STOCK MARKET
MOVEMENT PREDICTION

Background

Both my academic and professional experience for the last few years mainly revolve around mathematics, statistics, and finance. I am always intrigued to combine the three fields together by creating an algorithm using mainly Python language that is able to predict any movement in stock prices. Through this program, I hope to be able to make decisions on which companies that I should buy or sell. I believe this can be a challenging task as no one has been able to come up with the right algorithm that is able to work consistently.

For this project, my goal is to find a company where it will at least beat a benchmark, the stock market (in this case the SP500) by 1% as well as find the dates where the algorithm will recommend a buy and sell based on different features. The features can be the number of volumes on a day, the exchange rates between USD and another foreign currency, the Federal interest rate, P/E Ratio of a company and many more. In this project, I will use the first three mentioned earlier. I also want to caution myself not having too many features to start due to the Curse of Dimensionality.

Problem Statement

When you enter the field of Finance, the first question that you will get asked is “Is it Art or Science?” That is the question that I hope I can answer with the algorithm that I am trying to build. If my algorithm, which is based on scientific deductions and formulas, is able to predict the future price of stock’s prices, then I can conclude that Finance is a field of Science. If an intuition or a bit of luck is still required to make my decision, then I want to say that Finance is a field of Art.

Even though this sounds rather simple, I believe that no algorithm is close to being able to predict the future of the stock prices. The big question remains on which techniques and statistical methods will be the most efficient to be implemented for this project. After some research, I came up with different statistical methods, such as Moving Average, Linear Regression, Auto ARIMA, Prophet (from Facebook), and Long Short Term Memory. I have also attended a meetup where the group is trying to use different methods to find the best algorithm for their personal trading hobby.

This trading algorithm can be useful for hedging companies to mitigate short term as well as long term risk, and for investment firms to invest in the right stocks to ensure that their clients’ money grow and able to cover the retirements that they have planned.

Data

In order to be able to start the analysis and determine if the algorithm works, I start with the data. I sourced my data using Alpha Vantage by leveraging their API functionality with importing the library TimeSeries from the alpha_vantage timeseries package. Their documentation can be found here https://www.alphavantage.co/documentation/.

The timeseries package is able to process the data directly to the pandas table. In order to have the copy of the data, I also saved the pulled data into my own drive.

I am fortunate that the data that I sourced from Alpha Vantage can be considered clean. Some of my conclusions are:

- There is no missing values
- There are clear and descriptive column names
- There is no outliers
- All data types have 'float64' data types
- The index is set to date

The data wrangling that I performed on this dataset are:

1. Renaming the column names
   a. from with numbering to without numbering (from "1. open" to "open")
2. Adding new column called "days_since"
   a. number of days between today and the earliest date of the data

As for this capstone project, the time period that I will be using as my training data is from January 1, 2000 to December 31, 2019 while the time period as my test data is from January 1, 2020 to March 9, 2020.

In addition to the company data above, I also pulled the SP500 closing price from the same period of time mentioned above for benchmark. The features that are used also are being downloaded manually since all finance data providers with API requires paid subscriptions. The data was processed using Microsoft Excel and was added to the company's data.

Exploratory Data Analysis (EDA)

The goal of the algorithm is to be able to find the future stock's price for all companies, but to start, I will use one company (Disney, ticker: DIS) to create the basic infrastructure. The algorithm will be updated moving forward to accommodate companies from different industries.

Disney was founded in 1923 and will be celebrating its 100th year anniversary in 2023. It was started by Walt Disney and his brother Roy Disney. As of Feb 3, 2020, its market value is at $255 billion with $141.26 per share price. Since its inception, Disney had two public offerings in 1940 and 1957.

In this EDA, the date range that I will be referring to is 20 years period from Dec 16, 1999 to Dec 17, 2019. It spans 5,033 trading days.

