CS 584 Project

Aditya Varma Vetukuri
Computer Science
George Mason University
Fairfax, Virginia, USA
adityavarmavetukuri@gmail.com

Andrew Williams
Computer Science
George Mason University
Fairfax, Virginia, USA
mwilli90@gmu.edu

ABSTRACT

In this report we explore the use of machine learning and more traditional styles of prediction techniques for stock prices. Through data pre-processing, analysis, post-processing, and modeling a technical analysis is conducted and feasible approach for predicting change in prices given a five-day window of information of an undisclosed stock. With a variety of methods to include Moving Average, Linear Regression, k-Nearest Neighbors, Auto ARIMA, Prophet, and Long Short-Term Memory, can machine learning be used as a game changer in the domain of predicting stocks. In the end the report will show the contrary in favor of linear regression as a more traditional approach for predicting future returns.

1 INTRODUCTION

Predicting how a stock will perform can be one of the most difficult tasks because there are so many factors involved in the prediction including physical vs physiological factors, rational and irrational behavior, global/national/regional socioeconomic factors, etc. Finding the hidden sequence or signal in Megabytes of noisy, non-stationary data is like searching for a needle in the proverbial haystack. Additional challenges include predicting intra and end of day returns without being deceived by all of the noise from historical stock performance and the inevitability of masked features.

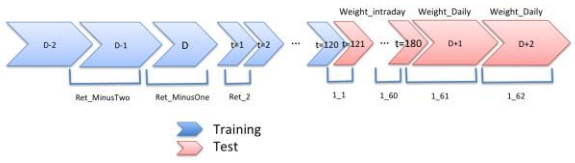

Figure 1: Individual Stock 5-day window

The dataset for this report includes 5-day windows of time, D-2, D-1, D, D+1, D+2 for thousands of individual and arbitrary stocks as shown in figure 1. This set of data provides D-2, D-1, and part of day D with the task of predicting the returns for the rest of day D, D+1, and D+2. During day D, there is an intraday return, which are the returns at different points in the day. The dataset includes 180 minutes of data, from t=1 to t=180. The training set includes the full 180 minutes, whereas the test set contains only 120 minutes. Additionally, each 5-day window includes 25 features: Feature_1 to Feature_25.

Broadly, stock market analysis is divided into two parts—Fundamental Analysis and Technical Analysis. Fundamental Analysis involves analyzing a company's future profitability on the basis of its current business environment and financial performance. Technical Analysis, on the other hand, includes reading the charges and using statistical figures to identify the trends in the stock market. Using primarily technical analysis we evaluate the dataset and determine the best approach for predicting D, D+1, and D+2 for each stock.

2 RELATED WORK

2.1 Moving Average

An attempt was made to use moving Average due to it being easily one of the most common things we use in our daily lives. Example, 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 are all routine tasks we do on a regular basis and the starting point for our analysis and experimentation.

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 used the moving average technique which uses the latest 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. Figure 2 is a simple example to better illustrate this concept with more clarity.

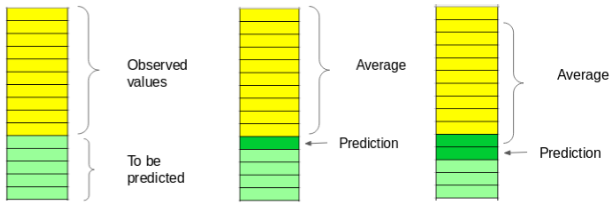

Figure 2: Moving Average

Implementing the moving average technique involved creating a dataframe consisting of only the columns Ret_2 through Ret_180, then split the train and validation sets to verify our predictions.

2.2 k-Nearest Neighbors

Another approach was to use a kNN based on the independent variables in order to find the similarity between new data points and old data points [1]. However as is seen in the data analysis section there was no significant improvement in RSME due to the dependence of features on Feature_7.

2.3 Auto ARIMA

ARIMA is a very popular statistical method for time series forecasting [2]. ARIMA models take into account the past values to predict the future values. Three important parameters include:

- P (pas 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 significant amount of time which can be reduced by setting auto ARIMA which automatically selects the best combination of (p, q, d) that provides the least error. With ARIMA using past data to understand the pattern in time series the model captures an increasing trend in the series [3]. Although these predictions using this technique are better than many implementations of machine learning models, these predictions are still not usually close to the real values. As is normally the case, the model will capture trends in the series but not focus on the seasonal part.

