SN: 21037837

Anonymous authors
Paper under double-blind review

Abstract

The S&P 500 index is a critical benchmark for the U.S. stock market, and understanding its trends is essential for informed investment decisions. This project aims to build a machine learning pipeline to predict the S&P 500 stock price. Using historical stock data from April 2017 to April 2024, including supplementary AAPL and MSFT stock data, a linear regression model is trained to forecast future prices. An ablation study is conducted to evaluate the impact of removing AAPL and MSFT data, demonstrating their significance in improving model accuracy. The resulting model follows the trend of the real S&P 500 stock reasonably well, and removal of the supplementary stocks AAPL and MSFT show a measurable decrease in measured accuracy.

1 Introduction

The S&P 500 index, which consists of the 500 leading publicly traded companies in the United States Kenton (2024), serves as a good indicator of the health of the U.S. stock market. Accurately predicting the movements of the S&P 500 is crucial for investors seeking to make informed decisions about their assets, and allow trading companies to maintain an edge over their competitors.

The end goal of this project is to build a machine learning pipeline that can predict the price of the S&P 500 using historical stock data. To increase model accuracy, supplemental data from AAPL and MSFT stocks (Apple and Microsoft) were also used. These two companies were chosen due to their leading positions in tech, a field that is quickly becoming one of the most valuable and important to the U.S economy. These two companies also have a significant influence on the S&P 500, with Apple alone contributing 6.8% to the index Kenton (2024).

2 Data description

The dataset used for this study consists of daily stock market data for the S&P 500 index (SPX), AAPL, and MSFT, spanning from 1st April 2017 to 1st April 2024. Each dataset contains the following attributes for each trading day:

- **Date**: The trading date.
- **Open**: Opening price of the stock/index.
- **High**: Highest price of the stock/index during the day.
- **Low**: Lowest price of the stock/index during the day.
- **Close**: Closing price of the stock/index.
- **Volume**: Number of shares traded during the day.
- **Dividends**: Dividends issued on the trading day (if any).
- **Stock Splits**: Stock split adjustments (if any).

Each trading day is given as a single row, and each day's attributes form the columns. The dataset for each stock contains 1759 rows, or 1759 trading days. Datasets were acquired using the yfinance Python library ranaroussi (2017) and saved as CSV files, before being converted to JSON objects and stored in a MongoDB collection. More details will be presented in the subsequent sections.

3 Data acquisition

As mentioned previously, the yfinance Python library was used to acquire all of the required stock data. The library's ease of use and Python integration allowed for quick and straightforward coding, while its direct connection to Yahoo Finance's database ensures that the stock data is always reliable. The library is also distributed under the Apache License Apache (2004), making it fully open source and usable under the scope of this project without any royalty fees.

To demonstrate yfinance's ease of use, the following is all that is required to import S&P 500 data for the required time frame and save it to a CSV file titled 'sp500':

```
sp500_data = get_stock_data("^GSPC", start_date, end_date, "sp500.csv")
```

Where ĜSSC is the Yahoo! Finance ticker for the S&P 500, start_date, end_date are variables that contain the dates required, and "sp500.csv" is the name of the CSV file that will contain the acquired dataset. This process is then repeated to acquire datasets for AAPL and MSFT stock. To ensure data consistency across multiple stocks, each CSV file can be manually inspected to make sure that all dates and attributes are lined up.

4 Data storage

To store the acquired datasets, MondoDB was chosen due to its document-oriented structure and quality-of-life features. While traditional relational databases like SQL would also be suitable for the consistent structure of stock data, the flexibility and ease of MongoDB seemed more valuable for the project. The CSV files for each stock are read and each row (trading day) is converted to a JSON format in Python, before being uploaded to the MongoDB collection. Once uploaded, the documents can be manually inspected using the MongoDB Atlas dashboard to check for errors, as shown in figure 1.

