CSC240
Final Project Report

Relationship between Stock and Unpredictable Events

Yingcan Chen
Xinyu Hu

As the Covid-19 pandemic hit the whole world extremely hard, the stock market has become a "Bermuda Triangle," even the best investor cannot predict the direction to go. Both my teammate and I are very interested in the stock market: we both invest some of our money into the market. As we enjoy the process of analyzing a company's economic performance, we always find it time-consuming. Also, humans are too easy to be influenced and would make irrational decisions: we were easily trucked by the falling price because we could not stop loss. Thus, we came up with the idea to build a project that can "predict" the stock price. Although, as we know, it's impossible to predict stock price accurately based on current technology because there are too many things that will affect the stock price and it's impossible to take them all into account, we want to build something that can give investors an idea on how the stock might behave. At first glance, it sounds like many people have already done things like this, using some sort of algorithm to predict the stock price. But we want to take an extra step: we want to let the algorithm takes the unpredictable event into account, such as the COVID-19, to make a more accurate prediction rather than just using past stock prices.

In order to find out an effective solution to predict the stock, it's essential to understand what causes the problem. As we take the Covid-19 as an example, the chaos in the stock market right now begins when the OPEC and Russia did not reach an agreement to cut down the oil production at the beginning of March. In order to force Russia into the agreement, the OPEC countries start to increase the oil production massively. This action leads to the crush crude oil price as it decreases as much as 35% percent. This may not sound horrible enough, but as oil prices start to drop dramatically, it drives the whole market down. What's even worse is that this sudden drop causes a great panic among the investors as they start to dump their stocks altogether. The chain reaction, along with the COVID-19 pandemic, triggers four meltdowns in

the next two weeks, which has never happened before. One of the most important indexes of the stock market, Dow Jones Index, drop below 20,000.

It is not possible to know what will happen in politics, or foreign relations, or what's the next natural disaster. However, if we watch the crude oil price closely, we will probably be able to notice, before the market behaves, that a great collapse is going to come. In reality, the oil price is not the only thing we can watch to make the right prediction. There are a lot more numbers we need to take into account to predict investment.

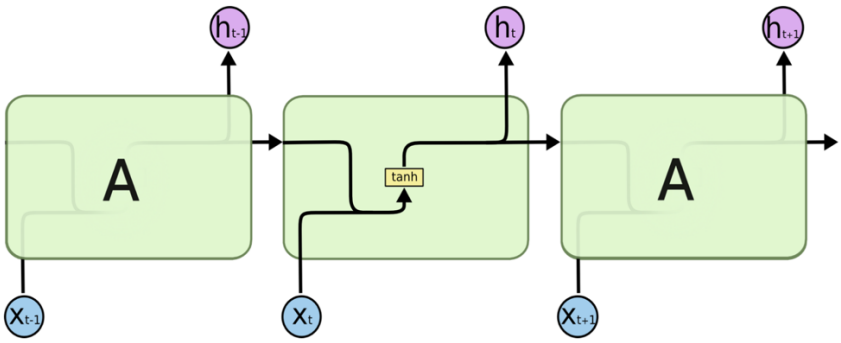
Data and procedure

In order to make some predictions about the stock price, we first want to collect our stock data. As for stock prices, the data is easy to collect. To make it easy for everyone interested in this project, we used Yahoo as our data source since it's easily accessible. Besides, we decide to choose the close-price as our standard price to predict because close-price reflects what happened in the market of that day. If some big things are occurring, the close price will reflect that while the open price will not. We don't take after-market and pre-market hours into account because the trading volume will become very small during the after-market, which means it is not representative. When there are big things that happen to the company or the market, the effect will show in the close price. So, it is not necessary to consider after-market hours and pre-market hours.

