Capstone Project

Paulo Mota

Data Scientist Nanodegree

May 2nd, 2023

Definition

Project Overview

The stock market is a dynamic and volatile environment, with prices constantly fluctuating based on a variety of factors such as economic conditions, company performance, and investor sentiment. Accurately predicting stock prices is a challenging task, but it can provide significant benefits to investors and traders. This project aims to develop a stock price predictor forecasting model that utilizes machine learning algorithms to forecast future prices based on historical data.

Problem Statement

The primary objective of this project is to develop an accurate and reliable stock price predictor forecasting model. The model will utilize machine learning algorithms such as regression, time-series analysis, and deep learning to analyze historical data and predict future stock prices. The specific objectives of the project are as follows:

- Gather and preprocess historical stock data from yahoo API
- Develop and train machine learning models for stock price prediction using Prophet
- Evaluate model performance and select the most accurate model
- Deploy a web application to user predict a close price from a stock symbol
- Extra: Create a stock comparator with historical data.

Metrics

The performance of each model will be evaluated using appropriate metrics such as mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE). The model with the best performance will be selected for the deployment model. It measures the average absolute difference between the actual values (y) and the predicted values (yhat).

The formula for calculating MAE is:

$$MAE = (1/n) * \Sigma|y - yhat|$$

Where:

- n is the number of data points
- Σ represents the sum over all data points
- | | denotes the absolute value

To calculate MAE in Python, we can use the scikit-learn library's mean_absolute_error() function. The function takes two arguments: the actual values (test_data['y']) and the predicted values (predictions['yhat']).

The Root Mean Squared Error (RMSE) is another metric used to evaluate the accuracy of a predictive model. It measures the square root of the average of the squared differences between the actual values (y) and the predicted values (yhat).

The formula for calculating RMSE is:

RMSE = sqrt((1/n) * Σ(y - yhat)^2)

Where:

- n is the number of data points
- Σ represents the sum over all data points
- sqrt denotes the square root

To calculate RMSE in Python, we can use the numpy library's sqrt() function and the scikit-learn library's mean_squared_error() function. The function takes two arguments: the actual values (test_data['y']) and the predicted values (predictions['yhat']).

Analysis

Data Exploration

I used Yahoo Finance API to get the stock data by symbol.

The stock data generally contains information on the performance of a particular stock, and typically includes several columns such as open, high, low, close, volume, and adj.