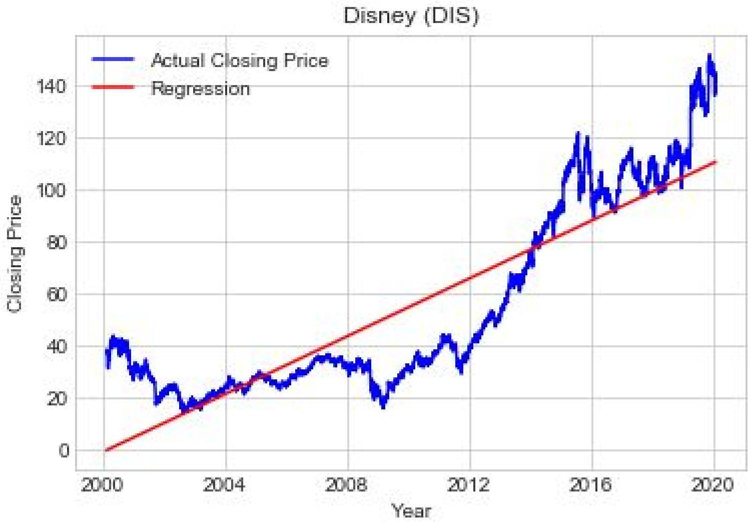
Pearson correlation

To see if any of the columns are correlated between each other, I checked their Pearson correlation leveraging the .corr command within the pandas library. Since all the columns are mainly different numbers within the same trading day, they are strongly positively correlated with each other.

Linear regression

Based on 20 years of data, I used the closing price of each trading day to find the regression line. By leveraging the polyfit function within the numpy library, I found the slope and intercept based with one degree of polynomial.

The graph below shows the stock prices in blue vs the linear regression line in red.



Looking at the graph, we can see that the regression line underfit the actual price in the first 4 years, overfit the actual price in the next 9 years, and underfit again in the last 7 years. I also calculated the Root Mean Square Error (RMSE) and got 17.69, which is quite high. It can be implied that the regression line does not fit well with the actual data. A better algorithm needs to be addressed.

Stock Performance

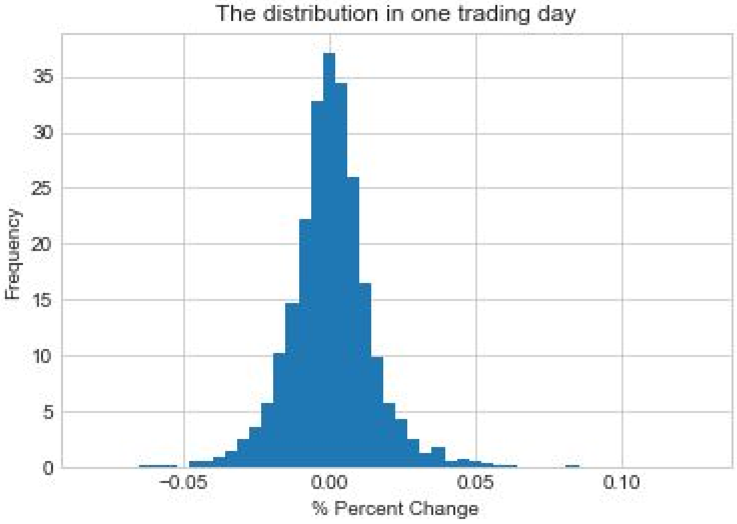
Over the last 20 years, Disney stock price has risen 1,001.23% or 50.06% per year. It means that if you invested $100 back in 1999, you will have $1,001.23 now without even considering the dividends paid during those years.

Given recent purchases of competitor studios like Fox and Marvel while ABC is also under their belt, it can be safely assumed that their performance will be good as well.

Open vs Close

We would like to check the volatility of the price within one trading day to see how erratic one trading day can be for Disney. In one day, the highest increase is 12.7% and the highest decrease is -16.23%.