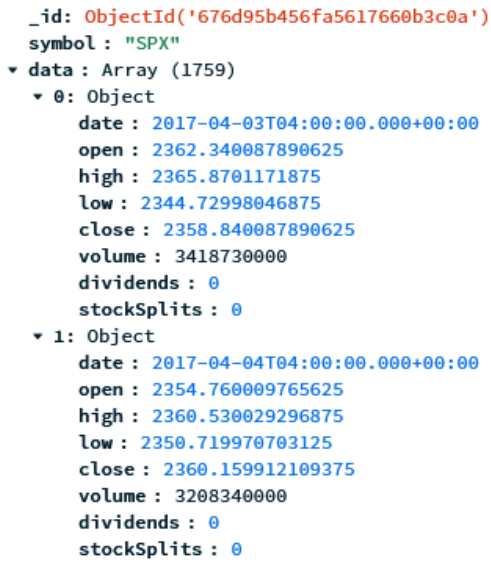

Figure 1: Excerpt of MongoDB database for SP500 stocks.

5 Data preprocessing

After storing the data, basic data cleaning was done in the form of outlier and missing value/duplicate detection. As expected from a reliable data source like Yahoo!, there were no missing values or duplicates. The only outliers found were in the trading volume attribute, caused by days that saw an unusually high number of transactions. While this may be an interesting topic to investigate, it is not within the scope of this project and was disregarded. Therefore, the code that was used to calculate the inter-quartile range and detect outliers was omitted from the submission, but will be included in the appendix of this report.

Preprocessing of the dataset to prepare it for model training involved merging the S&P 500, AAPL, and MSFT datasets along the date column to align all the datasets chronologically. As the pipeline only aims to predict future closing prices, unnecessary attributes such as open, high, low, volume, dividends, and stock splits were removed. Additional features were also created to improve predictions. Lagged values of all three stocks' closing price from 1 day and 7 days prior (SP500_Lag1 and SP500_Lag7, etc.), as well as 7-day and 30-day moving averages of the S%P 500 (SP500_MA7 and SP500_MA30).

Lagged values, essentially past values of the data, are included to allow the model to learn patterns from historical data, which is vital for predicting future values. Moving averages show the average value of the data within the specified window (e.g. 7 days). This reduces noise and smooths out short-term fluctuations and instead focuses on long-term trends, allowing the model to 'see the big picture' Brownlee (2020). However, lagged features are still effective in providing the model with an idea of short-term trends and fluctuations, as none of these are smoothed out.

6 Data exploration

Several methods were used to analyse the acquired data. The closing prices were plotted over time for all three stocks, then normalised for better legibility. A seasonality analysis was also conducted, in order to look for any cyclical trends. Finally, a correlation matrix was plotted after the data preprocessing stage to evaluate the feature selection.

6.1 Visualisation

To verify that no major errors were made during the data acquisition and storage process, separate time series plots of each stock were made and manually compared to real charts. This was done using the matplotlib Python library. Next, a plot with all three stocks together was produced to visualise AAPL and MSFT's contribution to the S&P 500 index, shown in figure 2.

