Predict the S&P 500 index price

Author: Sathish Manthani

Date: 5/25/2020

Contents

Introduction

Predicting stock prices is a popular usecase for machine learning and predictive analytics.

S&P 500 (Standard and Poor 500) is the stock market index that measures the stock performance of 500 large companies in the US. I want to predict the stock prices for it based on the available variables and historical data. I know the historical prices are no clear indicators of whether a price will go up or down in future. With that risk, I will use predictive analytics and various machine learning algorithms like Decision trees, Random forests, Gradient boosting, Neural networks, and Facebook's prophet to predict the stock price.

We also know that, apart from the historical price changes of a stock - a lot of other factors influence the stock prices. Like, the current overall market conditions (Bull market/Bear market), the conditions of the industry (e.g. oil industry), leadership changes within the company or any new acquisitions etc., So, in order to do predict the stock price, we also have to analyze the sentiment of the market by scraping the published articles, news and social media data. However, I will just stick to technical and fundamental analysis of the stocks to predict its price in this project.

Scope

In this project, I will be evaluating different machine learning methods for the best prediction outcome based on the historical data of the S&P 500. What I will not be considering is the sentiment of the market. Sentimental analysis usually involves analyzing the current overall market trends, published articles, political news and greed/fear factor of the people. So, I will

limit the scope of the project to historical data analysis of the stocks to predict the S&P performance.

Data Sources

I looked at multiple datasets on Kaggle and other data sites for stock market data. The amount of data available on the web related to stock market is overwhelming. I chose Yahoo Finance API to fetch historical stock market data. The good thing about this API is it has data since 1990s and I would use that data for better modeling.

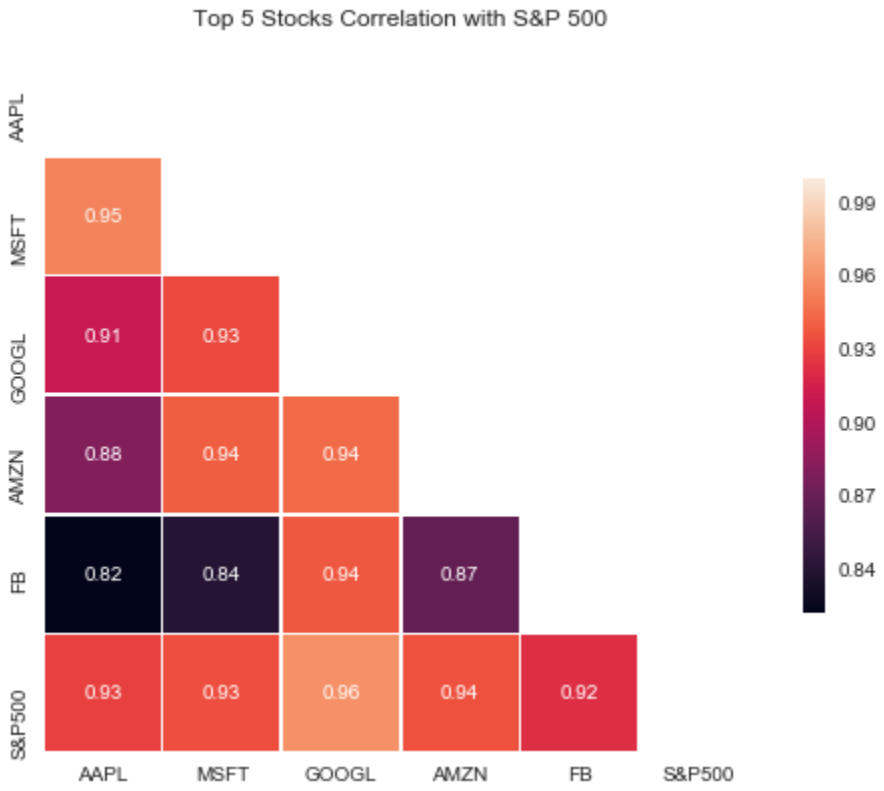
I used web.DataReader method to fetch data directly from **Yahoo Finance API** into dataframe.

Features available in the dataset:

Variable name	**Description**
Date	Date of the stock trade
High	Highest price of the stock on the day
Low	Lowest price of the stock on the day
Open	Opening price of the stock on the day
Close	Closing price of the stock on the day
Volume	Volume (number of shares exchanged) on the day
Adj Close	Adjusted closing price of the stock on the day. Adjusted price accounts into dividends and other price adjustments happened on the day. So that's why this variable is used as the final price of the stock.

