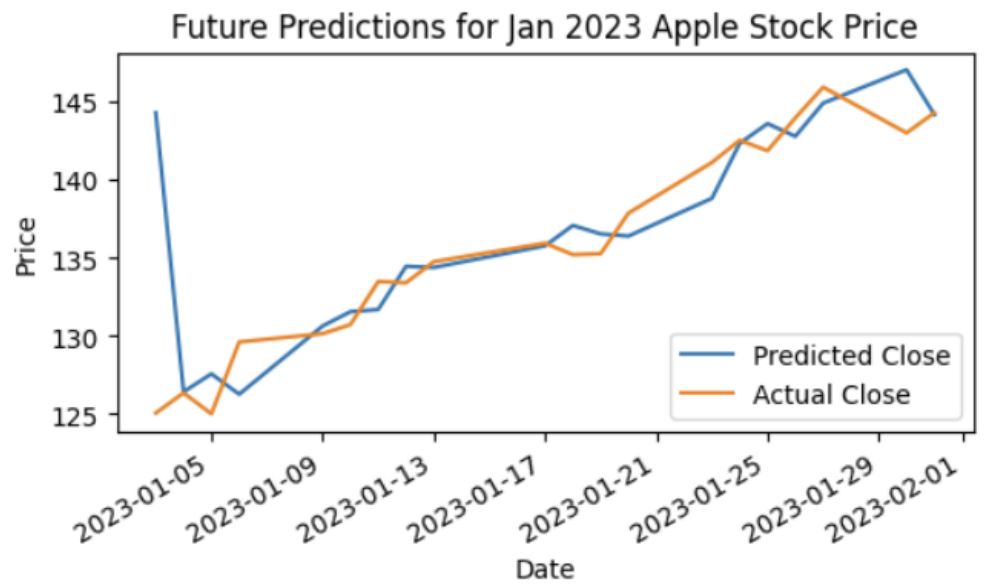
```
Best model for Apple is Linear Regression
Best model for Microsoft is Random Forest
Best model for Google is Linear Regression
Best model for Amazon is Linear Regression
Best model for Meta is Linear Regression
```