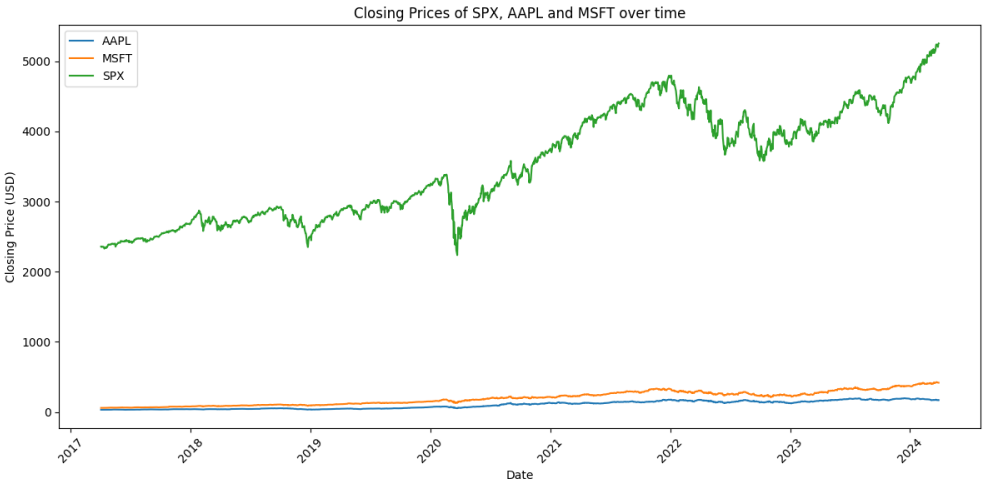

Figure 2: S&P 500, AAPL, MSFT plotted on absolute scale.

As expected, AAPL and MSFT trends are difficult to analyse due to the S&P 500 index skewing the scale of the chart. Therefore, all stock prices were individually normalised. This is done by scaling the closing price of each trading day to a value between 0 and 1, with 0 being the minimum value and 1 being the maximum:

$$\text{normalised price} = \frac{\text{closing price} - \text{min price}}{\text{max price} - \text{min price}}$$

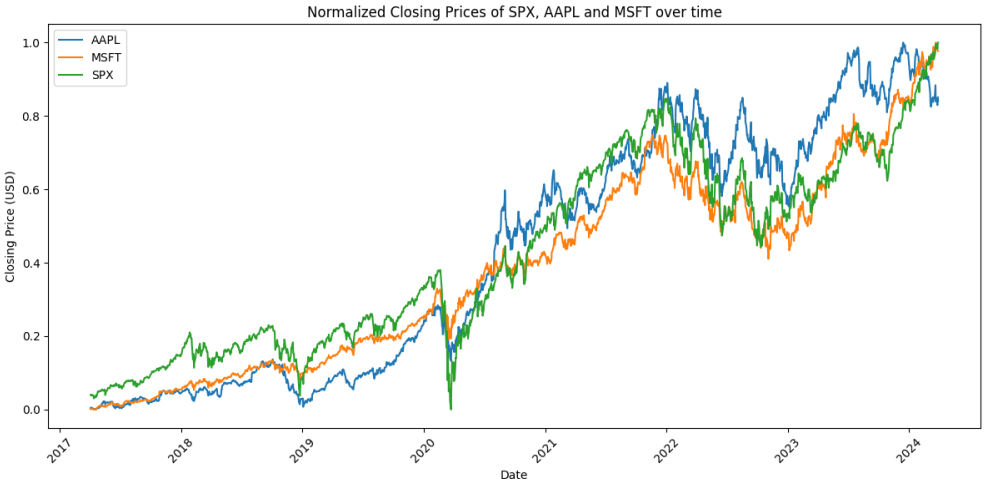

Figure 3: Normalised prices of all stocks plotted together.

As shown in figure 3, normalisation allows the trends of the three stocks to be compared more clearly. All three stocks follow the same general trends, notably, all are equally affected by the initial lockdown caused by the COVID-19 pandemic in March of 2020.