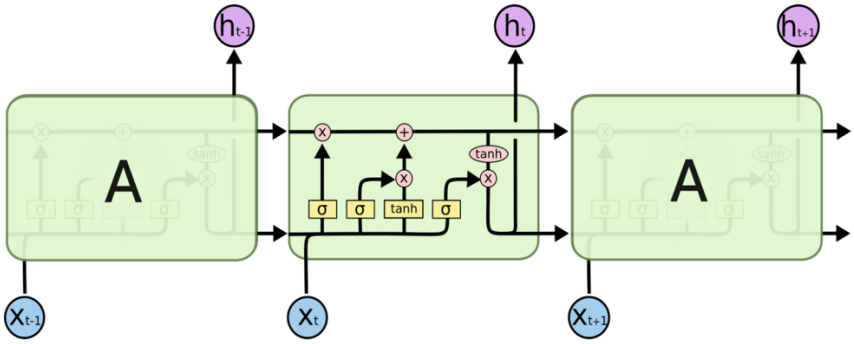
In order to build an accurate prediction about the stock, the first step is to create a basic prediction model. We start by implementing a proper neural network. We think the neural network is the route to go because we have an enormous amount of data, which is great to train

the algorithm. Moreover, it can be applied to any stocks, whereas some of the algorithms may be limited to a specific situation.

LSTM is the neural network that was implemented in this case. The reason is that predicting stocks is like predicting a sequence, and we need to remember the past information to predict the future. Although a normal recurrent neural network also seems to fit in this scenario, LSTM has the advantage of learning long-term dependencies regardless of the gap between relevant information, while RNN can only retrieve the relevant information for the short-term.



RNN



LSTM

The difference is shown in the graphs above as there is more mechanism in LSTM to control for what value it should keep and what it should abandon: it can keep essential information for the future.

We chose five stocks in total to implement our program, and they are Amazon(AMZN), Apple(AAPL), Disney(DIS), Starbucks(SBUX) and Microsoft(MSFT).

(1) We first grab all the close stock price from Yahoo Finance of Amazon from 2010-01-01 to 2020-05-02 and eliminated other columns but keep the "Close" column for the model.

```
df = web.DataReader('AMZN', data_source='yahoo', start='2010-01-01', end='2020-05-02')
```

(2) Then we tried to clean the data. In this case, we rescale all the data to make them between 0 and 1. The benefit is that it will speed up the process as we have such a large amount of data to train in the model. The next thing we did is that we keep 80% of the data we download for training and then use the model to predict the rest of the dates. We used 90 days as our timestamp. Originally, we were thinking about 60 days, but after running the program, the squared error of prediction using 60 days is not small enough and doesn't meet our needs. In the case of AMZN, the squared error will be more than 100 while it will remain around 70 with a timestamp of 90 days.

(3) The third step is we feed the data into the model, for every 90 days of stock data, we put it in the training data list x, we put a corresponding price of the 91 days like shown in

the graph so that we put the two list of data into the training model for it to do machine learning.

```
train_data = scaled_data[0:training_data_len,:]
x_train = []
y_train = []



for i in range(90,len(train_data)):
    x_train.append(train_data[i-90:i,0])
    y_train.append(train_data[i,0])
```

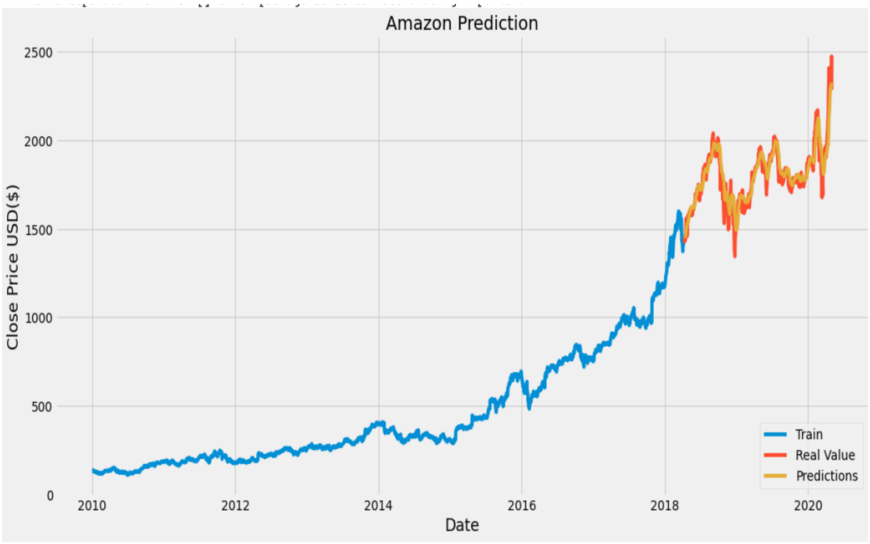
(4) After training the data, remember that we only use 80% of the data to train, then we are going to predict the rest of the data and compare it with reality.

```
squarederror = np.sqrt(np.mean((predictions-y_test)**2))
squarederror
```

```
65.3108430130059
```