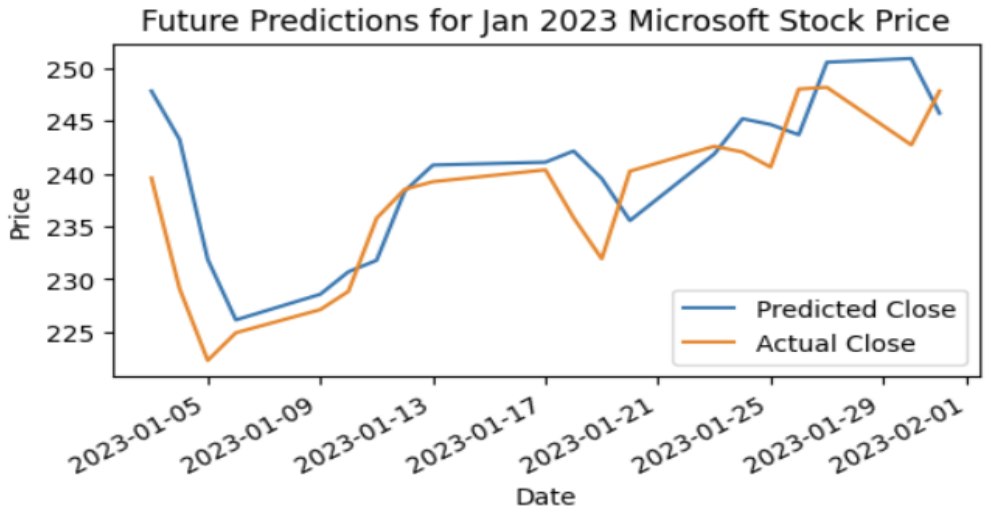
Predicción

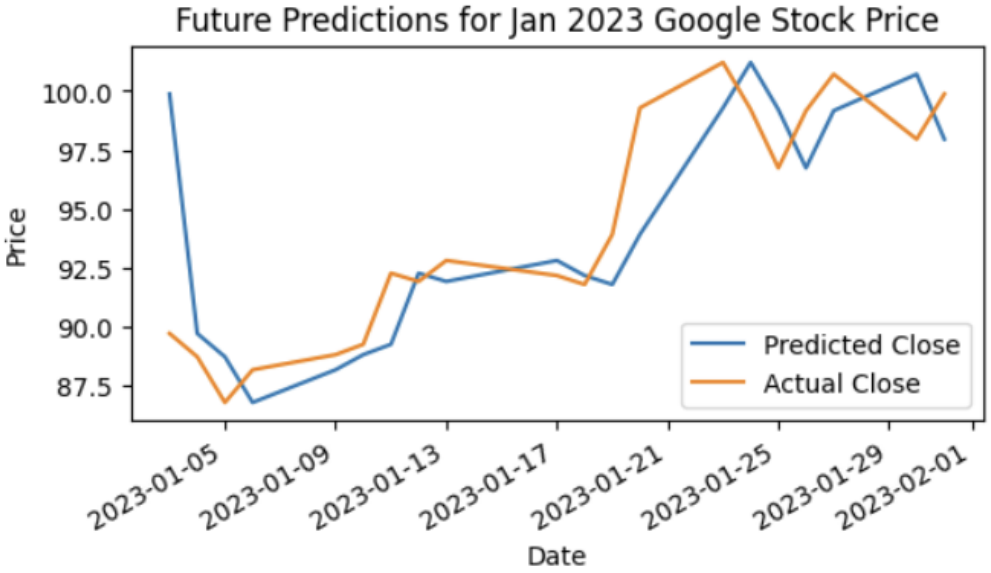
Con los mejores modelos seleccionados para cada compañía, vamos a analizar qué pasará en enero de 2023. Cabe destacar, que los modelos no han tenido ningún tipo de información sobre este mes, ya que la última entrada de los datasets consta del 31 de diciembre del 2022.

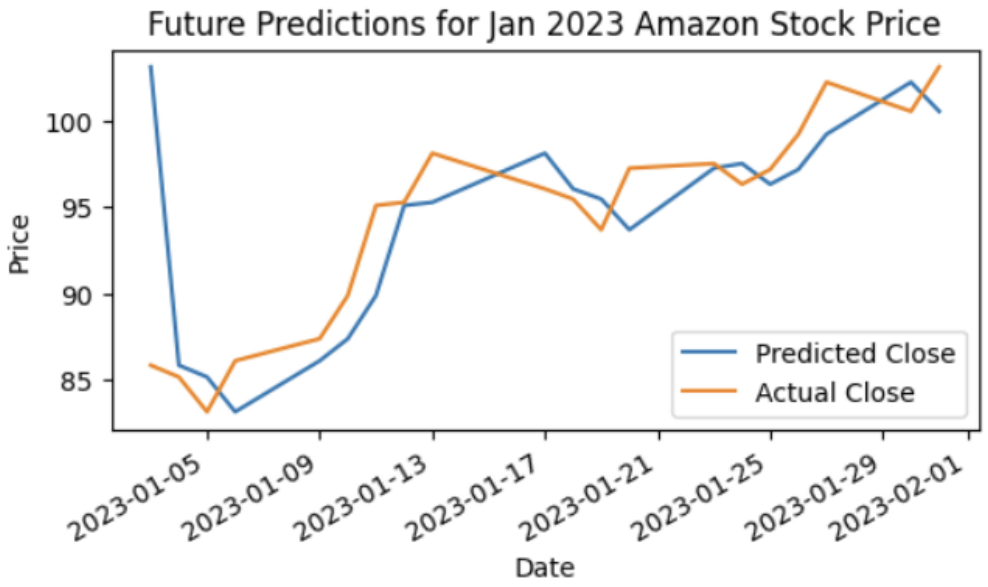
```
FUTURE_AAPL = pd.read_csv('./datasets/future/AAPL (Jan 2023).csv')
FUTURE_MSFT = pd.read_csv('./datasets/future/MSFT (Jan 2023).csv')
FUTURE_GOOG = pd.read_csv('./datasets/future/GOOG (Jan 2023).csv')
FUTURE_AMZN = pd.read_csv('./datasets/future/AMZN (Jan 2023).csv')
FUTURE_META = pd.read_csv('./datasets/future/META (Jan 2023).csv')

PRE_FUTURE_AAPL = preprocessing(FUTURE_AAPL)
PRE_FUTURE_MSFT = preprocessing(FUTURE_MSFT)
PRE_FUTURE_GOOG = preprocessing(FUTURE_GOOG)
PRE_FUTURE_AMZN = preprocessing(FUTURE_AMZN)
PRE_FUTURE_META = preprocessing(FUTURE_META)
```

