Syed Adil Ali

Capstone Project Report

I. Definition

A) Project Overview

The dataset that was presented for this problem was the Bitcoin value for last year. Mainly from January 1st 2017 to December 31st 2017. The features that were presented in this dataset were when Bitcoin was Opened, when it was Closed, what was the highest value on that day, what was the lowest value that day, what was the adjusted price for that day, as well as the volume for that day. Since Bitcoin is open 24/7, I decided that Bitcoin Opens was 12:00:00 and Bitcoin Closes at 11:59:59.

Bitcoin is a cryptocurrency. This is almost like stocks where people could invest their money so that they can make more money. It almost like stocks but it isn't, its a digital currency in which encryption techniques are used to regulate the generation of units of currency and verify the transfer of funds, operating independently of a central bank. So the goal is to buy low and sell high like stocks to make money.

There are many machine learning problem that are just like mines. Instead of using this algorithm on bitcoin, you can actually put it in the stock market like SP500, Apple,Google, Microsoft, etc. You still want o make money, so you would invest money in stocks and sell it when you want. The best thing about the stock market is that there is a set timing when the stocks are open and when the stocks are closed, rather than bitcoin which is open 24/7. With that in mind it would be easier for you to buy and sell stocks and make money.

B) Problem Statement

The problem was clearly defined, as well as the reader should be able to figure out what I'm trying to do. If they don't I am trying to figure out the next day prices for when Bitcoin is Open.

I have thoroughly discussed on how I will attempt to solve the problem. If they haven't figured it out. I can tell you. What I did was first import the data into Jupyter. Next I got all the graph I needed, as well as statistic value for each one. After that I normalized the data. Next thing after that was set up a bench mark model, which was my linear regression model. Next I did was build a neural network model to help make

the model even more efficient, and try to be a little bit more accurate it can by teaching the data from the train test model and repeating the process multiple time so that it can learn correctly rather than used the linear regression model.

The anticipated solution has been clearly defined because I wrote in the program specially “The next day prices for when the market open is: “ as well as “Accuracy: “ since these are the main thing the reader will be specially looking for . So the reader should be able to figure out what the result that they are looking for.

C) Metrics

The metric that I’ve chosen to performance is accuracy. Accuracy is a big thing when it comes to investing money. The higher the accuracy the better is it, because when your investing money, you want to make money not lose money.

So its better if the accuracy is closer to 100% because you wouldn’t lose any money at all. You would know ahead of time when it invest and when to sell.

II. Analysis

A) Data Exploration

Since Bitcoin is open 24/7, I decided that Bitcoin Opens was 12:00:00 and Bitcoin Closes at 11:59:59. This would help me for when the markets opens and when its closed. I mainly used the value for when the markets is Open to predict the next day outcome. The dataset can be provided if you go to yahoo finance and click on Bitcoin USD, and adjust the time period, if your not able to do I’ve provided the link here: https://finance.yahoo.com/quote/BTC-USD/history?period1=1483257600&period2=1514707200&interval=1d&filter=history&frequency=1d.

Since the dataset was presented for this problem, I was able to get the statistic value for each feature. I calculated the mean, minimum, maximum and the standard deviation for each feature of the dataset. What I was able to notice is the the value for when market is opened and closed is the same because they are one second apart, so they had to be the same. Other than that everything else was completely different.

```
The Statistic of When the Market is Open

The minimum price of when the market was open was: $ 785.429993
The maximum price of when the market was open was: $ 19346.599609
The mean of when the market was open was:  3945.74981474
The standard deviation of when the market was open was:  3951.07964561
```

```
The Statistic of When the Market is Closed

The minimum price of when the market was closed was: $ 785.429993
The maximum price of when the market was closed was: $ 19345.490234
The mean of when the market was closed was:  3981.07152553
The standard deviation of when the market was closed was:  3981.70966239


The Statistic of When the Market was at its Highest

The minimum price of when the market was at its highest: $ 826.429993
The maximum price of when the market was at its highest: $ 19870.619141
The mean of when the market was at its highest:  4124.41227024
The standard deviation of when the market was at its highest:  4180.16477
853


The Statistic of When the Market was at its Lowest

The minimum price of when the market was at its lowest: $ 739.549988
The maximum price of when the market was at its lowest: $ 18750.910156
The mean of when the market was at its lowest:  3765.69586917
The standard deviation of when the market was at its lowest:  3681.583905
51

The Statistic of When the Market for the Adjusted Closed

The minimum price of the Adjusted Closed : $ 785.429993
The maximum price of the Adjusted Closed : $ 19345.490234
The mean of the Adjusted Closed:  3981.07152553
The standard deviation of the Adjusted Closed :  3981.70966239


The Statistic of Volume

The minimum price of the Volume:  10214368
The maximum price of the Volume:  6245731508
The mean of the Volume:  502642902.567
The standard deviation of the Volume: 748665681.327
```