As we can see here, in this case, we got the squared error equal to about 65, which means, on average, the difference between our prediction price and the actual price is 65 dollars. This is not perfect but considering that the price of Amazon stock is over 2200 dollars per stock, it's about a 2% of error.

Let’s look at the graph:



Though the model did not predict every rise and fall perfectly, it's able to catch the overall trend of the stock. While it's not sagacious to entirely rely on this model for investment decisions, it's able to serve as a reference for possible future change.

(5) After testing the model, we apply it to make some predictions about the real future value of the stock. As for predicting the future, the problem we have is that the error tends to get big if we want to look further into the future. This is because when we are doing the testing, though we are predicting something unknown, the data we used for prediction is 100% correct. However, when we try to look further into the future, we are using our previous prediction to predict the future. By each round, the error gets bigger and bigger. Thus, we limit the future prediction to only one week, which means we only predict one week of the stock price.

Here is the test we tried to predict the stock price of Amazon in one week. The prediction was made on 2020/5/4

```
Price prediction for the next 7 trading days
[[2349.8667]]
[[2346.7415]]
[[2343.4404]]
[[2340.1968]]
[[2337.0767]]
[[2334.1038]]
[[2331.2793]]
```

IMPROVEMENT

In order to improve the accuracy of the algorithm, we try to implement the economics index. To achieve this effect, as we mentioned before, we want to take into consideration more variables to determine for the unforeseen events and make certain changes in response. In other words, when certain things such as the COVID-19 happens, it will affect the index in the stock market such as S&P500, Dow Jones. Because these indexed usually are stable, if we detect some degree of dramatic change in these indexes, we can assume there are unforeseen events happen. Then, based on the extent of the change, we adjust the prediction accordingly.

In the code, we compare the index we choose, such as S&P500, and compare it's value with yesterdays' value when running. When the difference is significant, for example, a considerable drop of 400, the program will turn down its prediction accordingly.

The benefit of doing it this way is that we can always update the program to include important indexes at the time. In other words, it can always include the most important indexes and make the most accurate prediction.

```
significant drop in SP, turn down expectation
Price prediction for the next 7 trading days
[[1813.8668]]
[[2203.0334]]
[[2169.8943]]
[[2144.933]]
[[2124.6433]]
[[2106.985]]
[[2090.9333]]
```

As we can see in the real testing, if we feed a fake S&P500 number into the program and let it believe the S&P 500 dropped significantly in a day, the program responded by predicting the price to be lower. The point is that even though it detects an abnormal drop in the market, it still remembers the previous performance of the Amazon stock and is still giving an upward prediction; instead of simply predict it will drop because of the mishap. We included "S&P 500 index, Dow Jones index" in this program to do this unforeseen event bias correction.