Big 5 Stocks and S&P 500

I checked how Big 5 stocks (Amazon, Microsoft, Facebook, Google, and Apple) are influencing S&P 500 index and looked at the correlation among these stocks



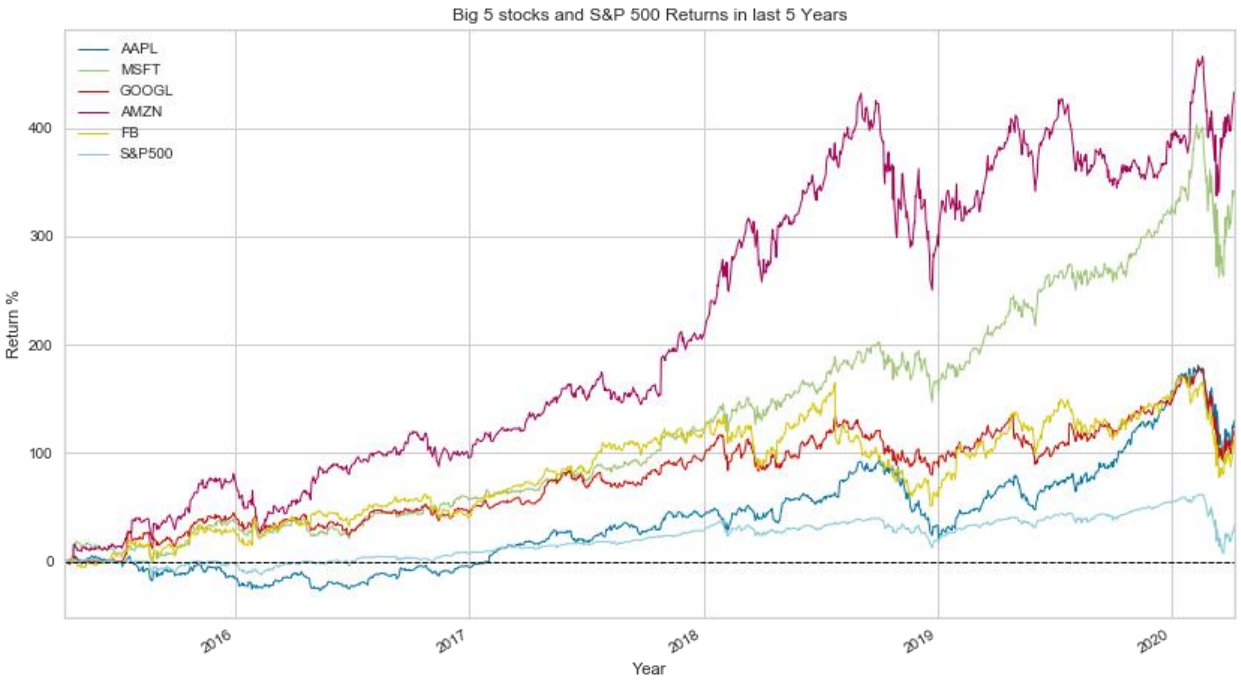
We can see that S&P 500 index's correlation is very strong with other stocks.

That means each of these big 5 stocks are moving in sync with S&P 500.

Returns - S&P 500 and Big 5

Rate of return tells what % is gained or lost over a period of time. It is calculated with the below formula:

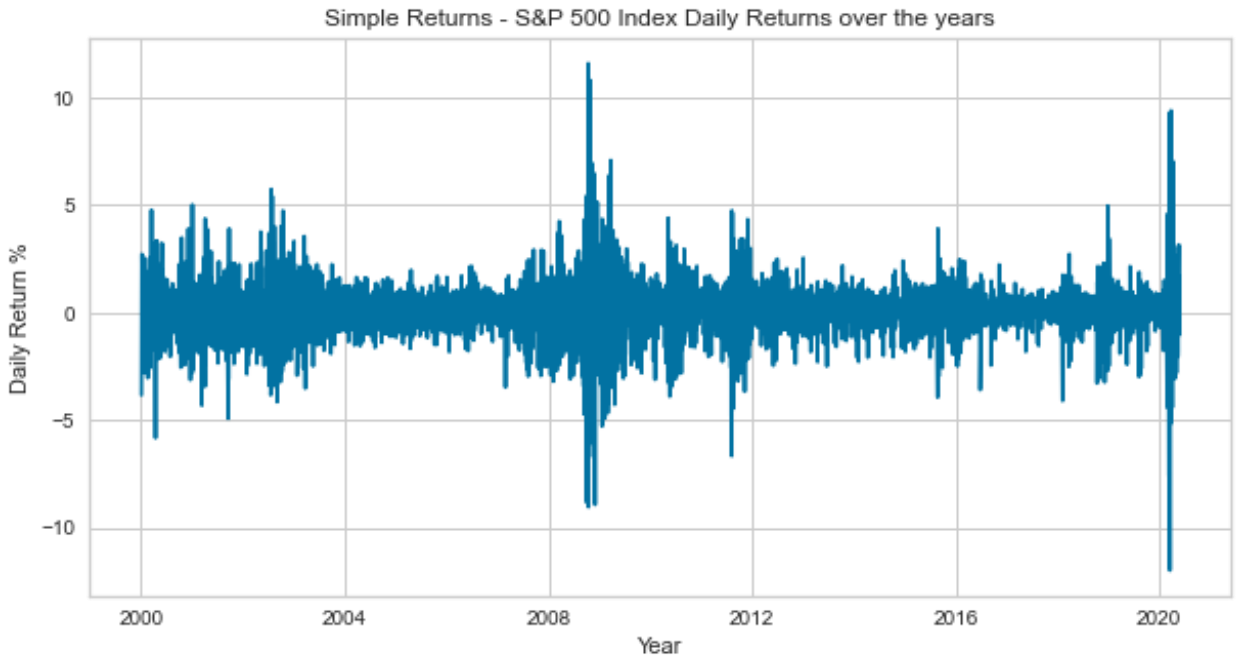
*Rate of Return % = (Curent Price - Starting Price)/ Starting Price * 100*



