Data 698 - Final report

Tze Fung Lung, Jim

April 17, 2019

Topic: Portfolio optimization and Machine learning with visualization analysis for S&P 500

10 Keyword:

- S&P500
- Stocks
- Return
- Risk
- Strategic
- Linear
- k-Nearest
- Moving-Average
- ARIMA
- LSTM

Table of contents

1. Introduction

1.1 Objectives

Predicting how the stock market will perform is one of the most difficult things to do. There are so many factors involved in the prediction – physical factors vs. physiological, rational and irrational behavior, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.

The S&P 500 is widely regarded as the best single gauge of large-cap U.S. equities. The index includes 500 leading companies and captures approximately 80% coverage of available market capitalization.

1.2 Description of the Problem

We'll look at the S&P 500, an index of the largest US companies. The S&P 500 is an American stock market index based on the market capitalization of 500 large companies having common stock listed on the NYSE, NASDAQ Exchange.

I will load all 500 dataset in S&P 500 for analysis by using portfolio optimization to get the possible several stocks with higher return and lower risk. And using the machine learning predict the investment trend for S&P 500 index.

- What are the top 20 higher monthly return among all 500 number of stocks in S&P500 by Mathematical programming? The target is to **find out the top valuable, higher return with lower risk of stocks**.
- Could I invest these top 20 stocks now by analysis for the **trend of S&P500 index by Machine learning**? It is to determine if I could **invest these stocks by choosing the most accuracy model with the trend**.

1.3 Why the problem is interesting

Automatic trading without anyone involved will be the trend of stock market near future. I would like to use the data science methods to make a strategic for investment.

I will study which method of machine learning would be more accurate, suitable for prediction by using root-mean-squared error, that the prediction will be more meaningful in use.

1.4 What other approaches have been tried

First of all, I will construct the portfolio optimization in order to achieve a maximum expected return given their risk preferences due to the fact that the returns of a portfolio are greatly affected by nature of the relationship between assets and their weights in the portfolio.

The top 20 monthly return of stocks will be get into the portfolio optimization. Then in order to get the higher return and lower risk of stocks, the portfolio optimization will be conducted to find out which are the best choose of investment and generate the visualization for returns and volatility.

For the next part, I will work with historical data about the S&P500 price index to understand if I can invest in market this moment. I will implement a mix of machine learning algorithms to predict the future stock price of this company, starting with simple algorithms like averaging and linear regression, and then moving on to advanced techniques like Auto ARIMA and LSTM.

And I will compare the models by using root-mean-squared error (RMSE) to measure of how model performed and measure difference between predicted values and the actual values.

1.5 Discussion on your hypothesis is and how you specific solution will improve

Stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

Fundamental Analysis involves analyzing the company's future profitability on the basis of its current business environment and financial performance. Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.

We'll scrape all S&P 500 tickers from Wiki and load all 500 dataset to be in cleaning and appending the adjusted closing price from 2008 to 2018.

- **Moving Average** - The predicted closing price for each day will be the average of a set of previously observed values. Instead of using the simple average, we will be using the moving average technique which uses the latest set of values for each prediction.
- **Linear Regression** - The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable.
- **K-Nearest** - Neighbors Another interesting ML algorithm that one can use here is kNN (k nearest neighbours). Based on the independent variables, kNN finds the similarity between new data points and old data points.
- **ARIMA** - ARIMA is a very popular statistical method for time series forecasting. ARIMA models take into account the past values to predict the future values.
- **Long Short Term Memory (LSTM)** - LSTMs are widely used for sequence prediction problems and have proven to be extremely effective.

2. Literature review

2.1 S&P 500 Index

The S&P 500 or Standard & Poor's 500 Index is a market-capitalization-weighted index of the 500 largest U.S. publicly traded companies. The index is widely regarded as the best gauge of large-cap U.S. equities.

The S&P 500 uses a market capitalization weighting method, giving a higher percentage allocation to companies with the largest market capitalizations. The market capitalization of a company is calculated by taking the current stock price and multiplying it by the outstanding shares.

Reference:

- https://www.investopedia.com/terms/s/sp500.asp (https://www.investopedia.com/terms/s/sp500.asp)

2.2 Time series forecasting

Time series is a collection of data points collected at constant time intervals. These are analyzed to determine the long term trend so as to forecast the future or perform some other form of analysis. But what makes a TS different from say a regular regression problem? There are 2 things:

It is time dependent. So the basic assumption of a linear regression model that the observations are independent doesn't hold in this case. Along with an increasing or decreasing trend, most TS have some form of seasonality trends, i.e. variations specific to a particular time frame. For example, if you see the sales of a woolen jacket over time, you will invariably find higher sales in winter seasons.

Reference:

- https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/ (https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/)

2.3 portfolio construction framework

Modern portfolio theory (MPT) provides investors with a portfolio construction framework that maximizes returns for a given level of risk, through diversification. MPT reasons that investors should not concern themselves with an individual investment's expected return, but rather the weighted average of the expected returns of a portfolio's component securities as well as how individual securities move together. Markowitz consequently introduced the concept of covariance to quantify this co-movement.

It proposed that investors should instead consider variances of return, along with expected returns, and choose portfolios offering the highest expected return for a given level of variance. These portfolios were deemed "efficient." For given levels of risk, there are multiple combinations of asset classes (portfolios) that maximize expected return. Markowitz displayed these portfolios across a two-dimensional plane showing expected return and standard deviation, which we now call the efficient frontier.

Reference:

- https://www.windhamlabs.com/insights/modern-portfolio-theory/ (https://www.windhamlabs.com/insights/modern-portfolio-theory/)

2.4 Monte Carlo Simulation

Monte Carlo simulations are used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables. It is a technique used to understand the impact of risk and uncertainty in prediction and forecasting models.

The technique was first developed by Stanislaw Ulam, a mathematician who worked on the Manhattan Project. After the war, while recovering from brain surgery, Ulam entertained himself by playing countless games of solitaire. He became interested in plotting the outcome of each of these games in order to observe their distribution and determine the probability of winning. After he shared his idea with John Von Neumann, the two collaborated to develop the Monte Carlo simulation.

Reference:

- https://www.investopedia.com/terms/m/montecarlosimulation.asp (https://www.investopedia.com/terms/m/montecarlosimulation.asp)

2.5 Machine learning - Moving average

Smoothing is a technique applied to time series to remove the fine-grained variation between time steps.

The hope of smoothing is to remove noise and better expose the signal of the underlying causal processes. Moving averages are a simple and common type of smoothing used in time series analysis and time series forecasting.

Calculating a moving average involves creating a new series where the values are comprised of the average of raw observations in the original time series.

Reference:

- https://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/ (https://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/)

2.6 Machine learning - Linear regression

Linear regression is a very simple approach for supervised learning. Though it may seem somewhat dull compared to some of the more modern algorithms, linear regression is still a useful and widely used statistical learning method. Linear regression is used to predict a quantitative response Y from the predictor variable X.

Linear Regression is made with an assumption that there's a linear relationship between X and Y.

Reference:

- https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06 (https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06)
- https://www.datascience.com/blog/time-series-forecasting-machine-learning-differences (https://www.datascience.com/blog/time-series-forecasting-machine-learning-differences)

2.7 Machine learning - K-Nearest - Neighbors

K-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

Reference:

- https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/ (https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/)
- https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761)
- https://tomaszgolan.github.io/introduction_to_machine_learning/markdown/introduction_to_machine_learning (https://tomaszgolan.github.io/introduction_to_machine_learning/markdown/introduction_to_machine_learning

2.8 Machine learning - ARIMA

An ARIMA model is a class of statistical models for analyzing and forecasting time series data.

It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.

ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average. It is a generalization of the simpler AutoRegressive Moving Average and adds the notion of integration.

This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

- AR: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.
- I: Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA model are defined as follows:

- p: The number of lag observations included in the model, also called the lag order.
- d: The number of times that the raw observations are differenced, also called the degree of differencing.
- q: The size of the moving average window, also called the order of moving average.

Reference:

- https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/ (https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/)

2.9 Machine learning - Prophet

Prophet is a procedure for forecasting time series data. It is based on an additive model where non-linear trends are fit with yearly and weekly seasonality, plus holidays. It works best with daily periodicity data with at least one year of historical data. Prophet is robust to missing data, shifts in the trend, and large outliers.

Prophet is open source software released by Facebook's Core Data Science team.

The Prophet procedure is an additive regression model with four main components:

- A piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data.
- A yearly seasonal component modeled using Fourier series.
- A weekly seasonal component using dummy variables.
- A user-provided list of important holidays.

Reference:

- https://research.fb.com/prophet-forecasting-at-scale/ (https://research.fb.com/prophet-forecasting-at-scale/)

2.10 Machine learning - Long Short-Term Memory

Long Short-Term Memory usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is their default behavior.

All recurrent neural networks have the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. The key to LSTMs is the cell state which is acting like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to flow along it unchanged.

Reference:

- https://machinelearningtutorials.com/long-short-term-memory-lstm/ (https://machinelearningtutorials.com/long-short-term-memory-lstm/)

3. Methodology section

This project are separated 4 part of analysis from data exploration, visualization, correlation and monthly return for data extraction by mathematical programming, portfolio optimization and machine learning.

The following strategic analysis as follow:

3.1 Data Exploration (Getting 500 stocks data)

1. To use **Beautifulsoup** grap stocks symbol for S&P 500
2. To use API in the function of **"fix_yahoo_finance"** to load all dataset for 500 stocks share from January 2008 until Now, each stocks symbol will be used to load their own CSV dataset file.
3. To select each stocks of adjusted closing price and use the **"join"** function and rename the columns to create the joined CSV
4. To use both **"plotly"** and **"matplotlib"** for data visualization, "iplot" can be used to compare the detail price in certain period of time.

3.2 Higher monthly return (Choosing top 10 stocks correlation and monthly return with index from 500 stocks data)

1. To use **corr()** function to find out the **top 30 stocks** which are higher correlation with S&P 500 index from **500 stocks share**.
2. The Last part of this project will conduct the prediction of machine learning for S&P 500 index, so the correlation between stocks and index is important reference for the trend affected by index fluctuation. And it will provide the buy and sell signal when the prediction are generated.
3. After getting the top 30 stocks of highest correlation with index, we will calculate and **compare the higher monthly return for this 30 stocks**.
4. To calculate the **top 10 stocks of highest monthly return from the above 30**. we will sort and extract the joined dataset for the next part of Portfolio optimization.

