2020

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Predicting Stock Market from Financial Online News with LSTM

Instructor: Papageorgiou Haris

MSc in Business Analytics

Academic Year 2019-2020

Course: Machine Learning and Content Analytics

Table of Contents

1.Introduction

Description

News media has a tremendous effect on the stock market prices. Financial news articles are read every day by millions, interpreted by analysts, and analyzed by automated stock algorithms at quantitative trading firms. Key news and data about company trades, revenues, market climates, and geopolitical conflicts strongly influence trading and market transactions. In our project, we address the problem of predicting stock price trends using news headlines. We want to determine the extent to which we can predict the movement of stock prices using features extracted from online news.

Related Work Stock market prediction is an area of very active research. Attempts have been made to predict stock prices using technical information such as company technical and price history recent work has begun to incorporate natural language text data into prediction – for example, using full-text of news articles hidden Markov model incorporating news headlines and sentiment in social media . Most of these works has worked with short term price movements (on the order of minutes), while we will focus on end of day stock prices; we find that a longer time horizon is less susceptible to noise, and more beneficial to the common investor.

Mission

As financial institutions begin to embrace artificial intelligence, machine learning is increasingly utilized to help make trading decisions. Although there is an abundance of stock data for machine learning models to train on, a high noise to signal ratio and the multitude of factors that affect stock prices are among the several reasons that predicting the market is difficult. At the same time, these models don't need to reach high levels of accuracy because even 60% accuracy can deliver solid returns. One method for predicting stock prices is using a long short-term memory neural network (LSTM) for times series forecasting. An approach with the accuracy described above would be a very helpful toolset on hands of investors and especially in trading Companies. Knowing the next day's stock price, the investor/trader would be able to know where to put his/her money. Gradually with an accuracy more than 50% this business workflow would be profitable.

Multiple implementations have been proposed in the near past, most of them are mentioned in our bibliography. Many researchers have used Neural Networks to predict stock prices, along with Twitter posts about the Stock or online news from different sources. In this project we have tried to apply a 360^{0} approach of predicting Open Stock Price of the biggest technology companies (Apple and Microsoft) and try to extract the best model in terms of predicting ability and performance.

For solving the above business scenario we suggest the following methodology: We have used Neural Networks for predicting Apple and Microsoft Stock Prices, feeding our LSTM Network both with the sentiment of Financial News referring the stock and the previous days actual Open price. We suggest the following models to combat the above project:

- LSTM with Historical Stock Prices and TextBloB Sentiment Score Analysis
- LSTM with Finbert Sentiment Analysis
- LSTM with Finbert Sentence Embeddings
- Finally, we compare the above implementation with ARIMA and make some predictions

Data Sources

Our data came from 2 data sources. The online news dataset containing titles and headlines about stock news is located publicly on Kaggle(https://www.kaggle.com/gennadiyr/us-equities-news-data). The stock prices of the two companies we have analyzed (Apple and Microsoft) came from Yahoo Finance public API. Our Datasets of stock price seems to be imbalanced since stock prices of both companies are growing gradually over the last years.

Exploratory Data Analysis

At a first glance, we conducted some exploratory data analysis to see how out data are distributed and take a first look of the datasets we were going to use:

	Open	High	Low	Close	Adj Close	Volume
count	2428.000000	2428.000000	2428.000000	2428.000000	2428.000000	2.428000e+03
mean	29.069151	29.328832	28.809544	29.079849	27.325662	2.806688e+08
std	14.768329	14.903127	14.661120	14.797974	15.133855	2.194149e+08
min	6.478929	6.612857	6.453571	6.459286	5.581721	4.544800e+07
25%	18.008660	18.137589	17.842768	17.952144	15.900101	1.194640e+08
50%	26.181249	26.445000	25.988751	26.273750	24.317713	2.046454e+08
75%	38.966249	39.278126	38.585001	38.961249	37.482624	3.833529e+08
max	81.112503	81.962502	80.837502	81.800003	81.432350	1.880998e+09

Figure 1: stats of Apple features

	Open	High	Low	Close	Adj Close	Volume
count	2769.000000	2769.000000	2769.000000	2769.000000	2769.000000	2.769000e+03
mean	63.963868	64.561322	63.333496	63.983063	59.910382	4.037384e+07
std	45.953716	46.474798	45.373649	45.957649	47.325218	2.307689e+07
min	23.090000	23.320000	22.730000	23.010000	18.133703	7.425600e+06
25%	29.660000	29.889999	29.420000	29.709999	24.187122	2.483380e+07
50%	45.930000	46.340000	45.570000	45.939999	41.130180	3.473280e+07
75%	85.430000	86.000000	84.879997	85.500000	82.166878	5.035250e+07
max	229.270004	232.860001	227.350006	231.649994	231.649994	3.193179e+08

Figure 2: stats of Microsoft features

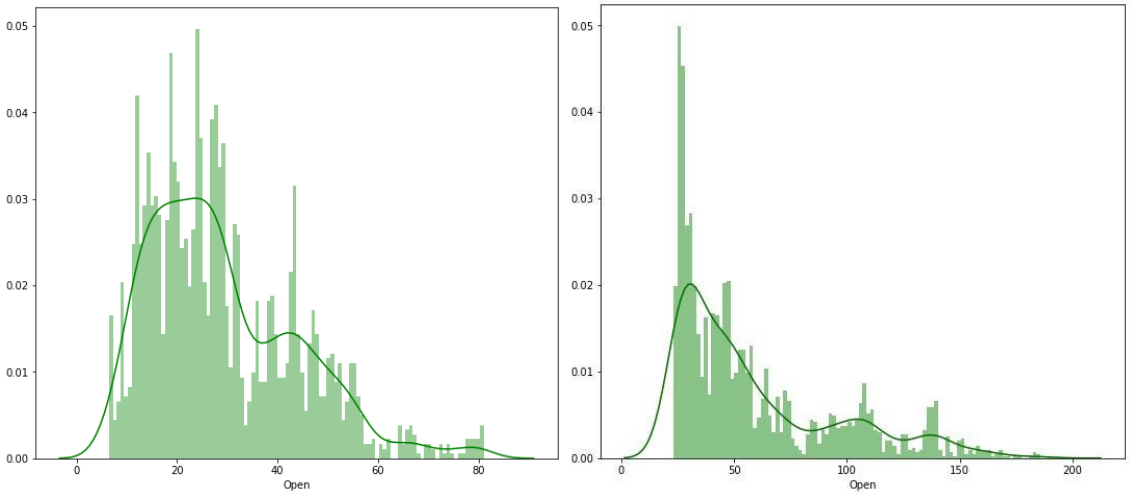


Figure 3,4: distribution of Open Price (left: Apple, right: Microsoft)

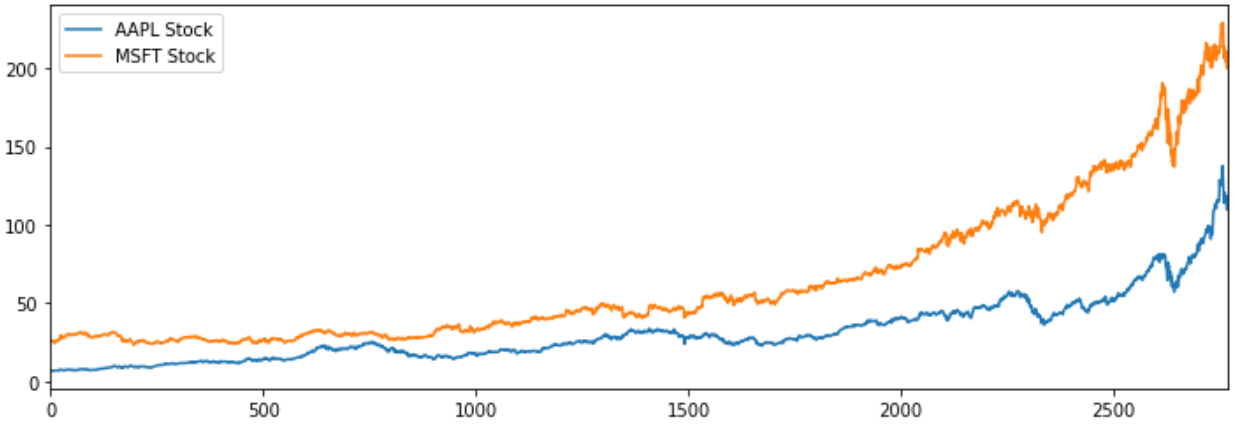


Figure 5: timeseries of both stock Prices

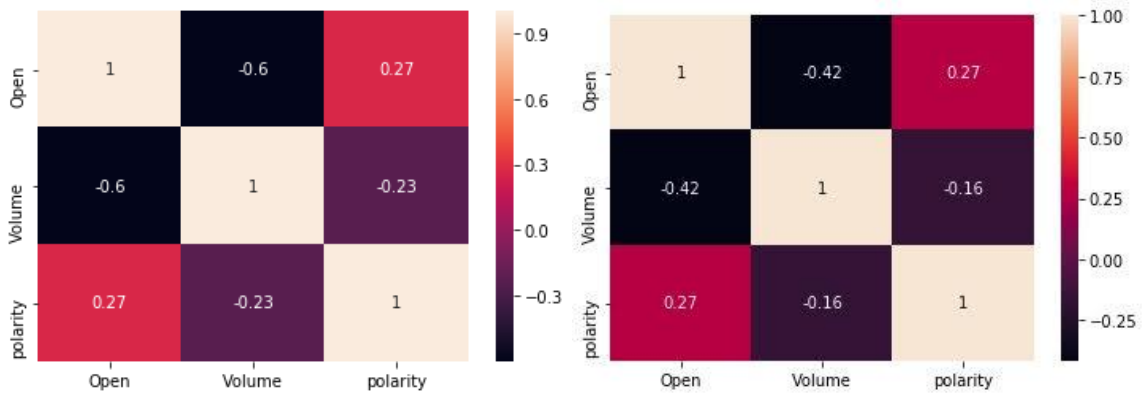


Figure 5,6: Correlation between Open Price and Sentiment Polarity(left: Apple, right: Microsoft)

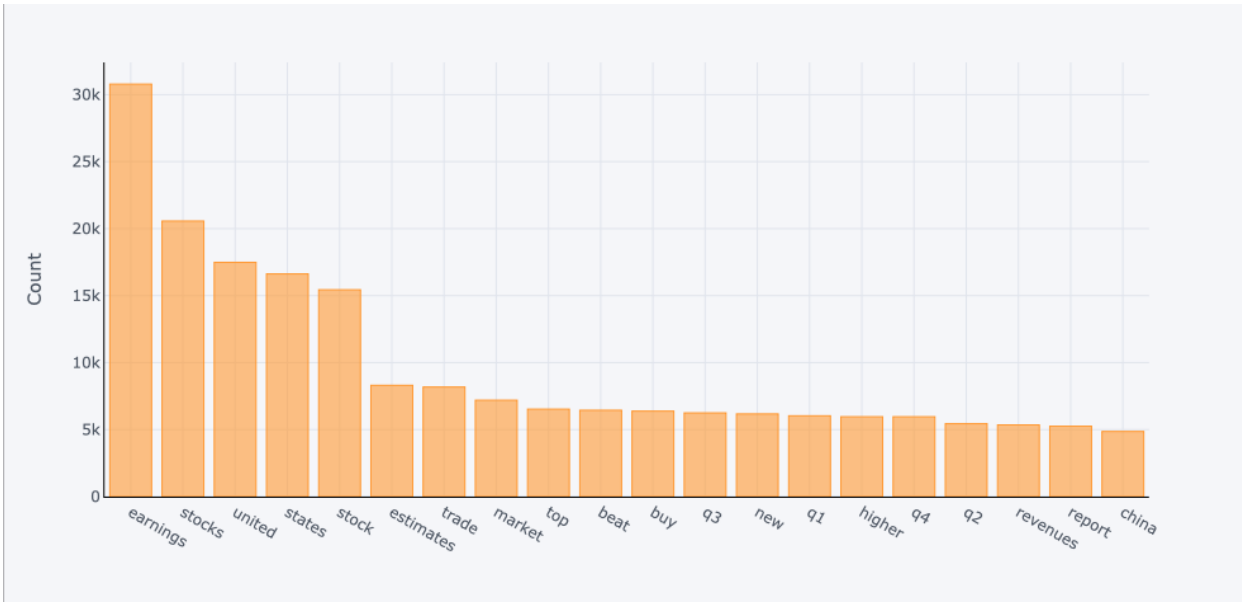


Figure 6: Top 20 Most common words in news titles.

Data Normalization

Normalization is changing the values of numeric columns in the dataset to a common scale, which helps the performance of our model. To scale the training dataset, we use Scikit-Learn's MinMaxScaler with numbers between zero and one.

Implementation 1: LSTM with Historical Stock Prices and TextBloB Sentiment Score Analysis.

Data Preparation

For this particular implementation, we have used both previous years stock prices (all the indicators) and sentiment analysis polarity score comes from the library *Textblob*[1]. In order the data to have the same shape we have created a dataset contains dates which include both stock prices values and sentiment extracted from news titles. For each date we join all titles of news together, we apply cleaning on text and keeping only english stopwords and finally we apply on them Textblob Sentiment Analysis Library. The output is two new columns on the final dataframe, named "*polarity*" having the score and "*subjectivity*". We keep only these dates having both for stocks and news data. The range of data are from 2009-09-21 to 2020-02-13.

release_date	title	Date	Open	High	Low	Close	Adj Close	Volume	polarity	subjectivity
2009-09-21	UPDATE 4 China buys Noble stake seeks commodit...	2009-09-21	6.581786	6.612857	6.486429	6.572143	5.679245	437715600	0.056597	0.380208
2009-09-22	US STOCKS Wall St open commods rise weak dolla...	2009-09-22	6.613929	6.620714	6.530357	6.588572	5.693443	356753600	0.018947	0.319649
2009-09-23	Nikkei climbs U S economy optimism Fed,Nikkei ...	2009-09-23	6.621428	6.746428	6.608214	6.625000	5.724923	593563600	0.042859	0.348008
2009-09-24	Nikkei gains 1 2 pct holiday JAL Aiful sink,Ni...	2009-09-24	6.685714	6.703571	6.527500	6.565000	5.673074	550880400	0.075398	0.305545
2009-09-25	US STOCKS Wall St drops recovery worry RIM res...	2009-09-25	6.500357	6.625000	6.480000	6.513214	5.628324	445239200	0.131481	0.248148

Table 1.1: Shape of final dataframe

The data were normalized to be in same range and then were given to LSTM input. The feature columns that we keep are ['Open','High','Low','Close','Adj Close','Volume','polarity'] and we want to predict the next day's Open price. To achieve that we use a time window of 60 days that LSTM look backs and tries to predict the next day's Open price.

Timesteps Window

We then input our data in the form of a 3D array to the LSTM model. First, we create data in 60 timesteps before using Numpy to convert it into an array. Finally, we convert the data into a 3D array with X_train samples, 60 timestamps, and one feature at each step. Moreover, we are trying to predict the next day price, so the are look up step is 1 day. For this reason we have shifted the open prices to 1 day up in front, assuming that previous days news influence next day's open price.