Amazon had a great run in last 5 years followed by Microsoft. Another interesting observation is - MSFT and AMZN were far apart in terms of returns in 2018-19 but MSFT performed amazingly well later. S&P 500 had lower returns than other individual stocks. That is expected because I took the big 5 stocks for comparision which are more likely be the top performers.

Daily Simple Returns of S&P 500

There is an alternative way of showing returns. Let's also use Log transformation to show the returns.



From the plot above, we can interpret that - except on few occasions (-4%,4%) seems to be the common range of daily returns.

We also know S&P 500 lost large % during 2008-09 financial crisis and during 2020 March.

Large -ve single day Percentage Change occured in 2020 March which is -11.98%.

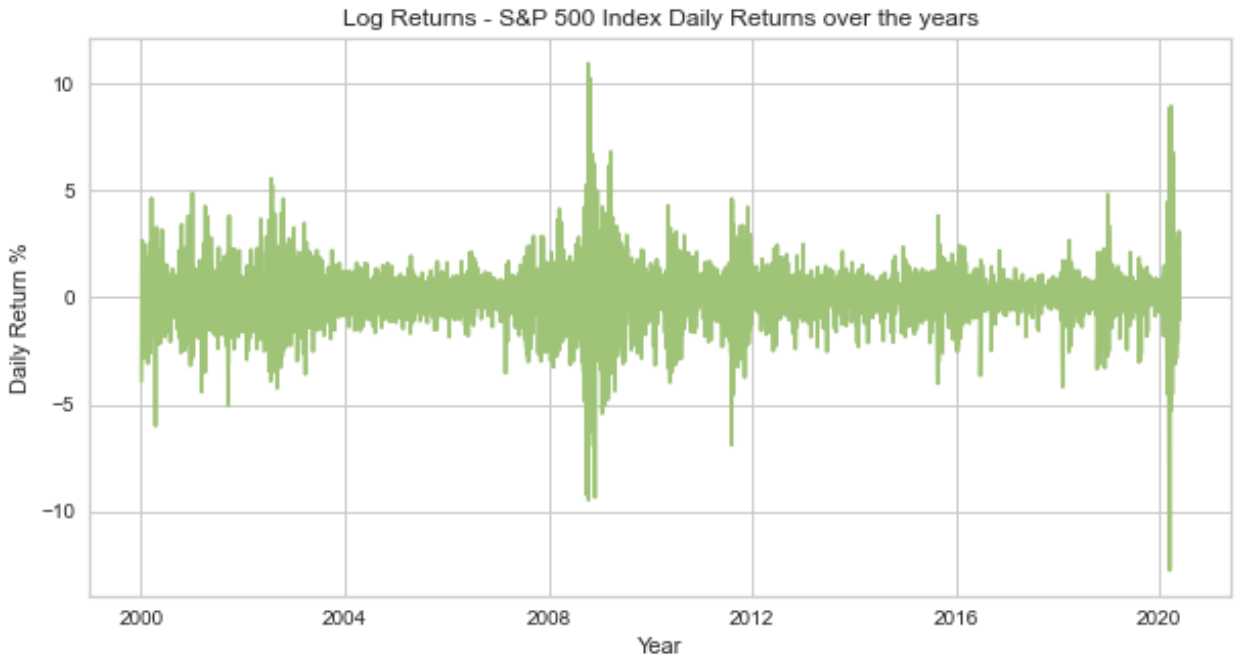
Average daily % change over the last 20 years is 0.033%.