3.3 Portfolio Optimization (Testing top 10 stocks from part 2 for portfolio optimization)

1. After getting the top 10 stocks from part 2 by using Highest correlation and monthly return without consideration the factor of risk, but this stage we would like to use the portfolio optimization method to decide the investment strategic.
2. Calculating the **top 10 higher monthly return** of stocks share as Medium or long term investment
3. Using **portfolio optimization** to calculate the **top 10 higher monthly return** of stocks share as Medium or long term investment which are higher monthly return and lower risk as investment strategic.

3.4 Machine Learning

After portfolio optimization analysis, it assume we get the proportion of investment strategic, we will expect to know if it is time to invest this moment or when is the best momnent for buying or selling. Therefore, the machine learning for S&P 500 index will be conducted in machine learning process by comparing different methods and model.

The command process as follow:

```
- Dropping NAN with dropna in **pandas**
- splitting into train and validation
- Measuring root mean square error (RMSE) for the standard deviation of resi
duals
- Plot the prediction

1. Moving average

2. Linear Regression
- Using the function of **linear_model** in **sklearn**

3. k-Nearest Neighbours
- Using the function of **neighbors, GridSearchCV, MinMaxScaler** in **sklea
rn** to find the best parameter

4. Auto ARIMA
- Using the function of **auto_arima** in **pmdarima** which automatically s
elects the best combination of (p,q,d) that provides the least error.

5. Prophet
- Using the function of **Prophet in fbprophet** which Prophet, designed by
 Facebook, is a time series forecasting library that The input for Prophet i
s a dataframe with two columns: date and target.

6. Long Short Term Memory
- Using the function of **MinMaxScaler in sklearn** and **LSTM in keras** to
 create and fit the LSTM network
```

Loading libraries

Out[65]: The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.

4. Definition of data collection method

4.1 Data Exploration and Description

In order to **get the total 500 stocks shares from January 2008 until Now (April 20 2019) around 10 years data**. The BeautifulSoup method has been used for scrapying the dataset in web of Nasdaq (http://www.nasdaq.com/symbol/ (http://www.nasdaq.com/symbol/)), but it found that the "id" will be changed automatically in certain period, it is not a stable method to get the dataset in our research project, the API of yahoo finance will be the best way to load large range of data.

Getting the stocks symbol

- Stock Symbols are scraping by **"BS4 - BeautifulSoup" in Web of Wiki (http://en.wikipedia.org/wiki/ (http://en.wikipedia.org/wiki/))**
- Adding the S&P index symbol into the list for measuring the correlation of stocks share.

Getting the data

- Before we analyze stock data, we need to get it into some workable format. Stock data can be obtained from Yahoo! Finance, Google Finance, or a number of other sources. These days I recommend **getting data from Yahoo! Finance**, a provider of community-maintained financial and economic data. Yahoo! Finance used to be the go-to source for good quality stock data.
- Then to **input stocks symbols scraped in API in the function of "fix_yahoo_finance", it loads all dataset for 500 stocks share from January 2008 until Now.**

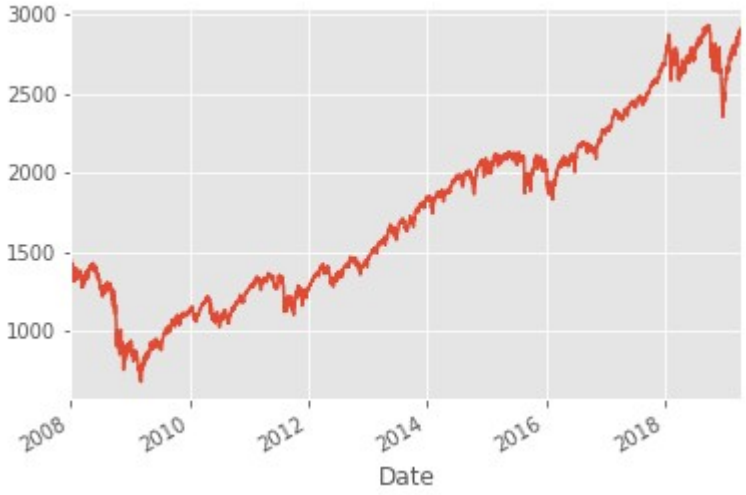
Detail of dataset

- **Open** is the price of the stock at the beginning of the trading day (it need not be the closing price of the previous trading day), **high** is the highest price of the stock on that trading day, **low** the lowest price of the stock on that trading day, and **close** the price of the stock at closing time. **Volume** indicates how many stocks were traded. **Adjusted prices** (such as the adjusted close) is the price of the stock that adjusts the price for corporate actions. While stock prices are considered to be set mostly by traders, stock splits (when the company makes each extant stock worth two and halves the price) and dividends (payout of company profits per share) also affect the price of a stock and should be accounted for.

Visualization

- We have stock data we would like to visualize it. I will use the **matplotlib** package. And The **plotly** will also be used for visualization for comparing the trend between stocks and index. It will be more easy and clear if many stocks share are plotted at the same graph.

```
[**********100%**********]  1 of 1 downloaded
```



Date	Open	High	Low	Close	Adj Close	Volume
2019-04-12	2900.860107	2910.540039	2898.370117	2907.409912	2907.409912	3688490000
2019-04-15	2908.320068	2909.600098	2896.479980	2905.580078	2905.580078	3088330000
2019-04-16	2912.260010	2916.060059	2900.709961	2907.060059	2907.060059	3402210000
2019-04-17	2916.040039	2918.000000	2895.449951	2900.449951	2900.449951	3602300000
2019-04-18	2904.810059	2908.399902	2891.899902	2905.030029	2905.030029	3506850000

4.2 Data Preparation - Stock Tickers and Dataset

To get all company pricing data in the S&P 500, and now we're going to pull stock pricing data on all of them.

Process for data preparation:

1. Using "bs" function to extract the symbol of S&P 500
2. Removing the error symbol or ticket
3. Adding the "^GSPC" index of S&P 500 for comparation purpose

	name	symbol
0	3M Company	MMM
1	Abbott Laboratories	ABT
2	AbbVie Inc.	ABBV
3	ABIOMED Inc	ABMD
4	Accenture plc	ACN

After getting the tickers:

1. Then it use datetime to specify dates from 2008 until now for the Pandas datareader, os is to check for, and create, directories.
2. The datasets for each stocks are downloaded by the function of Pandas datareader and use "to_csv" to generate each dataset csv files.

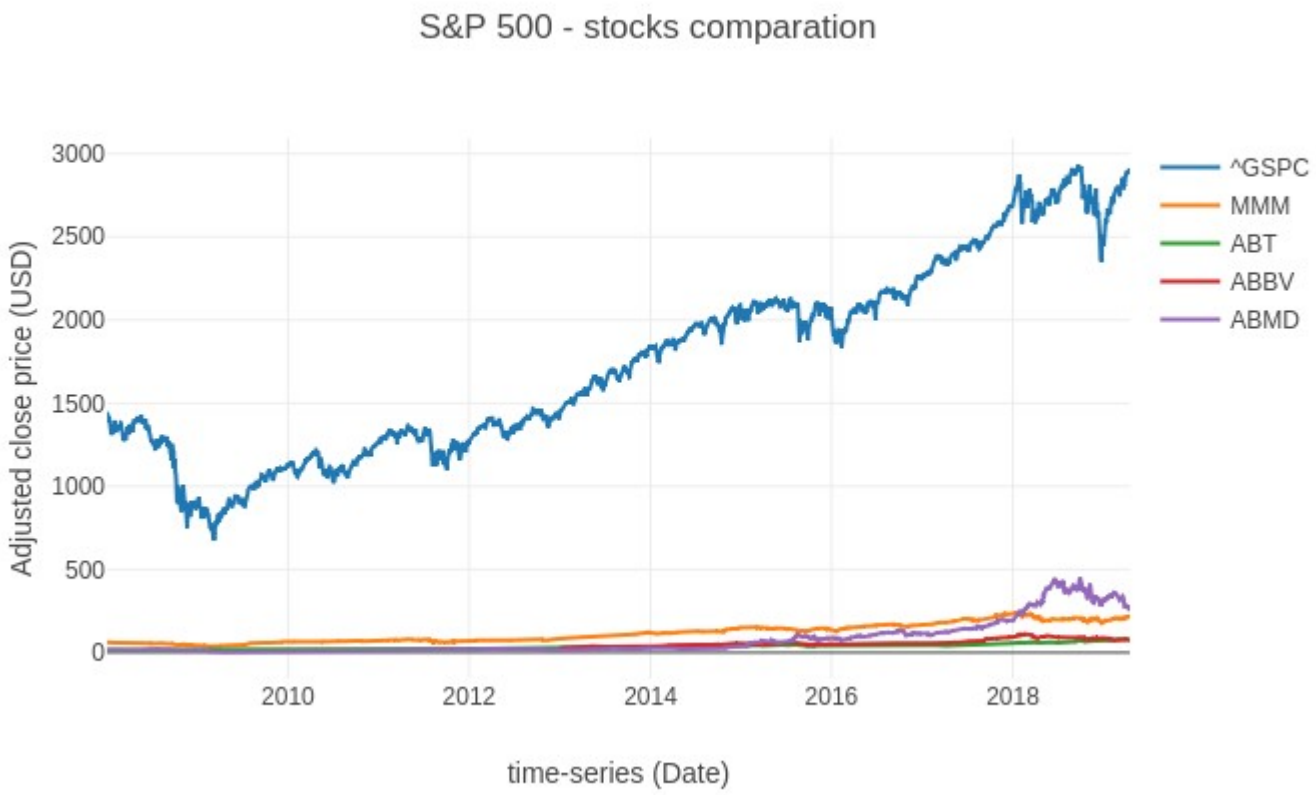
Compiling all the CSV stock data files by using the adjusted close price and rename the columns to each stock tickers and export to a "Joined" csv file.

We collected the total 504 number of stock shares with S&P 500 index value from January 2008 until Now (03-14-2019)

4.3 Below is a visualization by iplot of the comparation stock shares with S&P 500 index.

We considered if it plot all 500 stock shares, it can't appear a meaningful comparation graph. Initially, we view the first 5 column including S&P 500 index in the time series from 2008 to 2019 (Now). We will further see the top 10 monthly return, top 10 for higher correlation with S&P 500 index, and top 10 investing stocks strategic from the portfolio optimization in our data time series.

The **offline plotly** has been used for visualization instead of using "matplotlib" function to plot the graph, because it include too many stocks share and also the higher closing price is not equal to earn higher return in investment. It is not a valuable comparing method.