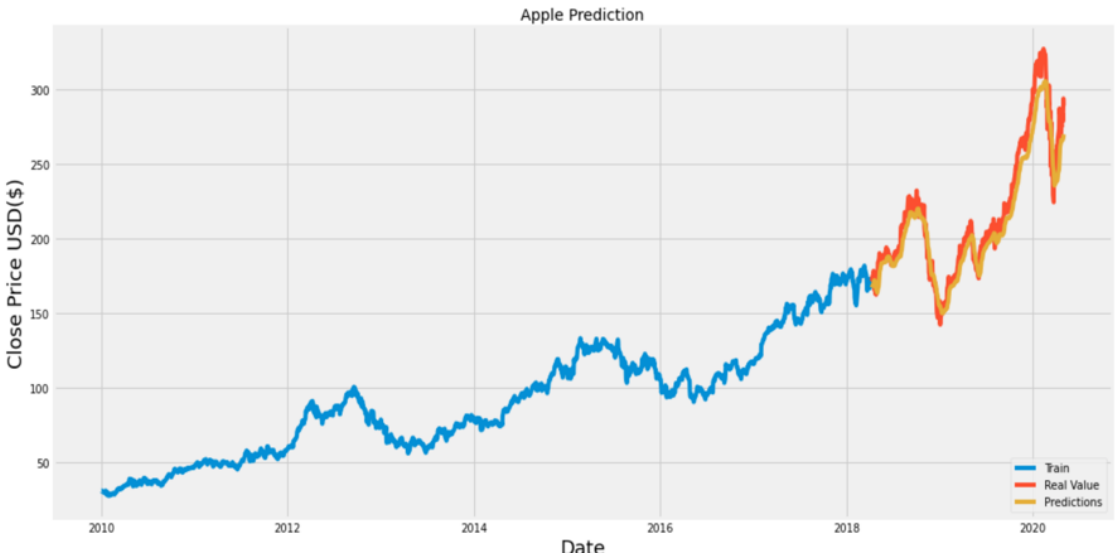
Further Testing

In order to further test the performance, we run the program on four more company stocks to see how it performed.

Apple

```
squarederror = np.sqrt(np.mean((predictions-y_test)**2))
squarederror
```

```
11.247768749430682
```

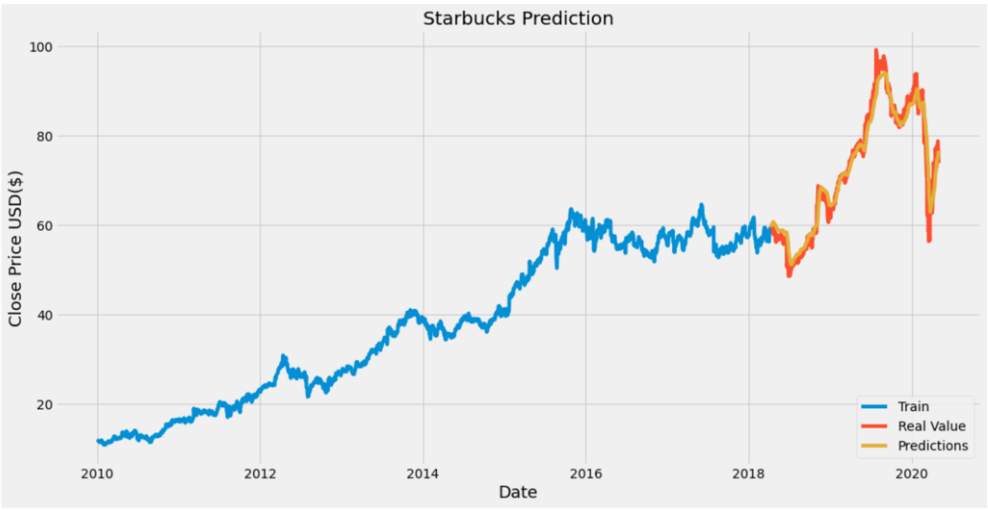



```
Price prediction for the next 7 trading days
[[271.45]]
[[271.08817]]
[[269.84113]]
[[268.11365]]
[[266.14215]]
[[264.06125]]
[[261.94577]]
```

Starbucks

```
squarederror = np.sqrt(np.mean((predictions-y_test)**2))
squarederror
```

```
2.7232733027711795
```