Daily Log Returns of S&P 500

Log returns are usually preferred when you are dealing with a large time frame. Let's also calculate it and see.

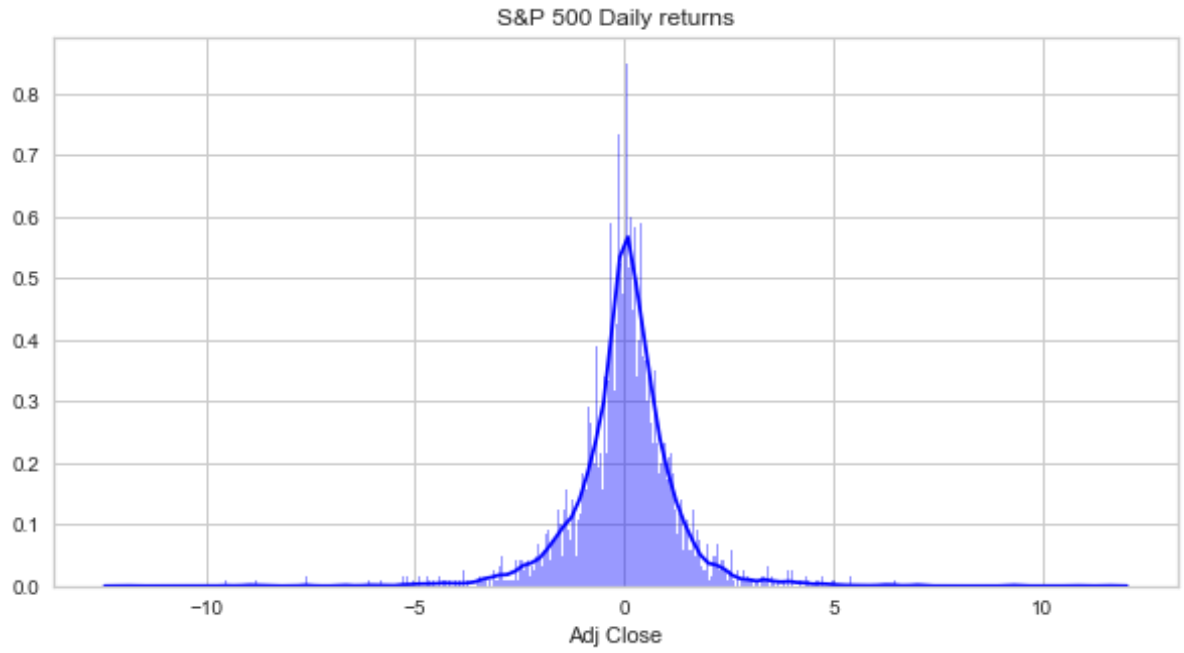
Formula:

Log returns = log(current_price / previous_price)



From the plot, Daily Log returns look very similar to Daily Simple Returns but if you look at mean, max values. Log values are smaller than Simple returns, that is expected with log function.

Distribution of S&P 500 Daily returns



The distribution of daily returns looks normal distribution with Mean of 0.033 and standard deviation is 1.25.

Volatility

Volatility is basically the dispersion of the financial asset returns over time

Volatility is often computed as the standard deviation of the stock price returns. Standard deviation basically tells us the spread of the data around the mean.

Its important to remember that the higher the volatility,the riskier the asset is. So, let's calculate daily, monthly and annual volatility for S&P 500.

Daily volatility: 1.26%

Monthly volatility: 5.75%

Annual volatility: 19.93%

Annualized S&P 500 volatility is very high. So what it means is the actual S&P 500 returns can fluctuate over 19% around the mean returns.

Defining new Features and Target for S&P 500

As part of exploratory data analysis, we have seen how big 5 are influencing S&P 500. To predict the price of S&P 500 index, I defined new derived features and targets for S&P 500:

- Future price changes - I chose **14-day future price** and **14-day future price percentage** as features
- Simple moving averages - I chose **14-day** and **200-day moving average** as features
- RSI(relative strength index) - **Relative strength index** measures the magnitude of recent price change:

$$\text{RSI} = 100 - \frac{100}{1 + \text{Relative Strength (RS)}}$$

$$\text{RS} = \frac{\text{An average of the 'up closes' in a given period of time}}{\text{An average of the 'down closes' in a given period of time}}$$

- Exponential Moving Average (EMA) - **Exponential moving average(EMA)** places more weight on recent price and is defined as below:

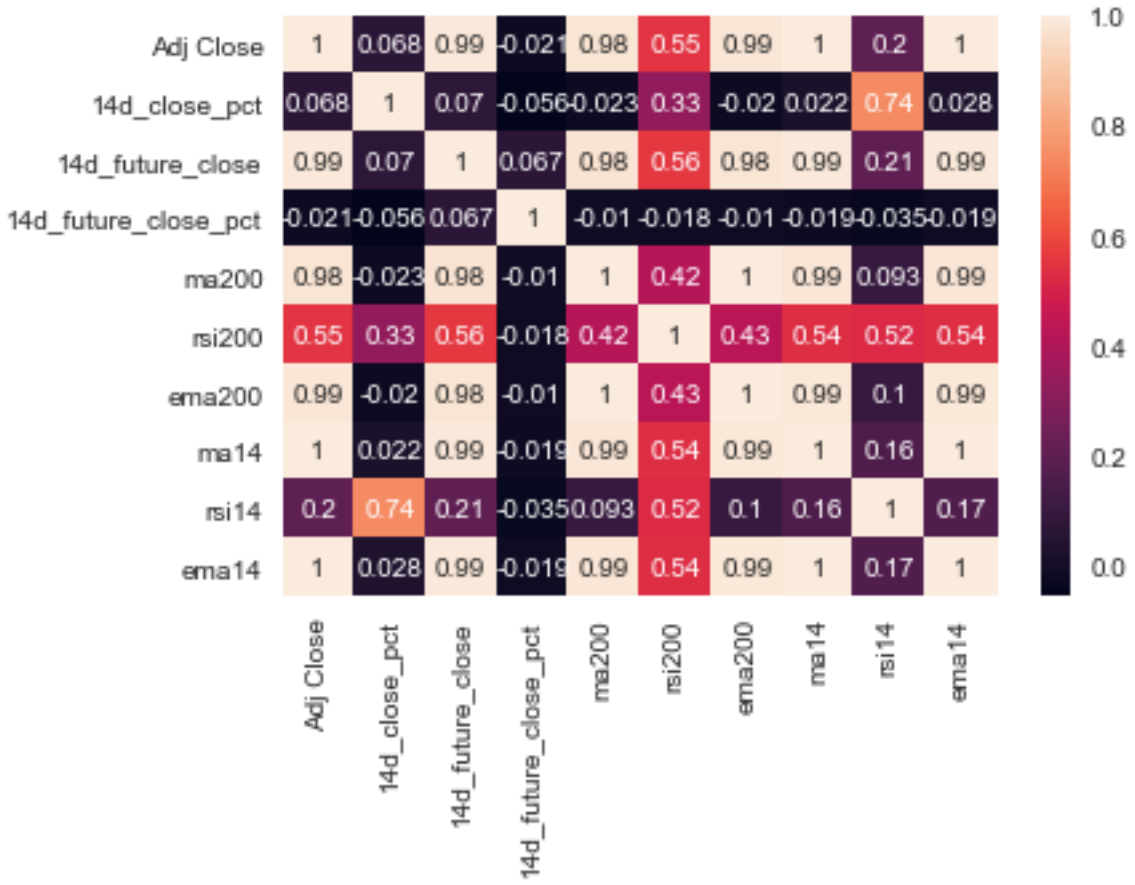
$$EMA_{\text{Today}} = \left(\text{Value}_{\text{Today}} * \left(\frac{\text{Smoothing}}{1 + \text{Days}}\right)\right) + EMA_{\text{Yesterday}} * \left(1 - \left(\frac{\text{Smoothing}}{1 + \text{Days}}\right)\right)$$