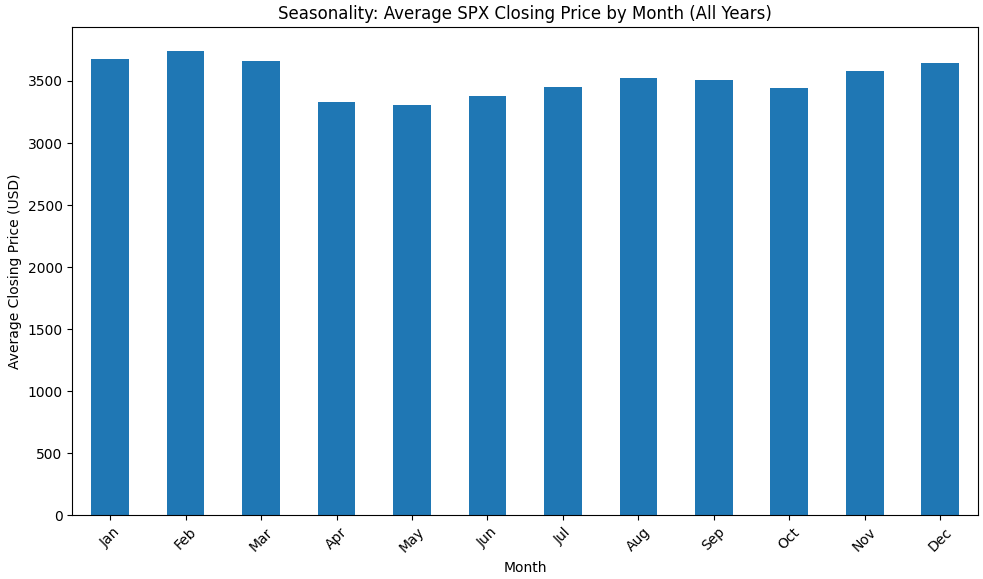

Figure 4: Seasonality analysis of S&P 500 closing prices.

6.2 Seasonality

Seasonality refers to recurring patterns or trends evident in the data at regular time intervals, i.e. one calendar year. In the context of the S&P 500, the seasonality analysis is visualised by plotting the average closing price of each calendar month as a bar chart. This is demonstrated above, in figure 4.

Most notably, the highest closing prices are in the first quarter, which could be linked to investor optimism going into the new year, known as the 'January effect' Menton (2025). Another phenomenon, known as 'sell in May and go away', is also evident here. Between the months of May and October, investors sell their stocks and opt for lower risk investments during the summer months Segal & Scott (2024). However, the credibility of these phenomena are widely disputed, and only cause relatively small fluctuations of under 10%.

6.3 Correlation analysis

After the preprocessing and feature engineering stage, a correlation matrix was created to visualise the relationship between the different features and the target variable. This was done by calculating the correlation coefficient of each feature using the '.corr()' function in the pandas library, then plotting the results on a heatmap using the seaborn library. The correlation coefficient can also be manually calculated using the following formula:

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

The result is a value between 0 and 1, known as the correlation coefficient. A value of 0 suggests no correlation, while a value of 1 suggests perfect correlation. In figure 5, strongest correlation can be seen between the close_sp500 variable and the other features that were derived from it, such as the lagged and moving averages, as expected. Relatively speaking, the correlations between the close_sp500 variable and the AAPL and MSFT stocks are lower, but still quite high.

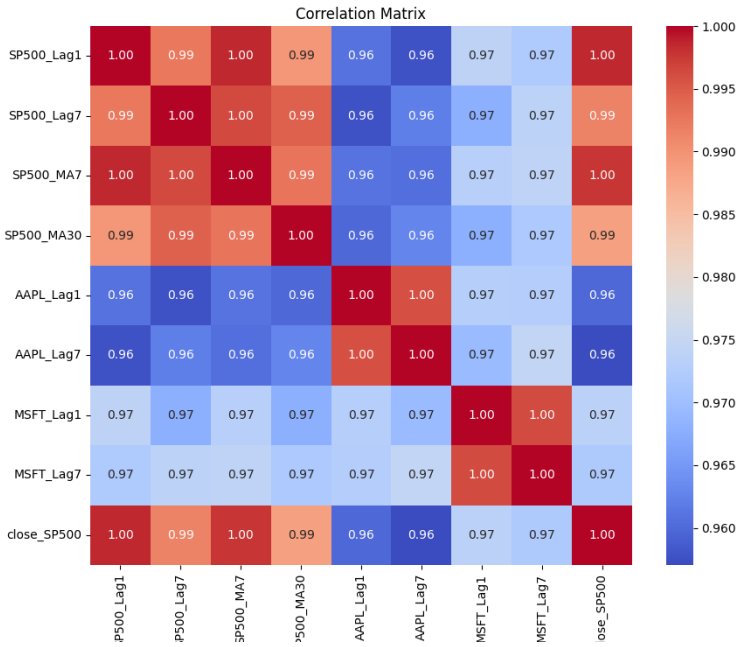

Figure 5: Correlation matrix between features and target variable

7 Training and inference

7.1 Sampling strategy

Data was sampled using the a time-based split, where the split dates were specified in the assignment brief. This consists of all stock data before 01/02/2024 being used for training, and all stock data after that date used for testing.