Methodology – Tuning

What we have finally tried are two implementations for both of companies : 1) Giving LSTM model all the feature columns (past stock prices) and predict next day's Open price and 2) Giving LSTM model all the feature columns (past stock prices) and the Sentiment score of certain day's financial news to predict next day's Open price, assuming that sentiment score will improve our model performance.

Our LSTM has been built having 3 layers, can be Bidirectional and we have tried 128 and 256 units/neurons to explore its capabilities in terms of performance. We have used 2 different drop

outs values (0.1 ,0.4) and two different batch sizes [32,64] to get the best parameters for our architecture. After each layer, we are adding dropout and for activation function we are using *Relu* function. Since *Adam* optimizer is known as the best optimizer in terms of performance we are keeping *Adam* as default and use *Bidirectional* LSTM as they have the ability to show very good results as they can understand context better, backwards and upwards We are training our model in 100 epochs for each implementation and finally get these parameters that minimize the Mean Absolute Error.

Learning Curves

To confirm that loss is reducing during epochs we are plotting it using Tensorboard the epoch and validation loss. Along with Tensorboard callback we are also using two more callbacks : 1) *ReduceLROnPlateau*: reduce learning rate when the metric (MAE: mean absolute error) has stopped improving and 2) *EarlyStopping*: that stop training when a monitored quantity has stopped improving.

Apple

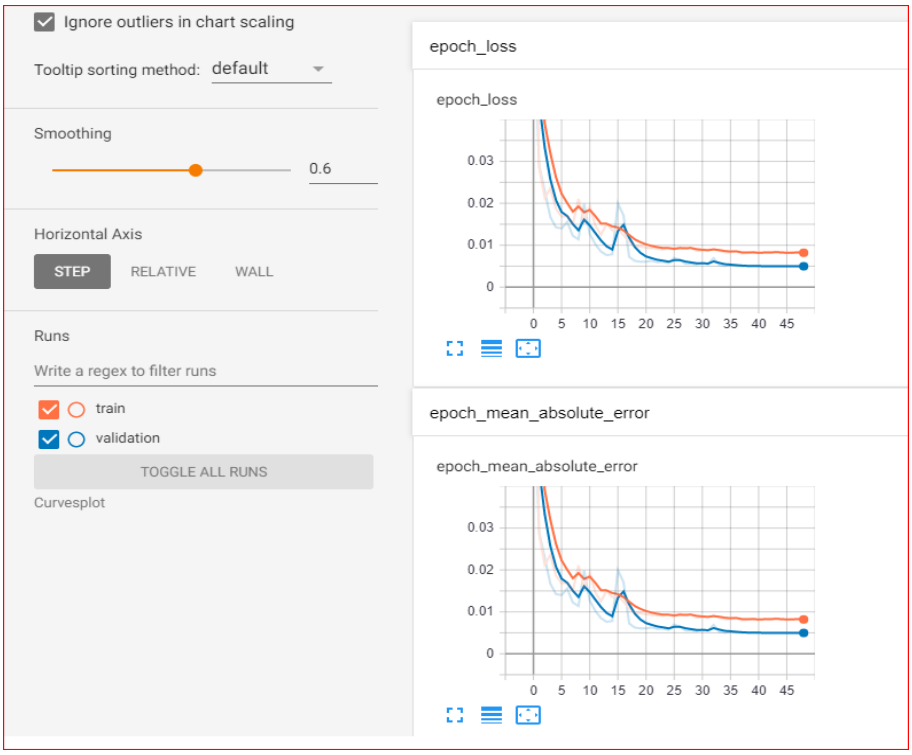


Figure 1.1: Learning Curves for Stocks &Sentiment model

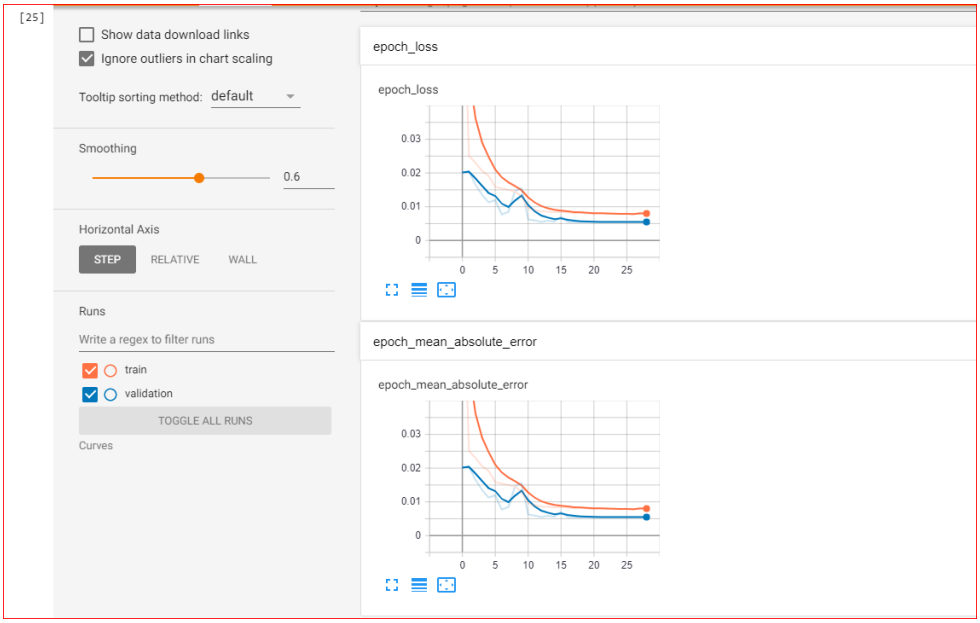


Figure 1.2: Learning Curves for Stocks only model

Microsoft

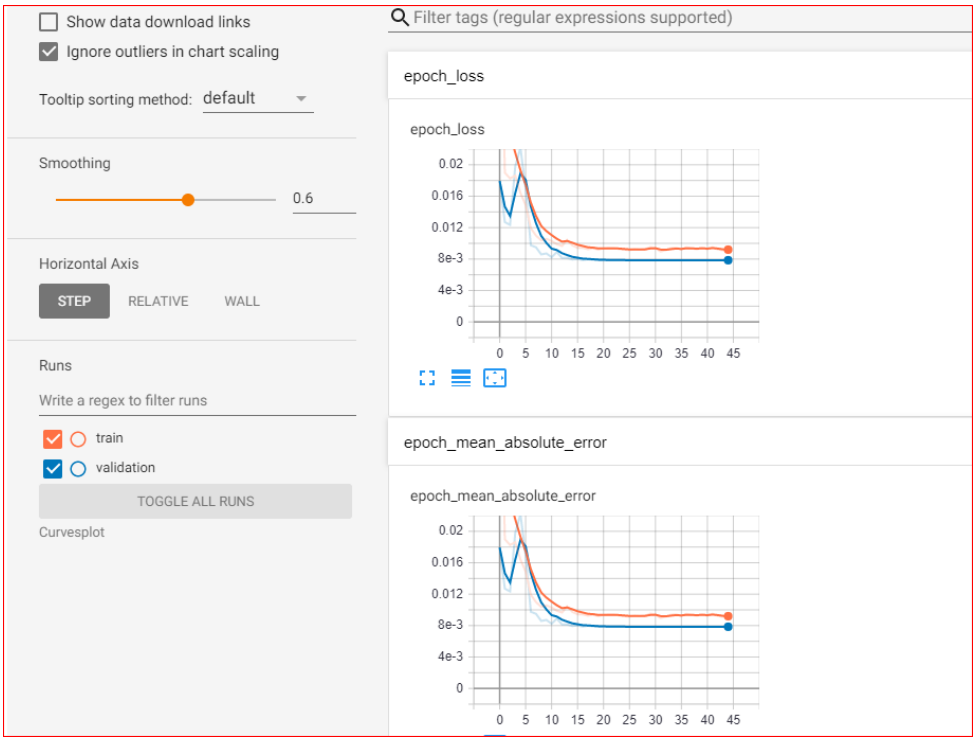


Figure 1.3: Learning Curves for Stocks &Sentiment model

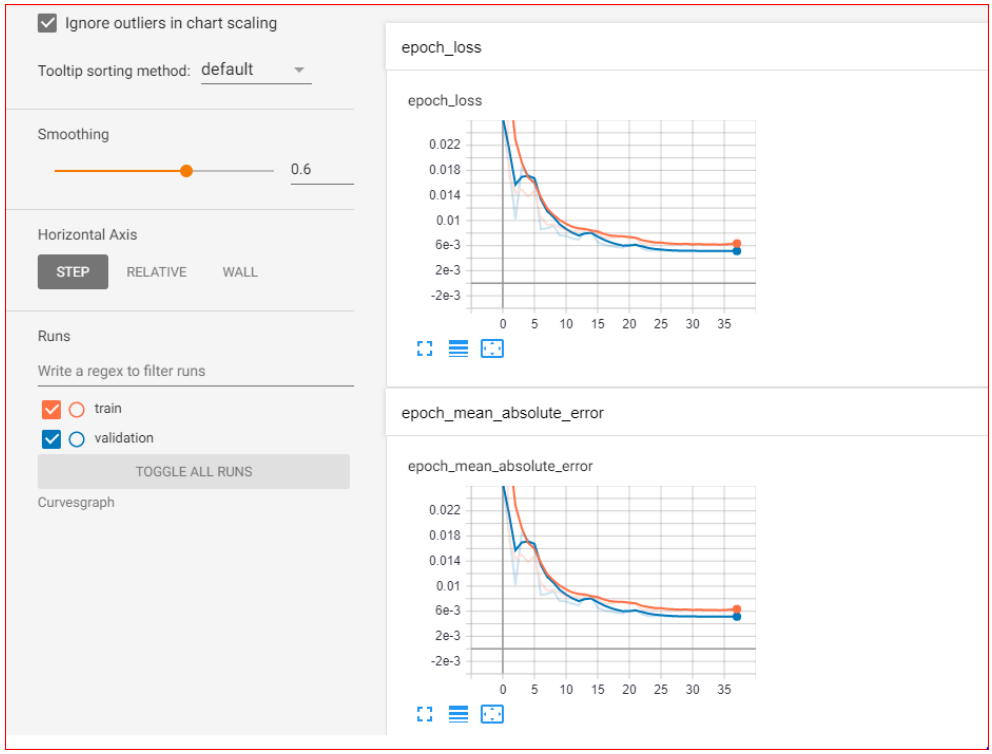


Figure 1.4: Learning Curves for Stocks Only model

Results

In order to interpret the performance of the above implementations we focus on the interpretation of Mean Absolute Error (MAE) we choose as evaluation metric. The MAE is the absolute value of the difference between the forecasted stock price and the actual price. MAE tells us how big of an error we can expect from the forecast on average. After running our models with different parameters of our choice as we mentioned above for tuning, we choose these who minimize the MAE. Best model was chosen with **MAE 6.86** (32 batch size, 0.1 dropout, 256 units) for only stocks Apple implementation and **MAE 7.09**(32 batch size, 0.1 dropout, 256 units) with stocks and sentiment for APPLE implementation. This means that on average our predicted values are 6.86 and 7.09 units away from real price. It is lower than Standard deviation of Open price of Apple (14.76) and it could be better if the Open stock price was not imbalanced as we noticed above (big difference year over year). It is clear also that sentiment did not had a positive impact on minimizing MAE (7.09 > 6.86). In the other hand, for Microsoft we choose also the best hyperparameters based on MAE minimization , **23.79** (32 batch size, 0.1 dropout, 256 units) for only stock model and **24.47**(32 batch size, 0.1 dropout, 128 units) for stocks and sentiment model. Comparing with standard deviation of Microsoft open price (45.95) the MAE is much lower for both models which is quite good, still high though, but seems logical as Microsoft prices has higher stock prices.

batch_size	dropout	units	mae
32.0	0.1	128.0	6.987520
32.0	0.1	256.0	6.863086
32.0	0.4	128.0	7.131041
32.0	0.4	256.0	7.088577
64.0	0.1	128.0	6.986449
64.0	0.1	256.0	6.899435
64.0	0.4	128.0	7.244094
64.0	0.4	256.0	7.357203

batch_size	dropout	units	mae
32.0	0.1	128.0	7.164780
32.0	0.1	256.0	7.098525
32.0	0.4	128.0	7.381705
32.0	0.4	256.0	7.401638
64.0	0.1	128.0	7.409796
64.0	0.1	256.0	7.272293
64.0	0.4	128.0	7.471470
64.0	0.4	256.0	7.352279

Figure 1.5: Apple only with stocks - Apple with sentiment and stock prices

batch_size	dropout	units	mae
32.0	0.1	128.0	24.053155
32.0	0.1	256.0	23.796392
32.0	0.4	128.0	24.743522
32.0	0.4	256.0	24.629776
64.0	0.1	128.0	24.503716
64.0	0.1	256.0	23.957179
64.0	0.4	128.0	24.900420
64.0	0.4	256.0	24.477669

batch_size	dropout	units	mae
32.0	0.1	128.0	24.474606
32.0	0.1	256.0	24.688490
32.0	0.4	128.0	24.954283
32.0	0.4	256.0	24.854168
64.0	0.1	128.0	24.642374
64.0	0.1	256.0	60.699660
64.0	0.4	128.0	25.049769
64.0	0.4	256.0	24.788361

Figure 1.6: Microsoft only with stocks - Microsoft with sentiment and stock prices

We are choosing the hyperparameters (batch size, dropout, units) that minimizing the Mean Absolute Error we created our final model for both companies. Setting these parameters to our final model we end up with a model ready to predict the next day's Open Price. Our last day of having stock Open Price for both companies is 13-02-2020. So, finally we wanted to predict Open Stock Price for 14-02-2020. The table below presents the final results:

Apple Actual Open Price 14/2/2020	Stock and Sentiment Model	Stock Only Model
81.18	80.63	81.03
Microsoft Actual Open Price 14/2/2020	Stock and Sentiment Model	Stock Only Model
183.25	176.45	177.78

Table 1.2: Final Results of 4 models

Comments – Next Steps

One of the main problem was to join two data sources (news and stocks) in one dataframe having dates with information for both of sources. Cleaning and preparing news title from stopwords, keeping only English language words was also a challenge to have a clearer polarity output. Each of the model needed around 40 mins to be trained and tuned working with Google Colab and using GPU runtime. Overall, our model has very good performance, but finally, sentiment score does not give additional information to enhance model's performance. As future steps, we could improve performance, including more company's performance indicators as growth in terms of employees, new offices, market capitalization and social media consumer's opinion.

Implementation 2: LSTM with Finbert Sentence Embeddings

Data preprocessing

The approach that will be used will try to predict if a stock will rise or fall based on the previous day's news which makes it a typical classification problem.

The first thing that needed to be done was to filter the dataset for the specific companies and then clean the dataset, by removing stopwords or pointless blanks. At this point the news were also grouped based on their release date. This was needed in order to have 1 row for each date and a list of all the headlines in that same row.

After this is done, we examine the days of each date in order to shift Saturdays and Sundays to the date of Friday. This workaround is needed because when we try to predict how the opening price of Monday will behave, we need all the news that were released within the weekend. If we shift the dates of weekends to Friday, then we will always be able to use the previous date's news to predict the next.

Later the stock prices of Apple and Microsoft for the relevant dates were downloaded from Yahoo Finance. In order to integrate these 2 datasets only the dates that were common between the headlines and the stock price datasets were kept.