Smoothing factor =2 is often used

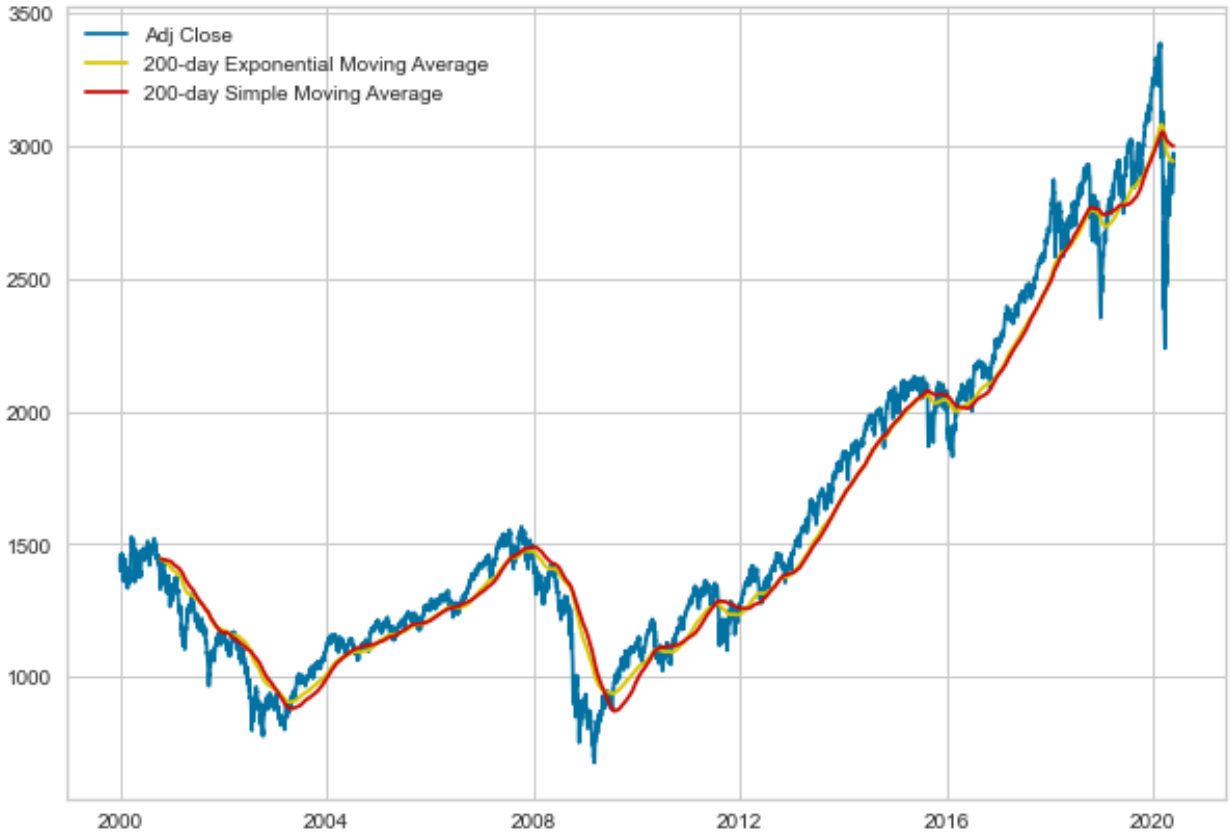
I took the target variable as 14-day closing percent change. The basic idea is to predict the closing price change of the index after 14-days for current day.

Correlation

Below is the correlation matrix among these new features and Adjusted close.



14-day future closing price, 14-day and 200-day moving average and 14-day and 200-day EMA are highly correlated with Adjusted Closing price of the stock.



Simple moving average and Exponential moving averages smoothes out the stock price curve and they are moving in sync with Adjusted close.

Test/Train data split

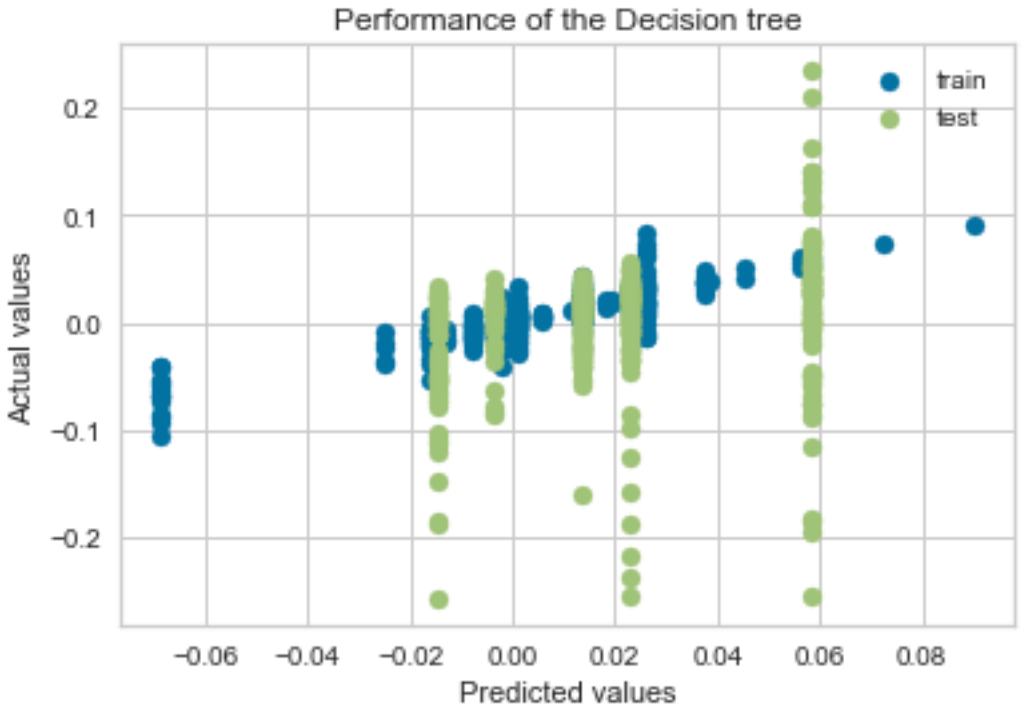
Time series data can't be sliced randomly. So, I sliced first 85% data to train the model and remaining 15% (latest data) to test the models

