Stock Forecasting Using Yahoo Finance

Udacity Capstone Report
Machine Learning Engineering with AWS

By Abhijith Anil Vamadev

1. Introduction

The stock market has always been a center of attention for investors, analysts, and researchers. Predicting the movement of stocks, while lucrative, is also quite challenging given the myriad of factors influencing the prices. Historically, tools like fundamental analysis, technical analysis, and more recently, artificial intelligence and machine learning have been employed to provide a more accurate forecast of stock prices. This project is an endeavor in the same direction, focusing on leveraging historical stock data from leading tech giants to anticipate future stock prices.

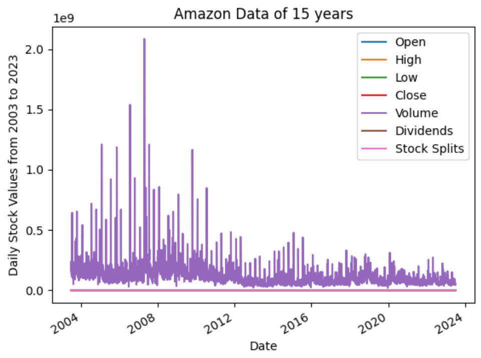
2. Problem Statement

The primary aim of this research is to craft a predictive model that can precisely forecast 5% of the future price movement of a given stock, whether it's the next day, within the subsequent 7 days, 14 days, or up to 30 days. The linchpin predictor for this task will be the Adjusted Close value. It's noteworthy that the testing data is relevant up to the date of 3/23/2023.

3. Data Sources

The foundation of this project is the historical stock data derived from Yahoo Finance, encapsulating the following attributes:

- Open Price: The inception trading price for a given period.
- Close Price: The concluding trading price for that period.
- High Price: Peak price attained during a period.
- Low Price: The nadir price of a period.
- Adjusted Close Price: This is the close price, fine-tuned for elements like dividends and stock splits.
- Date: The exact date of the stock data.



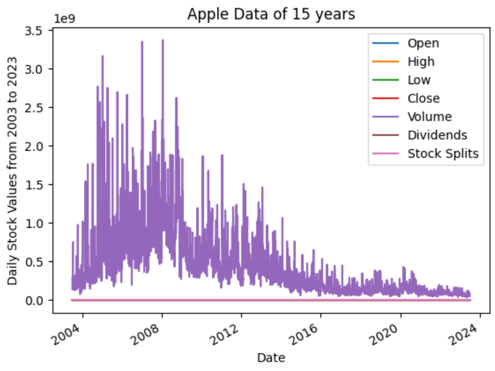
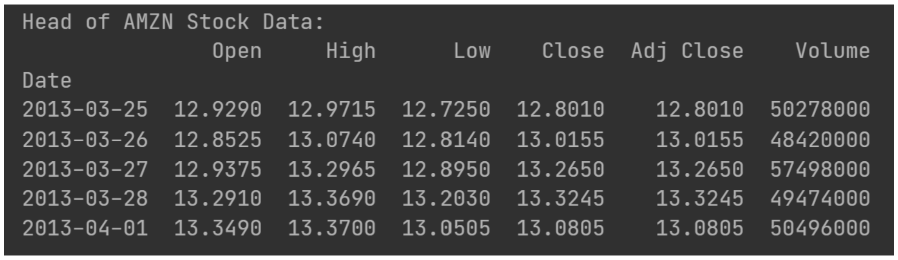


Figure 1: Shows Amazon and Apple values from 2003 to 2023.