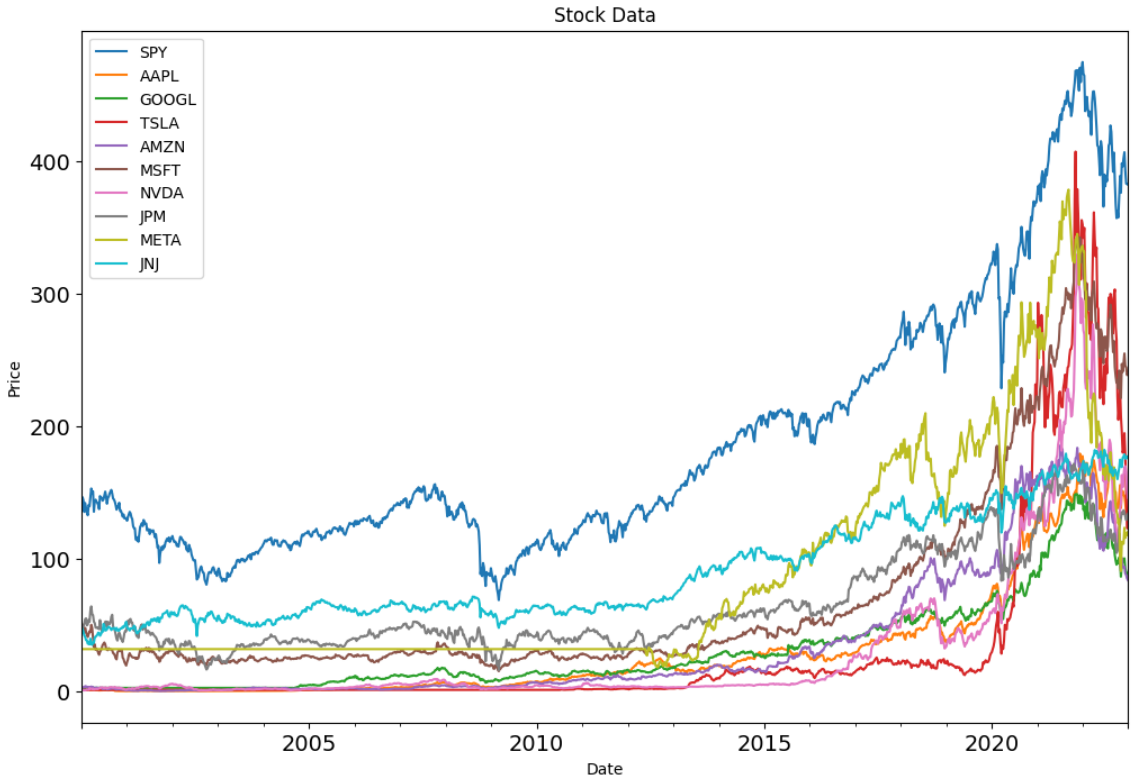
- Open: the price of the stock at the opening of the trading day.
- High: the highest price of the stock during the trading day.
- Low: the lowest price of the stock during the trading day.
- Close: the price of the stock at the end of the trading day.
- Volume: the number of shares of the stock that were traded during the trading day.
- Adj: short for adjusted, this column represents the adjusted stock price that takes into account any changes to the stock such as stock splits or dividends.
-

These data points are often used by investors, analysts, and traders to analyze the performance of a particular stock, and to make informed decisions about buying or selling that stock. For example, the high and low values can indicate the volatility of the stock, while the volume can provide an indication of how active the market is for that particular stock. The adjusted price is often used to analyze the long-term performance of a stock, as it reflects any changes to the stock that may have occurred over time.

In terms of domain understanding, I realized to make two cleanings:

- backfill and forward fill (because there is no stock data on weekends and which company starts and ends stock activities in different time periods.)

Exploratory Visualization



The following plot displays the closing prices for a selection of stock symbols, after the data processing stage where backfill and forward fill techniques were utilized to fill in any gaps in the stock data that may have occurred within the user's specified date range.