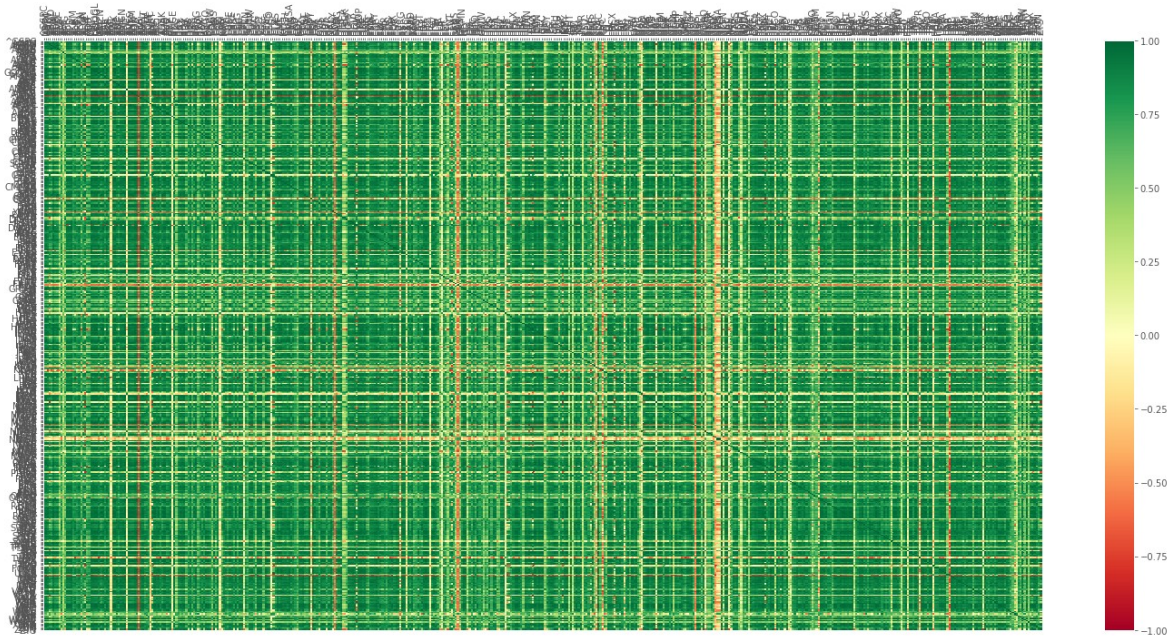


If only checking the adjusted closing price on the graph, it does not get any trend or prediction result. Therefore, the following parts will conduct portfolio optimization and machine learning analysis for getting the best strategic to have a highest return investment.

Correlation

Does it have strong relation with S&P 500 index fluctation when we invest some stocks?

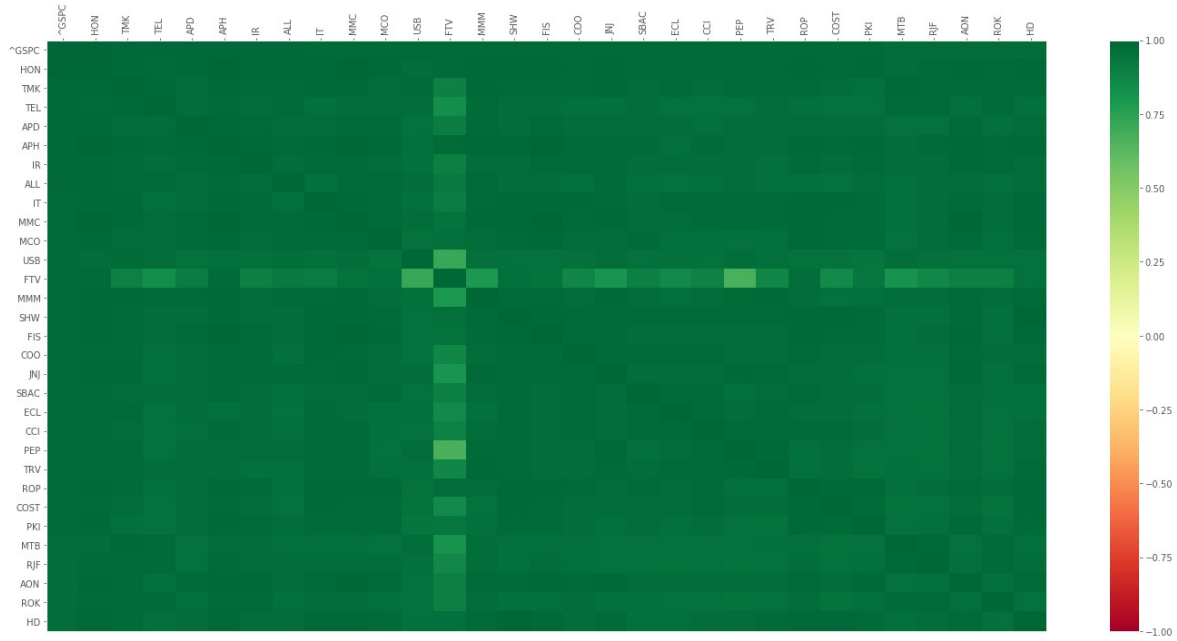
Most people think that they will get higher return when the main index going up, so we can calculate the correlation to check which stocks share are strong relationship and low relationship.



5. Experimentation and Results

5.1 The highest Correlation (Sort out top 30 from all 500 stocks share)

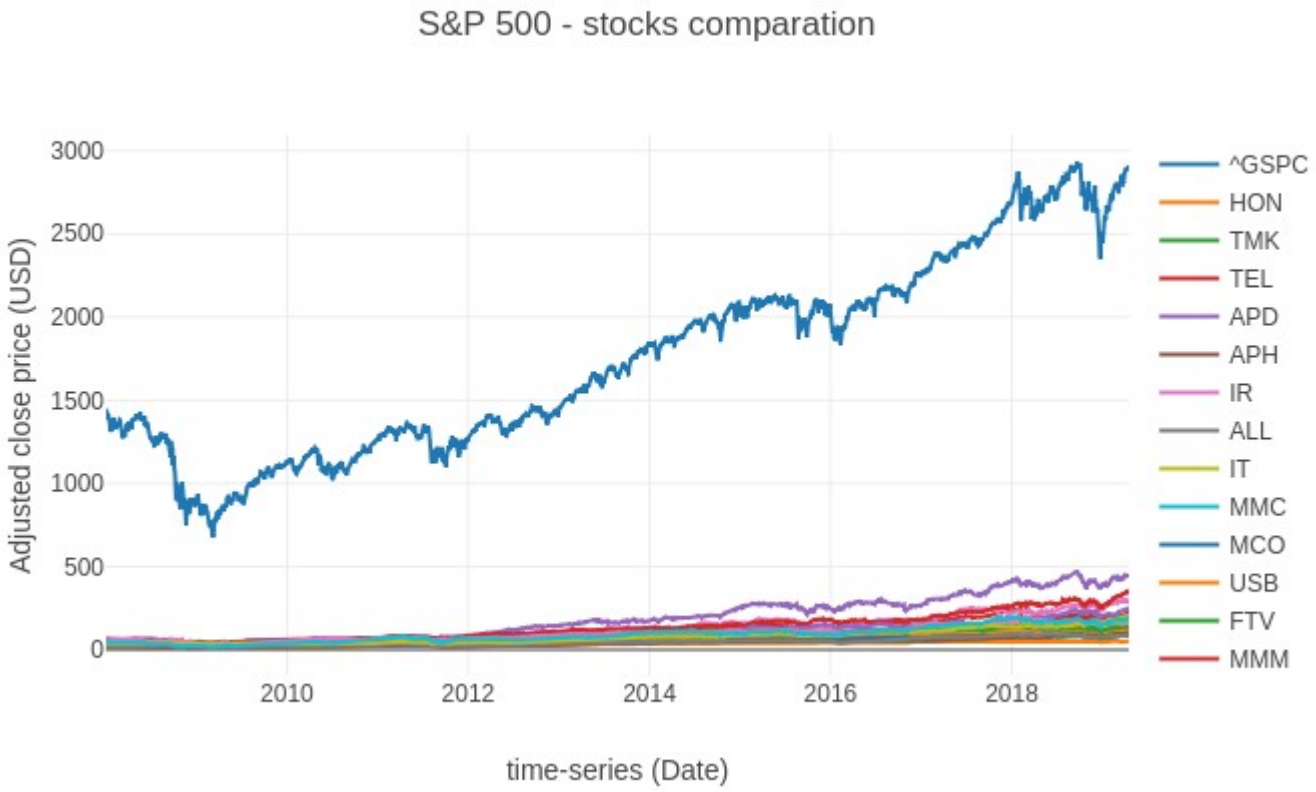
- To use corr() function to find out the top 30 stocks which are higher correlation with S&P 500 index from 500 stocks share.
- The Last part of this project will conduct the prediction of machine learning for S&P 500 index, so the correlation between stocks and index is important reference for the trend affected by index fluctuation. And it will provide the buy and sell signal when the prediction are generated.
- After getting the top 30 stocks of highest correlation with index, we will calculate and compare the higher monthly return for this 30 stocks.
- To calculate the top 10 stocks of highest monthly return from the above 30. we will sort and extract the joined dataset for the next part of Portfolio optimization.



Out[59]:
```
['^GSPC',
 'HON',
 'TMK',
 'TEL',
 'APD',
 'APH',
 'IR',
 'ALL',
 'IT',
 'MMC',
 'MCO',
 'USB',
 'FTV',
 'MMM',
 'SHW',
 'FIS',
 'COO',
 'JNJ',
 'SBAC',
 'ECL',
 'CCI',
 'PEP',
 'TRV',
 'ROP',
 'COST',
 'PKI',
 'MTB',
 'RJF',
 'AON',
 'ROK',
 'HD']
```

We create a new dataframe for higher correlation stocks share with S&P 500 index to further analysis.

We can believe if choosing the above stocks share for investment, the S&P index will be a higher relative reference.



Assuming we pick the tickers which are the most higher correlation with S&P 500 index (^GSPC), the trend of machine learning will be meaningful and significate for prediction

If the other stock shares which is less correlation with S&P 500 index, it is no prediction value when we build the machine learning model for S&P 500 index.

5.2 The highest monthly return (Sort out top 10 from 30 stocks share)

A “better” solution, though, would be to plot the information we actually want: the stock’s returns. This involves transforming the data into something more useful for our purposes. There are multiple transformations we could apply.