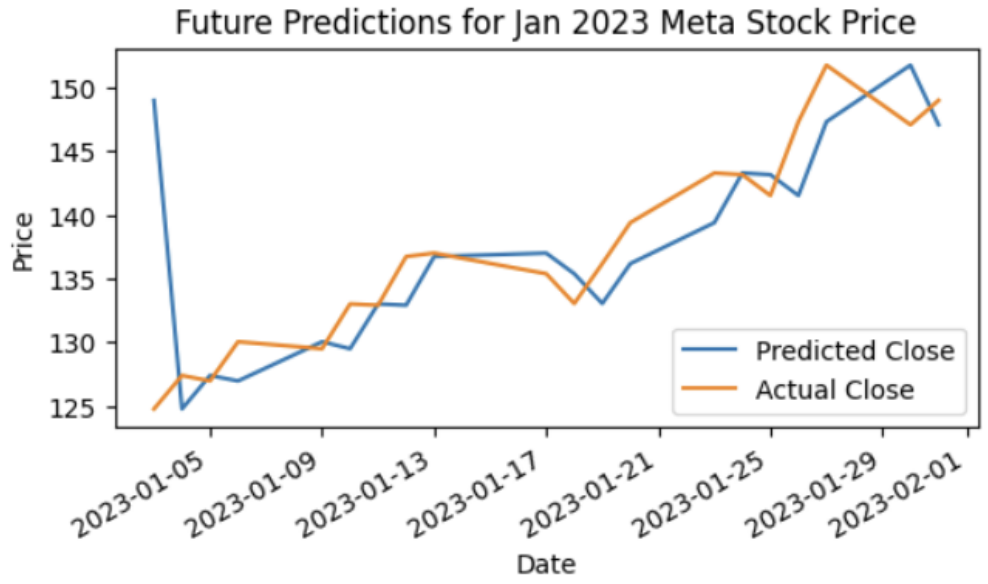
```
def predict_with_best_model(original_df, company, lr):
    df = original_df.copy()
    X = df.drop(['Close'], axis=1)
    lr_pred = lr.predict(X)
    lr_pred = pd.DataFrame(lr_pred, index=df.index, columns=['Close'])
    lr_pred['Close'] = lr_pred['Close'].shift(1)
    lr_pred['Close'].iloc[0] = df['Close'].iloc[-1]
    lr_pred['Close'].plot(figsize=(6,3), label='Predicted Close')
    df['Close'].plot(label='Actual Close')
    plt.legend(loc='best')
    plt.title("Future Predictions for Jan 2023 " + company + " Stock Price")
    plt.ylabel('Price')
    plt.xlabel('Date')
    plt.show()
    # calculate the root mean squared error
    rmse = np.sqrt(mean_squared_error(df['Close'], lr_pred['Close']))
    print('RMSE del mejor modelo de ' + company + ': %.3f' % rmse)


predict_with_best_model(PRE_FUTURE_AAPL, 'Apple', BEST_AAPL)
predict_with_best_model(PRE_FUTURE_MSFT, 'Microsoft', BEST_MSFT)
predict_with_best_model(PRE_FUTURE_GOOG, 'Google', BEST_GOOG)
predict_with_best_model(PRE_FUTURE_AMZN, 'Amazon', BEST_AMZN)
predict_with_best_model(PRE_FUTURE_META, 'Meta', BEST_META)
```