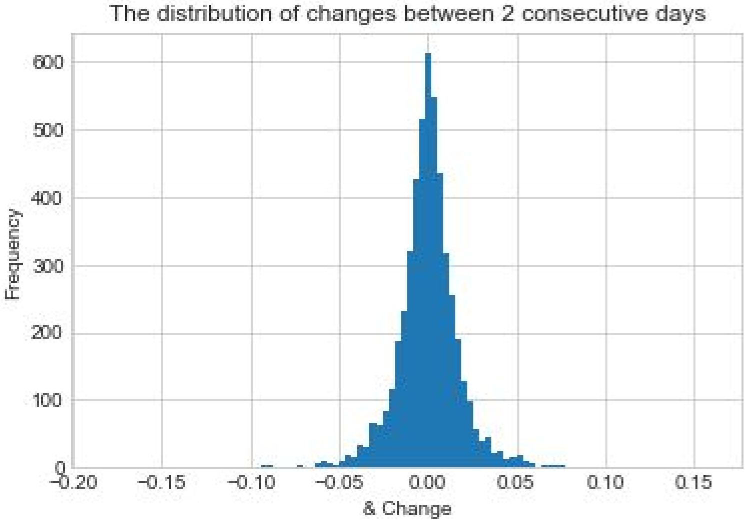
Below is the histogram to show the distribution of the change in one day. Visually, we can see that the distribution is close to normal. The actual mean of this distribution is 0.05% and its standard deviation is 1.52%.



Trading Days

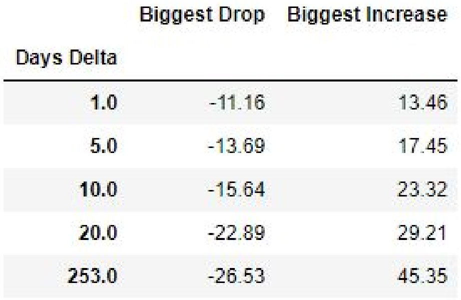
While we have seen the movement within one day, we also want to see the behaviour of the change between two consecutive trading days. Between two closing prices of consecutive trading days, the biggest jump is 15.97% while the biggest drop is -18.36%.

Below is the histogram to show the distribution of the change in two days. As with the distribution of change in one trading day, the distribution of change between two closing trading days is normal with mean of 0.04% and standard deviation of 1.83%.



Trading Days by Custom Period

As we have seen in the section above, we were looking for the changes between two trading days. Furthermore, I would like to extend the period to different periods. Below is the summary of the results.



	Biggest Drop	Biggest Increase
Days Delta		
1.0	-11.16	13.46
5.0	-13.69	17.45
10.0	-15.64	23.32
20.0	-22.89	29.21
253.0	-26.53	45.35

In any given period of time listed above, we can see what the biggest increase and drop are between the two periods of time. This could help us to determine if the company is recommended for short-term or long-term investing.

INFERENTIAL STATISTICS

In this section, we can use different methods to infer findings from our data. However, since my project involves time regression data, it is rather difficult to use any of the methods (Frequentist Inference, Bootstrap Inference and Bayesian Inference) to infer any findings from the data.

MACHINE LEARNING

Setup

As my data is time-dependent, I can not split into train and test set using the scikit-learn module. Instead, I use:

- Train data: dataset from the beginning of 2000 to the end of 2019
- Test data: dataset from the beginning of 2020 to present

Even though at the moment the test dataset is relatively small compared to the train dataset, it will be updated continuously.

In order to start, I use 3 different features (Volume, Effective Federal Interest Rates, and EUR/USD Exchange Rates), which can be expanded later. These features are assigned to the X_train variable.