7.2 Model selection and training

A linear regression model was chosen mainly due to its simplicity, but has also been shown to be effective in predicting market trends Sangeetha & Alfia (2024). Linear regression can be especially useful for predicting numerical trends that follow relatively linear patterns, such as large stock indexes. It is also computationally efficient and easily available through Python libraries such as scikit-learn, allowing for quick recalculations and the ability to run large datasets without access to powerful hardware Walker (2024).

After training the model using the extracted features, it was asked to predict the prices of the testing split, and the results plotted against the actual prices of the testing period.

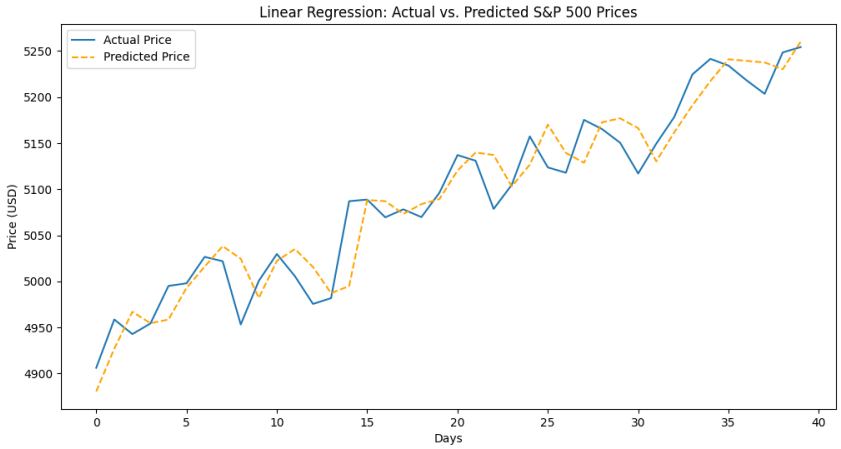

Figure 6: Testing result of linear regression model.

7.3 Ablation study

To test the effectiveness of the model and the impact of supplementary stock data on its accuracy, an ablation study was conducted. This was done by removing all AAPL and MSFT stock data from the feature set, and running the same model. The resulting prediction can be seen below in figure 7:

When visually comparing the results of the ablation study to the original model, there appear to be some instances where the ablation model deviates more from the actual price. To quantify the impact of removing AAPL and MSFT stock data, numerical metrics such as mean absolute error (MAE) and R-squared (R^2) are calculated using the associated functions provided by scikit-learn.

	MAE	R^2
Control	23.83	0.90
Ablation	25.00	0.89

Table 1: Evaluation scores of ablation study.

There is a measurable difference in the mean absolute error where the ablation model scored slightly higher, suggesting that its predictions are slightly further off than the control model. The lower r-squared value also suggests that there is less linearity between the predicted values of the ablation model and the actual values. While the differences are very small, they do suggest that adding supplementary stocks AAPL and MSFT to the feature set indeed yield subtle improvements to model accuracy.