Then the opening price difference with the previous day's opening price was calculated and eventually the negative difference was replaced with 0 and a positive difference which meant that the stock price went up, was replaced with 1.

MSFT

	Date	headlines	Open
1	2009-10-01	[US STOCKS SNAPSHOT Nasdaq falls 2 pct selloff...	0
2	2009-10-22	[US STOCKS Futures mixed ahead data earnings, ...	1
3	2009-10-23	[US STOCKS SNAPSHOT Futures rise Microsoft res...	0
4	2009-10-26	[US STOCKS Wall St gains tech energy shares, U...	1
5	2009-10-28	[US STOCKS Wall St falls data lowered Goldman ...	0
6	2011-03-30	[Microsoft co founder Allen blasts Gates book]	0
7	2011-03-31	[US STOCKS Futures suggest slightly lower open...	0
8	2011-04-04	[Google could target FTC antitrust probe report]	1
9	2011-04-05	[US STOCKS S P hovers near resistance volume s...	1
10	2011-04-06	[US STOCKS S P500 index edges higher light vol...	1

APPLE

	Date	headlines	Open
1	2012-07-19	[Mid Year Update U S And Canadian Stock Market...	1
2	2012-07-23	[Summer Heat Scorches Europe And U S, Apple Ea...	0
3	2012-07-24	[Market Bait And Switch]	0
4	2012-07-27	[Will AAPL Fall From The Tree]	1
5	2012-07-30	[Bulls Snatch Victory From Jaws Defeat]	1
6	2012-07-31	[What Driving China Real Estate Rally Part 3, ...	1
7	2012-08-02	[As Expected Apple Fills Post Earnings Gap, Po...	0
8	2012-08-07	[AAPL With An Eye On The Upcoming iPhone 5, Ch...	1
9	2012-08-10	[Good Knight Public Markets]	1
10	2012-08-14	[VIX Is Under 14 Now What, Largest USA Tech Co...	1

Finbert Sentence Embeddings on the news Headlines

The implementation that will be used includes creating sentence embeddings for each headline and feed them into an RNN LSTM network to see if the stock price behavior can be predicted using only these headlines. Embeddings are low-dimensional, continuous vector representations of discrete variables that are learned when reading a text. The dimensionality of categorical variables is reduced and that allows us to meaningfully represent categories in the transformed space.

For the sentence embeddings on the headlines of the news the BERT base uncased corpus was used that was trained in a vocabulary of 30,000 words. FinBERT uses the BERT corpus and has been specifically trained on financial data but let us be more specific as to what that means.

BERT stands for Bidirectional Encoder Representations Transformers. It is a state-of-the-art NLP model that trains bidirectionally on text, which has proven to have a deeper sense of language context that common language models. That allows the model to spot the difference e.g between “near the river’s bank” and “the bank near the river”.

The fact that each day has a different number of news headlines was a problem, since the embeddings had to be created in a way that all dates would have an equal number of embeddings and the embeddings should be of equal size. To solve this problem and avoid dropping ay information 3 metrics were used to map the different number of embeddings for each day: The mean, minimum and maximum value array. This way we lose less information, and each date now has a list of 3 arrays that represent the mean, minimum and maximum value of all the embeddings from this specific date. On top of that Finbert embeddings use embeddings of 768 length regardless of the sentence’s length.

Model hyperparameter tuning

For the hyperparameter tuning of the model 3 parameters were tested: a) batch size, b) dropout and c) modes of Cyclic learning rate.

The cyclic learning rate is a callback that allows the learning rate of the model to be adjusted in a cyclical way. This allows the model to escape possible local optimum or minimum that might be encountered during training. Local optimum / minimum traps the model in a specific area and that way there is no way to know if the model reached the true optimal point or if there is a better one.

After implementing a thorough hyper parameter tuning 120 different models were built, all following a wide architecture with no additional LSTM layers. Below the parameters of the best model for Microsoft and Apple can be seen. The accuracy that was obtained from both models is approximately 60%.

Microsoft

```
batch_size              16
dropout                0.8
cyclic_mode      triangular2
test_accuracy        60.2787
test_loss           0.883939
Name: 17, dtype: object
```

Apple

```
batch_size              32
dropout                0.8
cyclic_mode      triangular2
test_accuracy        60.6322
test_loss           0.802002
Name: 41, dtype: object
```

Model Results

Afterwards a confusion matrix and a classification report were created to see more details regarding the accuracy of the model on each class.

Microsoft Confusion matrix

```
Test accuracy: 60.279 %
    0    1
0  32   94
1  20  141
```

Microsoft Classification Report

	precision	recall	f1-score	support
0	0.62	0.25	0.36	126
1	0.60	0.88	0.71	161
accuracy			0.60	287
macro avg	0.61	0.56	0.54	287
weighted avg	0.61	0.60	0.56	287

Apple Confusion matrix

Test accuracy: 60.632 %

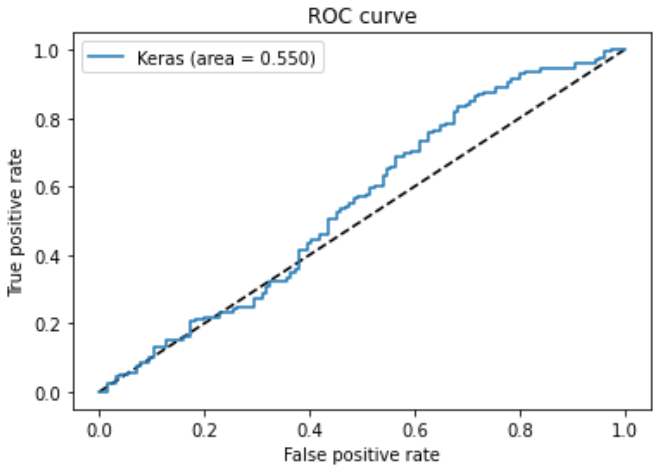
	0	1
0	21	127
1	10	190

Apple Classification Report

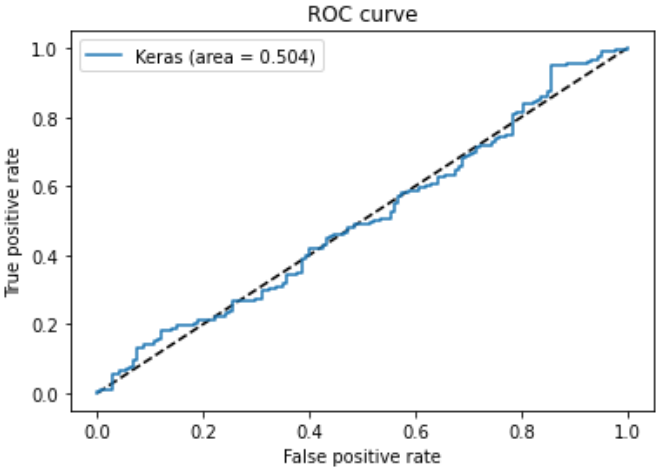
	precision	recall	f1-score	support
0	0.68	0.14	0.23	148
1	0.60	0.95	0.74	200
accuracy			0.61	348
macro avg	0.64	0.55	0.48	348
weighted avg	0.63	0.61	0.52	348

In the plots below we can see the ROC Curves for Apple and Microsoft. The ROC Curve plots the true positive rate versus the false positive. It is considered normal for a low accuracy of 60% to not have an exact curve. Nonetheless for Microsoft it seems that the true positive rate is better than Apple.

Microsoft ROC Curve

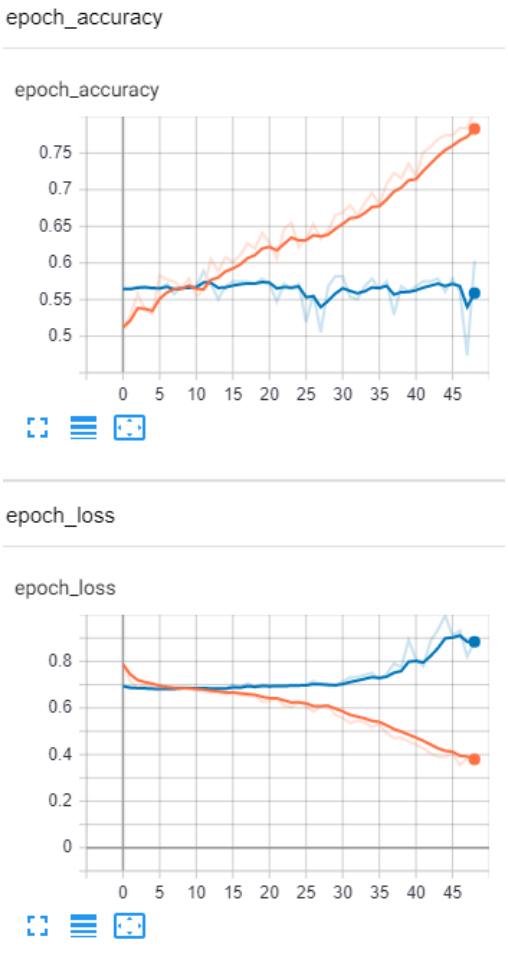


Apple ROC Curve



One more useful plot to inspect would be the tensorboard plots of the loss and the accuracy of the model as it was trained. The purple line refers to the train dataset and the blue to the validation dataset.

Microsoft



Apple

From the above analysis we can conclude that the Microsoft model seems to produce higher quality predictions despite the fact that both models have the same accuracy percentage. To make this clearer imagine that a similar dataset with 0 and 1 as target variables comprises of 1s by 57% and we have 2 models with the same accuracy of 57%. The one model predicts everything to be 1 and the other model predicts both classes of the dataset. The model we would choose would be the model that is able to predict both classes more accurately.

Implementation 3: LSTM with Finbert Sentiment Analysis

The purpose of this side assignment is, initially, to calculate the sentiment of each headline of the dataset, regarding only Apple Inc. (AAPL) and Microsoft Inc. (MSFT). Afterwards, the headline sentiments must be grouped in such a way that each calendar day from the dataset is associated with the average sentiment of the corresponding daily headlines. Finally, defining the daily sentiments as input, a Long Short-Term Memory (LSTM) neural network will be implemented in order to create a classification model, that given a set of past daily sentiments, it will be able to decide if the next day's opening stock price will increase or decrease.

Methodology

In an attempt to create such a model, capable of predicting a stock price's surge or fall having news sentiment as input, we utilized the BERT (Bidirectional Encoder Representations from Transformers) model. BERT is a Natural Language Processing (NLP) model used for pre-training language representations. It leverages an enormous amount of plain text data publicly available *(Wikipedia and Google Books)* on the web and is trained in an unsupervised manner [*Sun, C., Huang, L., & Qiu, X. (2019). Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence*]. Because the news' headlines were in finance context and the meaning of a word or a sentence might differ from the meaning BERT would normally assign, we employed a pre-trained FinBERT model for the stock-related headline sentiment analysis. FinBERT is itself a model, able to analyze sentiment of financial text. It is built by further training the BERT language model in the finance domain, using a large financial corpus and thereby fine-tuning it for financial sentiment classification. The FinBERT implementation we applied, relies on HuggingFace's "pytorch_pretrained_bert" library [2] and their implementation of BERT for sequence classification tasks. The resulting sentiment labels were added next to its corresponding headline, in a new column in the existing dataset.

In order to obtain the daily stock prices for the stocks in question, we created a Python script and with the assistance of Yahoo Finance API, we created another column in the aforementioned dataset, occupying the Open prices for each stock in the corresponding date.

The resulting dataset will comprise of two (2) columns, the daily average sentiment and the opening price of the stock in question that day.

Model Implementation

Having loaded the dataset file into a Pandas DataFrame , we retrieved its subset, which contained the stock tickers, headlines and the release date of each headline. Next, we filtered the DataFrame above, so as to retain only the news regarding Apple and Microsoft. The next course of action would be to handle the news published on weekends. We decided to make the assumption that the weekend news would be attributed to the Fridays in order to associate them more efficiently with the opening stock prices, which are affected by the news of the previous days. With the display, which was described above, we used the FinBert model in order to assign the sentiment to each headline, with the possible options being Negative, Neutral and Positive (*Figure 1*). When the sentiment labeling was completed, we transformed the sentiment values, in order to be in a numeric form in order to calculate their average sentiment. Thus, we assigned to the rows having Negative sentiment the number 1, the rows with Neutral sentiment the number 2 and those with Positive sentiment the number 3. Next, we grouped the headline sentiments by each date in such a form that each date was associated with their average and finally utilizing the yahoo-api script we merged the opening stock prices in their corresponding day. After the merge we transformed the prices into a binary categorical variable of 0,1, with 0 being if that day's price decreased from the previous one, and 1 if the price increased.

In summary, the final form of the dataframe had the columns "sentiment" and "Open", which were the average sentiment of each day and the opening stock price the same day (*Figure 2*).

ticker	title	release_date	sentiment
MSFT	GLOBAL MARKETS U S stocks retreat on recession fears oil falls	2008-10-15	neutral
MSFT	GLOBAL MARKETS U S stocks oil slide on global recession fears	2008-10-15	neutral
MSFT	INTERVIEW UPDATE 2 Nintendo Wii DS sales strong	2008-12-08	neutral
MSFT	Nintendo Dentsu to distribute video on Wii Nikkei	2008-12-24	neutral
MSFT	UPDATE 1 Nintendo Dentsu to offer video on Wii	2008-12-24	neutral
MSFT	COLUMN What Apple loses without Steve Eric Auchard	2009-01-15	negative
MSFT	INTERVIEW Namco Bandai aims to double game profit in 3 yrs	2009-01-19	neutral
MSFT	WRAPUP 3 Qimonda Samsung hit by chip market crash	2009-01-23	neutral
MSFT	UPDATE 2 Microsoft to open own stores take on Apple	2009-02-12	neutral
MSFT	ANALYSIS Video gaming could be bargain buy of UK equities	2009-03-25	neutral

Figure 1

	sentiment	Open
1425	2.200000	1.0
1426	2.000000	0.0
1427	2.500000	1.0
1428	2.300000	0.0
1429	2.000000	1.0
1430	2.000000	0.0
1431	1.700000	1.0
1432	2.000000	1.0
1433	2.274194	1.0
1434	2.500000	0.0

Figure 2

For the LSTM classification model, we implemented, the input will be the sentiments for a specific number of days back and the target variable will be the binary categorical variable. In order to decide which time window of news sentiment provides the best accuracy in the increase / decrease prediction, we introduced multiple time windows, looking 60, 30, 15 and 5 days back. Due to the imbalance of both Apple (*Figure 3) and* Microsoft datasets (*Figure 4*), there has been observed a certain level of failure to correctly predict the days that the stock prices will fall, with the exception of relatively correction predictability, when the model had a timestep of only five (5) days. Regarding the aforementioned imbalance, instead of a Receiver Operating Characteristic (ROC) curve, the visualized evaluation method of choice was the Precision-Recall curve of the model compared to a no-skill classifier. A no-skill classifier is one, that cannot discriminate between the classes and would predict a random class or a constant class in all cases. The no-skill line changes based on the distribution of the positive to negative classes. It is a horizontal line with the value of the ratio of positive cases in the dataset. For a balanced dataset, this is 0.5. In conclusion, of the models implemented, the most reliable in terms of both accuracy and low error rate is the one with the 5-day timestep, which produces ~56% accuracy, and the type-2 error is at its lowest value.