Building Models and Evaluation

1 - Decision Tree

I started my modeling with Decision tree. When I ran Decision tree modeling on the training dataset, I got accuracy of 0.78 which is excellent but on the test dataset the accuracy is 0.013

That means the tree did not perform well for the test dataset. Decision tree probably overfit for the training dataset.



The graph looks good for training dataset, but the predicted points are way off for the test dataset.

2 - Random forests

Decision tree didn't do well do well with test data. So, I tried modeling with random forests. The basic idea is that random forests supposed to reduce the variance of decision trees.

With random forests, I achieved accuracy of 0.023 on the test dataset and it is better accuracy than decsion trees.

3 - Gradient boosting

Boosted models are general class of machine learning algorithms. These work by iteratively fitting models such as decision trees to the data. They work by taking residual error of the first model to the next model and so on.

Gradient boosting gave us slightly better performance (0.029) than previous models but its still not enough to predict the stock prices accurately.

4 - Neural networks

I tried neural networks next. Neural network requires lot of customization to achieve better performance. With the parameters I used, neural network performed poorly with the dataset. I got -ve accuracy. That is because I didn't tune the model well, there's a lot of scope for improvment. Like creating custom loss functions and using different activation functions etc., I could have tried tuning it but due to lack of time, I'm going to try one more forecasting model.

5 - Facebook's prophet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best

with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

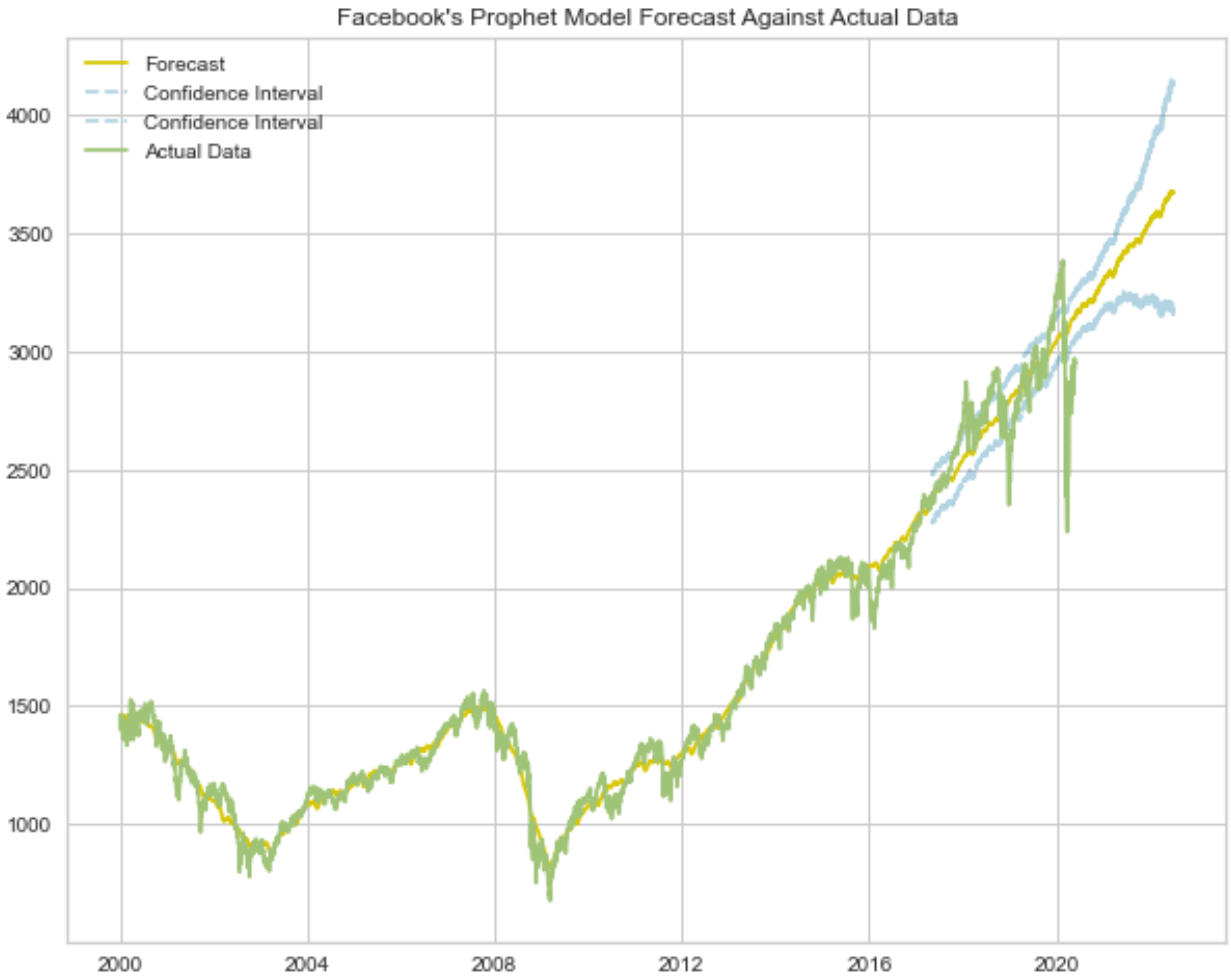
When I ran Prophet model with target variable as “Adjusted close”, R-square value came out to be 0.98 for the fitted model which means the model is a great fit. To find out if its an overfit using Mean squared error and Mean absolute error.