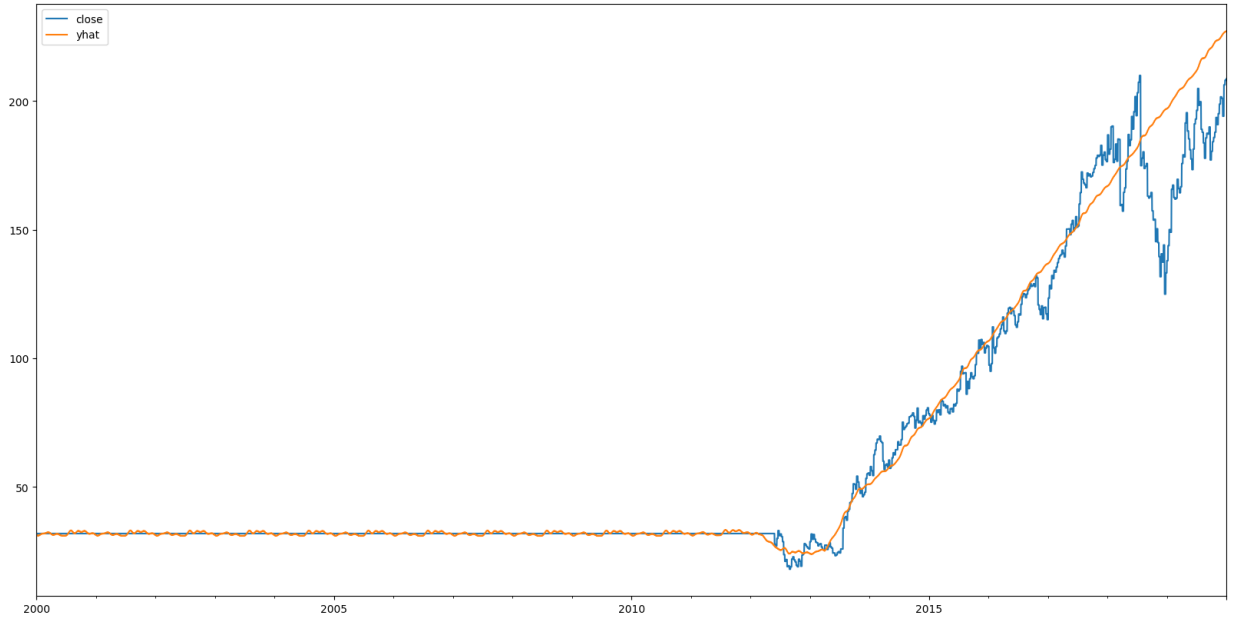
Algorithms and Techniques

The model is a time series forecasting model developed by Facebook called Prophet. It is designed to be easy to use and to provide accurate forecasts for a variety of time series data in our case Stock Price data.

For training the model I needed to prepare my data in a specific format. The data should be in a pandas DataFrame with two columns: 'ds' and 'y', where 'ds' is a datetime column containing the dates and 'y' is the column containing the values you want to forecast.

For testing my predictions I used “Backtesting” , which involves applying a trading strategy to historical stock price data to see how well it would have performed if it had been used in the past.

The figure below shows, the result of predictions about **META**(Facebook stock symbol)



Methodology

Data Preprocessing

The preprocessing done, consists of the following steps:

1. Connection with Yahoo Finance API
2. Get JSON data into Pandas Dataframe
3. Fill stock price data using backfill and forward fill.
4. Create dataframe for train and test

Implementation

The implementation process can be split into two main stages:

1. Prophet training stage
2. The application development stage

During the first stage, the classifier was trained on the preprocessed training data.

1. Implement main functions:
    a. get_stock_data_by_symbol(...): Get pandas dataframe with stock data from a symbol and specific start date and end date.
    b. get_data(...): Get pandas dataframe with stock data from several stock symbols to be plot easier in the line chart.
    c. get_data_for_training(...). Get train dataset and test dataset to be processed in the Prophet model.
2. Train the model with stock data
3. Test the predictions with error metrics mentioned before.
4. Use Streamlit library to receive the user input to be processed dynamically.
5. Deploy the application using Streamlit cloud.

As intermediate solutions I have chosen to predict 1 year forward just because the domain subject stock prices are really flexible and there are many external factors that affect them drastically.

Here are a list of intermediate solutions:

- Use symbols with more stock price data (old companies)
- Try with symbols with less stock price data (new companies)

- Companies with more data of course the model perform better.