One transformation would be to consider the stock’s return since the beginning of the period of interest.

$$\text{return}_{t,0} = \frac{\text{price}_t}{\text{price}_0}$$

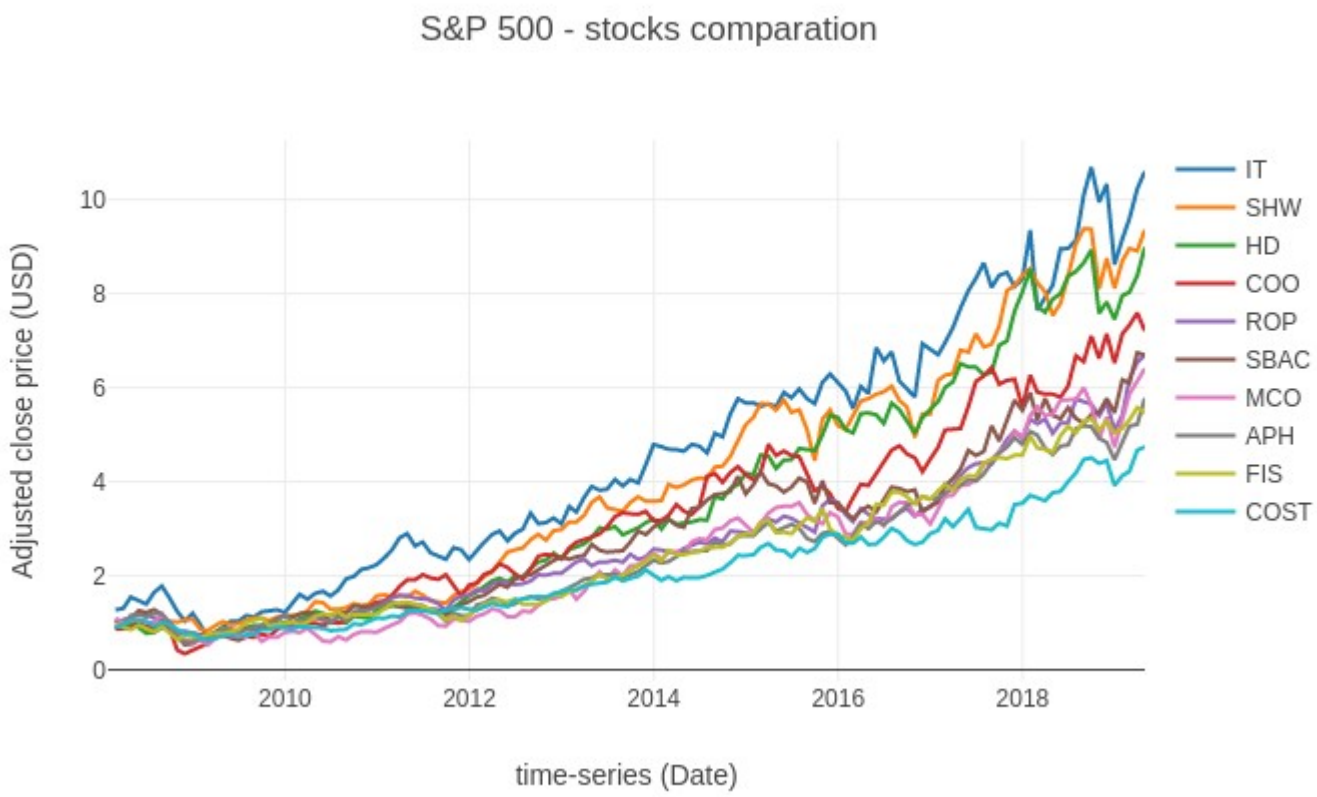
I am using a **lambda function**, which allows me to pass a small function defined quickly as a parameter to another function or method

---

Out[63]:

Date	IT	SHW	HD	COO	ROP	SBAC	MCO	APH	FIS	COST
2018-07-31	9.132164	9.055837	8.472125	6.675123	5.756390	5.344478	5.750779	5.129259	5.036642	4.187018
2018-08-31	10.098449	9.379355	8.655667	6.554177	5.689082	5.242485	5.998048	5.187952	5.282784	4.474156
2018-09-30	10.687795	9.371737	8.930724	7.101767	5.647898	5.424856	5.633489	5.169493	5.342351	4.507741
2018-10-31	9.947404	8.100652	7.582601	6.619007	5.401586	5.476866	4.901674	4.920970	5.098915	4.387793
2018-11-30	10.329737	8.748771	7.820522	7.144816	5.682071	5.768659	5.375525	4.835197	5.287492	4.449202
2018-12-31	8.620364	8.117271	7.451876	6.521378	5.088832	5.467409	4.732435	4.466833	5.038099	3.918829
2019-01-31	9.163183	8.696162	7.959740	7.143824	5.417782	6.164471	5.356600	4.847247	5.135373	4.128901
2019-02-28	9.595415	8.960154	8.029566	7.329108	6.190303	6.097940	5.867519	5.180798	5.313217	4.219332
2019-03-31	10.227916	8.908651	8.384287	7.589993	6.540701	6.742992	6.137644	5.219292	5.574535	4.670700
2019-04-30	10.600135	9.357900	8.985942	7.206867	6.697916	6.693009	6.407770	5.779133	5.465607	4.741492

Using 10 years cumulative return for selecting higher return, it may be some problem, like it does not reflect the trend of industry, because new industry may have large return only these few years, it may use only 3 years cumulative return, it would be more realistic. But now in research level, we keep using 10 years sum of return.



Apart from the risk consideration, the percentage of increasing return can be 9.8 times over 10 year if just only invest "IT" tickers of stock. Of course, that can be a part of investment strategic we can notice.

```
['IT', 'SHW', 'HD', 'COO', 'ROP', 'SBAC', 'MCO', 'APH', 'FIS', 'COS
T']
```

```
Gartner Inc
Sherwin-Williams
Home Depot
The Cooper Companies
Roper Technologies
SBA Communications
Moody's Corp
Amphenol Corp
Fidelity National Information Services
Costco Wholesale Corp.
```

5.3 Portfolio Optimization - Random Portfolios

“Modern Portfolio Theory (MPT), a hypothesis put forth by Harry Markowitz in his paper “Portfolio Selection,” (published in 1952 by the Journal of Finance) is an investment theory based on the idea that risk-averse investors can construct portfolios to optimize or maximize expected return based on a given level of market risk, emphasizing that risk is an inherent part of higher reward. It is one of the most important and influential economic theories dealing with finance and investment.

In order to choose from 10 number of stocks share from the result of part 1 which are higher monthly return and lower risk as our final medium or long term investment strategic.

We have **10 stocks in our portfolio**. One decision we have to make is how we should allocate our budget to each of stock in our portfolio. If our total budget is 1, then we can decide the weights for each stock, so that the sum of weights will be 1. And the value for weights will be the portion of budget we allocate to a specific stock. For example, if weight is 0.5 for Amazon, it means that we allocate 50% of our budget to Amazon.

“portfolio_annualised_performance” will calculate the returns and volatility, and to make it as an annualised calculation into 252 trading days in one year. “random_portfolios” function will generate portfolios with random weights assigned to each stock, and by giving num_portfolios argument, you can decide how many random portfolios you want to generate.

Out[25]:

Date	IT	SHW	HD	COO	ROP	SBAC	MCO	APH	FIS	COST
2008-01-02	17.139999	47.881672	19.503134	37.653690	56.886524	33.139999	29.759460	20.039412	19.284296	51.994942
2008-01-03	16.830000	47.856274	19.286514	38.030224	56.504227	32.290001	29.088890	19.921530	19.389524	51.216621
2008-01-04	15.980000	45.790310	18.644131	38.099590	53.800243	31.170000	28.214615	19.459085	19.069073	50.094925
2008-01-07	16.510000	47.001087	18.950386	37.554592	50.723274	30.330000	29.105852	19.427345	18.466436	50.346737
2008-01-08	16.790001	45.714100	18.479799	37.564507	48.541424	29.559999	28.265533	18.747278	18.203388	49.576035

Firstly I download daily price data for each of the stocks in the portfolio, and convert daily stock prices into daily returns. And then it need to calculate annualised portfolio return and annualised portfolio volatility.

Portfolio standard deviation

The first is the calculation for portfolio's volatility in "portfolio_annualised_performance" function.

$$\sigma_{portfolio} = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2Cov_{1,2}}$$

This formula can be simplified if we make use of matrix notation.

$$\sigma_p^2 = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} \\ \sigma_{2,1} & \sigma_2^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_1\sigma_1^2 + w_2\sigma_{2,1} & w_1\sigma_{1,2} + w_2\sigma_2^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$
$$= w_1^2\sigma_1^2 + w_1w_2\sigma_{2,1} + w_1w_2\sigma_{1,2} + w_2^2\sigma_2^2$$
$$= w_1^2\sigma_1^2 + 2w_1w_2\sigma_{1,2} + w_2^2\sigma_2^2$$

For matrix calculation, we get the part inside the square root in the original formula. Same as the annualised return, I took into account of 252 trading days to calculate the annualised standard deviation of a portfolio.

Sharpe ratio

For the Sharpe ratio, Risk-adjusted return refines an investment's return by measuring how much risk is involved in producing that return, which is generally expressed as a number or rating. There could be a number of different methods of expressing risk-adjusted return, and the Sharpe ratio is one of them.