Future Predictions for Jan 2023 Apple Stock Price
Price
145
140
135
130
125
Predicted Close
Actual Close
2023-01-05
2023-01-09
2023-01-13
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
Date


Future Predictions for Jan 2023 Microsoft Stock Price
Price
250
245
240
235
230
225
Predicted Close
Actual Close
2023-01-05
2023-01-09
2023-01-13
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
Date


Future Predictions for Jan 2023 Google Stock Price
Price
100.0
97.5
95.0
92.5
90.0
87.5
Predicted Close
Actual Close
2023-01-05
2023-01-09
2023-01-13
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
Date


Future Predictions for Jan 2023 Amazon Stock Price
Price
100
95
90
85
Predicted Close
Actual Close
2023-01-05
2023-01-09
2023-01-13
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
Date


Future Predictions for Jan 2023 Meta Stock Price
Price
150
145
140
135
130
125
Predicted Close
Actual Close
2023-01-05
2023-01-09
2023-01-13
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
Date

4. Conclusiones

Al explorar los datos, fue interesante ver cómo cambiaron las cotizaciones bursátiles de las distintas empresas debido a la pandemia y cómo las cotizaciones bursátiles de las empresas tecnológicas se recuperaron más rápidamente que las de las demás empresas consideradas. Estos son algunos de los aspectos más destacados de la sección de exploración de datos:

- 2019: Antes de la pandemia, las acciones de la mayoría de las empresas iban relativamente bien, con Apple y Microsoft a la cabeza.
- 2020: Al inicio de la pandemia, en torno a la primavera, se produjo una caída de las cotizaciones bursátiles de todas las empresas, pero después las compañías tecnológicas como Amazon, Apple, Microsoft y Google empezaron a crecer de nuevo.
- 2021: Empresas como Google y Microsoft crecieron. En general, hubo una mejora en los precios de las acciones de todas las empresas que consideramos.

Future work

El mercado bursátil ha sido siempre el tema más candente cuando se trata de prever series temporales o de intentar intuir hacia dónde se dirige el mercado en general. Es imposible encontrar la fórmula "a seguir" para predecir la dirección del mercado de valores, debido a la constante volatilidad del mercado, la incertidumbre de las variables móviles que podrían afectar a la volatilidad del mercado de valores, desde el riesgo asociado a la inestabilidad política y los factores macroeconómicos, bueno, la lista podría continuar. Sorprendentemente, estos modelos nos han dado muy buenos resultados. Han predicho con bastante precisión la evolución y tendencia del valor de los stocks. Sin embargo, si esto fuese verdad, todo el mundo sería rico. La realidad es mucho más dura, y aunque nuestro modelo refleja parcialmente un sistema tan variable como el mercado bursátil, este está lleno de aleatoriedad y estadística muy compleja que es imposible de reflejar completamente en un proyecto académico. Un punto de mejora sería dedicar más tiempo a afinar los parámetros de los distintos modelos, así como a incluir más características que puedan ser relevantes para la predicción del precio de las acciones, y revisar bien los algoritmos de aprendizaje para comprobar que están usando la información realmente necesaria.

Con estas predicciones acabamos nuestro proyecto de MLE. Durante la realización de este proyecto, hemos buscado, leído y procesado datasets para adecuarlos a nuestro objetivo concreto. Hemos generado medidas estadísticas derivadas de

nuestros datos, analizando y visualizando distintas gráficas para poder obtener información del dominio del mundo bursátil. Hemos creado y evaluado 4 modelos de aprendizaje distintos, quedándonos con el mejor para cada caso concreto de las 5 empresas analizadas, y para finalizar hemos realizado una predicción futura de los valores de sus acciones.