There wasn't any abnormalities or weird characters that needed to be addressed in the dataset. There weren't any missing values since I got all the values from yahoo finance. There also wasn't any outliers because Bitcoin made an increasingly steadily growth last year, which is why there wasn't any outliers.

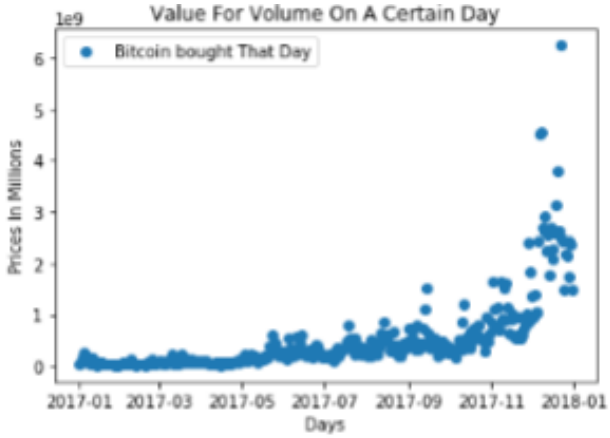
B) Exploratory Visualization

The most relevant characteristic is that is started below $2500 and thought the year its got up to the point where its value was about $20,000 and then started to dip back down. This is frequent when you look at the graph for when its open, for when its closed, the highest point that day, the lowest point that day, and the adjust volume. All 5 features basically have the same graph. Later on I mainly focused on the Open value to get the prediction, so you will be able to see the graph for Open only later on, as well as the graph results for when I normalized the data for each feature.

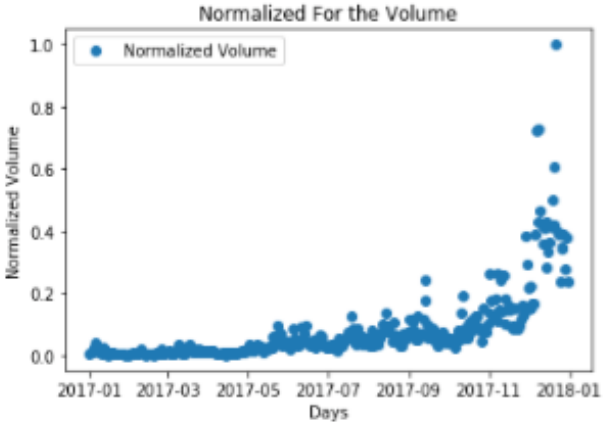
I believe that the visualization has been through analyzed since the first thing I did was to introduce the graph first for each feature. Later on I then got all the statistic value for each of the feature.

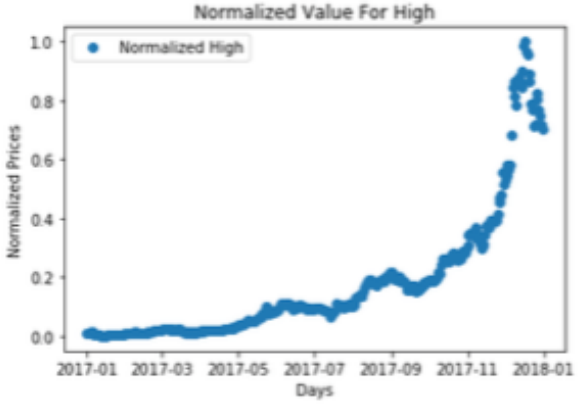
For each plot the data the axioms, title and the data would be clearly defined. If you couldn't tell, The X coordinate are the dates, the Y coordinate is the prices, and the data on the graph is the prices for each date.