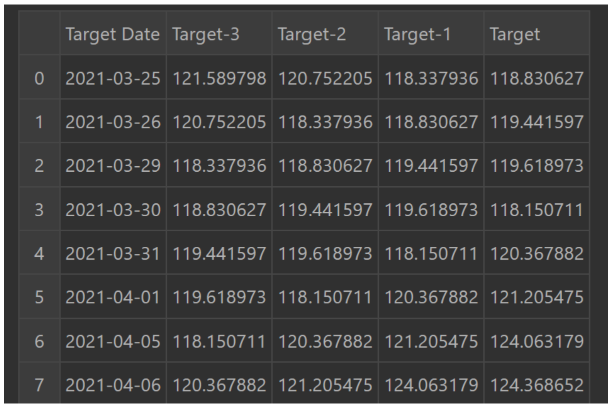
The initial stocks under the lens are:
- Apple (AAPL)
- Amazon (AMZN)
- Microsoft (MSFT)
- Google (GOOGL)

These stocks have been compartmentalized into three datasets:
- Raw Data - found in the raw folder from github
- Processed Data - found in the processed folder from github
- Model Predictions - found in the results folder from github



```
Head of AMZN Stock Data:
               Open     High      Low    Close  Adj Close    Volume
Date
2013-03-25  12.9290  12.9715  12.7250  12.8010    12.8010  50278000
2013-03-26  12.8525  13.0740  12.8140  13.0155    13.0155  48420000
2013-03-27  12.9375  13.2965  12.8950  13.2650    13.2650  57498000
2013-03-28  13.2910  13.3690  13.2030  13.3245    13.3245  49474000
2013-04-01  13.3490  13.3700  13.0505  13.0805    13.0805  50496000
```

Figure 2: Shows AMZN raw data points that were collected.



	Target Date	Target-3	Target-2	Target-1	Target
0	2021-03-25	121.589798	120.752205	118.337936	118.830627
1	2021-03-26	120.752205	118.337936	118.830627	119.441597
2	2021-03-29	118.337936	118.830627	119.441597	119.618973
3	2021-03-30	118.830627	119.441597	119.618973	118.150711
4	2021-03-31	119.441597	119.618973	118.150711	120.367882
5	2021-04-01	119.618973	118.150711	120.367882	121.205475
6	2021-04-05	118.150711	120.367882	121.205475	124.063179
7	2021-04-06	120.367882	121.205475	124.063179	124.368652

Figure 3: Shows Apple processed Adj Close data, the previous 3 days were used to predict the next day's Adj Close Value.

4. Approach and Methodology - Data Preprocessing and EDA

During the Exploratory Data Analysis (EDA) phase, it became evident that a univariate model would be more efficacious given the high correlation among the variables like Open, Close, High, Low, and Adjusted Prices. During the EDA process all the data was looking more heavily right skewed, and this was seen for all the Adj Close values for the stocks, but the High column was left skewed for all stocks respectively. As a result, the Adjusted Close was used out for analysis. The model leverages a time window (or time steps) of 3 days, implying the preceding 3 days of Adjusted Close data would be harnessed to predict the subsequent day's data. The data was stratified into an 80-10-10 split for training, validation, and testing respectively.

The preprocessing of stock data in the given code can be summarized as follows:

1. Data Downloading and Organization: The code starts by defining a function download_data that fetches historical stock data from Yahoo Finance for specified symbols (like AAPL, AMZN) within a date range. The downloaded data is saved in a local folder named data/raw.
2. Exploratory Data Analysis (EDA): The visualize_stock_data function is defined to visualize stock data. It checks for the presence of necessary columns like 'High', 'Low', 'Close', and 'Volume'. It produces multiple plots such as line graphs of high/close prices, histogram of close prices, candlestick chart (if data allows), and moving averages. Additionally, the head of the dataset, variable information, summary statistics, and correlation matrix are printed.
3. Null Value Handling: The handle_null_values function processes the dataset to manage any missing values. Columns with more than 60% missing values are removed, while those with less are filled with their mean values.
4. Windowed Data Generation: The df_to_windowed_df function is designed to transform the dataset into a windowed format, where each row represents a sequence of data leading up to a target date, facilitating time-series prediction tasks. This is used for determining the input shape of the LSTM model, for the model in this project the input shape will be 3,1 - time-steps, features.
5. Data Splitting: Data is split into training, validation, and test sets. The training set consists of the first 80% of data points, validation is the next 10%, and the test set comprises the final 10%.