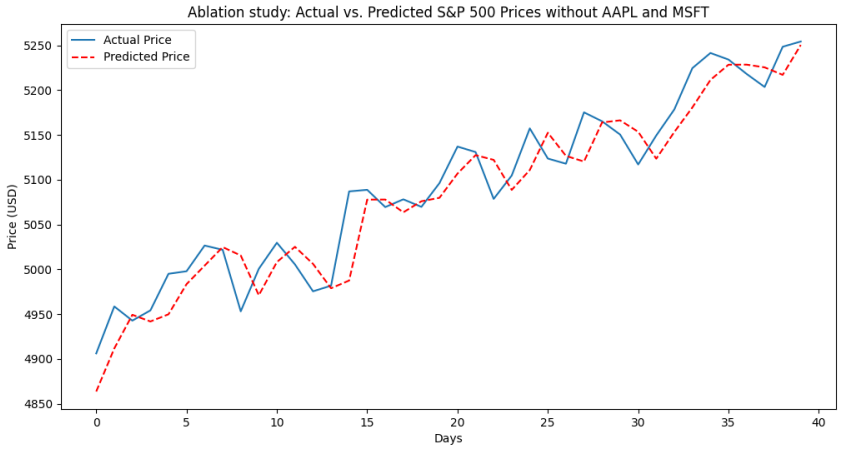

Figure 7: Ablation study results.

8 Conclusion

This project aimed to create a data pipeline that could predict the closing price of the S&P 500 index using historical stock data. Stock data, acquired from Yahoo! Finance, was stored in a No-SQL database in the form of JSON documents. This data can then be pulled, visualised, and its features extracted. A Linear Regression model was trained using features such as lagged prices, moving averages, and external stock data (AAPL and MSFT). The results showed that lagged values and moving averages were effective in predicting future trends. An ablation study was conducted, showing that removing AAPL and MSFT stock prices had a minor but measurable impact on the model 's accuracy, suggesting that S&P 500 movements can largely be predicted using its own historical data but still benefit from external data.

A major drawback to the feasibility of this project is the inherent unpredictability of stock prices, which are influenced by a myriad of factors that are outside the scope of this dataset. Additionally, the Linear Regression model assumes a linear relationship between features and the target variable, which may not always be the case when predicting more volatile stocks. For the purposes of this project, it can be said that the "Money Money Money!" company would be smart to buy S%P 500 stocks, judging by the continued upwards trend that was predicted by the model.

Based on the model's results, the S&P 500 index shows a continued upward trend, suggesting that the "Money Money Money!" would be successful if they were to long (buy) shares.

References

Apache. Apache license, version 2.0, 2004. URL https://www.apache.org/licenses/LICENSE-2.0.

Jason Brownlee. Moving average smoothing for data preparation and time series forecasting in python, 2020. URL https://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/.

Will Kenton. S&p 500 index: What it's for and why it's important in investing, 2024. URL https://www.investopedia.com/terms/s/sp500.asp.

Jess Menton. Is the stock market's 'january effect' real?, 2025. URL https://www.bloomberg.com/news/articles/2025-01-05/is-the-stock-market-s-january-effect-real.

ranaroussi. Download market data from yahoo! finance's api, 2017. URL https://github.com/ranaroussi/yfinance.

J. Margaret Sangeetha and K. Joy Alfia. Financial stock market forecast using evaluated linear regression based machine learning technique. *Measurement: Sensors*, 31:100950, 2024. ISSN 2665-9174. doi: https://doi.org/10.1016/j.measen.2023.100950. URL https://www.sciencedirect.com/science/article/pii/S2665917423002866.

Troy Segal and Gordon Scott. "sell in may and go away": Definition, statistics, and analysis, 2024. URL https://www.investopedia.com/terms/s/sell-in-may-and-go-away.asp#toc-the-bottom-line.

Stephen M. Walker. What is linear regression?, 2024. URL https://klu.ai/glossary/linear-regression.

A Appendix

Code for IQR calculation and outlier detection:

```
# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = msft_data.quantile(0.25)
Q3 = msft_data.quantile(0.75)
IQR = Q3 - Q1  # Interquartile range

# Identify outliers
outliers = (msft_data < (Q1 - 1.5 * IQR)) | (msft_data > (Q3 + 1.5 * IQR))

print("Outliers:")
print(outliers.sum())
```

In case of an error, the link to the Github repository containing this project is: https://github.com/UCL-ELEC0136/final-assignment-zceesur