	Count
Open	
0	797
1	942

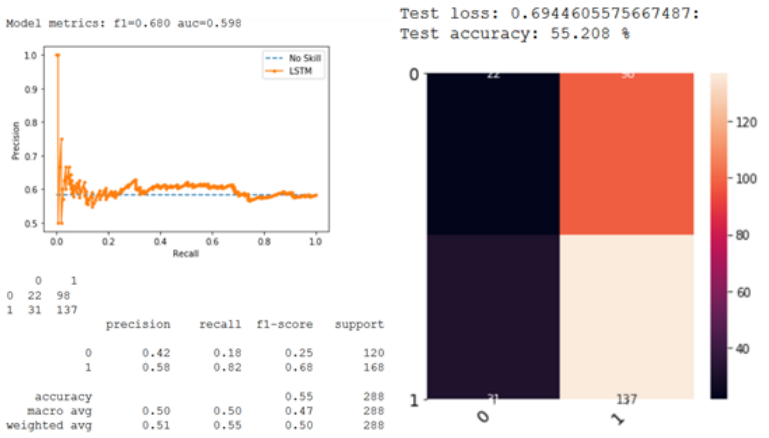
Figure 3

	Count
Open	
0.0	641
1.0	793

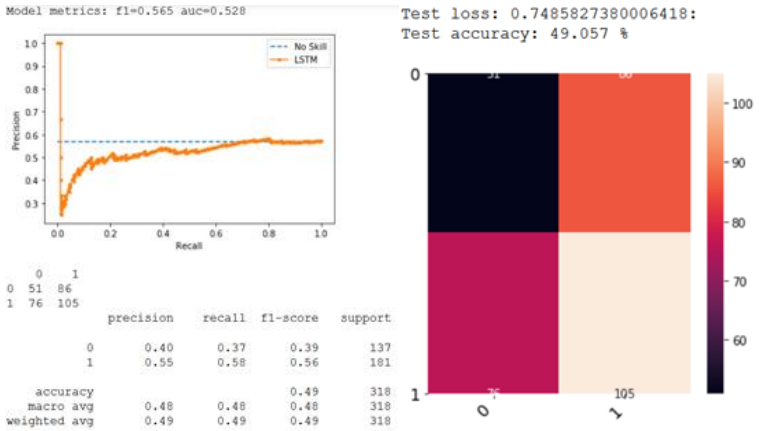
Figure 4

Apple Dataset Metrics:

- 60-day timestep

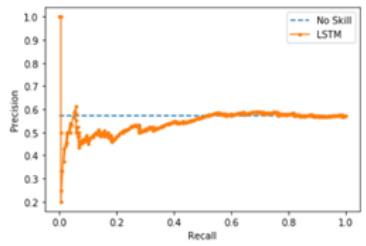


- 30-day timestep



- 15-day timestep

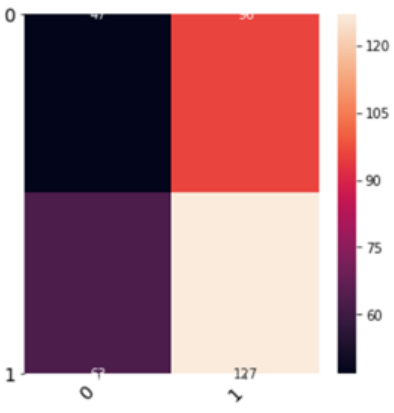
Model metrics: f1=0.615 auc=0.546



	0	1
0	47	96
1	63	127

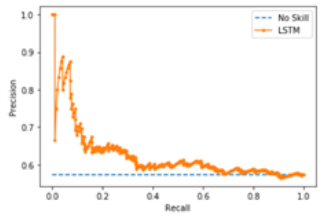
	precision	recall	f1-score	support
0	0.43	0.33	0.37	143
1	0.57	0.67	0.62	190
accuracy			0.52	333
macro avg	0.50	0.50	0.49	333
weighted avg	0.51	0.52	0.51	333

Test loss: 0.7519956873701857:
Test accuracy: 52.252 %



- 5-day timestep

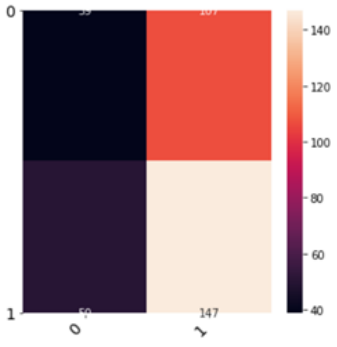
Model metrics: f1=0.652 auc=0.625



	0	1
0	39	107
1	50	147

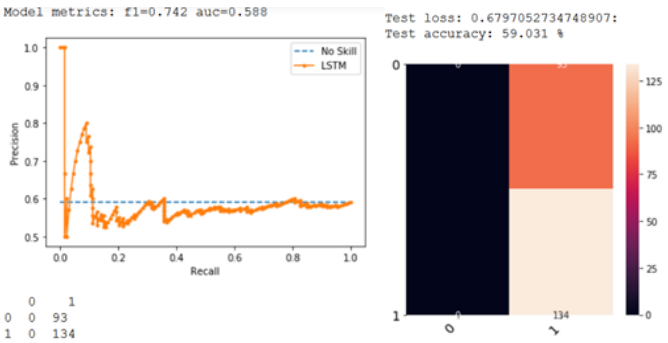
	precision	recall	f1-score	support
0	0.44	0.27	0.33	146
1	0.58	0.75	0.65	197
accuracy			0.54	343
macro avg	0.51	0.51	0.49	343
weighted avg	0.52	0.54	0.52	343

Test loss: 0.6803394168528454:
Test accuracy: 54.227 %



Microsoft Dataset Metrics

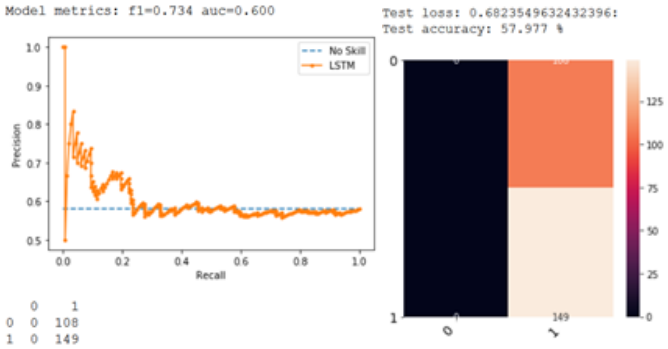
- 60-day timestep




```
     0    1
0    0   93
1    0  134
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	93
1	0.59	1.00	0.74	134
accuracy			0.59	227
macro avg	0.30	0.50	0.37	227
weighted avg	0.35	0.59	0.44	227

- 30-day timestep

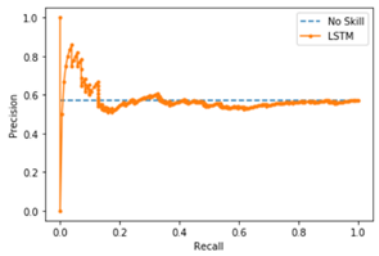



```
     0    1
0    0  108
1    0  149
```

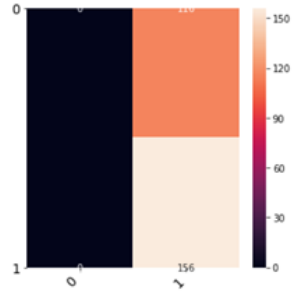
	precision	recall	f1-score	support
0	0.00	0.00	0.00	108
1	0.58	1.00	0.73	149
accuracy			0.58	257
macro avg	0.29	0.50	0.37	257
weighted avg	0.34	0.58	0.43	257

- 15-day timestep

Model metrics: f1=0.729 auc=0.573



Test loss: 0.6833967952167287:
Test accuracy: 57.353 %

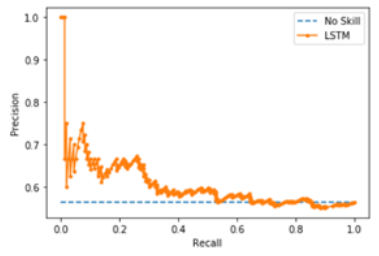


	0	1
0	0	116
1	0	156

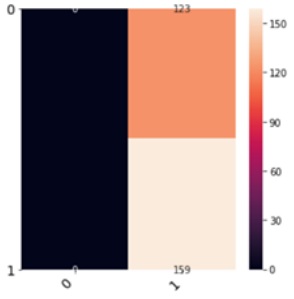
	precision	recall	f1-score	support
0	0.00	0.00	0.00	116
1	0.57	1.00	0.73	156
accuracy			0.57	272
macro avg	0.29	0.50	0.36	272
weighted avg	0.33	0.57	0.42	272

- 5-day timestep

Model metrics: f1=0.721 auc=0.603



Test loss: 0.6853475727088062:
Test accuracy: 56.383 %



	0	1
0	0	123
1	0	159

	precision	recall	f1-score	support
0	0.00	0.00	0.00	123
1	0.56	1.00	0.72	159
accuracy			0.56	282
macro avg	0.28	0.50	0.36	282
weighted avg	0.32	0.56	0.41	282

Implementation 4: Arima Model - Forecasting Using Time Series Analysis

Since stock prices are not randomly generated values, they can be treated as a discrete-time series model which is based on a set of well-defined numerical data items collected at successive points at regular intervals of time. In this implementation we are going to use Autoregressive Integrated Moving Average Model (ARIMA) instead of simple forecasting, because it is essential to analyze trends of stock prices in order to identify a model.

Autoregressive Integrated Moving Average Model

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 stands for the number of autoregressive terms, i.e. the number of observations from past time values used to forecast future values. e.g. if the value of p is 2, then this means that two previous time observations in the series are being used to forecast the future trend.
- d denotes the number of differences needed to make the time series stationary (i.e. one with a constant mean, variance, and autocorrelation). For instance, if d = 1, then it means that a first-difference of the series must be obtained to transform it into a stationary one.
- q represents the moving average of the previous forecast errors in our model, or the lagged values of the error term. As an example, if q has a value of 1, then this means that we have one lagged value of the error term in the model.

A linear regression model is constructed including the specified number and type of terms, and the data is prepared by a degree of differencing in order to make it stationary, i.e. to remove trend and seasonal structures that negatively affect the regression model.

Before we can build a model, we must ensure that the time series is stationary. Stationarity is a very important concept in Time Series Analysis. In order to apply a time series model, it is important for the Time series to be stationary; in other words all its statistical properties (mean, variance) remain constant over time. This is done basically because if we consider a certain behavior over time, it is important that this behavior is same in the future in order to forecast the series. In practice we can assume the series to be stationary if it has constant statistical properties over time and these properties can be:

- constant mean
- constant variance
- an auto co-variance that does not depend on time.

There are two primary way to determine whether a given time series is stationary.

- Rolling Statistics: Plot the rolling mean and rolling standard deviation. The time series is stationary if they remain constant with time (with the naked eye look to see if the lines are straight and parallel to the x-axis).
- Augmented Dickey-Fuller Test: The time series is considered stationary if the p-value is low (according to the null hypothesis) and the critical values at 1%, 5%, 10% confidence intervals are as close as possible to the ADF Statistics

Methodology

To the begin with, data was transposed in a pandas Series with date as index. We were interested in forecasting opening stock values, so from the original datasets only Open values and dates were kept.

Date	Open
2009-09-21	6.581786
2009-09-22	6.613929
2009-09-23	6.621428
2009-09-24	6.685714
2009-09-25	6.500357

Date	Open
2009-09-21	25.110001
2009-09-22	25.400000
2009-09-23	25.920000
2009-09-24	25.920000
2009-09-25	25.690001

Figure 4.1, 4. 2: Open Values (Left: Apple, Right: Microsoft)

From the Figure 1 it's obvious that data are not stationary. We can also assume that from the autocorrelation plots.

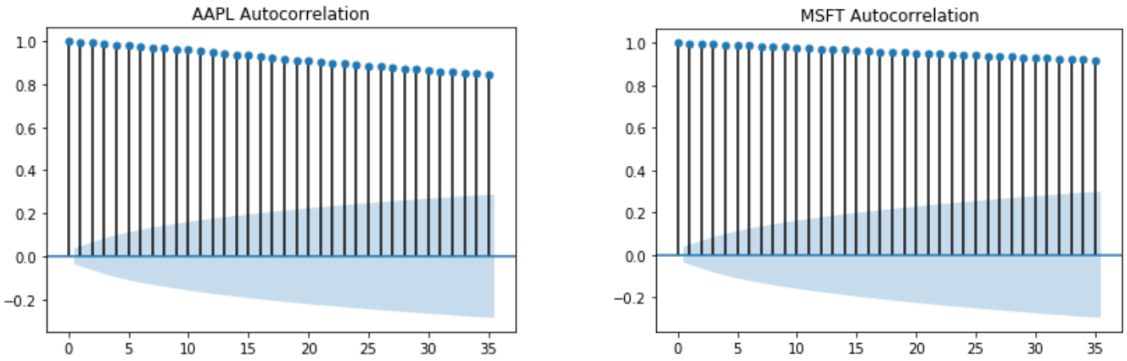


Figure 4.3, 4.4: Autocorrelation Plots (Left: Apple, Right: Microsoft)

When autocorrelation is a function of time (decreasing or increasing smoothly following a pattern) data is not stationary. Data should be transformed in order to become stationary and proceed with our analysis. The most common transformation used in time series is to calculate the n-th order discrete difference along time. In our case, the discrete difference with the previous value (n=1), was enough to make the data stationary. The value of d, therefore, is 1, the minimum number of differencing needed to make the series stationary

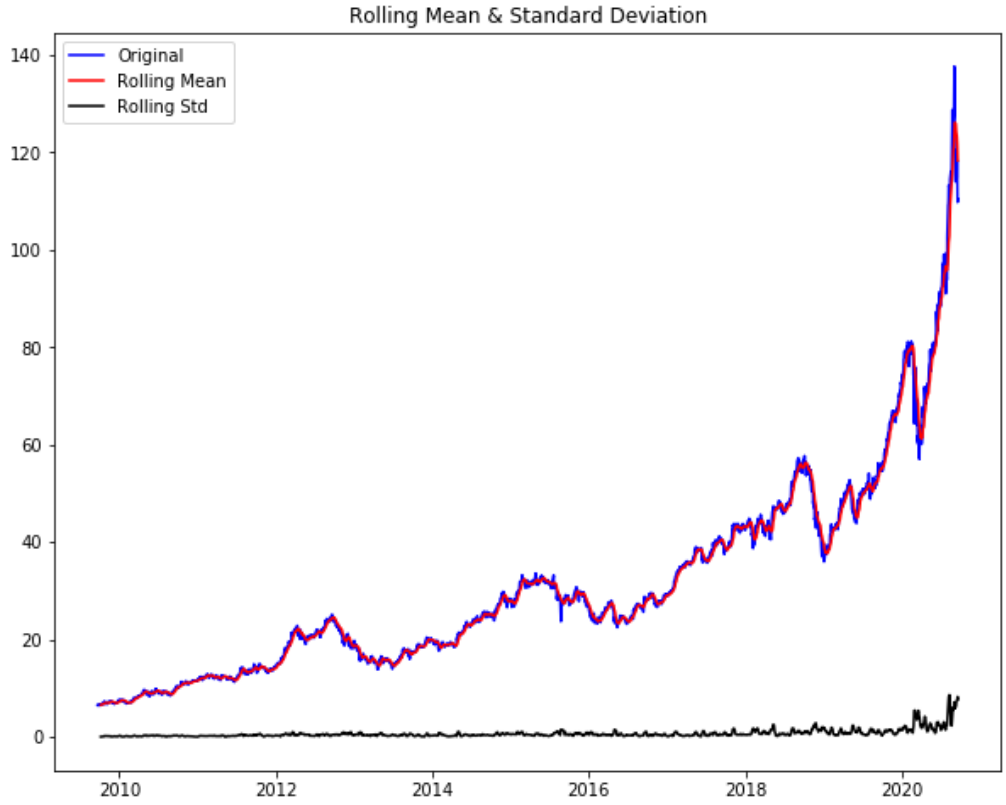
Date	Open
2009-09-22	0.032143
2009-09-23	0.007499
2009-09-24	0.064286
2009-09-25	-0.185357
2009-09-28	0.066429

Date	Open
2009-09-22	0.289999
2009-09-23	0.520000
2009-09-24	0.000000
2009-09-25	-0.229999
2009-09-28	-0.090001

Figure 4.5, 4.6: Open Values Differences (Left: Apple, Right: Microsoft)

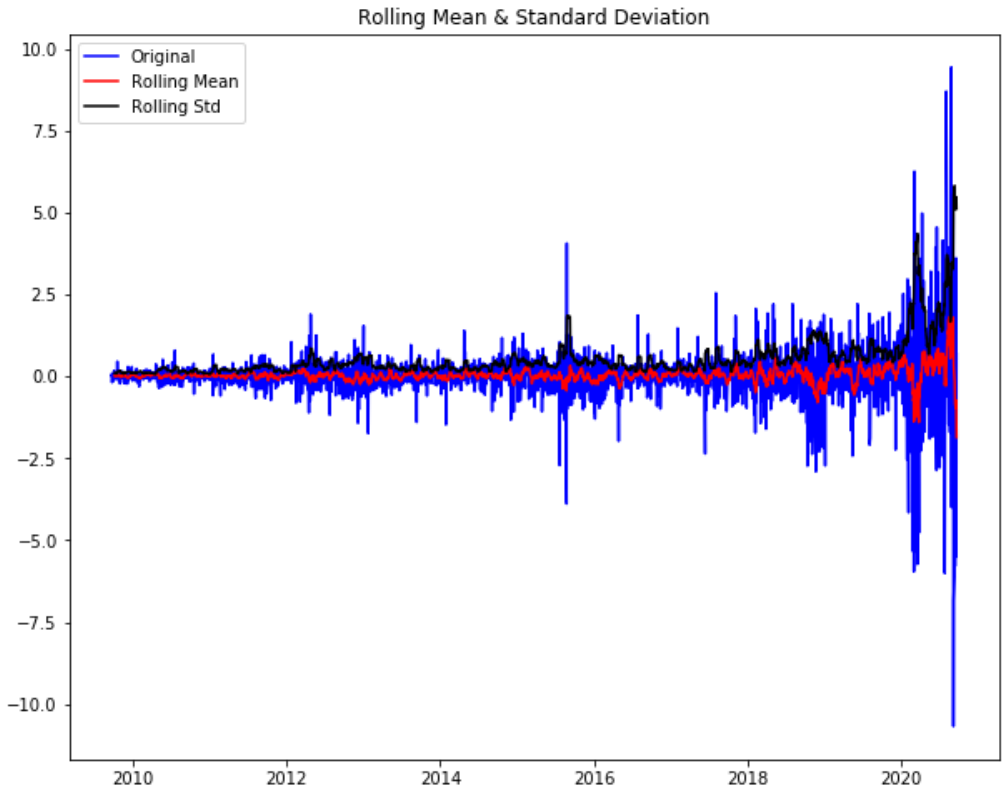
In python, a function was built to examine rolling statistics of data and apply the Augmented Dickey-Fuller Test to check the stationarity of data. If p-value of hypothesis test is not greater than the level of significance (0.05), then we can assume that data is stationary.

In the following figures the results of this function are presented. Figures 4.7, 4.9 show the results for the initial data, were the stationarity hypothesis should be rejected, while figures 4.8, 4.10 show the results for the difference values, where null hypothesis can be rejected.



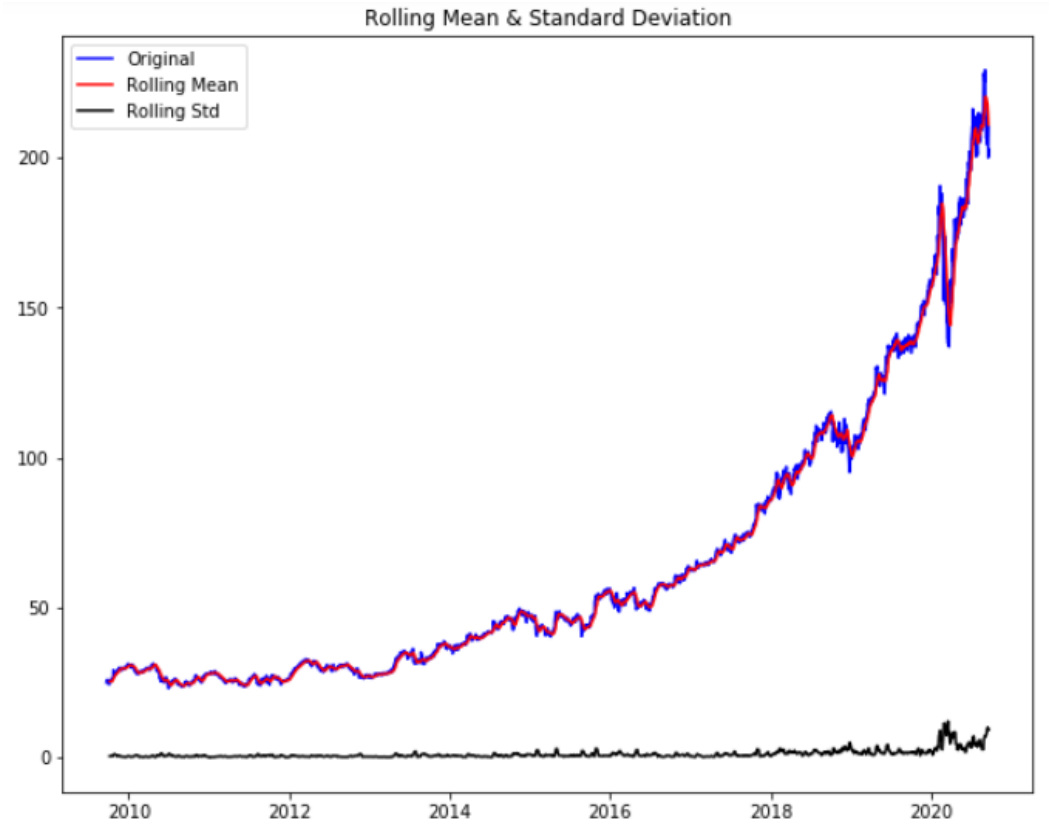