6. Data Saving: Processed data is organized into structured folders. The save_data_to_folder function ensures that each stock's processed data (train, validation, test) is saved into its respective directory.
7. Visualization of Data Split: For each stock, the code visualizes the different data splits. This provides a visual confirmation of the data segmentation and helps understand the distribution of data points across the timeline.
8. In essence, this preprocessing pipeline ensures that raw stock data is fetched, cleaned, transformed, and organized efficiently for subsequent modeling tasks.

5. Model Architecture

The model employed is the LSTM (Long Short-Term Memory) model, a variant of the recurrent neural network (RNN). LSTMs excel at discerning long-term dependencies in sequential data, making them an ideal fit for the intricacies of stock forecasting. The training phase ingested data spanning from 2018 to 2022, validation phase from 2022 to half of 2023 and the rest for the test set, the last date for the test date is 08/23/2023.

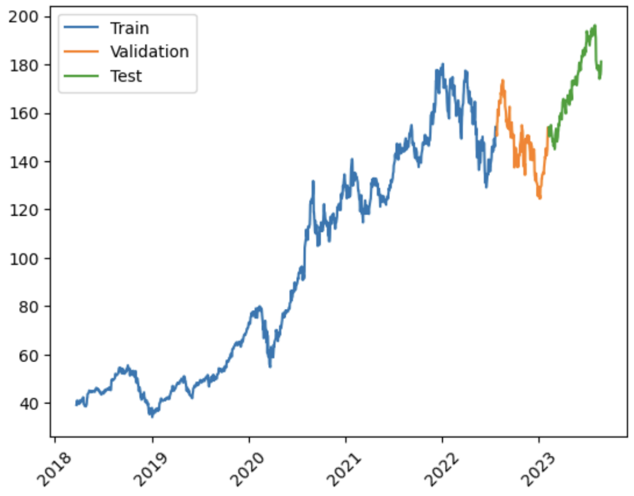


Figure 4: Shows visualization for Apple train-validation-test split values.

6. Benchmark Model

For a comparative assessment, the benchmark model has the following configuration:
- Simple LSTM Model
- Input layer configuration of 3,1 (time steps, features)

- 64 LSTM units
- Adam Optimizer with a learning rate of 0.001
- Training for 50 Epochs
- Metrics: Mean Absolute Error, Validation Loss
- During the benchmark model, the model performed fairly well having a Validation loss of 30~25. This improved as the number of epochs increased.

5. Evaluation Metrics

To gauge the efficacy of the predictive model, two primary metrics were deployed:

- **Mean Absolute Error (MAE)**: Given its resilience against outliers, MAE is instrumental, especially when stock prices can exhibit sudden and unpredictable fluctuations.
- **Validation Loss:** This metric offers insights into how adeptly the model is generalizing its learnings to new, unseen data. Tracking validation loss can preempt overfitting.

Model	Metric	Benchmark	AAPL	AMZN	MSFT	GOOGL
Loss	Training Loss	4.2685	49.9805	161.6090	14.3575	2.9566
	Validation Loss	8.0298	201.2620	487.2097	33.1002	6.2774
MAE	Training MAE	1.4890	5.5669	9.9878	2.5964	1.1957

	Validation MAE	2.2233	12.7025	18.4932	4.4802	1.9085

Table 1: Showing the final evaluation metrics for the baselinemodel and the final models

AAPL: The final model for AAPL has significantly higher training and validation losses compared to the benchmark. The mean absolute error (MAE) for both training and validation is also considerably higher. This suggests that the model for AAPL might be overfitting or may not be well-optimized for the given dataset.

AMZN: The AMZN model has the highest difference from the benchmark, with extremely high training and validation losses. The MAE values for both training and validation are much higher than the benchmark, indicating poor performance and potential overfitting.

MSFT: The MSFT model also shows higher training and validation losses compared to the benchmark. The MAE for both training and validation is higher than the benchmark but not as drastic as AAPL and AMZN.