2.4 Prophet

There are a number of time series techniques that can be implemented on a stock market dataset, but most of these techniques require large amount 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 [4]. The input for Prophet is a dataframe with two columns: date and target (ds and y). With this information Prophet attempts to capture the seasonality in the past data and works well when the dataset is large. With this approach however Prophet (like most time series forecasting techniques) fails to live up to its reputation. This is due to stock prices not having a particular trend or seasonality. Instead it highly depends on what is currently going on in the market and therefore the prices rise and fall. As a result forecasting techniques like ARIMA, SARIMA, and Prophet will not show reliable results for stocks in general and especially with the dataset of this report [5].

2.5 Long Short Term Memory (LSTM)

LSTMs are widely used for sequence prediction problems and have been proven to be extremely effective. The reason they work so well is because LSTM is able to store past important information and forget information that is irrelevant [6]. With this in mind LSTM has three gates:

- The input gate: The input gate adds information to the cell state
- The forget gate: It removes the information no longer required by the model
- The output gate: Output Gate at LSTM selects information to be shown as output

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 [7]. As useful as this model may seem there is a key element from the introduction that cannot be ignored. Stock prices are affected by the news about the company and other factors like demonetization or mergers/demergers of a company. These intangible and not easily quantifiable measures which can make it impossible to predict beforehand.

3 SOLUTION

Considering all approaches and techniques researched what led to the lowest WMAE and RMSE was the simplest of machine learning algorithms, linear regression. The linear regression model determines the relationship between independent variables and the dependent variable. The equation can be written as follows:

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

Here, $X_1, X_2, \ldots X_n$ represent the independent variables while the coefficients $\theta_1, \theta_2, \ldots \theta_n$ represent the weights. As will be seen in the next section though linear regression is a simple model with a few obvious disadvantages with thorough analysis and preprocessing we mitigate overfitting.

4 EXPERIMENTS

4.1 Data

The data used for these experiments and report consists of two files train.csv and test.csv for a total of 444 Megabytes. Train.csv (https://cs584.s3-us-west-2.amazonaws.com/train.csv) is comprised of 40,000 instances (40,000 unique stocks) with 211 columns:

- Feature_1 – Feature_25
- Ret_MinusTwo, Ret_MinusOne
- Ret_2 – Ret_120
- Ret_121 – Ret_180: target variables
- Ret_PlusOne, Ret_PlusTwo: target variables
- Weight_Intraday, Weight_Daily

Test.csv (https://cs584.s3-us-west-2.amazonaws.com/test.csv) is comprised of 120,000 instances with 147 columns as follows:

- Feature_1 – Feature_25
- Ret_MinusTwo, Ret_MinusOne
- Ret_2 – Ret_120

4.2 Experimental Setup

Training and test data was prepared with the use of Pandas dataframes for data processing throughout the analysis, experimentation, and finally for the solution. Feature_7 is used to sort the data as will become apparent later on in this report. Aggregation of the intraday returns Ret_2 Ret_121 and saved as a new feature called Ret_Agg and the target labels are grouped into a separate variable reducing the training and test sets to 30 columns.

The Performance metrics used throughout the experiments were the Root Mean Squared Error (RMSE) method and Mean

Absolute Percentage Error (MAPE) and the baseline is the lower the value the better the prediction.

4.3 Analysis and Experimental Results

4.3.1 Analysis Constraints: The intraday one min data was found to be too noisy to be predicted by the data given in both the training and tests sets and therefore our focus was mainly on predicting Ret_PlusOne and Ret_PlusTwo, meaning you will only see outputs predictions for {id}_61 and {id}_62 and output for all of {id}(1-60).

4.3.2 Data Analysis: For initial data analysis each feature is observed for the number of missing values, how many values are unique, how imbalanced the values are, how many potential outliers are contained in the values and if values in the training and test sets are disjoint. For the purposes of this report potential outliers are values which differ from the mean by more than 3 standard deviations.