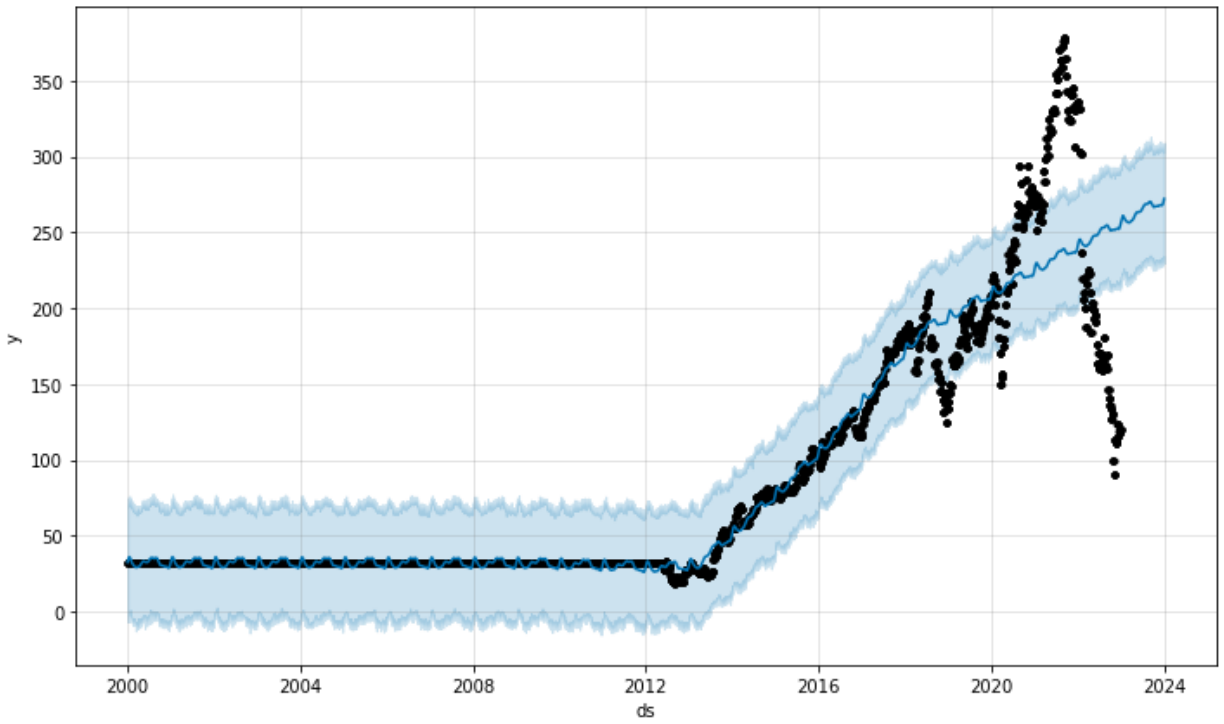
The interesting point is that I trained the model in the as go per symbol , each symbol has its own model.

Refinement

Refining a Prophet model involves making adjustments to the model's settings, such as its hyperparameters, to improve its accuracy and performance.

If the initial results suggest that the model could benefit from additional information, such as the impact of external factors like holidays or economic indicators, I can add external regressors to the model. This involves including additional data in the form of covariates, which can help the model better capture the relationship between the predictor variables and the target variable.

The plot below displays the real and predicted close values for **META** stock price. Divergence between the two lines indicates periods where the model's predictions were not accurate, such as during the COVID-19 pandemic. To improve the model's accuracy during these specific periods, it is recommended to incorporate this information into the model by providing additional data or adjusting the model's settings. This will help the model better understand the impact of these events on stock prices and improve the accuracy of its predictions.(y = close value and ds = date)



Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.

The error metrics MAE and RMSE will depend on the stock symbol data that was used to train the model. Because some companies have more data than others and it affects the performance.

To evaluate the model properly I used the cross validation available on prophet.

Horizon (days)