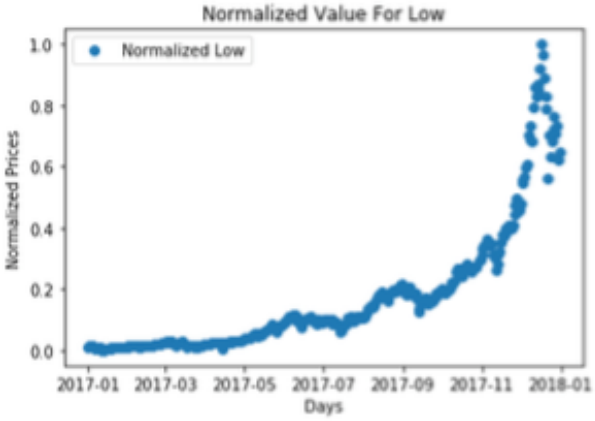
These are the graph of the dataset when not normalized:



These are the graph when the data is normalized:







C) Algorithm and Technique

The algorithm that I'll be using is Linear Regression and Neural Network. I'll be using the Linear Regression for my benchmark model, while I used my Neural Network model has my main model to use. The default variable that I used was the feature Open. Since the bitcoin is open 24/7 I had decided to focus on Open which open at 12:00:00 everyday.

The technique for Linear Regression will allow me to predict the next day price by having a line being fitted through the graph and will predict the next day price. You would do this by getting the data, decide what feature you would want, normalized a data, split the data to an 80-20 split so that you can have the algorithm learn 80% of the data while testing on 20% of the data, create the regression line and predict the accuracy and the future pricing. For neural network it would be more or less the same but it will allow me neural network to learn what it's doing and predict future prices, you would pretty much do the same as linear regression until you split the data to an 80-20 split, once that is done you would convert an array of values into a dataset matrix and reshape. Next you would reshape the input and create and fit an LSTM model. Once you done with that you can make the predicting from your model and inverse the numbers because you had normalized it. Once all that is done, then you can print out the predicted price.

The input data will be the Open feature. It will be handled very thoroughly from the beginning. I'll be loading the data first and then start my algorithm with using Time Series for Linear Regression and Neural Network, since they are in dates. I wouldn't want to do a test train split with random dates everywhere, always being different.

D) Benchmark

The results had been provided for the benchmark model. Since the benchmark model was linear regression i was able to measure the accuracy. The main goal is see how well Linear Regression performance is how good the accuracy is.

To get the accuracy you would need to compute the R2 Score. The R2 score tells you how accurate it is. You must first import R2 Score library from sklearn. Once that is done and you have trained the model with linear regression you can type reg.score(X_test,y_test) to get the R2 score. With that you can get the accuracy of the model.

III. Methodology

A) Data Preprocessing

To run both Linear Regression and Neural Network Long Short Term Memory I must normalized the data because the prices for the data are in the thousand.

So to make it easy for the program I normalized the data to make it easier. To do that I had done a MinMax Scaler to normalized the data. Once I had finished model was done training with the normalized data, I had converted it back to its original numbers in thousands using scaler.inverse. I had done that for all the feature: open, close, high, low, adjusted close, and volume. When it came down to run the data through the algorithm I only had normalized the data for open only.

From the data exploration sections I had checked to see if theres was any abnormalities of the data that needed to be check and I had saw none. I believe that the main reason for this was because the data was slowly and steadily increasing and if it did decrease it wasn't as big one, it was a very minimal one. Which is why there isn't any outlier or any other abnormalities in the data set.

I had to preprocess the data so that I can run the algorithm, especially when it came down to the LSTM neural network.

B) Implementation

The algorithm that I will be using are Linear Regression and Neural Network. My benchmark model is the Linear Regression while my main neural network model I'll be using a Long Short Term Memory has my main model. The default variable that I will be using is the data used from the feature Open.