The ratio describes how much excess return you are receiving for the extra volatility that you endure for holding a riskier asset. The Sharpe ratio can be expressed in below formula.

$$= \frac{\bar{r}_p - r_f}{\sigma_p}$$

Where:

$\bar{r}_p$ = Expected porfolio return

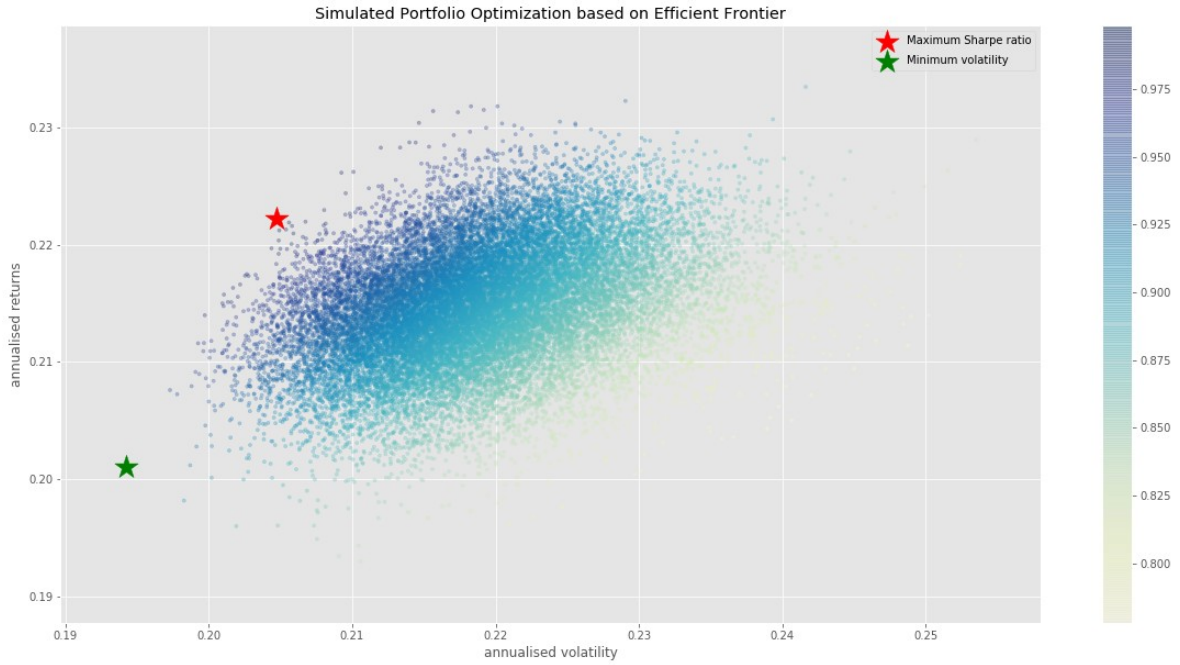
r_f = Risk free rate

σ_p = Portfolio standard deviation

I can get daily returns by calling **pct_change** on the data frame with the price data. And the **mean daily returns**, the covariance matrix of returns are needed to **calculate portfolio returns and volatility**. We will generate **25,000 random portfolios**. Finally, the risk-free rate has been taken from U.S. Department of The Treasury. The rate of **1.78% is the 52week treasury bill rates** at the start of 2018.

And does similar steps for minimum volatility portfolio, and displays it as a green star on the plot.

```
--------------------------------------------------------------------
-----------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.22
Annualised Volatility: 0.2


                IT    SHW     HD    COO   ROP  SBAC   MCO   APH   FIS
COST
allocation  15.46  22.04  15.61  16.67  7.35  1.68  0.35  1.92  8.99
9.93
--------------------------------------------------------------------
-----------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.2
Annualised Volatility: 0.19


               IT    SHW   HD    COO   ROP  SBAC   MCO   APH    FIS
COST
allocation  2.79  17.67  2.8  18.34  4.43  1.55  2.42  1.77  14.25  3
3.99
```



For minimum risk portfolio, we can see around 30% of our budget is allocated to "Cost" - Costco. If you take another look at the daily return plot from earlier, the Costco is the least volatile among these stocks, so allocating a large percentage to Costco for minimum risk portfolio makes sense.

If we are willing to take higher risk for higher return, one that gives us the best risk-adjusted return is the one with maximum Sharpe ratio. In this scenario, we are allocating a significant portion to "SHW" - Sherwin-Williams and "IT" - Gartner Inc, which are quite volatile stocks from the previous plot of daily returns. And "Cost" - Costco which had around 30% in the case of minimum risk portfolio, has only 10% budget allocated to it.

5.4 Portfolio Optimization - Efficient Frontier

From the plot of the randomly simulated portfolios, we can see it forms a shape of an arch line on the top of clustered blue dots. This line is called efficient frontier.

Points along the line will give you the lowest risk for a given target return. All the other dots right to the line will give you higher risk with same returns. If the expected returns are the same, the option with lower risk should be taken.

The way for two kinds of optimal portfolio above was by simulating many possible random choices and pick the best ones (either minimum risk or maximum risk-adjusted return) by **using Scipy's optimize function**.

Scipy's optimize function is doing the similar task when given what to optimize, and what are constraints and bounds.

Below functions are to get the maximum Sharpe ratio portfolio. In Scipy's optimize function, there's no 'maximize', so as an objective function you need to pass something that should be minimized and "neg_sharpe_ratio" is computing the negative Sharpe ratio and use this as our objective function to minimize.

In "max_sharpe_ratio" function:

constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) — 1})

The above constraint is saying that sum of x should be equal to 1.

np.sum(x) == 1 has become np.sum(x)-1

It simply means that the sum of all the weights should be equal to 1. It cannot be allocated more than 100% of budget in total.

"bounds" is giving another limit to assign random weights, by saying any weight should be inclusively between 0 and 1. You cannot give minus budget allocation to a stock or more than 100% allocation to a stock.

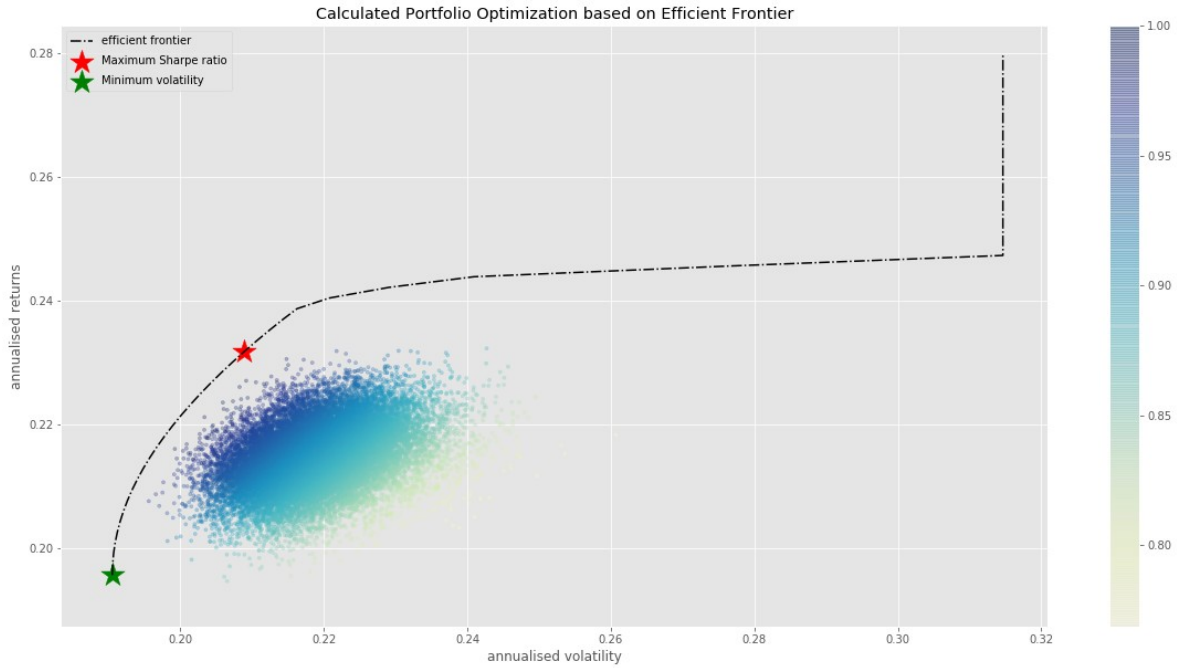
The **efficient frontier** is to draw a line which depicts where the efficient portfolios for a given risk rate.

The first function "efficient_return" is calculating the most efficient portfolio for a given target return, and the second function "efficient_frontier" will take a range of target returns and compute efficient portfolio for each return level.

Let's try to plot the portfolio choices with maximum Sharpe ratio and minimum volatility also with all the randomly generated portfolios.

But this time, it does not pick the optimal ones from the randomly generated portfolios, but we are actually calculating by using Scipy's 'minimize' function. And the below function will also plot the efficient frontier line.

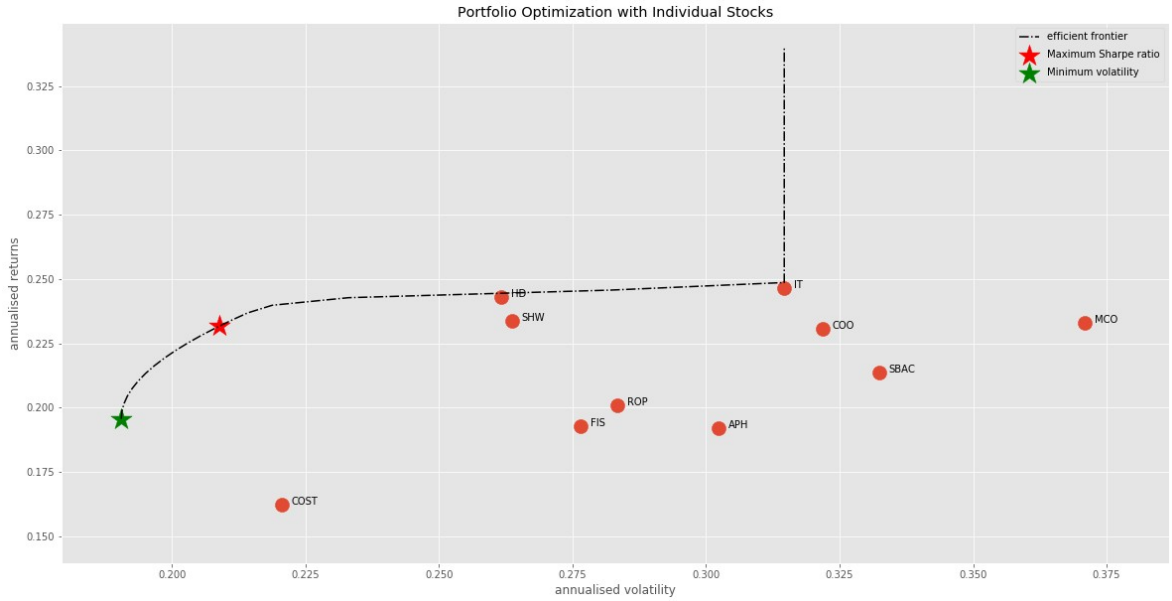
```
------------------------------------------------------------------------
-----------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.23
Annualised Volatility: 0.21


                IT    SHW     HD    COO  ROP  SBAC  MCO  APH   FIS  CO
ST
allocation  16.21  29.92  27.65  17.41  0.0   0.0  0.0  0.0  0.87  7.
94
------------------------------------------------------------------------
-----------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.2
Annualised Volatility: 0.19


               IT    SHW    HD    COO   ROP  SBAC  MCO  APH   FIS    CO
ST
allocation  3.93  20.48  2.47  12.69  0.79  1.37  0.0  0.0  12.4  45.
87
```



The slight difference is that the **Scipy's "optimize" function** has not allocated any budget at all for **Costco** on maximum Sharpe ratio portfolio, while one we chose from the randomly generated samples has **0.45% of allocation for Costco**. There are some differences in the decimal places but more or less same.