GOOGL: The GOOGL model performs closest to the benchmark. The training loss is lower than the benchmark, but the validation loss is slightly lower. The MAE for training is lower than the benchmark, but the validation MAE is slightly lower.

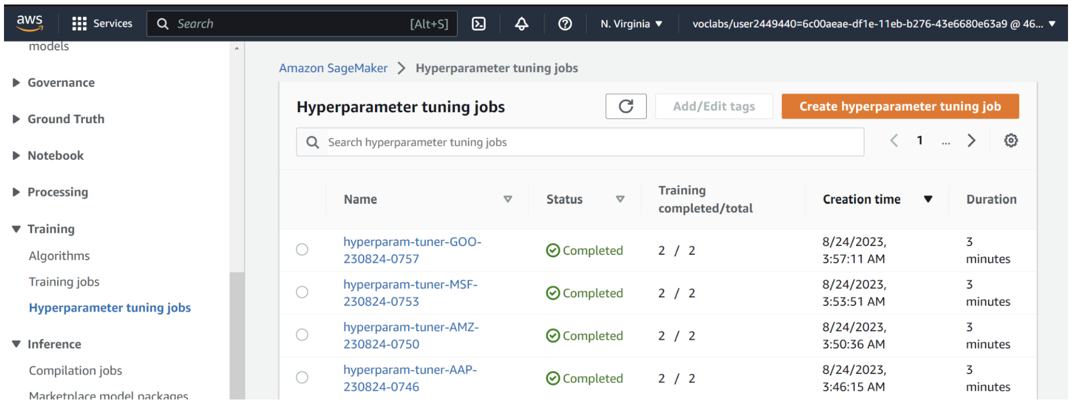
The benchmark model seems to provide a more generalized solution with lower loss and MAE values across training and validation sets.

The final models for AAPL and AMZN particularly seem to be underperforming significantly when compared to the benchmark. This might be due to several reasons such as overfitting, lack of sufficient model tuning, or non-optimal architecture choices.The MSFT model, though not as efficient as the benchmark, has a relatively better performance compared to AAPL and AMZN. The GOOGL model is the closest in performance to the benchmark, suggesting that it might be the most optimized among the final models.

6. Hyperparameter Tuning

Hyperparameters play a pivotal role in refining the model's performance. The key hyperparameters tuned include:

- **Dense Units:** Pertains to the number of neurons in the Dense layer.
- **Learning Rate**:Dictates the step size during iterations while optimizing the loss function.
- **LSTM Units**: Determines the memory capacity of the LSTM layer.


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2449440=6c00aeae-df1e-11eb-b276-43e6680e63a9 @ 46...
models
Governance
Ground Truth
Notebook
Processing
Training
Algorithms
Training jobs
Hyperparameter tuning jobs
Inference
Compilation jobs
Marketplace model packages
Amazon SageMaker > Hyperparameter tuning jobs
Hyperparameter tuning jobs
Add/Edit tags
Create hyperparameter tuning job
Search hyperparameter tuning jobs

Name	Status	Training completed/total	Creation time	Duration
hyperparam-tuner-GOO-230824-0757	Completed	2 / 2	8/24/2023, 3:57:11 AM	3 minutes
hyperparam-tuner-MSF-230824-0753	Completed	2 / 2	8/24/2023, 3:53:51 AM	3 minutes
hyperparam-tuner-AMZ-230824-0750	Completed	2 / 2	8/24/2023, 3:50:36 AM	3 minutes
hyperparam-tuner-AAP-230824-0746	Completed	2 / 2	8/24/2023, 3:46:15 AM	3 minutes

Figure 5: Shows Hyperparameter tuning jobs for each of the stocks.

The finalized hyperparameters for all models are documented in the results folder under *hpo*. For hyperparameter tuning, multi-instance training was utilized as many stocks have to be predicted, and for each stock, two training jobs are utilized for each of the hyperparameter jobs.