The technique I used for the linear regression was noted was very simply. I first imported all the libraries I needed to use. Next I printed out the last 4 data entries by using print (bitcoin.tail()). Next I decided to focus on only the focus feature which was open by writing bitcoin=bitcoin[['Open']]. After that I sent a variable to predict the future by writing Future=int(1), and make an extra column for that by saying bitcoin['Prediction']=bitcoin[['Open']].shift(-Future). Next I normalized the data. Next I set a variable to the last day and then removing the last day from X, you can do this by writing X_Future=X[-Future:] and right after you would write X=X[:-Future]. After that I define the Y value, by writing y=np.array(bitcoin['Prediction']) and right after that you would write y=y[:-Future] . Once I've done that I start the linear regression with a train test split an create a linear regression as well as train it. You would have to do this in a TimeSeriesSplit since there's date included. You can do this by writing train_size = int(len(bitcoin) * 0.80) right after you would write train, test = bitcoin[0:train_size], bitcoin[train_size:len(X)]. The you would create a linear regression by writing reg=LinearRegression(). After that train it by writing reg.fit(X_train,y_train). I then needed to see how accurate it could be you can do this by writing accuracy=reg.score(X_test,y_test) and then print it out by writing **print** ("Accuracy of the Linear Regression is : ",accuracy). Finally I printed to the accuracy and the value for the next day by writing Future_Prediction=reg.predict(X_Future) and right after to print it out you would write **print** ("Tomorrow Bitcoin Will Open up at :" ,Future_Prediction).

For the neural network LSTM(Long Short Term Memory) I did it completely different. I first imported all the libraries. I then only focus on one feature which was Open by writing Bitcoin = pandas.read_csv('BTC-USD.csv', usecols=[1], engine='python', skipfooter=3) and on the next time write bitcoin = Bitcoin.values and on the next time write bitcoin = bitcoin.astype('float32'). Once that was done I normalize that dataset in doing so you would write scaler = MinMaxScaler(feature_range=(0, 1)) and on the next time write bitcoin = scaler.fit_transform(bitcoin). After that i did a Train Test Split by writing train_size = int(len(bitcoin) * 0.80) and on the next time write test_size = len(bitcoin) - train_size, train and on the next time write test = bitcoin[0:train_size,:], bitcoin[train_size:len(bitcoin),:] and on the next time write print(len(train), len(test)) . Next a converted the array of array of values into a dataset and then reshape it into X=t and y=T+1. You would do this by writing:

```
def create_bitcoin(bitcoin, look_back=1):
    dataX, dataY = [], []
    for i in range(len(bitcoin)-look_back-1):
        a = bitcoin[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(bitcoin[i + look_back, 0])
    return np.array(dataX), np.array(dataY)
```

. Next you would still need to reshape it so you would write something along the line: look_back = 1 and on the next time write X_train, y_train = create_bitcoin(train, look_back), and on the next time write X_test, y_test = create_bitcoin(test, look_back). That would convert your array vow values into data sets Then i had to reshape the input so that its goes from samples, time steps and then features you would write this by typing X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1])) and on the next time write X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1])). Then I created the model and fitted the model. Once the model was done I was able to make a prediction. An example of that would be:

```
#create and fit the LSTM Model

#import model
model = Sequential()

#Input Layer
model.add(LSTM(32,input_dim=1,return_sequences=True))

#Hidden Layers
model.add(LSTM(16,return_sequences=False))

model.add(Dropout(0.2))

#Output layer
model.add(Dense(1,activation='linear'))

#complier
model.compile(loss='mse', optimizer='rmsprop')
```

```
model.fit(X_train, y_train, batch_size=25, nb_epoch=250, validation_split=.15,verbose=1)
```

Since it was still normalized I then had to convert it to regular numbers by writing trainPredict = model.predict(X_train) and on the next line testPredict = model.predict(X_test). Once that was done I was able to then calculate the accuracy, mean square and the test score to get the accuracy you would write R2=r2_score(y_test[0],testPredict[:,0]), for the mean squared error you would write trainScore = math.sqrt(mean_squared_error(y_train[0], trainPredict[:,0])) and finally for the test score you would write testScore = math.sqrt(mean_squared_error(y_test[0], testPredict[:,0])). I was also able to calculate the next day predictions by writing futurePredict = model.predict(np.asarray([[testPredict[-1]]])) and after that would be futurePredict = scaler.inverse_transform(futurePredict), and once that was compiled I would go on to write print("Bitcoin price for tomorrow: ", futurePredict)

The input of the data was cleared when i typed Bitcoin= pandas.read_csv('BTC-USD.csv'), which meant that that the data for bitcoin has been successfully been loaded to Jupyter. Once that was done I was able to type a few line of codes to use focus on one feature and one feature only.