Feature	Missing	Missing %	Unique	Unique %
Feature_1	33313	83.2825	11	0.0275
Feature_2	9146	22.865	30855	77.1375
Feature_3	1237	3.0925	38764	96.91
Feature_4	7721	19.3025	32280	80.7
Feature_5	0	0	10	0.025
Feature_6	1933	4.8325	38068	95.17
Feature_7	0	0	824	2.06
Feature_8	469	1.1725	33	0.0825
Feature_9	1875	4.6875	37	0.0925
Feature_10	19471	48.6775	7	0.0175
Feature_11	987	2.4675	39014	97.535
Feature_12	1096	2.74	102	0.255
Feature_13	594	1.485	11	0.0275
Feature_14	728	1.82	39273	98.1825
Feature_15	2141	5.3525	921	2.3025
Feature_16	610	1.525	3	0.0075
Feature_17	646	1.615	39355	98.3875
Feature_18	568	1.42	39433	98.5825
Feature_19	1190	2.975	38811	97.0275
Feature_20	7826	19.565	10	0.025
Feature_21	1018	2.545	38983	97.4575
Feature_22	1345	3.3625	38656	96.64
Feature_23	1711	4.2775	38290	95.725
Feature_24	726	1.815	39275	98.1875
Feature_25	655	1.6375	39346	98.365
Ret_MinusTwo	0	0	40000	100
Ret_MinusOne	0	0	40000	100
Ret_Agg	0	0	40000	100
Ret_Agg_Std	0	0	40000	100
Ret_Std	0	0	40000	100
Ret_PlusOne	0	0	40000	100
Ret_PlusTwo	0	0	40000	100

Table 1: Stock.py analysis1 (Feature Analysis)

Feature	Imbalance	Imbalance %	Outlier	Outlier %	Disjoint
Feature_1	2651	6.6275	0	0	False
Feature_2	1	0.0025	62	0.155	True
Feature_3	1	0.0025	269	0.6725	True
Feature_4	1	0.0025	80	0.2	True
Feature_5	6943	17.3575	0	0	False
Feature_6	1	0.0025	1018	2.545	True
Feature_7	114	0.285	0	0	True
Feature_8	4178	10.445	0	0	False
Feature_9	5863	14.6575	237	0.5925	False
Feature_10	14437	36.0925	778	1.945	False
Feature_11	1	0.0025	605	1.5125	True
Feature_12	1596	3.99	0	0	False
Feature_13	4715	11.7875	0	0	False
Feature_14	1	0.0025	552	1.38	True
Feature_15	61	0.1525	888	2.22	False
Feature_16	39100	97.75	290	0.725	False
Feature_17	1	0.0025	486	1.215	True
Feature_18	1	0.0025	894	2.235	True
Feature_19	1	0.0025	15	0.0375	True
Feature_20	7008	17.52	0	0	False
Feature_21	1	0.0025	950	2.375	True
Feature_22	1	0.0025	118	0.295	True
Feature_23	1	0.0025	1434	3.585	True
Feature_24	1	0.0025	798	1.995	True
Feature_25	1	0.0025	1527	3.8175	True
Ret_MinusTwo	1	0.0025	635	1.5875	True
Ret_MinusOne	1	0.0025	689	1.7225	False
Ret_Agg	1	0.0025	673	1.6825	True
Ret_Agg_Std	1	0.0025	611	1.5275	True
Ret_Std	1	0.0025	684	1.71	True
Ret_PlusOne	1	0.0025	625	1.5625	NaN
Ret_PlusTwo	1	0.0025	655	1.6375	NaN

Table 2: Stock.py analysis1 (Feature Analysis continued)

4.3.2.1 Analysis 1: We can infer from the above output of Stock.py method analysis1 that Feature_1 has a very high number of missing values. As a result of dropping this feature the model performance improved as seen later on in this report. Feature_1, Feature_5, Feature_7, Feature_8, Feature_9, Feature_12, Feature_13, Feature_16, and Feature_20 contains few unique values. These features were as a result considered as categorical in nature. Feature_15 his highly imbalanced. Besides being categorical, Feature_6 and Feature_7 were also distinct between the training and test data sets. However, Feature_6 is highly unique, whereas Feature_7 only contains relatively few unique values. This let Feature_7 as a special feature. In an attempt to better understand how this feature could help in predictions we investigated this feature more thoroughly by grouping returns by values of Feature_7 and looked for a relationship in the sign of the returns to see if equal values of Feature_7 had correlated returns.