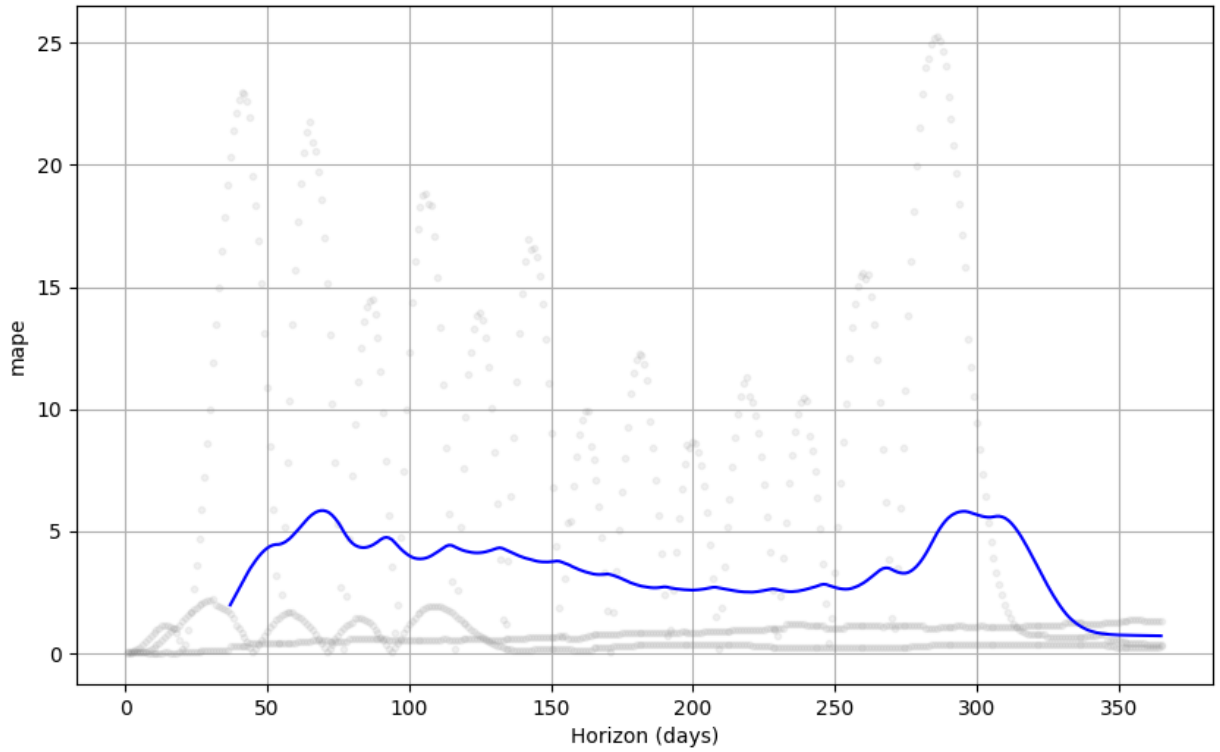
In [137]:
```python
# Python
import itertools
import numpy as np
import pandas as pd

param_grid = {
    'changepoint_prior_scale': [0.001, 0.01, 0.1, 0.5],
    'seasonality_prior_scale': [0.01, 0.1, 1.0, 10.0],
}

# Generate all combinations of parameters
all_params = [dict(zip(param_grid.keys(), v)) for v in itertools.product(*param_grid.values())]
rmses = []  # Store the RMSEs for each params here

# Use cross validation to evaluate all parameters
for params in all_params:
    m = Prophet(**params).fit(df_train)  # Fit model with given params
    df_cv = cross_validation(model, cutoffs=cutoffs, horizon='30 days', parallel="processes")
    df_p = performance_metrics(df_cv, rolling_window=1)
    rmses.append(df_p['rmse'].values[0])

# Find the best parameters
tuning_results = pd.DataFrame(all_params)
tuning_results['rmse'] = rmses
print(tuning_results)
```

As a result I've trained 15 models and automatically got the best one.

Seasonality has period of 365.25 days which is larger than initial windo

	changepoint_prior_scale	seasonality_prior_scale	rmse
0	0.001	0.01	58.554334
1	0.001	0.10	58.554334
2	0.001	1.00	58.554334
3	0.001	10.00	58.554334
4	0.010	0.01	58.554334
5	0.010	0.10	58.554334
6	0.010	1.00	58.554334
7	0.010	10.00	58.554334
8	0.100	0.01	58.554334
9	0.100	0.10	58.554334
10	0.100	1.00	58.554334
11	0.100	10.00	58.554334
12	0.500	0.01	58.554334
13	0.500	0.10	58.554334
14	0.500	1.00	58.554334
15	0.500	10.00	58.554334

plot about cross validation:



Justification

I successfully deployed the web application that allows the user to play with the stock symbols and get the predictions. try it out on the link https://paulowiz-udacity-stock-predictor-main-3gk95u.streamlit.app/\

Time-series data is unique in that it has a temporal component that defines the relationship between data points. This temporal structure can include seasonality, trends, and other patterns that are specific to the data being analyzed. When dealing with missing values in time-series data, it is important to preserve this temporal structure to accurately represent the underlying data. The forward fill and backfill techniques do just that by filling in missing values with the nearest observed value in the time-series. This method ensures that the imputed values are consistent with the overall pattern of the data, which is important for any subsequent analysis or modeling.

In contrast, using the mean technique to fill in missing values assumes that the missing values are similar to the other data points in the time-series. However, this may not always be the case, and the resulting imputations may not accurately represent the underlying data. Additionally, using the mean technique can introduce bias into the data if the mean value is influenced by outliers or extreme values. Therefore, the forward fill and backfill techniques are generally preferred over the mean technique for completing missing values in time-series data because they preserve the temporal structure, provide a better representation of the underlying data, and have a lower risk of introducing bias.

The figures below show a snapshot of the application running in production:

Closing price and daily returns comparator

Start date

1993/01/01

End date

2023/05/01

Enter Symbols Separated by comma (,)

GOOG,TSLA,MSFT

⚠ The chart will not display invalid symbols or symbols that are not listed in the USA market.

Run

Closing price by stock symbol

Closing price predictor with Prophet model

More info about Prophet: https://facebook.github.io/prophet/

Why should you predict stock value?

Stock value prediction is to forecast the future price or value of a stock, which is influenced by various factors such as company performance, market conditions, and global economic trends. By accurately predicting the future value of a stock, investors can make informed decisions about whether to buy, sell, or hold a particular stock, potentially resulting in significant financial gains.

Stock Symbol

TSLA

Predict Stock

Closing price forecast - TSLA

Conclusion

My project successfully utilized the Prophet model to accurately predict stock price data for a period of one year. By integrating the Streamlit framework, I was able to provide users with an interactive platform to visualize and analyze the predicted data. My project can serve as a valuable tool for investors and traders looking to make informed decisions based on reliable stock market predictions. I believe that our project has the potential to make a significant impact in the field of finance and can contribute towards the development of more advanced predictive models in the future.

Training a time-series model can be challenging due to the unique aspects of time-series data. One of the main challenges is ensuring data quality. Time-series data can be affected by various factors such as measurement errors, data gaps, and outliers, which can introduce noise and distortions in the data. Therefore, it is important to carefully pre-process the data and remove any inconsistencies to ensure that the model is trained on high-quality data. Additionally, temporal dependencies between data points must be considered when training a time-series model. This includes identifying seasonality, trends, and other patterns in the data that can be used to inform the model.

Selecting the right model for a particular dataset is another challenge in training a time-series model. There are several types of time-series models, each with their own strengths and weaknesses. Choosing the appropriate model for a given dataset can be challenging, especially if the data has multiple factors that affect it. Furthermore, time-series models often require tuning of several hyperparameters, such as the window size, number of hidden layers, and learning rate, to achieve optimal performance. Tuning these hyperparameters can be time-consuming and require extensive experimentation to find the best values. Despite these challenges, developing accurate and reliable time-series models is critical for many applications, and careful attention to data quality, temporal dependencies, model selection, and hyperparameter tuning can help overcome these challenges.

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, yahoo finance api was found
2. The connection between the application and Yahoo finance API.
3. The Prophet was trained using the data from stock symbols
4. Streamlit web framework was used to create the system interface and interaction
5. All plots in the experiment were rewritten to streamlit.

Improvement

To achieve the optimal user experience, we can add more inputs in the web application to the user control the period of training and how much time they would like to predict about the stock price, also we can improve the current model checking more params that work better with holidays, and specific events in the historical time which affects the predictions.