The second variable that needs to be set up is the y_train variable. Since the goal of this algorithm is to beat the market (in this case the SP500), I copied a function from robertmartin8's github page.

```
def status_calc(stock, sp500, outperformance):
    """A simple function to classify whether a stock outperformed the S&P500
    :param stock: stock price
    :param sp500: S&P500 price
    :param outperformance: stock is classified 1 if stock price > S&P500 price +
    outperformance
    :return: true/false
    """
    if outperformance < 0:
        raise ValueError("outperformance must be positive")
    return stock - sp500 >= outperformance
```

The function checks if the stock performance on any given day is higher than the market on the same day by the outperformance constant (any number >0 where 1% is 0.01). Thus, setting up the y_train with the following line code.

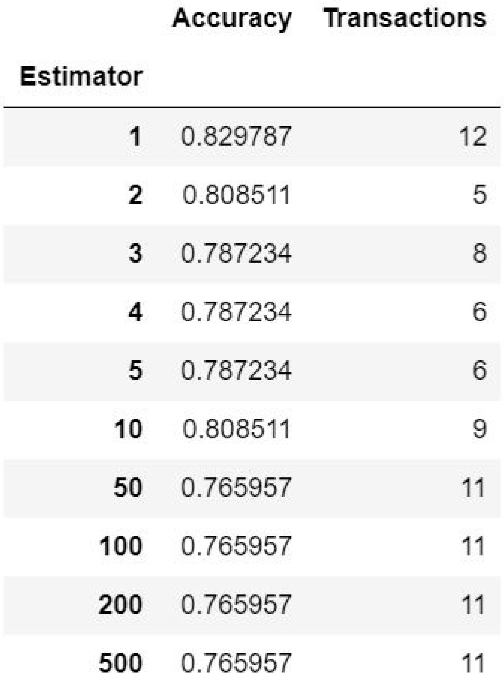
```
y_train = status_calc(disney_.Change, sp500_.Change, 0.01)
```

Method

For this project, I use the RandomForestClassifier method from scikit learn library. The challenge is to determine how many clusters are appropriate for the model. I created a function to test different n clusters to see which has the best accuracy score. The function does:

1. Initialize the RandomForestClassifier module
2. Fit the training data
3. Calculate the accuracy score
4. Predict the decision

Below are the results of the function with estimator of 1, 2, 3, 4, 5, 10, 50, 100, 200 and 500.



	Accuracy	Transactions
Estimator		
1	0.829787	12
2	0.808511	5
3	0.787234	8
4	0.787234	6
5	0.787234	6
10	0.808511	9
50	0.765957	11
100	0.765957	11
200	0.765957	11
500	0.765957	11

As we can see, the highest accuracy is the one with estimator 1, but it will overestimate the number of transactions which may lead to wrong decision to purchase the stock. 2 clusters will also give a good accuracy number, but it underestimates the number of transactions. In this case, however, underestimating is better than overestimating. In my opinion, the best method will be to use 10 clusters using the 3 features mentioned above. If additional features are added, I will then check again if 10 are still the best clusters.

Results

By using 10 clusters, we can see that in 2020 (47 trading days), the algorithm predicts that 9 of those days are good to buy Disney stock and sell it the next day. The dates are as follow:

1. January 14, 2020
2. February 4, 2020
3. February 24, 2020
4. February 25, 2020
5. March 2, 2020
6. March 3, 2020
7. March 5, 2020
8. March 6, 2020
9. March 10, 2020

CONCLUSION

This project is a good starting point for anyone who is willing to learn on how to use Machine Learning on a question that they have. I started with collecting the data, cleaning the data itself, performing Exploratory Data Analysis, and creating algorithms for Machine Learning.

Given the current condition of the economy, it is hard to determine if the algorithm is good enough to be used for production. Even though it is unfortunate, this new data will be a good training set because it has never happened before in the history of the stock market.

As of now, I can say that finance is a mixture between art and science. I am able to use scientific methods such as regression and classification to determine a decision based on different features that were set but to set up the algorithm itself, I can considered it to be an art in a sense that there is no right or wrong features to be chosen.

REFERENCE

https://github.com/robertmartin8/MachineLearningStocks