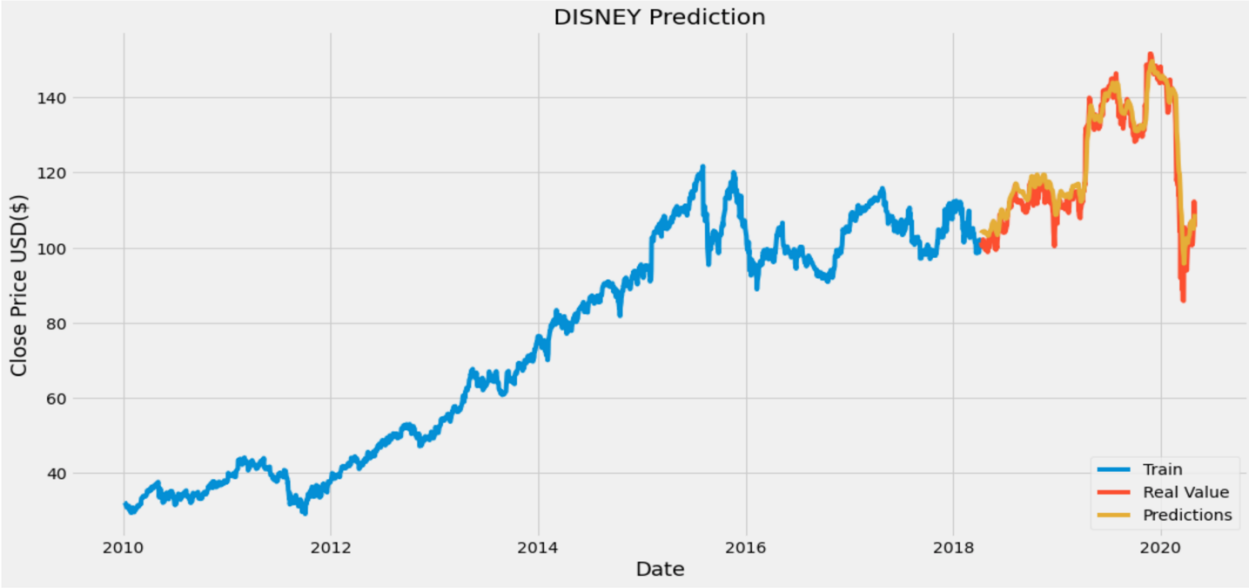



```
Price prediction for the next 7 trading days
[[76.47938]]
[[76.48488]]
[[76.50896]]
[[76.541695]]
[[76.577675]]
[[76.61428]]
[[76.650314]]
```

Disney

```
[54] squarederror = np.sqrt(np.mean((predictions-y_test)**2))
     squarederror
```

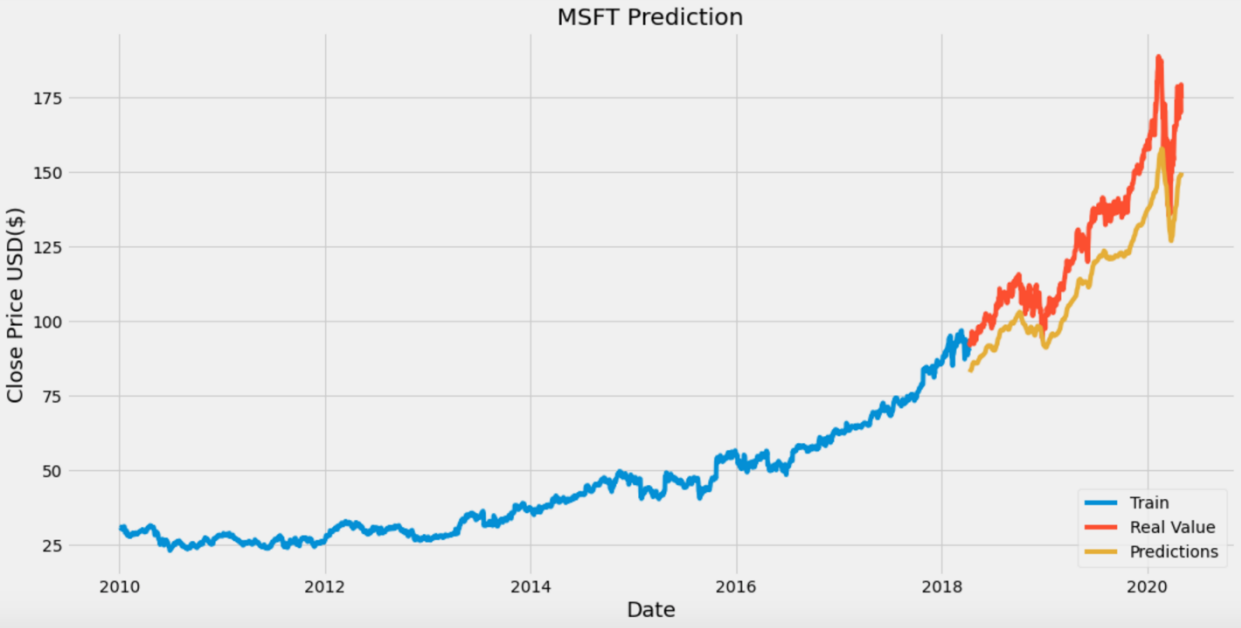
```
3.8736761990078863
```




```
Price prediction for the next 7 trading days
[[109.48062]]
[[110.16582]]
[[110.900154]]
[[111.650154]]
[[112.39903]]
[[113.138466]]
[[113.864265]]
```

```
squarederror = np.sqrt(np.mean((predictions-y_test)**2))
squarederror
```

```
15.976475987542715
```




```
Price prediction for the next 7 trading days
[[150.04282]]
[[148.50787]]
[[146.07153]]
[[143.20004]]
[[140.15395]]
[[137.07692]]
[[134.04579]]
```