Instead of plotting every randomly generated portfolio, we can plot each individual stocks on the plot with the corresponding values of each stock's annual return and annual risk. This way we can see and compare how diversification is lowering the risk by optimising the allocation.

```
------------------------------------------------------------------------
-----------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.23
Annualised Volatility: 0.21


               IT    SHW     HD    COO  ROP  SBAC  MCO  APH   FIS  CO
ST
allocation  16.21  29.92  27.65  17.41  0.0   0.0  0.0  0.0  0.87  7.
94
------------------------------------------------------------------------
-----------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.2
Annualised Volatility: 0.19


              IT    SHW    HD    COO   ROP  SBAC  MCO  APH   FIS   CO
ST
allocation  3.93  20.48  2.47  12.69  0.79  1.37  0.0  0.0  12.4  45.
87
------------------------------------------------------------------------
-----------
Individual Stock Returns and Volatility

IT : annuaised return 0.25 , annualised volatility: 0.31
SHW : annuaised return 0.23 , annualised volatility: 0.26
HD : annuaised return 0.24 , annualised volatility: 0.26
COO : annuaised return 0.23 , annualised volatility: 0.32
ROP : annuaised return 0.2 , annualised volatility: 0.28
SBAC : annuaised return 0.21 , annualised volatility: 0.33
MCO : annuaised return 0.23 , annualised volatility: 0.37
APH : annuaised return 0.19 , annualised volatility: 0.3
FIS : annuaised return 0.19 , annualised volatility: 0.28
COST : annuaised return 0.16 , annualised volatility: 0.22
------------------------------------------------------------------------
-----------
```



As you can see from the above plot, the stock with the least risk is **COST at around 0.22**, but the return is only around 0.16. If we will to take slightly more risk around 0.225 return, we should consider to choose **HD and SHW** rather than **IT** with higher risk with portfolio optimisation.

6. Machine Learning

We will work with historical data about the stock prices of a publicly listed company. We will implement a mix of machine learning algorithms to predict the future stock price of this company, starting with simple algorithms like averaging and linear regression, and then move on to advanced techniques like Auto ARIMA and LSTM.

1. Moving Average
2. Linear Regression
3. k-Nearest Neighbors
4. Auto ARIMA
5. Prophet
6. Long Short Term Memory (LSTM)

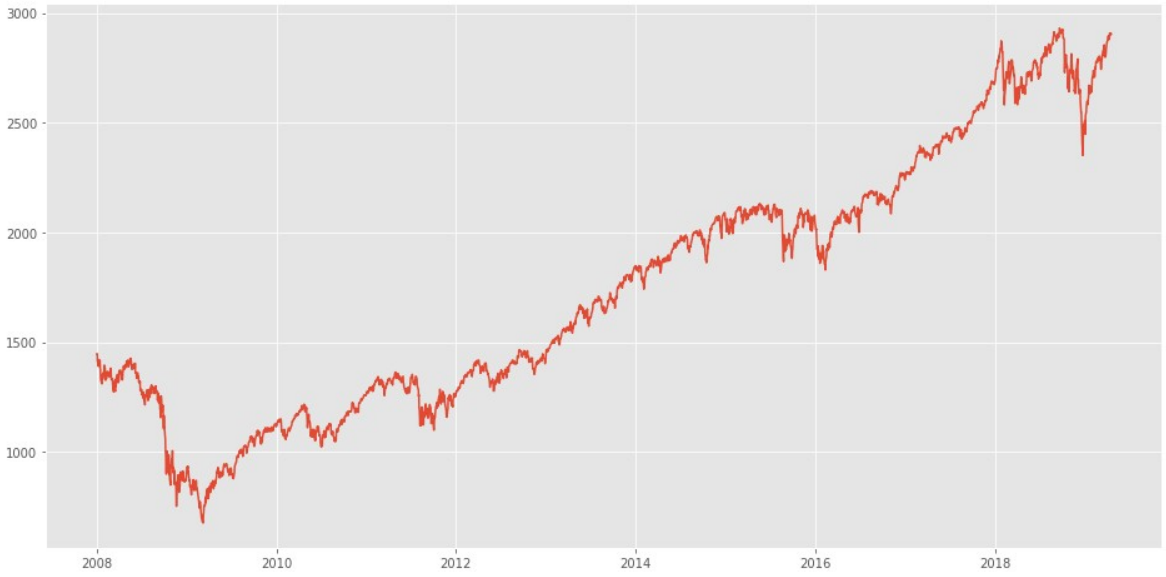
In this technical analysis part, we will use a dataset from **Part1 - Data Exploration** and load the S&P 500 index for analysis.

In order to confirm the trend of market, we can determine if it is time to invest the **stocks which we obtain from Part 3 - portfolio optimization**. Because it is the highest correlation with S&P 500 index, it is an important technical analysis if the trend of model is upward, that mean we can invest this moment.

Out[22]:

	Date	Close
2839	2019-04-12	2907.409912
2840	2019-04-15	2905.580078
2841	2019-04-16	2907.060059
2842	2019-04-17	2900.449951
2843	2019-04-18	2905.030029

The profit or loss calculation is usually determined by the closing price of a stock for the day, hence we will consider the closing price as the target variable. Let's plot the target variable to understand how it's shaping up in our data:



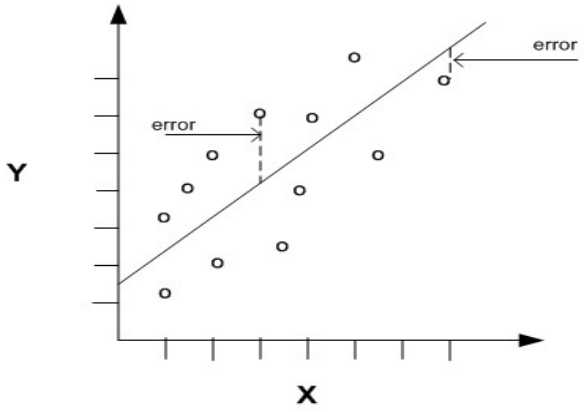
In the upcoming sections, we will explore these variables and use different techniques to predict the daily closing price of the stock.

Comparsion by RMSE

The root-mean-squared error (RMSE) is a measure of how well your model performed. It does this by measuring difference between predicted values and the actual values.

Let's say you feed a model some input X and your model predicts 10, but the actual value is 5. This difference between your prediction (10) and the actual observation (5) is the **error term: (y_prediction - y_actual)**.

The error term is important because we usually want to minimize the error. In other words, our predictions are very close to the actual values.



But there are different ways we could minimize this error term. We could minimize the squared error. Or minimize the absolute value of the error.

In our case, we take the square root of the squared difference (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

In our example above, we would get:

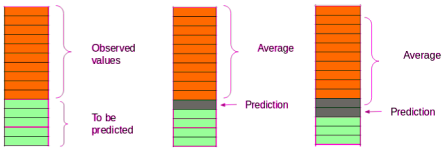
$$\sqrt{(10-5)^2} = 5$$

6.1 Machine Learning - Moving Average

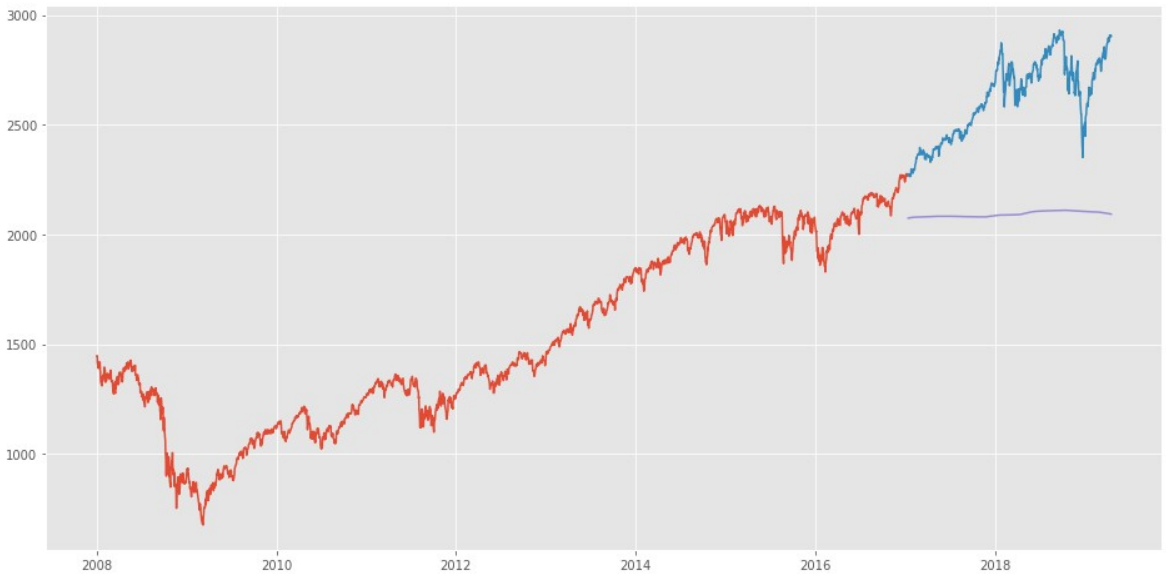
Introduction

'Average' is easily one of the most common things we use in our day-to-day lives. For instance, calculating the average marks to determine overall performance, or finding the average temperature of the past few days to get an idea about today's temperature – these all are routine tasks we do on a regular basis. So this is a good starting point to use on our dataset for making predictions.

The predicted closing price for each day will be the average of a set of previously observed values. Instead of using the simple average, we will be using the moving average technique which uses the latest set of values for each prediction. In other words, for each subsequent step, the predicted values are taken into consideration while removing the oldest observed value from the set.



```
The shape is (2844, 2) (2275, 2) (569, 2)
2008-01-02 00:00:00 2017-01-12 00:00:00 2017-01-13 00:00:00 2019-04-1
8 00:00:00
The root mean square error is 556.7202646158338

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:30: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:31: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy


(556.7202646158338, 2092.7282045845936)
```



Result

The RMSE value is close to 556.7202646158338 but the results are not very promising (as you can gather from the plot). The predicted values are not the same range as the observed values in the train set (there is not obvious an increasing trend and then a slow decrease).