```
Results of Dickey-Fuller Test:
Test Statistic                    1.973091
p-value                           0.998637
#Lags Used                       27.000000
Number of Observations Used    2741.000000
Critical Value (1%)              -3.432738
Critical Value (5%)              -2.862595
Critical Value (10%)             -2.567332
dtype: float64
```

Figure 4.7: Apple Open Values Stationarity Test (Up: Rolling Statistics, Down: ADF Test)



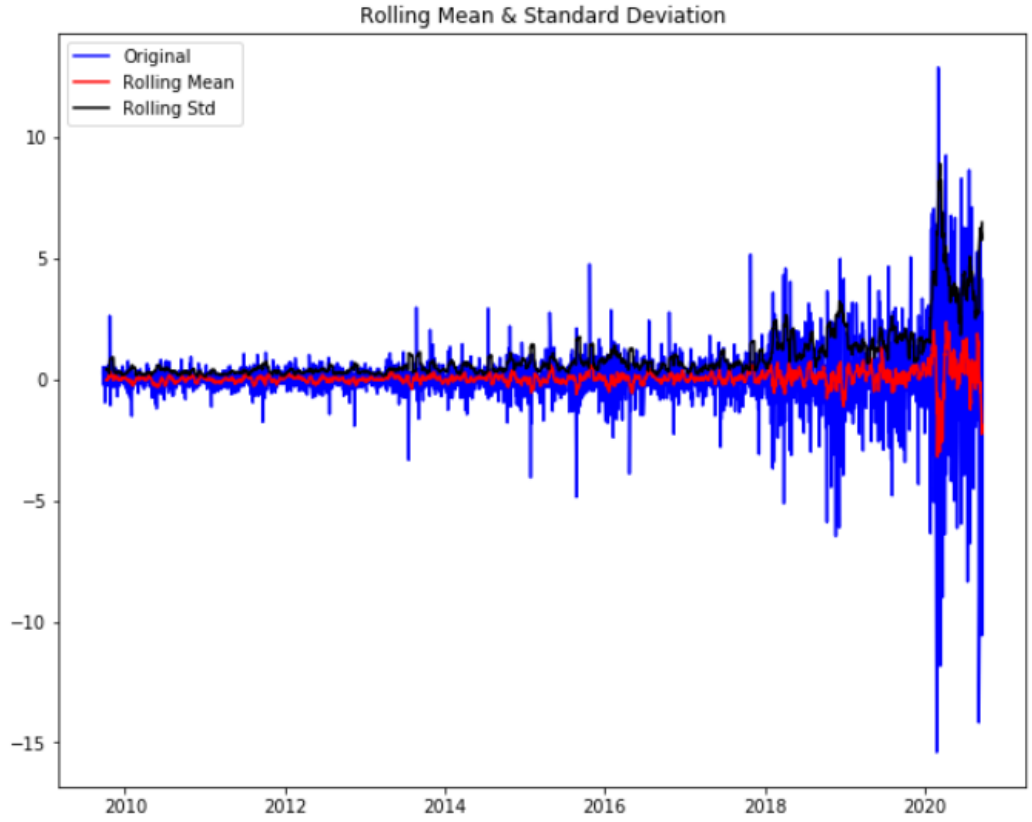

```
Results of Dickey-Fuller Test:
Test Statistic                -9.990012e+00
p-value                        2.007736e-17
#Lags Used                     2.600000e+01
Number of Observations Used    2.741000e+03
Critical Value (1%)           -3.432738e+00
Critical Value (5%)           -2.862595e+00
Critical Value (10%)          -2.567332e+00
dtype: float64
```

Figure 4.8: Apple Open Values Difference Stationarity Test (Up: Rolling Statistics, Down: ADF Test)




```
Results of Dickey-Fuller Test:
Test Statistic                    4.287284
p-value                           1.000000
#Lags Used                       28.000000
Number of Observations Used    2740.000000
Critical Value (1%)              -3.432739
Critical Value (5%)              -2.862595
Critical Value (10%)             -2.567332
dtype: float64
```

Figure 4.9: Microsoft Open Values Stationarity Test (Up: Rolling Statistics, Down: ADF Test)