4.3.2.2 Analysis 2: Observing the output from Stock.py method analysis2 we see a frequency of return signs: 0.9886 and determine that the sign of the returns within each group is almost 100% correlated. Regardless of what the nature of Feature_7 is, we account for this relationship when doing model cross-validation to avoid data leakage. Our assumption is that Feature_7 likely has a relationship to time, which explains why returns within this group are correlated.

4.3.3 Visual Data Analysis: In order to get a better understanding of the properties of the different features we plot the features mutual correlations, the distributions of the features and regression plots for the features to see how each relate to the targets (for sake of readability only Ret_PlusOne in the regression plots). As a compromise in computation time only 10% of the sample data is provided in the below plots.

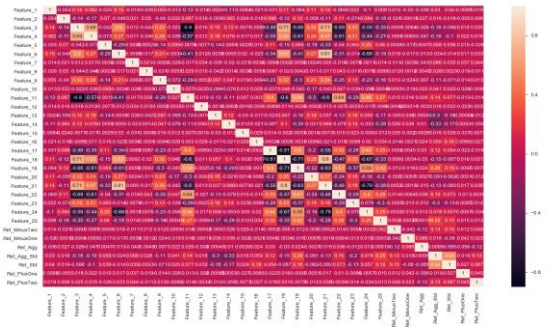

Figure 2: Correlation Heatmap (All features)[1]

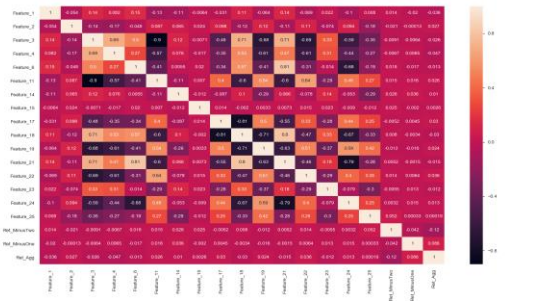

Figure 3: Correlation Heatmap (Numerical features)[2]

Figure 4: Correlation Heatmap (Categorical features)[3]

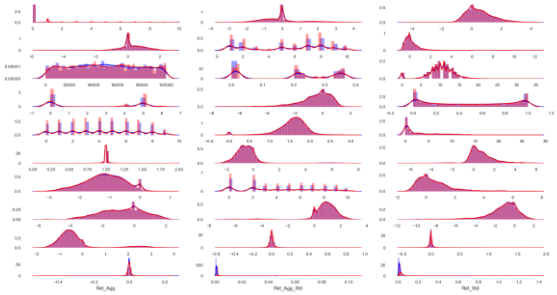

Figure 5: Raw Distributions (All features)[4]

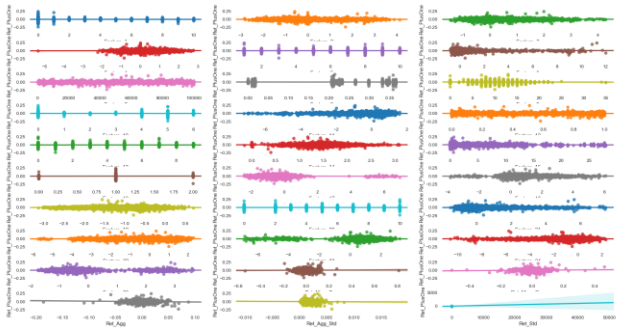

Figure 6: Regression Plots[5]

4.3.4 Analysis Conclusions: Observations of the visual data reveal that when attempting to predict stock returns each feature shows very little correlation with the targets (apparent from both the correlation heatmap and the regression plots). Further observation shows a high correlation between some of the numerical features and in particular there are three distinct clusters of correlation:

- [Feature_3 – Feature_11] correlates with [Feature_3 – Feature_11]
- [Feature_3 – Feature_11] correlates with [Feature_17 – Feature_25]
- [Feaure_17 – Feature_25] correlates with [Feature_17 – Feature_25]

Within the numerical features this is not coincidental, and utilizing these relationships resulted in the most optimal solution. It was also observed that most of the feature distributions did not appear to be gaussian. Finally, it was observed that Feature_13 had relatively smooth distribution due to this feature containing ordered information.

4.3.4 Preprocessing

4.3.4.1 Feature Selection: Using a robust preprocessing and modeling approach all features are included. Based on the previously mentioned analysis the features are split into numerical and categorical with categorical further split into ordered (Feature_13) and unordered.