6.2 Machine Learning - Linear Regression

Introduction

The most basic machine learning algorithm that can be implemented on this data is linear regression. The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable.

The equation for linear regression can be written as:

$$Y = \theta_1 X_1 + \theta_2 X_2 + \dots \theta_n X_n$$

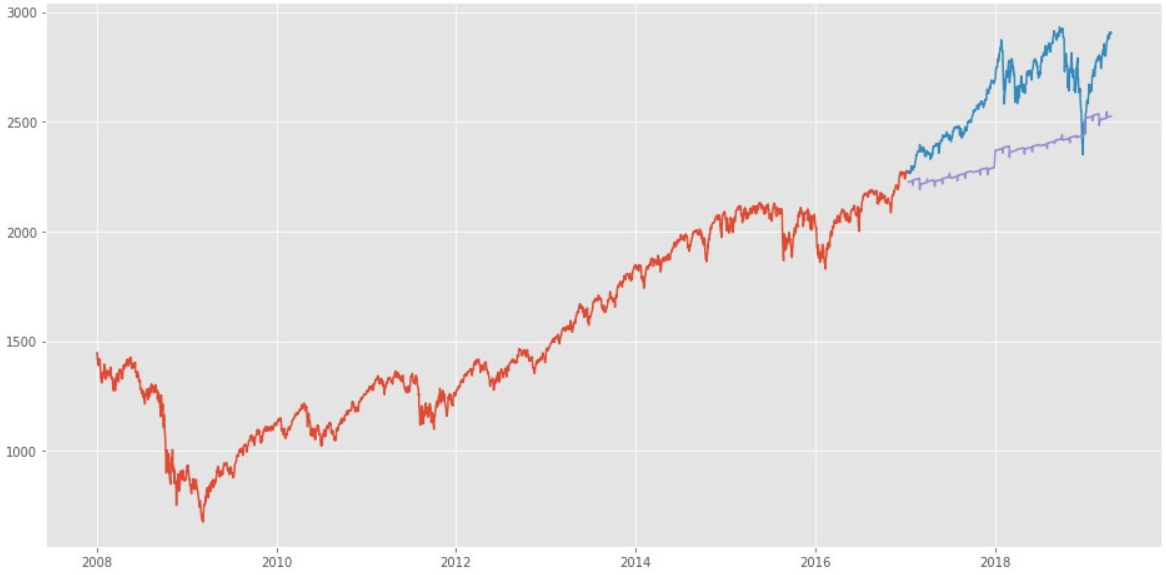
Here, x1, x2,….xn represent the independent variables while the coefficients θ1, θ2, …. θn represent the weights.

We will first sort the dataset in ascending order and then create a separate dataset with new feature created by using **add_datepart from fastai library**.

This creates features such as:

‘Year’, ‘Month’, ‘Week’, ‘Day’, ‘Dayofweek’, ‘Dayofyear’, ‘Is_month_end’, ‘Is_month_start’, ‘Is_quarter_end’, ‘Is_quarter_start’, ‘Is_year_end’, and ‘Is_year_start’.

Apart from this, we can add our own set of features that we believe would be relevant for the predictions. For instance, my hypothesis is that the first and last days of the week could potentially affect the closing price of the stock far more than the other days. So I have created a feature that identifies whether a given day is Monday/Friday or Tuesday/Wednesday/Thursday.



The RMSE value is lower than the previous technique, which clearly shows that linear regression has performed better.

Result

Linear regression is a simple technique and quite easy to interpret, but there are a few obvious disadvantages. One problem in using regression algorithms is that the model overfits to the date and month column. Instead of taking into account the previous values from the point of prediction, the model will consider the value from the same date a month ago, or the same date/month a year ago.

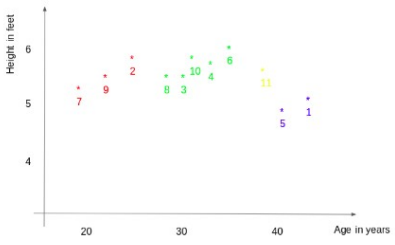
6.3 Machine Learning - k-Nearest Neighbours

Introduction

ML algorithm can use here is kNN (k nearest neighbours). Based on the independent variables, kNN finds the similarity between new data points and old data points.

Here is a simple example to explain the concept of KNN:

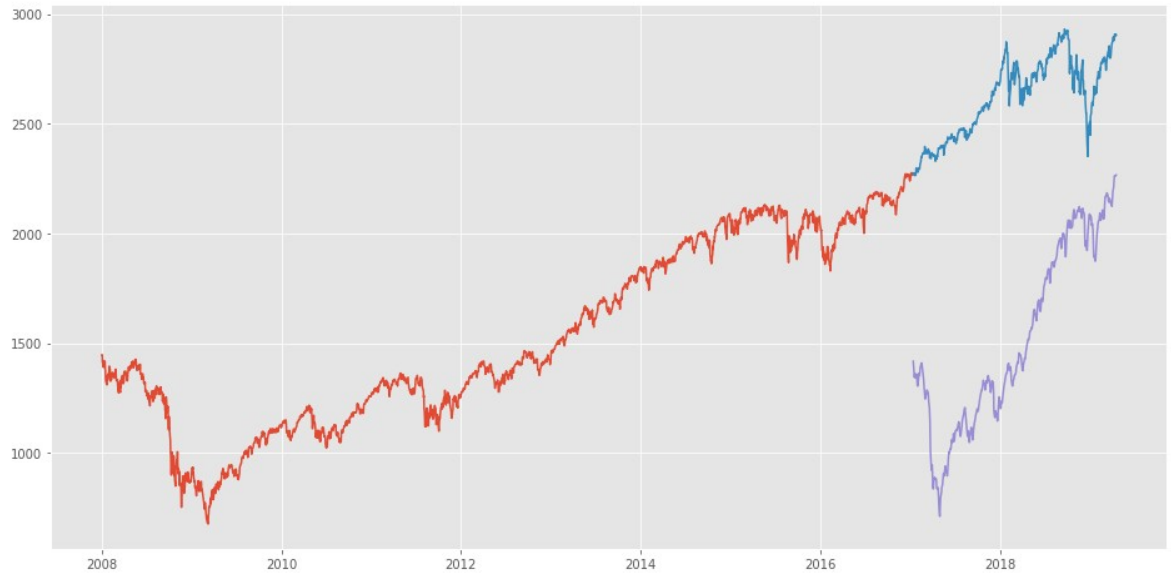
ID	Age	Height	Weight
1	45	5	77
2	26	5.11	47
3	30	5.6	55
4	34	5.9	59
5	40	4.8	72
6	36	5.8	60
7	19	5.3	40
8	28	5.8	60
9	23	5.5	45
10	32	5.6	58
11	38	5.5	?



To determine the weight for ID #11, kNN considers the weight of the nearest neighbors of this ID. The weight of ID #11 is predicted to be the average of it's neighbors. If we consider three neighbours (k=3) for now, the weight for ID#11 would be = (77+72+60)/3 = 69.66 kg.

The training error rate and the validation error rate are two parameters we need to access on different K-value.

```
The root mean square error is 1142.9905687152443

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:40: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:41: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy


(1142.9905687152443, 2266.327772111111)
```



Result

The RMSE value of KNN is higher than the linear regression model and the plot shows the same pattern.

Like linear regression, kNN also identified a raising trend from January 2019 since that has been the pattern for the past years. We can safely say that regression algorithms have not performed well on this dataset.

6.4 Machine Learning - Auto ARIMA

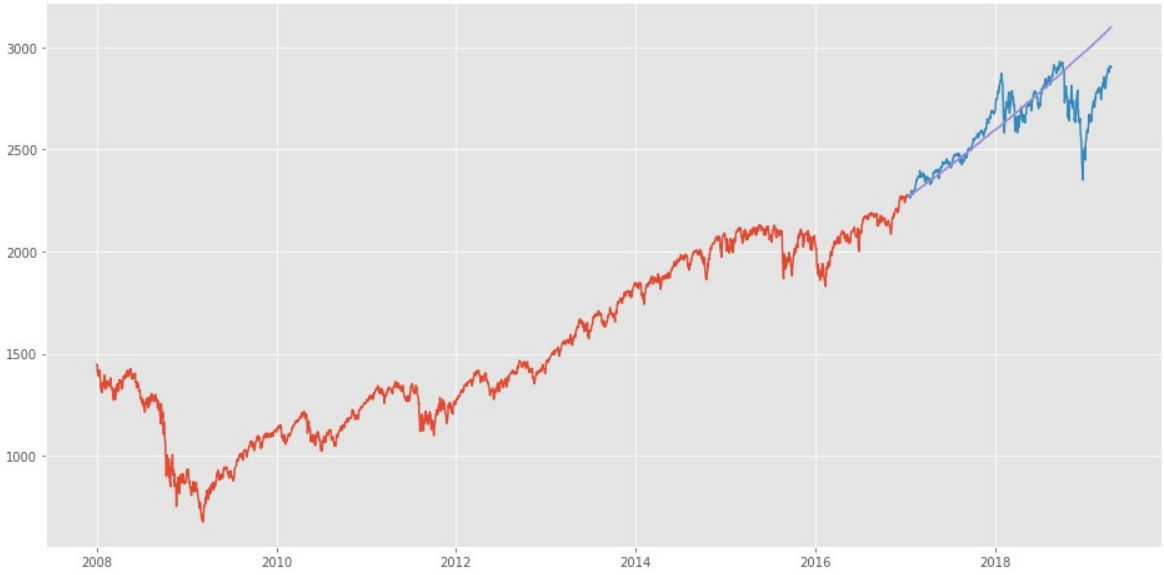
Introduction

ARIMA is a very popular statistical method for time series forecasting. ARIMA models take into account the past values to predict the future values. There are three important parameters in ARIMA:

- p (past values used for forecasting the next value)
- q (past forecast errors used to predict the future values)
- d (order of differencing)

Parameter tuning for ARIMA consumes a lot of time, so using auto ARIMA automatically selects the best combination of (p,q,d) that provides the least error.

```
numpy version: '1.16.2'
pmdarima version: '1.1.0'
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=19210.71
9, BIC=19239.339, Fit time=43.920 seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 12); AIC=20681.25
5, BIC=20692.703, Fit time=0.623 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 12); AIC=20035.31
6, BIC=20058.212, Fit time=28.039 seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=19213.80
4, BIC=19236.700, Fit time=22.085 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 1, 12); AIC=19211.85
8, BIC=19246.202, Fit time=48.456 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 0, 12); AIC=20561.78
7, BIC=20584.683, Fit time=19.264 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 2, 12); AIC=19211.90
4, BIC=19246.248, Fit time=161.274 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 2, 12); AIC=19214.30
7, BIC=19254.375, Fit time=222.436 seconds
Fit ARIMA: order=(2, 1, 1) seasonal_order=(0, 1, 1, 12); AIC=19217.57
0, BIC=19251.914, Fit time=93.027 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(0, 1, 1, 12); AIC=19214.81
5, BIC=19237.711, Fit time=21.214 seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=19212.21
1, BIC=19246.555, Fit time=76.623 seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 1, 12); AIC=19225.40
5, BIC=19242.577, Fit time=22.868 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 1, 12); AIC=19213.07
0, BIC=19253.139, Fit time=104.732 seconds
Total fit time: 864.590 seconds
The root mean square error is 149.4734994509333

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:27: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:28: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy


(149.4734994509333, 3098.415957538634)
```



Result

Auto ARIMA takes into account the AIC and BIC values generated (as you can see in the code) to determine the best combination of parameters. AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) values are estimators to compare models. The lower these values, the better is the model.