```
Results of Dickey-Fuller Test:
Test Statistic                -1.272080e+01
p-value                        9.837120e-24
#Lags Used                     2.700000e+01
Number of Observations Used    2.740000e+03
Critical Value (1%)           -3.432739e+00
Critical Value (5%)           -2.862595e+00
Critical Value (10%)          -2.567332e+00
dtype: float64
```

Figure 4.10: Microsoft Open Values Difference Stationarity Test (Up: Rolling Statistics, Down: ADF Test)

Equivalent is the change in the autocorrelation of the data.

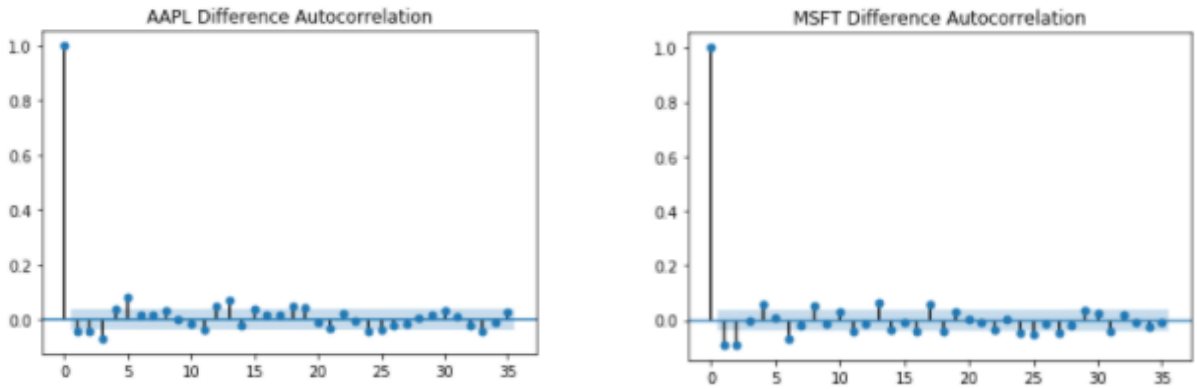


Figure 4.11, 4.12: Open Values Difference Autocorrelation Plots (Left: Apple, Right: Microsoft)

Since the parameter d is fixed, we can use the auto_arima function from pmdarima library to identify the most optimal parameters for an ARIMA model and fit the model. It works by conducting differencing tests (i.e. Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller or Phillips–Perron) to determine the order of differencing, d, and then fitting models within ranges of defined start_p, max_p, start_q, max_q ranges. The function provides the model which minimizes the AIC criterion.

But first a simple AR model was fitted, to be used as a comparison measure.

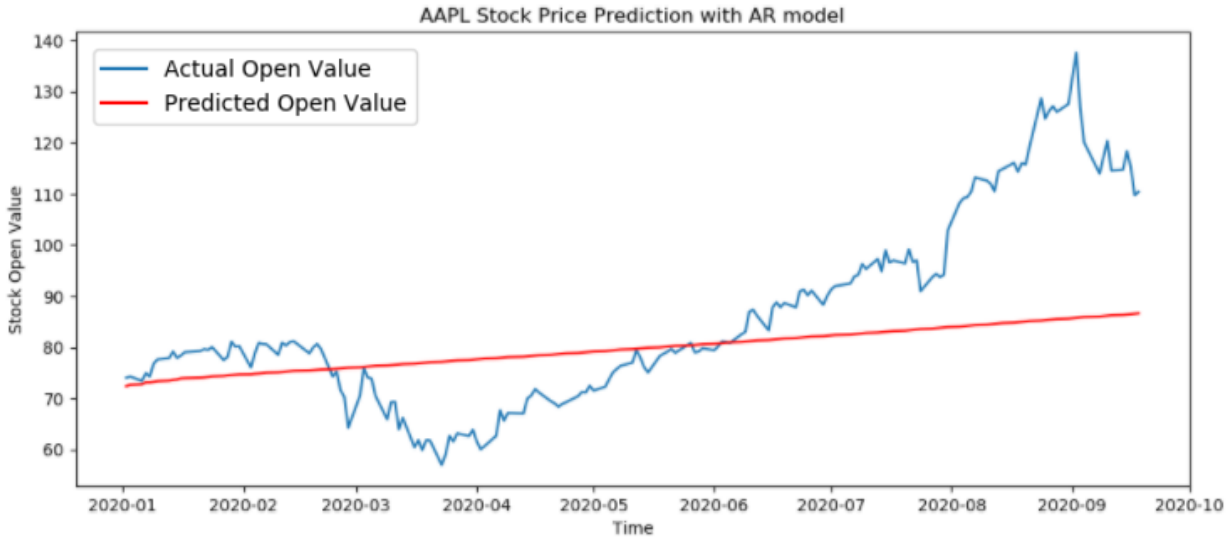


Figure 4.13: AR model predictions for Apple Open Values

The AR model indicates that the best fit is a simple line. We expect much better solutions from the ARIMA model.

The auto_arima function chose the parameters that minimize the AIC criterion. In figure 4.14 the model parameters are shown. Then the model was fitted and test values were predicted.

Original and predicted values were plotted in figure 4.15. Unfortunately the models doesn't seem a lot better than the AR model.

SARIMAX Results

Dep. Variable:	y	No. Observations:	2588
Model:	SARIMAX(3, 1, 2)	Log Likelihood	-1790.011
Date:	Tue, 20 Oct 2020	AIC	3594.023
Time:	01:01:35	BIC	3635.031
Sample:	0 - 2588	HQIC	3608.885
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0546	0.020	2.706	0.007	0.015	0.094
ar.L1	-0.2190	0.044	-5.015	0.000	-0.305	-0.133
ar.L2	-0.8541	0.042	-20.319	0.000	-0.936	-0.772
ar.L3	-0.0717	0.014	-4.985	0.000	-0.100	-0.044
ma.L1	0.1481	0.041	3.573	0.000	0.067	0.229
ma.L2	0.8892	0.036	24.436	0.000	0.818	0.961
sigma2	0.2336	0.003	79.537	0.000	0.228	0.239

Ljung-Box (Q):	78.87	Jarque-Bera (JB):	7730.69
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	5.07	Skew:	-0.35
Prob(H) (two-sided):	0.00	Kurtosis:	11.44

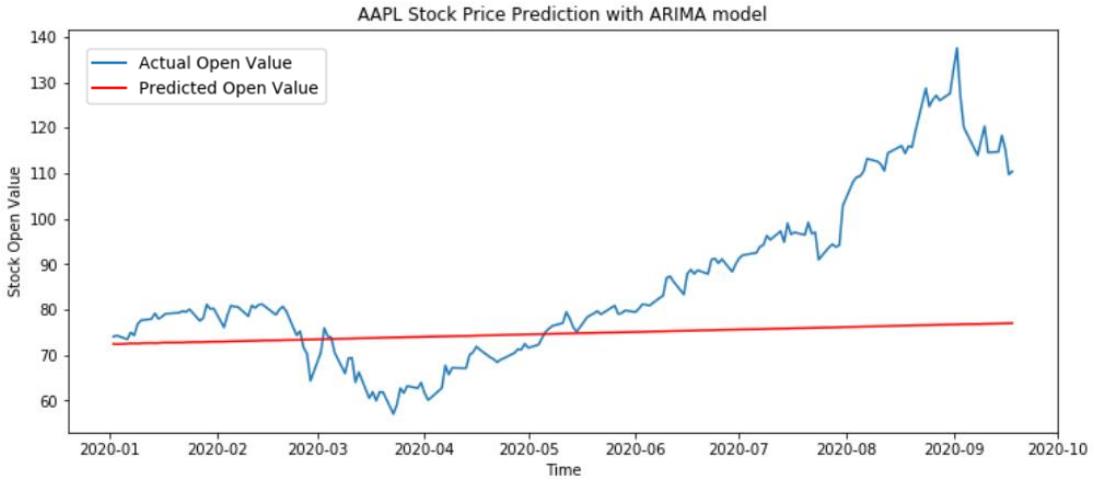
SARIMAX Results

Dep. Variable:	y	No. Observations:	2588
Model:	SARIMAX(4, 1, 1)	Log Likelihood	-3319.721
Date:	Tue, 20 Oct 2020	AIC	6653.442
Time:	01:45:17	BIC	6694.450
Sample:	0 - 2588	HQIC	6668.304
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0176	0.006	2.929	0.003	0.006	0.029
ar.L1	0.6343	0.077	8.288	0.000	0.484	0.784
ar.L2	0.0754	0.017	4.479	0.000	0.042	0.108
ar.L3	-0.0022	0.013	-0.166	0.868	-0.028	0.023
ar.L4	-0.0533	0.014	-3.704	0.000	-0.081	-0.025
ma.L1	-0.7526	0.077	-9.766	0.000	-0.904	-0.602
sigma2	0.7623	0.009	83.676	0.000	0.744	0.780

Ljung-Box (Q):	84.41	Jarque-Bera (JB):	8868.09
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	11.90	Skew:	-0.27
Prob(H) (two-sided):	0.00	Kurtosis:	12.05

Figure 4.14, 4.15: ARIMA models summary (Left: Apple, Right: Microsoft)



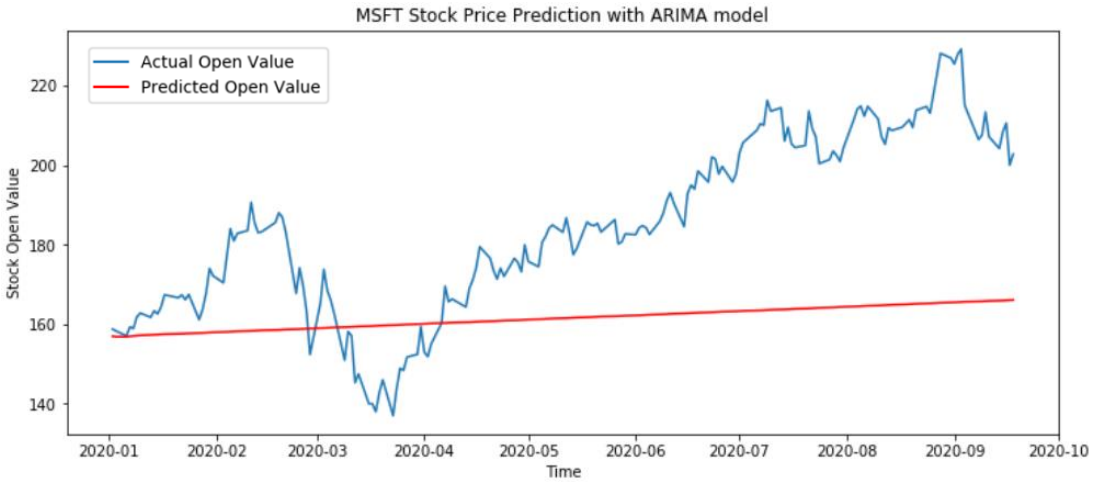


Figure 4.16, 4.17: ARIMA model predictions (Up: Apple, Down: Microsoft)

To fix the models results, we built a function that chooses the models, minimizing the Mean Squared Error of predictions. For the previous models, MSE=420.75 for Apple and MSE=922,29 for Microsoft. The minimum values we managed to reach is MSE=177.75 (MAE=10.32) for Apple and MSE=477.96 (MAE=18.5) for Microsoft.

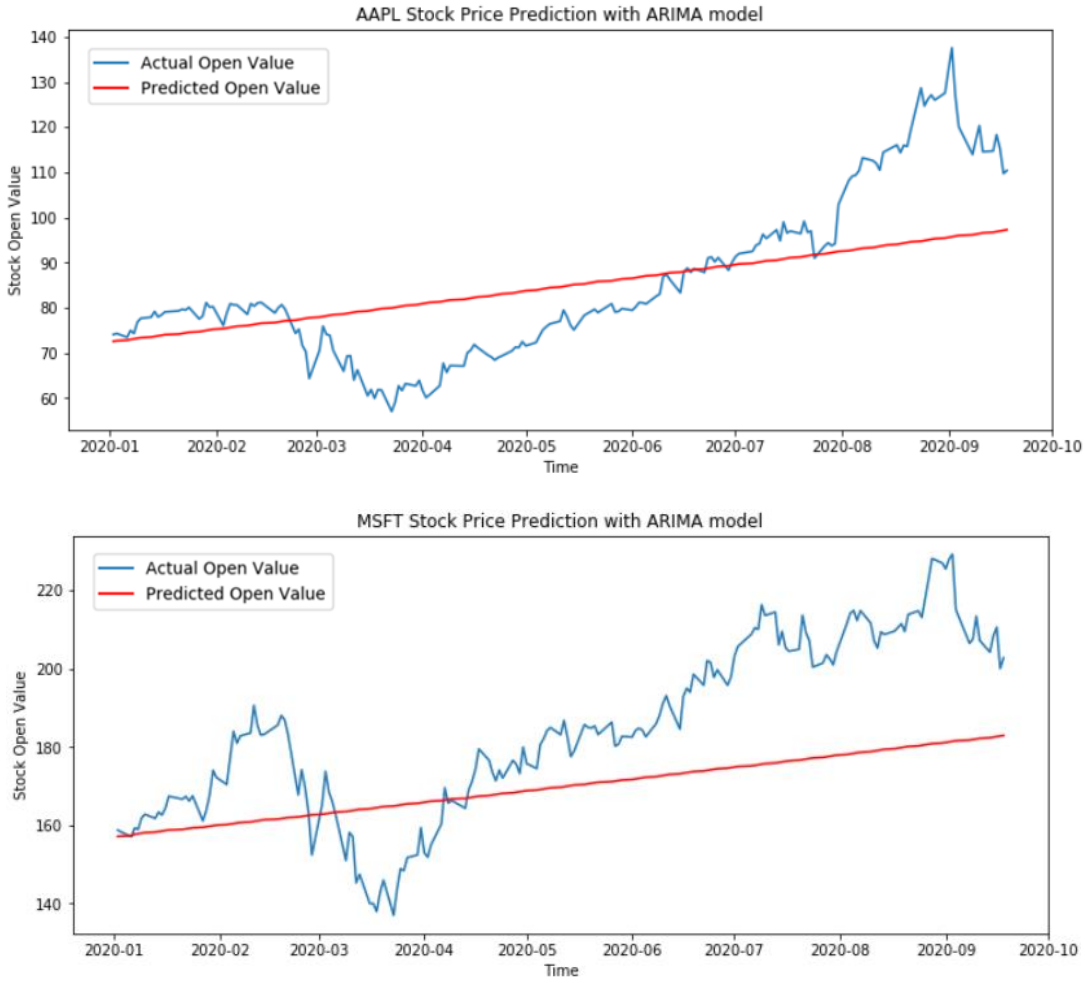


Figure 4.18, 4.19: ARIMA model predictions -min MSE (Up: Apple, Down: Microsoft)

Although the fit was improved, we can not claim we have valid predictions. This becomes more clear when we plot all data with predictions and the confidence level. For 95% confidence level the model fails to predict the rapid increase of the value in 2020. Maybe, some extraneous factors, like the pandemic, affected the stock market, and our model is incapable to predict them.

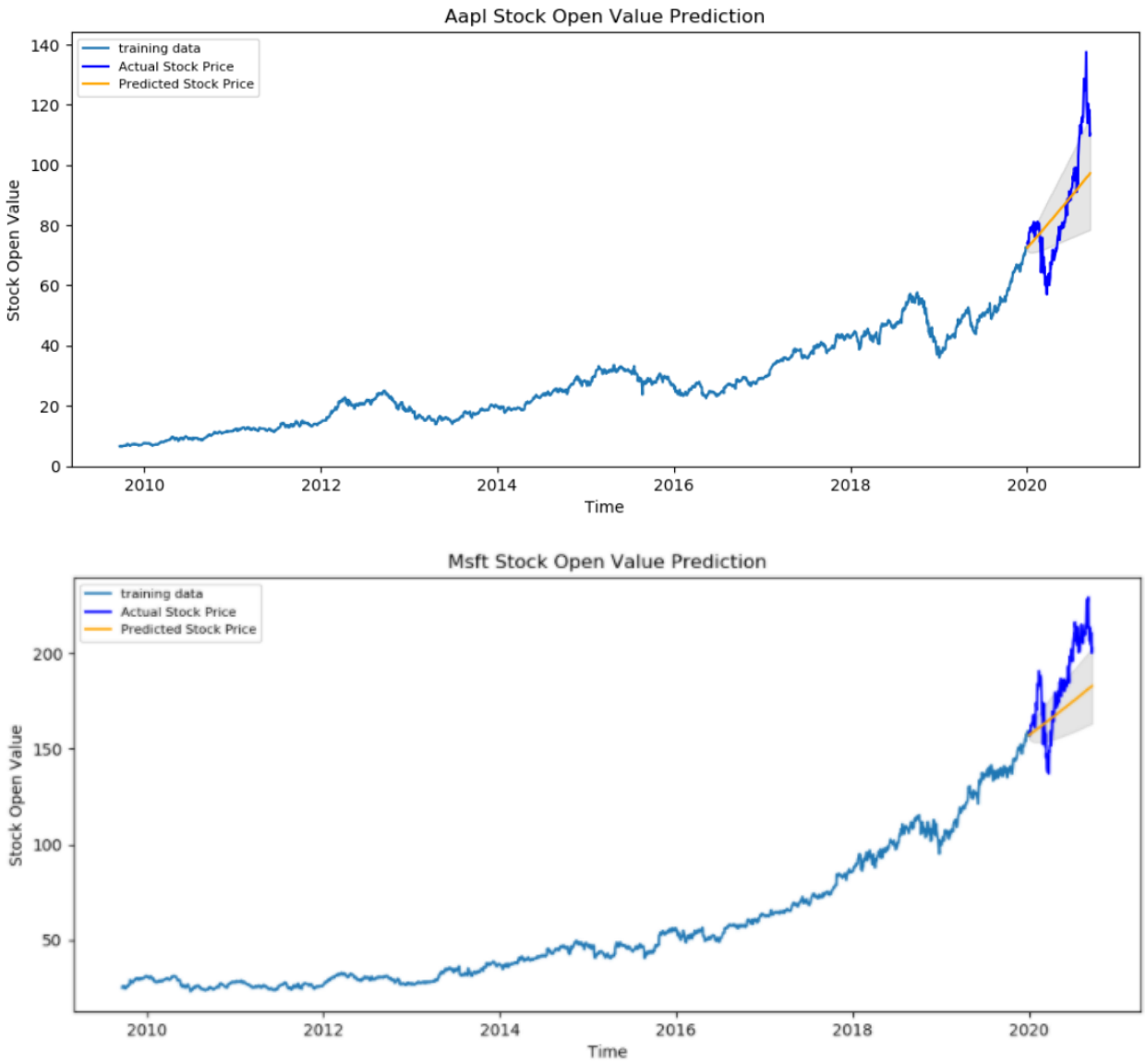


Figure 4.20, 4.21: ARIMA model predictions -min MSE (Up: Apple, Down: Microsoft)

Because ARIMA is a time series forecast model, the amount and variety of data lead the model to a smooth and more general solution. In order to achieve more precise results we should try short term prediction. In this implementation, we followed the same methodology, but the data used was the latest one, from July until September 2020. The train set included 56 days and the model would predict the Open Values for 14 days (test).

Date	Open
2020-07-01	203.139999
2020-07-02	205.679993
2020-07-06	208.830002
2020-07-07	210.449997
2020-07-08	210.070007

Date	Open
2020-07-01	91.279999
2020-07-02	91.962502
2020-07-06	92.500000
2020-07-07	93.852501
2020-07-08	94.180000

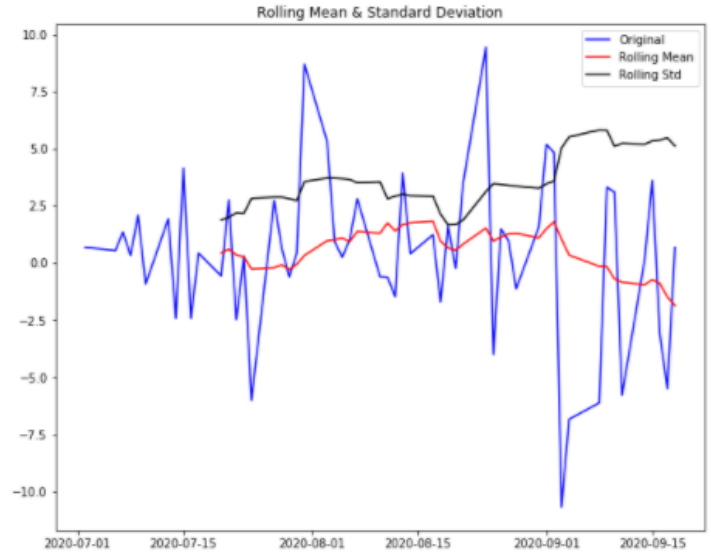
Figure 4.22, 4.23: Open Values since 2020-07-01 (Left: Apple, Right: Microsoft)

Once again the data should be differenced in order to be stationary.

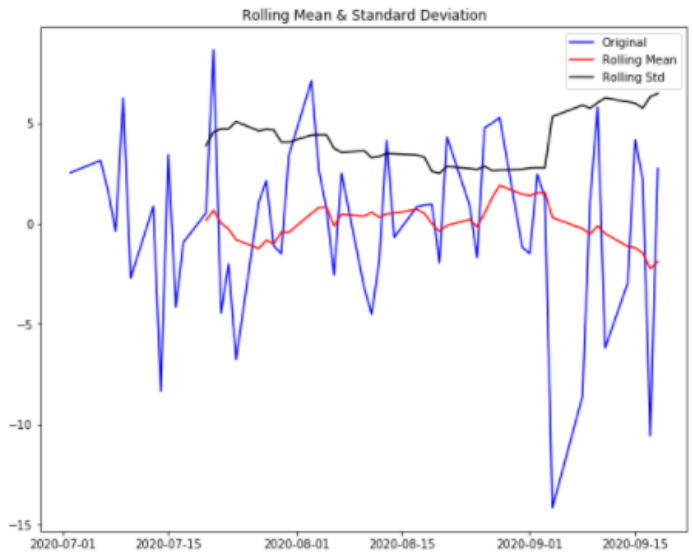
Date	Open
2020-07-02	0.682503
2020-07-06	0.537498
2020-07-07	1.352501
2020-07-08	0.327499
2020-07-09	2.082497

Date	Open
2020-07-02	2.539994
2020-07-06	3.150009
2020-07-07	1.619995
2020-07-08	-0.379990
2020-07-09	6.259995

Figure 4.22, 4.23: Open Values Differences since 2020-07-01 (Left: Apple, Right: Microsoft)