7. Project Execution

The project was executed in the following sequential phases:

1. Data Collection: Procured a decade's worth of historical stock data for the chosen symbols, through the use of finance and python. The starting date for data collection was set at *2013 - 03 - 25* and the end date was set to *2023 - 08 - 24*.
2. Data Preprocessing:The data was cleansed, formatted, and subjected to EDA. During EDA, the data was visualized, data was cleansed, the units for each column were looked at, along with duplicates in the data. Since, Adj Close prices was only used for analysis, standardization or normalization of data hasn't been conducted.

3. Model Development and Evaluation: Deep learning models were trained and their performance benchmarked. A basic LSTM model was initially utilized and evaluated on MAE and Val loss.
4. Model Fine-tuning: Hyperparameters were tweaked based on initial results.
    a. For AAPL the hyperparameters were:
        i. Dense Units: 23
        ii. Learning Rate: 0.047
        iii. LSTM units: 78
    b. For AMZN the hyperparameters were:
        i. Dense Units: 56
        ii. Learning Rate: 0.040
        iii. LSTM units: 103
    c. For AAPL the hyperparameters were:
        i. Dense Units: 19
        ii. Learning Rate: 0.0003
        iii. LSTM units: 39
    d. For AAPL the hyperparameters were:
        i. Dense Units: 18
        ii. Learning Rate: 0.0001
        iii. LSTM units: 34

5. Prediction and Analysis: The model was used to forecast future stock prices and juxtaposed with prevailing trends.The data was fit in a new model using LSTM hyperparameters and then the test predictions were saved in a csv to utilize later.
6. Interface and Visualization: The outcomes were visualized using matplotlib and streamlit, and a user-centric interface was developed using Streamlite and subsequently deployed on Streamlit's cloud environment. The project has been deployed with an UI in https://capstoneudacity.streamlit.app/ with more time the UI can be developed but this is a good start for the UI, there are some limitations with the UI so using something like React for the frontend and Flask for the back end may be more beneficial for a more complex project.

8. Augmenting Project Complexity

To further amplify the project's complexity and depth, the following avenues can be explored:
- Sentiment Analysis:Integrate news sentiment analysis to fathom the ripple effect of news on stock prices. The stock market is not solely governed by past prices; it's

influenced by a multitude of external factors, including current events. Harnessing news APIs could potentially bolster prediction accuracy.
- Macroeconomic Indicators:Enrich predictions by factoring in macroeconomic indices like GDP growth, interest rates, and unemployment rates.
- Real-time Prediction:Introduce a feature to predict stock prices in real-time using live data streams. Tools like Sagemaker Multi-model Endpoint can be harnessed for real-time predictions for an array of stocks, and the primary application could be hosted on an EC2 instance.

9. Conclusion

Predicting stock market movements is a complex and multifaceted challenge. This project, by leveraging advanced machine learning techniques and comprehensive data sources, aims to make a tangible impact in this domain. The fusion of data-driven insights and technological prowess holds the potential to revolutionize stock market predictions. Initially I set out to find stock information for about 30 stocks, but after research and planning realized that due to the budget constraint using a maximum up to 4 or 5 stocks would be sufficient for this project. Also, only one variable was used to conduct the analysis - Adj Close. In future, for analysis other variables like macro economic factors, or sentiment analysis can be done to further improve the models. Using only Adj Close is not a reliable source for Stock prediction, as it is nearly impossible to predict stocks with limited information. Although this is a good starting point for stock prediction, I would not personally use this technique to gauge stock price in real time, as seen with couple of stocks like AAPL and GOOGL the predictions were mostly off but MSFT and AMZN found better predictions.

Given more time, the complexity of the project could be to have real-time deployment to an Ec2 server and have aws endpoint make real-time predictions, and also include other variables for analysis like News sentiment analysis and macro-economic factors to improve analysis.

Sources:
1. https://neptune.ai/blog/predicting-stock-prices-using-machine-learning
2. https://colab.research.google.com/drive/1Bk4zPQwAfzoSHZokKUefKL1s6lqmam6S?usp=sharing#scrollTo=PGiLbQIYOawE
3. https://capstoneudacity.streamlit.app/