C) Refinement

There has been a few initial solution that I found online. When I did the bench mark model which was Linear Regression I saw a lot of the same model that goes through Linear Regression. For Linear Regression its very straight forward and simple so there weren't any changes at all. For Neural Network I saw a lot of different variation. For Neural Network 14 different types of variation when it came down to which kind of neural network I wanted to use. After that I had decided that I wanted to use

LSTM neural network for this project. For LSTM model majority of the code was almost the same except for when it came down to create and fit the model. For that I had to do a lot of trial and error to figure it what model would be best to use.

While writing the code I do believe that I may have been able to create an small improvement in the LSTM model. For the linear regression model there was no way for me to improve the model since it was very straight forward, but for the LSTM there were. It all depended on how i created the model. I believe I created a model that gave me a 92% accuracy. So I believe I may have improved on this model.

There were some intermediate and final solutions that I found the the process, and theres are some thing that I believe that could have been improved. For example the R2 score I believe could have been improved if I could have created a better model for both Linear Regression and for LSTM Neural Network. I also believe that for the final answer I could have tried been closer to the correct number, since for LSTM neural Network I was off by $300 and for Linear Regression I was off by $1300.

I believe the best thing to do if you are going to make any refinement into this model would be at my LSTM model. I believe that there could be better model than what I have build, from the input layer, to the output and when you write your model.fit() code. I believe for that, comes a lot of trial and error and I had to learn it the hard way because I had spend hours trying to figure the best model so that my accuracy score could get high, but it wouldn't get higher than 92%. So I believe that the one thing that could need some refinement.

IV. Results

A) Model Evaluation

I believe the at final model is pretty reasonable and made it possible to align with the solution expectation. I was at first though it was unreasonable possible to get an accuracy score of 80% or higher when I was building both the model, but luckily I was able to. I don’t need to add any final parameters of model because I was able to get an accuracy score of over 90%.

The model has been tested with various inputs, so that I could if the model generalized well to unseen data. I had tested it with new set of data that started from February 2017 to February 2018 and it gave me a very close accuracy of the March 1st price.

The model is robust enough for the problem. Small here and there changes won’t affect the results, but when you add more data, like I did for when Bitcoin was open till now, that when the accuracy went down and wasn't predicting it correctly. I believe that is because since I was getting a total of 5 years of data and its started off as $0.05 and went to $20,000 there were a lot of outliers, which messed up the model.

I do believe that this model can be trust. Mainly because of the accuracy score. It is above 90%, which is a good thing. Anything below a 90% especially in the stock market would be bad and you would be losing money. So since the accuracy is 92% than I believe that this model can be trusted.

B) Justification

I do believe that the final results is stronger that the benchmark results reported earlier. By the accuracy score you can see that it is not, but it is. That is because when i compared the prices for January 1st 2018 price when its open for both Linear Regression and LSTM neural network, Linear Regression was off by $1300 while the LSTM neural Network was off by $300. Which is why I believe that my final results are better than the benchmark results that was reported earlier.

I have analyzed the final solution thoroughly. The way to do this is to compare data from last year data to the value that you predicted. Another way to analyzed the data is to find out the R2 Score(accuracy score) as well as the mean squared error as well.

I believe the my final solution is significant enough to solve the problem, through the various things I talked about, as well as my emphasis of the accuracy score. With the accuracy score high enough I do believe that my final solution was able to solve the problem.