```
Results of Dickey-Fuller Test:
Test Statistic                -6.755884e+00
p-value                        2.872652e-09
#Lags Used                     0.000000e+00
Number of Observations Used    5.400000e+01
Critical Value (1%)           -3.557709e+00
Critical Value (5%)           -2.916770e+00
Critical Value (10%)          -2.596222e+00
dtype: float64
```




```
Results of Dickey-Fuller Test:
Test Statistic                -3.115479
p-value                        0.025429
#Lags Used                    11.000000
Number of Observations Used   43.000000
Critical Value (1%)           -3.592504
Critical Value (5%)           -2.931550
Critical Value (10%)          -2.604066
dtype: float64
```

Figure 4.24, 4.25: Open Values Difference Stationarity Test (Up: Rolling Statistics, Down: ADF Test, Left: Apple, Right: Microsoft)

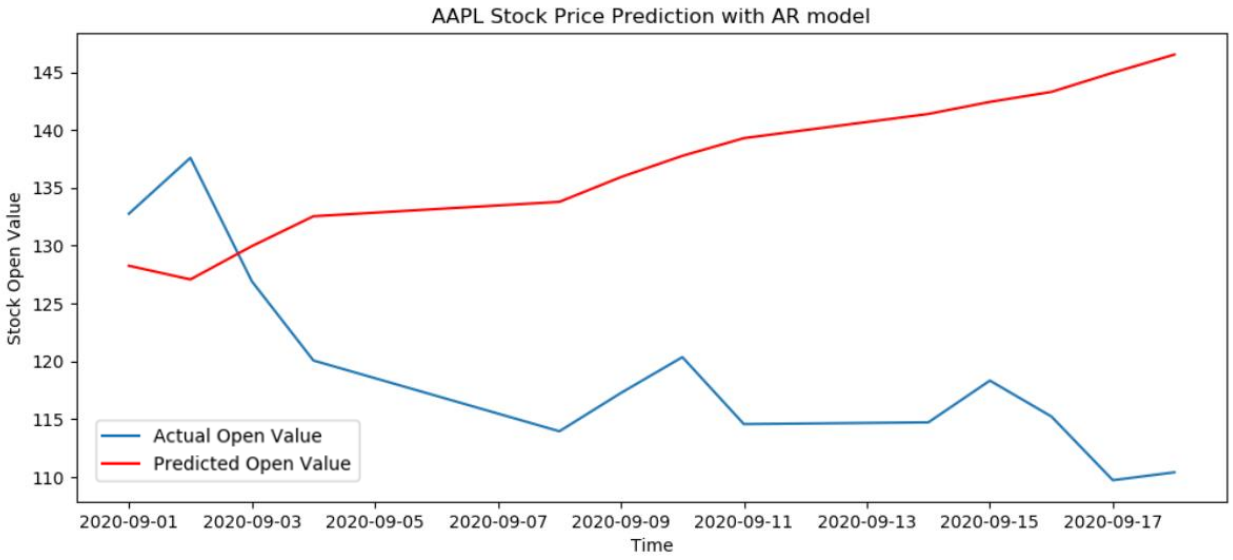


Figure 4.26: AR model predictions for Apple Open Values

The simple AR model fails to predict the change in the curve monotony and gives increasing values, when the actual stock value, decreases.

The auto_arima function gives a model close to the AR model.

SARIMAX Results

Dep. Variable:	y	No. Observations:	43
Model:	SARIMAX(2, 1, 2)	Log Likelihood	-108.841
Date:	Tue, 20 Oct 2020	AIC	229.682
Time:	17:39:34	BIC	240.108
Sample:	0	HQIC	233.504
	- 43		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	1.7130	1.869	0.916	0.359	-1.950	5.376
ar.L1	-1.5685	0.291	-5.393	0.000	-2.138	-0.999
ar.L2	-0.6472	0.275	-2.354	0.019	-1.186	-0.108
ma.L1	1.8095	21.279	0.085	0.932	-39.896	43.515
ma.L2	0.9991	23.477	0.043	0.966	-45.015	47.013
sigma2	9.4056	219.554	0.043	0.966	-420.912	439.724

Ljung-Box (Q):	40.00	Jarque-Bera (JB):	0.53
Prob(Q):	0.47	Prob(JB):	0.77
Heteroskedasticity (H):	0.61	Skew:	-0.09
Prob(H) (two-sided):	0.37	Kurtosis:	2.48

SARIMAX Results

Dep. Variable:	y	No. Observations:	43
Model:	SARIMAX(0, 1, 0)	Log Likelihood	-102.997
Date:	Tue, 20 Oct 2020	AIC	209.994
Time:	17:20:28	BIC	213.469
Sample:	0	HQIC	211.268
	- 43		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.8643	0.468	1.848	0.065	-0.052	1.781
sigma2	7.8990	1.313	6.015	0.000	5.325	10.473

Ljung-Box (Q):	15.76	Jarque-Bera (JB):	10.99
Prob(Q):	1.00	Prob(JB):	0.00
Heteroskedasticity (H):	2.37	Skew:	0.75
Prob(H) (two-sided):	0.12	Kurtosis:	5.01

Figure 4.27, 4.28: ARIMA models summary (Up: Apple, Down: Microsoft)

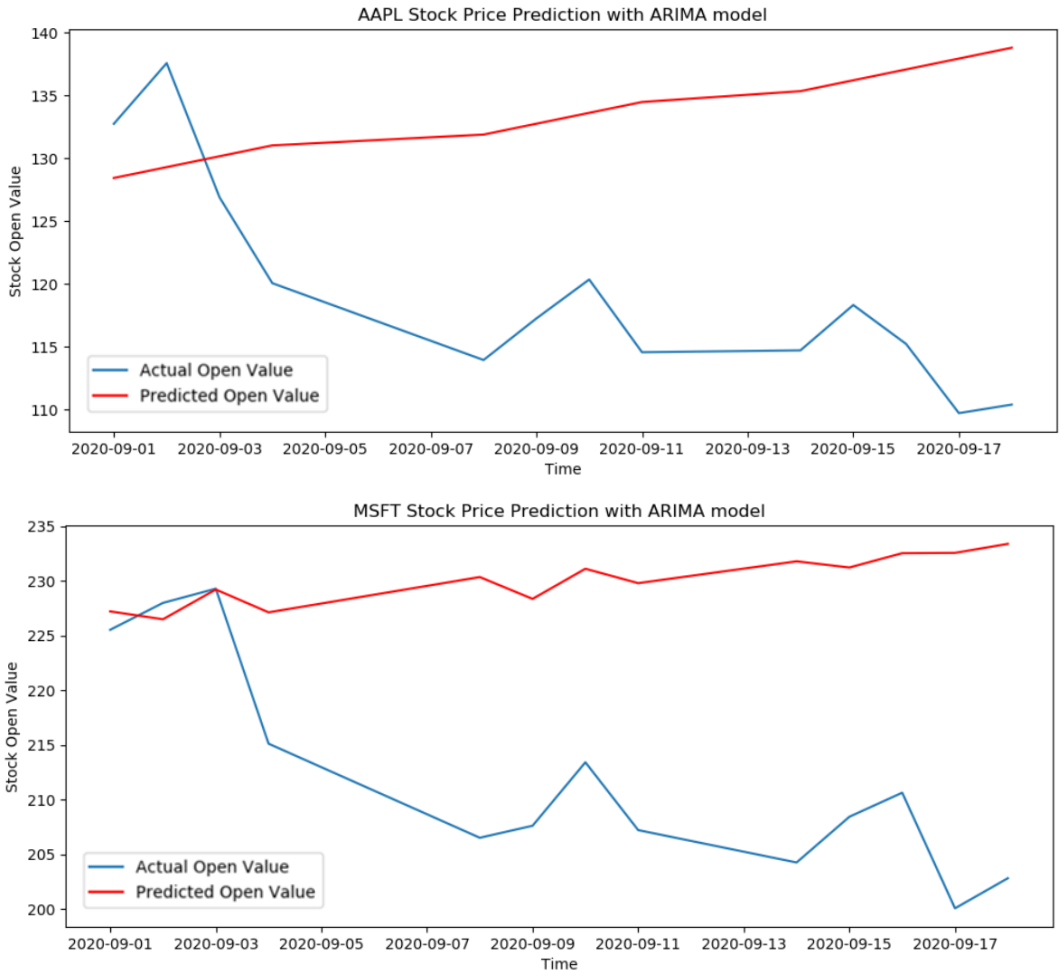


Figure 4.29, 4.30: ARIMA model predictions (Up: Apple, Down: Microsoft)

For the previous models, MSE=321.71 for Apple and MSE=922,29 for Microsoft. The minimum values we managed to reach is MSE=177.75 (MAE=4.99) for Apple and MSE=439.7 (MAE=3.82) for Microsoft. This time the forecast looks a lot better.

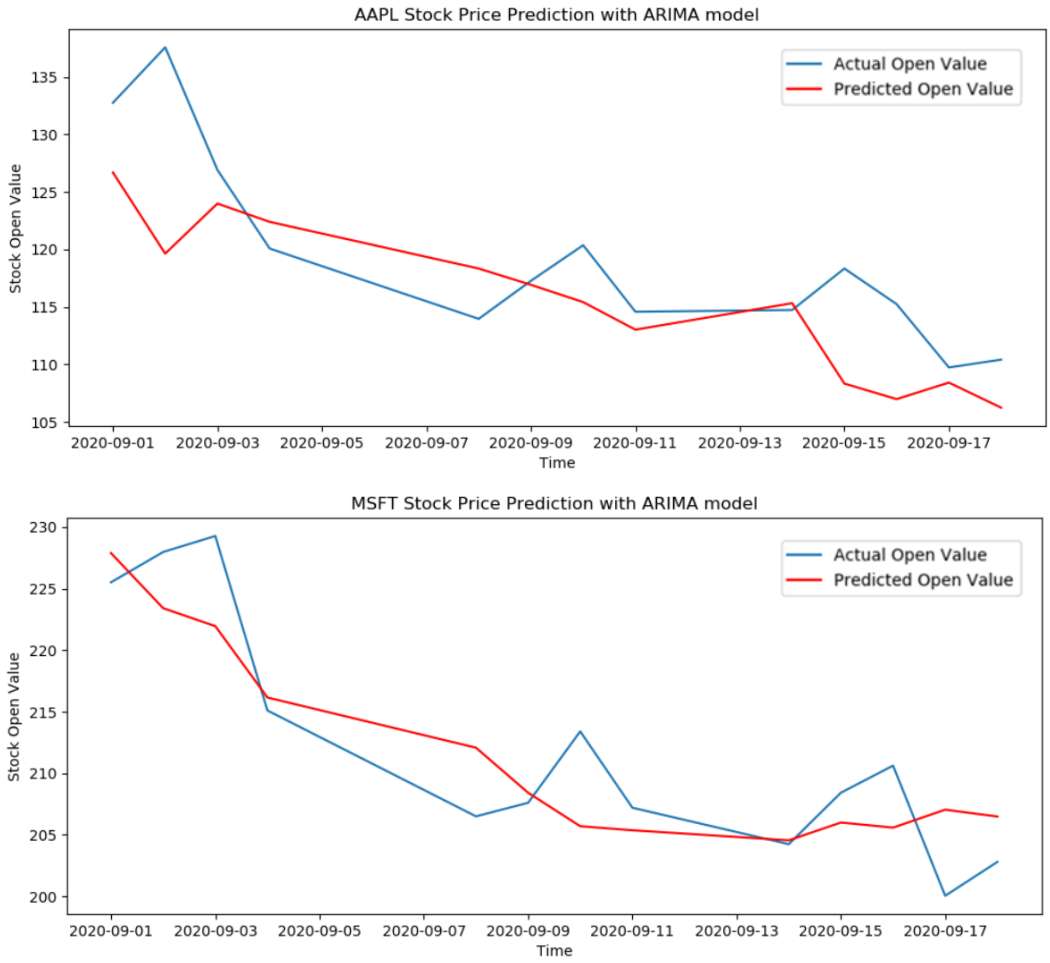


Figure 4.31, 4.32: ARIMA model predictions -min MSE (Up: Apple, Down: Microsoft)

Now, if we forecast the stock prices on the test dataset keeping 95% confidence level we can see that our model did quite handsomely, this time.

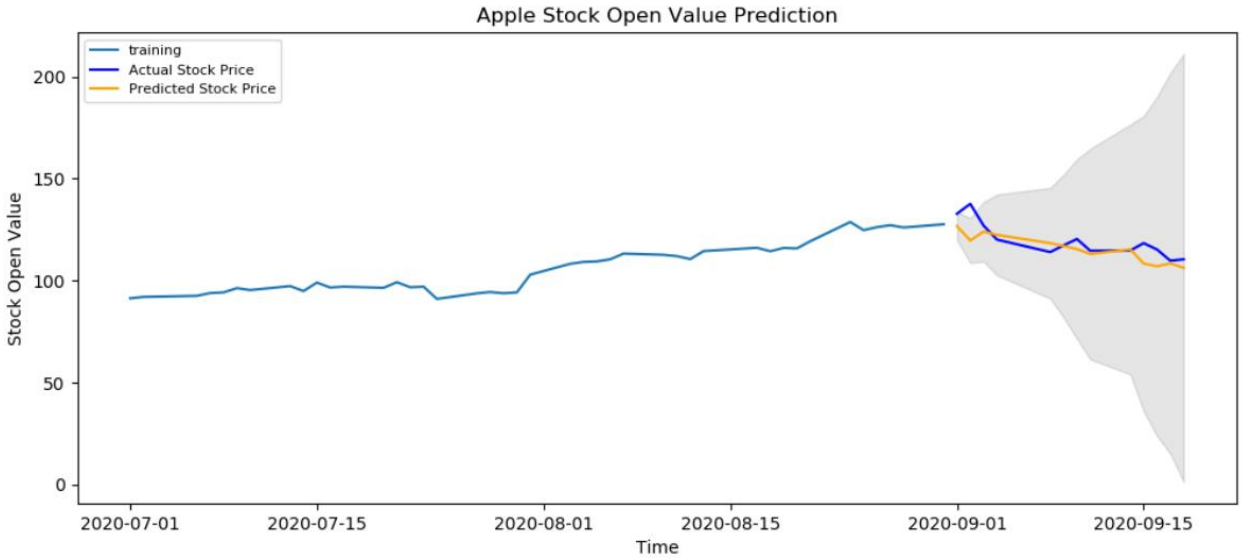


Figure 4.33: ARIMA model predictions -min MSE (Apple)

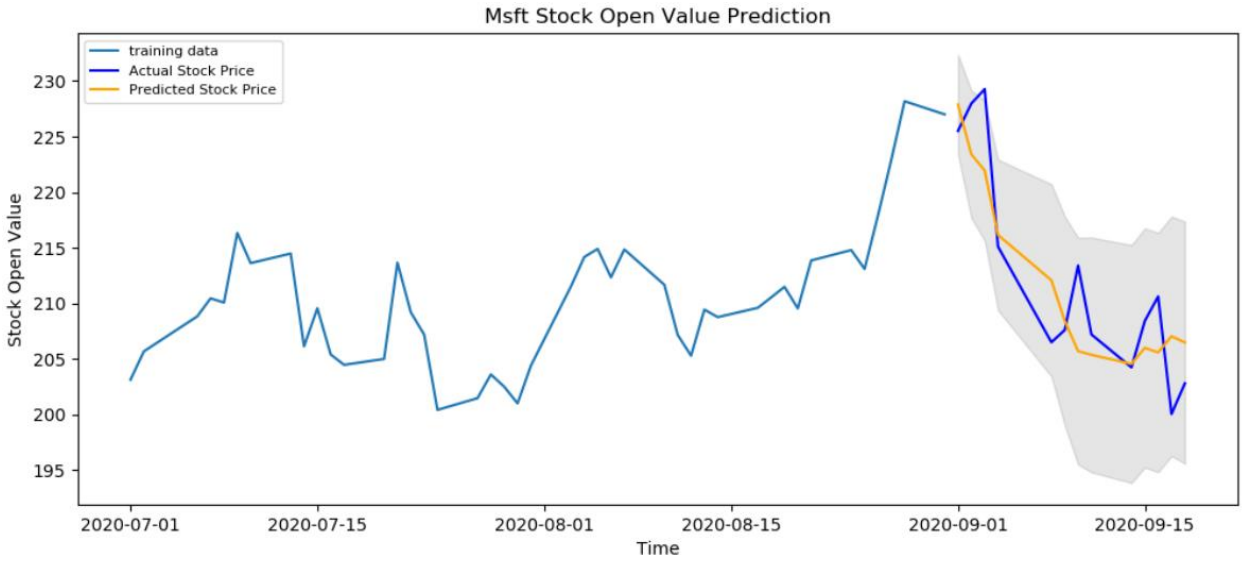


Figure 4.34: ARIMA model predictions -min MSE (Microsoft)

Finally, we should take a look in models' residuals (constant mean and variance).

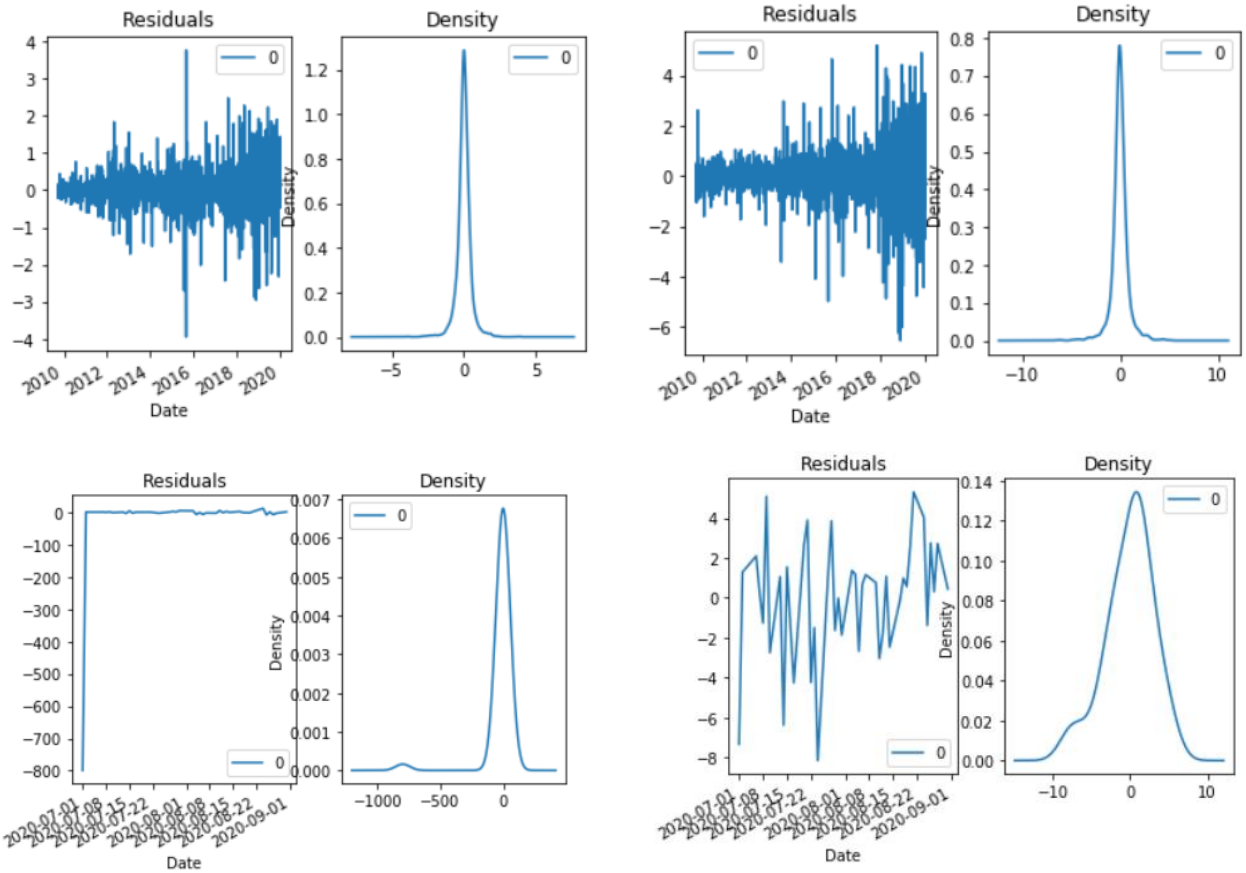


Figure 4.35, 4.36, 4.37, 4.38: ARIMA models residuals (Up: min AIC, Down: min MSE, Left: Apple Right: Microsoft)

From the plots we can ensure that there are no patterns with mean near zero and uniform variance.

Summarizing, forecast analysis with ARIMA may predict some respectful values under certain circumstances, especially in short term analysis. In long term analysis the model seems to predict correctly the overall monotony of the curve (positive or negative), but fails to predict local changes. However, stock values cannot be considered only as time series since they are affected from extraneous factors and econometric criteria. Moreover, there is no clear seasonality or trend in most occasions.

Conclusion

After conducting the above 4 implementations we end up with some thoughts about our business problem of stock price prediction. Using previous day's stock price when it comes to regression problem help us to predict very close to the real price using LSTM. It is important that model performs better than Arima when it comes to Apple, but worst when it comes to Microsoft and this is something that is correlated with standard deviation of open price where in Microsoft is higher.

On the other hand, using LSTM with Finbert embeddings and sentiment analysis for Classification, we have achieved to reach 60% accuracy on our best model, which is great, since its higher than the random choose of 0 or 1 (stock decrease or increase). Using this model, a trader who would invest for a long time his money using it, at the end of this long period, his/her portofilio would be marginally profitable.