Auto ARIMA model uses past data to understand the pattern in the time series. Using these values, the model captured an increasing trend in the series. Although the predictions using this technique are far better than that of the previously implemented machine learning models, these predictions are still not close to the real values.

6.5 Machine Learning - Prophet

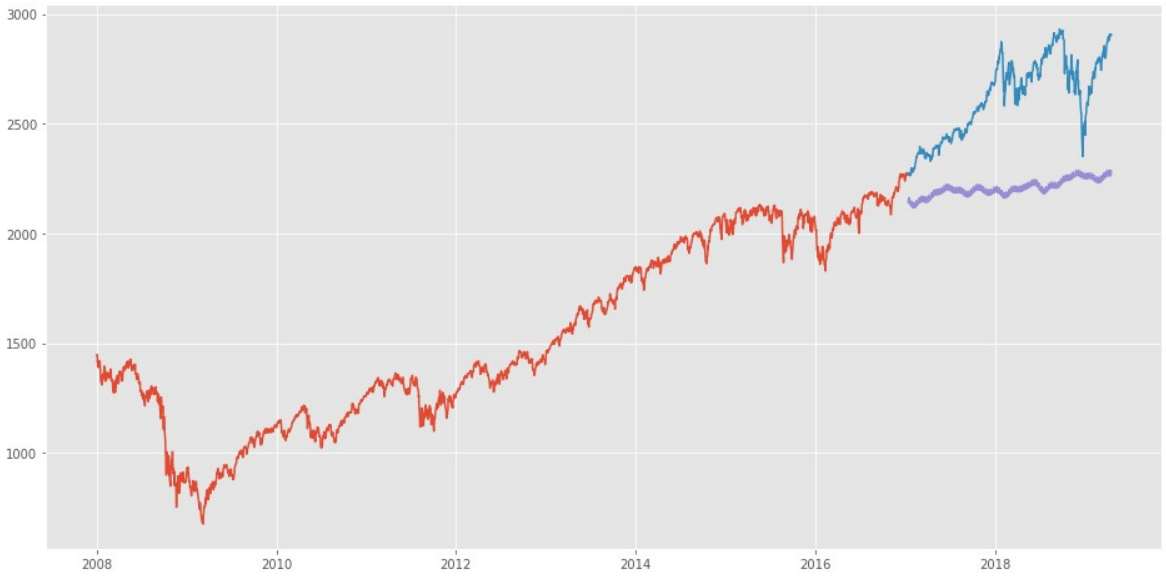
Introduction

There are a number of time series techniques that can be implemented on the stock prediction dataset, but most of these techniques require a lot of data preprocessing before fitting the model.

Prophet, designed and pioneered by Facebook, is a time series forecasting library that requires no data preprocessing and is extremely simple to implement. The input for Prophet is a dataframe with two columns: date and target (ds and y).

Prophet tries to capture the seasonality in the past data and works well when the dataset is large.

```
/home/jim/anaconda3/lib/python3.6/site-packages/fbprophet/forecaster.
py:880: FutureWarning:

Series.nonzero() is deprecated and will be removed in a future versio
n.Use Series.to_numpy().nonzero() instead

INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_se
asonality=True to override this.

The root mean square error is 446.4384233932304

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:30: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:31: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy


(446.4384233932304, 2285.3302977744615)
```



Result

Prophet (like most time series forecasting techniques) tries to capture the trend and seasonality from past data. This model usually performs well on time series datasets, but fails to live up to it's reputation in this case.

As it turns out, stock prices do not have a particular trend or seasonality. It highly depends on what is currently going on in the market and thus the prices rise and fall. Hence forecasting techniques like ARIMA, SARIMA and Prophet would not show good results for this particular problem.

6.6 Machine Learning - Long Short Term Memory (LSTM)

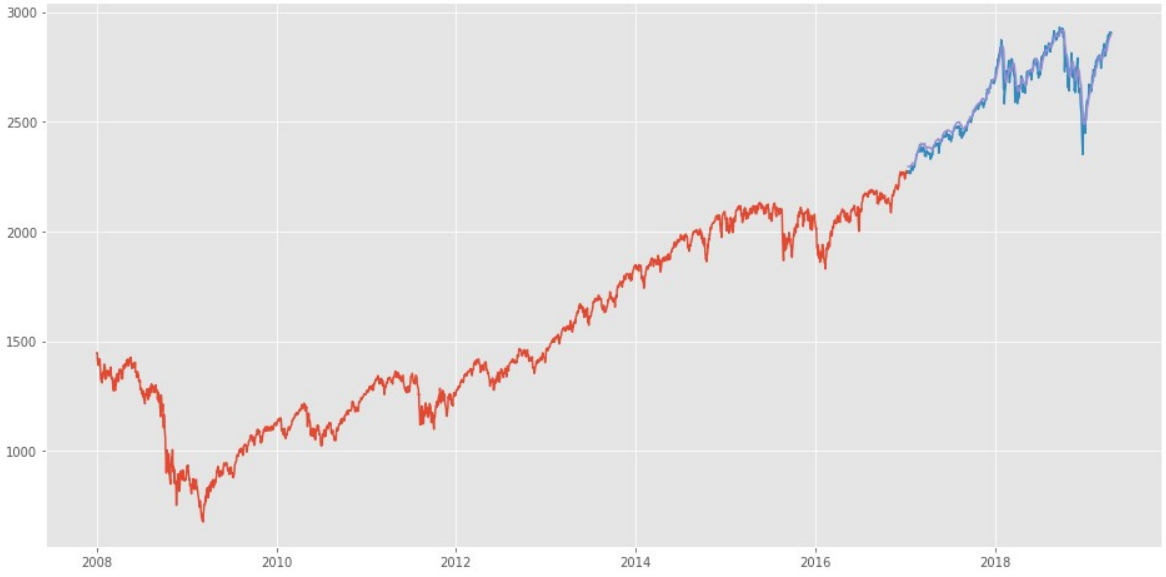
Introduction

LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is because LSTM is able to store past information that is important, and forget the information that is not.

LSTM has three gates:

1. The input gate: The input gate adds information to the cell state
2. The forget gate: It removes the information that is no longer required by the model
3. The output gate: Output Gate at LSTM selects the information to be shown as output

```
Epoch 1/1
 - 274s - loss: 8.2249e-04
The root mean square error is 36.88954759621163

/home/jim/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.p
y:62: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas
-docs/stable/indexing.html#indexing-view-versus-copy


(36.88954759621163, 2899.4375)
```



Result

The LSTM model can be tuned for various parameters such as changing the number of LSTM layers, adding dropout value or increasing the number of epochs.

Although the model seem to predict well with real data, but it is not enough to identify whether the stock price will increase or decrease. Stock price is affected by the news about the company and other factors like demonetization or merger/demerger of the companies.

There are certain intangible factors as well which can often be impossible to predict beforehand.

6.7 Model Comparsion

	RMSE	Prediction
ma	556.720265	2092.728205
linear	296.136189	2525.148140
knn	1142.990569	2266.327772
arima	149.473499	3098.415958
prophet	446.438423	2285.330298
lstm	36.889548	2899.437500

```
The last S&P 500 index is 2905.030029
```

```
To compare with the root mean square of error, the most lowest rmse o
f model is by using of Long Short Term Memory method, and the predict
ion value is 2899.4375 As we mentioned as last part, we should also c
onsider the news about the company and other factors like demonetizat
ion or merger/demerger of the companies.
```

7. Conclusion

From Part 1, we compare all the S&P500 stocks share and choose the top 30 highest correlation with index for comparsion from all 500 stocks share. After calculation and sort the correlation, the average value are around 0.8 - 0.9.

For the second part, we select the top 10 higest monthly return from above 30 stocks share by comparing the percentage of monthly return. These 10 stocks share selected will indicate the **higher monthly return for 10 years with higher correlation with S&P500 index, including Gartner Inc, Sherwin-Williams, Home Depot, The Cooper Companies, Roper Technologies, SBA Communications, Moody's Corp, Amphenol Corp, Fidelity National Information Services and Costco Wholesale Corp.**

The portfolio optimization has been used to generate the proportion of investment strategic, the stock with the least risk is COST at around 0.22, but the return is only around 0.16. If we will to take slightly more risk around 0.225 return, **we should consider to choose HD and SHW rather than IT with higher risk with portfolio optimisation**.

For the last part of machine learning, building the model for S&P 500 index has been implemented for the data of time series including Moving average, Linear regression, k-Nearest Neighbours, auto arima, Long Short Term Memory to generate the prediction and compare the root mean square error in order to get the less error model and most accuracy model. **From the result of LTSM with lowest rmse, the prediction index is to drop down from 2095 (present index) to 2899 (prediction value), so now is not a good investment time**.

As we mentioned above, it is not enough to identify whether the stock price will increase or decrease. Stock price is affected by the news about the company and other factors like demonetization or merger/demerger of the companies. Therefore, the sentiment analysis for financial news by NLTK should also be done to improve the factor of accuracy with trend of machine learning.