4.3.4.2 Data Preprocessing: In this part of the experiment we define the pipelines for preprocessing the input features and targets. Both numerical and categorical features are imputed using a constant as a more conservative option. Numerical features are scaled using quartile range = [5, 95] as this approach is robust to outliers. The distribution is then converted to gaussian using a quartile transformation. This is justified by the tendency of linear estimators to perform better on gaussian distributions. Values which lie more than 3 standard deviations from the mean are removed in order to mitigate skewing the results.

Ordered categorical features are encoded using OrdinalEncoder to preserve the ordering. Correlations between features using PCA whitening followed by one-hot encode is used for unordered categorical features so that models could be used in the model building step which does work natively with categorical features.

For target returns we also use quantile transformation to force the distribution of returns (which are not normally gaussian) into a

Full size images accessible by clicking each image.
Below Stock.py methods correspond to each plot:
1 – pre_plot1()
2 – pre_plot2()
3 – pre_plot3()
4 – pre_plot4()
5 – pre_plot5()

gaussian distribution. Log transformation was considered however quantile transformation yielded a better model performance. Note: It was expected that PCA whitening of the numerical features would be beneficial for model performance due to many of the numerical features being highly correlated, however, the opposite was true. It may be that the correlation clusters apparent in the correlation heatmap contains relevant and useful information that we felt could be further investigated with future experiments. Finally, for data preprocessing we normalize all features using I2-norm which improved I2-regularized regression significantly.

4.3.4.3 Visualize Data Preprocessing: For the visualizing data preprocessing step of the experiment we looked at the effects of preprocessing the features. As with before only 10% of the data is used due reduce the computational complexity and time. With the below plots we are able to determine how much the transformation affects the categorical values by reducing the dimension of the output using PCA (we use TruncatedSVD, which is suitable for sparse input).

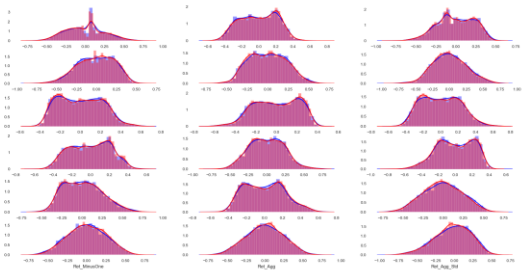

Figure 7: Distributions (Numerical Features)[1]

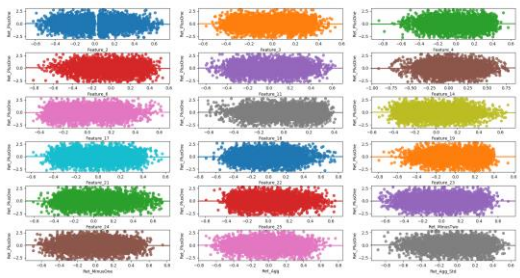

Figure 8: Regression Plots (Numerical Features)[2]

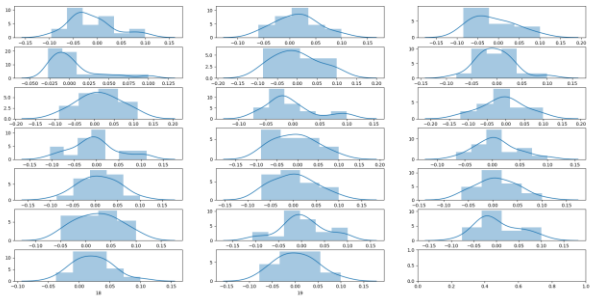

Figure 9: Distributions (Categorical Features)[3]

Full size images accessible by clicking each image.
Below Stock.py methods correspond to each plot:
1 – post_plot1()
2 – post_plot2()
3 – post_plot3()
4 – post_plot4()
5 – post_plot5()

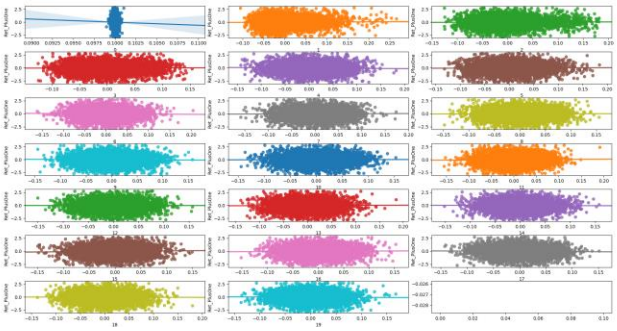

Figure 10: Regression Plots (Categorical Features)[4]