Conclusion:

Based on all the running information, we think that the program predicts the trend of the stock. Pair with the mechanism that tries to eliminate the effect of unpredictable events, we believe the tool can provide some useful suggestions regarding the investment plan. However, as we can see in the graph, the neural networks are far from perfect, so it should not be used as the only thing for people to make their decision.

In the hope of making this project sustainable, we came up with some possible improvements. When we are writing the project, we discover that our model is not perfect in predicting the influence of unforeseen events. We are not able to tailor different kinds of offset strategy for different companies and different stocks but was only able to apply a general strategy instead. We are thinking about adding elements to the program in the near future. The way it will work is that we want to categorize each stock based on its volume, profit, business type, and tailor the offset for each company. Thus it will be able to provide a more accurate prediction about the influence of sudden events such as natural disasters, political issues, and more. As time goes on, this platform needs a constant update; for instance, there may be someday that crude oil is not one of the most important resources. The crude oil price may no longer have such a great impact on the stock market as it has now. As we can put in more variables and data into consideration, it will make a more accurate prediction.

Code Running Instruction:

The data we used comes from "Yahoo" Finance, so there is no database attached.

We start our code on the Google colab environment and later transform it into pycharm.

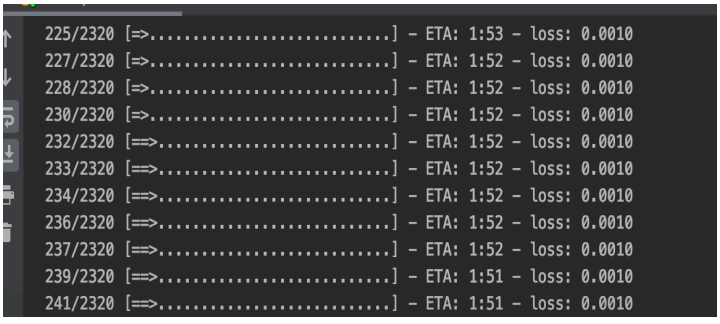
1. The package we used are in the screenshots shown below.

```
from datetime import date, timedelta
import datetime
import math
import numpy
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
```