Mean Squared Error: 6244.43

Mean square error is decent so the model is a good fit.

Mean Absolute Error: 51.49

It means the predicted value can be 51 basis points away from the actual value either side of the curve at the maximum.



From the above plot and R-square, mean square error and mean absolute error. I think Prophet predicts the stock prices with decent accuracy.

Summary/Conclusion

The goal of this project is to predict the S&P index price using machine learning models.

I started with data extraction of S&P 500 using Yahoo finance API. I extracted the data for the last 20 years.

I have also extracted Big 5 stocks(Apple, Amazon, Microsoft, Facebook and Google) data for last 5 years to see how they compare to S&P 500 index. Upon analysis, I saw a strong correlation

(more than 90% correlation) between the big 5 stocks to S&P 500. This was surprising, I know the correlation would be there but did not expect it to be this strong. So, that means all the big 5 stocks are moving in sync with the S&P 500 index.

The daily returns of S&P 500 index is generally in the range of (-4% to +4%) except on few occasions. We also know S&P 500 lost large % during 2008-09 financial crisis and during 2020 March. Large single day “loss” occurred in 2020 March which is -11.98%, we also know this is due to covid-19 pandemic. Another observation is that the average daily % change over the last 20 years is 0.033%. That means, S&P 500 index tend to have positive returns in a long run.

Before building models, I added few technical indicators as features to S&P 500 index.

These features are Future price change, Simple moving averages, Relative Strength Index, and Exponential moving average.

Simple moving average is basically calculating the rolling window average over a given time interval. Relative Strength index measures the magnitude of price change. Exponential moving average is a weighted moving average which places more weight on the recent price change.

I used 5 different models to predict the stock prices i.e. Decision trees, Random forests, Gradient boosting, Neural networks, and Facebook's prophet.

Decision trees fit the training data well but did not perform well with test data. Then I tried Random forests, it gave a slightly better performance but not great. Gradient boosting model further improved the accuracy on the test data but its not enough to predict the S&P 500 index price.

Later I tried Neural networks. Neural networks generally work great if you tune the parameters, improve the loss function and do the back propagation. I tried some basic modeling in neural networks, so it did not perform well. Finally, I tried Facebook's Prophet for forecasting the S&P index price which showed good results (please look at the last plot). From the plot, the S&P 500 index is going to go up in next 2-3 years.

Current S&P 500 index price is **2955.45** and according to the prophet forecasting model, its going to be at **3667.43** in 2 years from now (05/31/2022) with the confidence interval's price range is **(3183.69, 4121.55)**.

References

- [1] "Python for Finance: Investment Fundamentals and Data Analytics" - By 365 Careers in https://learning.oreilly.com/home/
- [2] Datacamp - https://campus.datacamp.com/courses/importing-and-managing-financial-data-in-python/
- [3] Look up https://www.investopedia.com/ for definition of technical indicators
- [4] Facebook's prophet https://pythondata.com/forecasting-time-series-data-with-prophet-part-1/
- [5] https://towardsdatascience.com/predicting-stock-prices-with-python-ec1d0c9bece1
- [6] https://medium.com/@randerson112358/stock-price-prediction-using-python-machine-learning-e82a039ac2bb