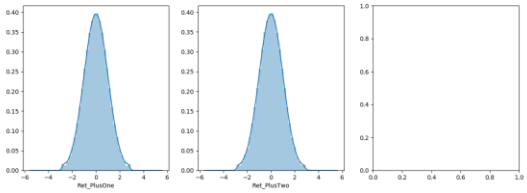

Figure 11: Distributions (target values)[5]

4.3.4.3 Data Postprocessing and Conclusion: From the above plots it is observed that most distributions of the numerical features as well as the targets now appear smooth and more gaussian-like. It is also observed that some features have retained their imbalance. In order to avoid losing other valuable data points we decide to refrain from removing more of the imbalanced rows. Though nothing significant shows from the transposed output it is important to note that no oddities standout that would prevent moving to the next step of building the model.

4.3.5 Modelling

4.3.5.1 Building Model: LinearSVR regression with L2 regularization was as the ideal candidate due to performing particularly well with data consisting of a low signal to noise ratio which is conventional for financial data. As a relatively fast algorithm a grid search with 5-fold GroupKFold cross-validation is used. As stated previously, due to returns reliance or lack of independence on Feature_7 cross-validations are grouped in order to avoid data leakage leading to overestimation of the CV performance. It is possible that use of a loss function would be suitable for optimizing WMAE (which is non differential at 0) however due to time constraints this will have to be further investigated in a future report. In either case reasonable results were gained without further optimizations.

4.3.6 Experimental Results

4.3.6.1 Model Evaluation: The model was evaluated using WMAE of the model prediction and compare this to the baseline model of just predicting the returns as the mean of the returns in the training data.

4.3.6.2 Model Results: From evaluating the model it is observed that its results perform significantly better than the baseline model.

5 ANALYSIS AND CONCLUSION

Despite producing a relatively favorable result in terms of a lower WMAE a greater amount of consideration must be factored in order to build a profitable trading model. Being able to beat a baseline model does not guarantee the model can be exploited due to market friction (trade costs, trade impact or overnight position costs) which may adversely impact the model performance in a real trading setup. Ideally, we would need to design a trading system using the model and optimize a trading metric such as Sharp Ratio. We would also need to test such a system by successively retraining and retesting this result on a given set of time windows moving forward in time. Moreover, we would need an adequate portfolio optimization and risk management system composed of each stock (or company's) financial data information about each stock's market of interest.

6 CONTRIBUTIONS

	Code	**Report**	**Presentation**
Andrew	50%	50%	50%
Aditya	50%	50%	50%
Total	100%	100%	100%

REFERENCES

[1] Teja Kodali. 2015. Using kNN Classifier to Predict Whether the Price of Stock Will Increase. (October 2001). Retrieved March 20, 2020 from https://datascienceplus.com/knn-classifier-to-predict-price-of-stock/

[2] Aishwarya Singh. 2018. Build High Performance Time Series Models using Auto ARIMA in Python and R. (August 2018). Retrieved March 23, 2020 from https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series-modeling-python-r/

[3] Pier Paolo Ippolito. 2019. Stock Market Analysis Using ARIMA. (May 2019). Retrieved May 1, 2020 from https://towardsdatascience.com/stock-market-analysis-using-arima-8731ded2447a

[4] Ankit Choudhary. 2018. Generate Quick and Accurate Time Series Forecasts using Facebook's Prophet. (May 2018). Retrieved April 15, 2020 from https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/

[5] Rakshit Ratan. 2019. Stock Price Prediction with the help of python and fbprophet. (August 2019). Retrieved April 23, 2020 from https://medium.com/datadriveninvestor/stock-price-prediction-with-the-help-of-python-and-fbprophet-prophet-library-part-1-3-f55ebff0b624

[6] Pranjal Srivastava. 2017. Essentials of Deep Learning: Introduction to Long Term Short Term Memory. (December 2017). Retrieved April 2, 2020 from https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

[7] Jiayu Qui. 2020. Forecasting stock prices with long-short term memory neural network based on attention mechanism. (January 2020). Retrieved April 7, 2020 from https://doi.org/10.1371/journal.pone.0227222