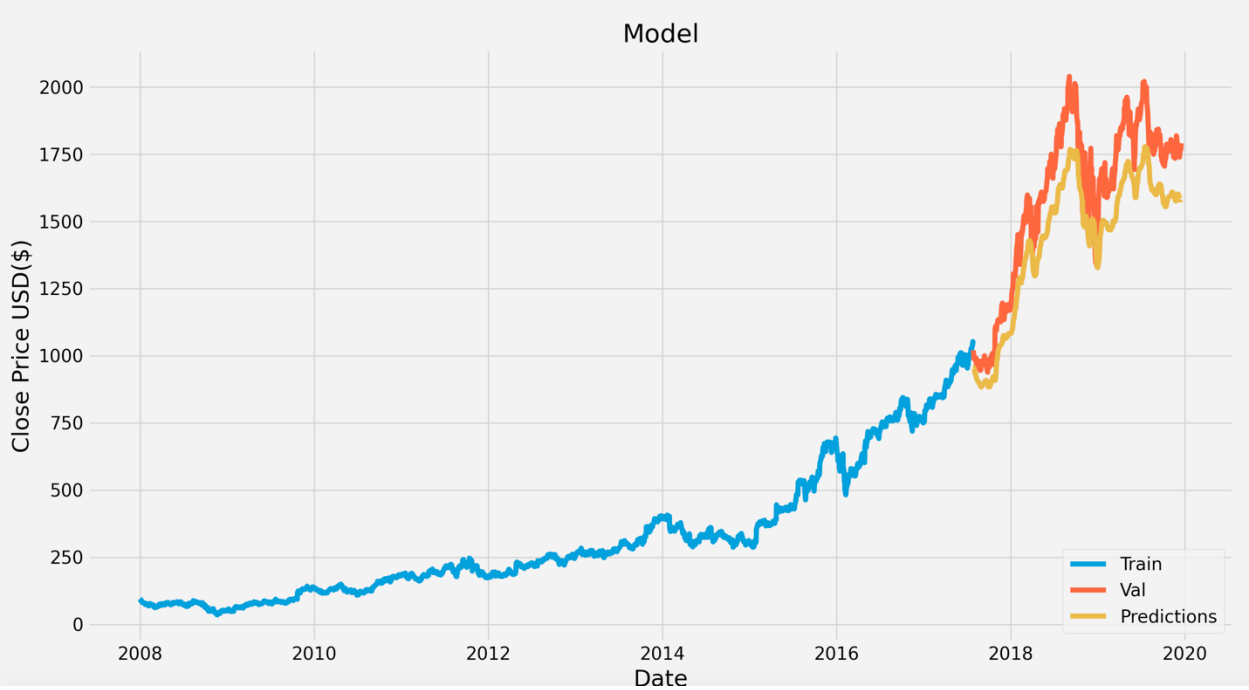
The stock Prediction preset is set to Amazon (AMZN).

2. When first running the project, a graph of the stock price will be shown like in the blow.

3. Close the graph, and you can see the model training.



4. After the training process is done, you will see a prediction test result shown in the graph.



5. Close the graph, and you will see the predictions of 7 days in the future.

```
Price prediction for the next 7 trading days
[[2070.944]]
[[2042.0859]]
[[2003.2714]]
[[1959.7201]]
[[1914.5215]]
[[1869.3889]]
[[1825.2324]]
```

6. You will also be able to see the prediction when taking other indexes into account.

```
below is prediction with event bias correction
S&P 500 today
                 Close
Date
2020-05-08  2929.800049
DOW Jones Today
                  Close
Date
2020-05-08  24331.320312
S&P500 yesterday
                 Close
Date
2020-05-06  2848.419922
Dow Jones Yesterday
                  Close
Date
2020-05-06  23664.640625
[[81.38012695]]
[[666.6796875]]
Price prediction for the next 7 trading days
[[2070.944]]
[[2042.0859]]
[[2003.2714]]
[[1959.7201]]
[[1914.5215]]
[[1869.3889]]
[[1825.2324]]
```

Bibliography

AAII. (n.d.). *The Top 10 Economic Indicators: What to Watch and Why*. Retrieved from American Association of Individual Investore: https://www.aaii.com/investing-basics/article/the-top-10-economic-indicators-what-to-watch-and-why

Nayak, A. (2019, 1). *Stock Buy/Sell Prediction Using Convolutional Neural Network*. Retrieved from towarddatascience: https://towardsdatascience.com/stock-market-action-prediction-with-convnet-8689238feae3

Olah, C. (2015, August 27). *Understanding LSTM Networks*. Retrieved from colah's blog: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Xavier, A. (2019, 1 22). *Predicting stock prices with LSTM*. Retrieved from Medium: https://medium.com/neuronio/predicting-stock-prices-with-lstm-349f5a0974d4