Of course, there is not a simple recipe of predicting stock market, but getting different traffic sources and signals, not only stock historical prices would add additional information gain in our model. For example, as future work, we could improve performance, including more company's performance indicators as company growth in terms of employees, new offices, market capitalization and social media consumer's opinion. Additionally, we need to extend our traffic sources from different and reliable sources, finding maybe an automated way of scraping everyday news and store it to our data warehouse. Last but not least, a weird but clever approach, is to use for the company do we consider, competitive company's stock as an additional feature since might be correlated.

Timeplan

Start	End	Task
8/5/2020	8/12/2020	Research for Subject. Subject chosen predict stocks movements from twitter sentiment analysis
9/1/2020	9/5/2020	Subject was transformed. We have started collecting previous years tweets with Library GetOldTweets but Twitter has aborted our connected endpoint

9/6/2020	9/8/2020	Then, we have found a new data source to extract sentiment, an online news Dataset on Kaggle. Started explore articles and papers about the subject
9/11/2020	9/18/2020	We have found the different implementations to be done, and we have separate tasks for every team member. Additionally, data cleaning and transformation was started
9/19/2020	9/26/2020	Everybody works on his implementation/tasks
9/27/2020	10/4/2020	Get in touch to aggregate functionalities /implementations, solve bugs, see each other code
10/5/2020	10/11/2020	Run our models and get results. Get everything together in a unified report

Members/Roles

Name	Role	Task
Mallios Charalampos	ML Engineer Data Analyst Business Analyst	• Implementation 1: LSTM with Historical Stock Prices and TextBloB Sentiment Score • Data Analysis.
Patrikalos Vangelis	ML Engineer Data Analyst	• Implementation 2: LSTM with Finbert sentence embeddings
Protopapas Nikolaos	ML Engineer Data Analyst	• Implementation 3: LSTM with Finbert sentiment analysis
Ralis Dionysios	ML Engineer Data Analyst	• Arima Model - Forecasting Using Time Series Analysis

Contact Person (phone & e-mail)

Name	Phone	Mail
Mallios Charalampos	6944137907	Mallioscharis@gmail.com

Bibliography

Michael Grogan, (August 28, 2019), SARIMA: Forecasting Seasonal Data with Python and R (Online), https://medium.com/analytics-vidhya/sarima-forecasting-seasonal-data-with-python-and-r-2e7472dfad83.

Cory Maklin, (May 25, 2019), ARIMA Model Python Example — Time Series Forecasting (Online), https://towardsdatascience.com/machine-learning-part-19-time-series-and-autoregressive-integrated-moving-average-model-arima-c1005347b0d7.

Jason Brownlee, (January 9, 2017), How to Create an ARIMA Model for Time Series Forecasting in Python (Online), https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/.

Jason Brownlee, (March 24, 2017), How to Make Out-of-Sample Forecasts with ARIMA in Python (Online), https://machinelearningmastery.com/make-sample-forecasts-arima-python/.

www.hackdeploy.com, (November 20, 2018), Augmented Dickey-Fuller Test in Python.

Jacob Stallone, (November 9, 2017), Time Series Forecast : A basic introduction using Python (Online), https://medium.com/@stallonejacob/time-series-forecast-a-basic-introduction-using-python-414fcb963000.

Selva Prabhakaran, (February 18, 2019), ARIMA Model – Complete Guide to Time Series Forecasting in Python (Online), https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/.datamites.com, ARIMA in Python - Time Series Forecasting Part 1 & 2, https://www.youtube.com/watch?v=D9y6dcy0xK8.

Nagesh Singh Chauhan, (January 20, 2020), Stock Market Forecasting Using Time Series Analysis, https://www.kdnuggets.com/2020/01/stock-market-forecasting-time-series-analysis.html

https://medium.com/@Currie32/predicting-the-stock-market-with-the-news-and-deep-learning-7fc8f5f639bc

[1] Textblob Library: extract sentiment from sentence :https://textblob.readthedocs.io/en/dev/

https://www.thepythoncode.com/article/stock-price-prediction-in-python-using-tensorflow-2-and-keras

https://towardsdatascience.com/getting-rich-quick-with-machine-learning-and-stock-market-predictions-696802da94fe

http://davidanastasiu.net/pdf/papers/2019-MohanMSVA-BDS-stock.pdf?fbclid=IwAR3fTE1lO_wYSUWdelBKyEXdan0CauOJ5aw7ULg2pnVAxJMpjfyQZJ1DjwM

http://davidanastasiu.net/pdf/papers/2019-MohanMSVA-BDS-stock.pdf?fbclid=IwAR3fTE1lO_wYSUWdelBKyEXdan0CauOJ5aw7ULg2pnVAxJMpjfyQZJ1DjwM

http://github.com/huggingface/transformers

http://github.com/yya518/FinBERT

https://towardsdatascience.com/adaptive-and-cyclical-learning-rates-using-pytorch-2bf904d18dee

https://github.com/bckenstler/CLR

https://github.com/Currie32/Predicting-the-Dow-Jones-with-Headlines/blob/master/Predict_Dow_with_News.ipynb

https://pypi.org/project/finbert-embedding/

https://medium.com/prosus-ai-tech-blog/finbert-financial-sentiment-analysis-with-bert-b277a3607101

https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

https://huggingface.co/bert-base-uncased

https://medium.com/analytics-vidhya/sentiment-classification-with-bert-embeddings-56607b45daad