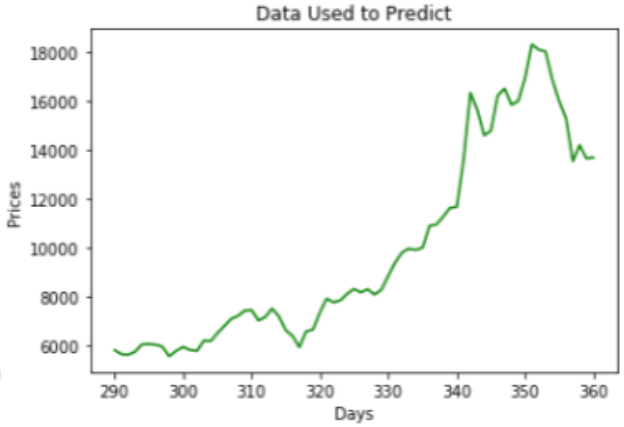
V. Conclusion

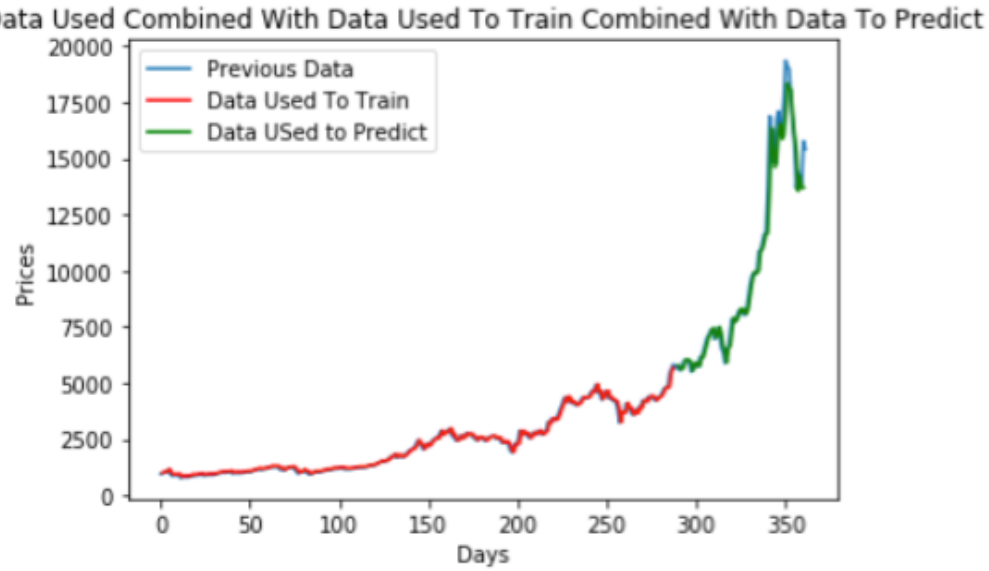
A) Free-Form Visualization

I had made a plot for everything when I started this project. I had made a plot for when the market opens, when the market closes, the highest point, the lowest point, the adjusted volume and its volume. These 5 features from the bitcoin I was able to graph it as well as when these data were normalized I had graph that part to. When it came down to the neural network I had graph a few stuff. I had graph last year data, data that was used to train, data that was predicted. I also compared the graph of all three of them with one another, so that you can get the idea of what's happening to the data.

With each data being plotted you can see what is happening. You can tell that from each plot the the graph goes higher and higher at a reasonable rate and once it reaches around 20,000 it starts to decrease steadily. You can tell from the graph that there's no spike of going super high one day or going super low another day, its going at a reasonable rate.

With the plot provided you can tell what each graph is talking about. I had put a title on each graph so a person can tell what they are looking at. I have also provided what the X-Axis is and what the Y-Axis is. The X-Axis on some graph represents the dates or the days, while the Y-Axis is telling you the prices for each point. I have also provided a legends telling you what the point represents because for some graph I had combine two or three data into one graph, so with that you can tell what each line color represents in the graph, so that no one could get confused.





B) Reflection

I personally thought throughout the entire process of the project that it was really interesting. I was able to predict the price of bitcoin in real time, which was really interesting. Most project and example are done by using fake data, but this time I got the whole data from yahoo finance.

It also interesting to get to predicted real time prediction on real time data. I was able to predict data and was off by a few hundred, not a few thousands. It was also interesting to see that better the accuracy the more closer the prediction would be.

I believe the most difficult part of this project would be building the neural network. I had to do a lot of research to figure out how I wanted to build my neural network model architecture. To me that was the hardest part, since I had to figure out what I wanted to put in my hidden layer, if I needed one. I also thought that trying to get the accuracy closer to 100% was pretty hard. I wasn't able to get to 100% but I believe that I got pretty close.

I believe the final model has fit my expectation for the problem. I had found out what the next day price would be and how accurate it would be. I believe that this model can be applied to the real life problem, especially for common people like me or you who would try to make more money. They can use the model to find out when the best time to invest money and when the best time to sell the bitcoin to make money since you tell from the model when the price would go high or low.

C) Improvement

I believe that there could be further improvement that could be made on the LSTM Model, when I was creating it, for this project. That is because you can play

around with it for a very very long time so that you can increase the accuracy score to be better than 92%. When I was playing around with it, I was not able to get a score better than 92%. The lowest accuracy score that I got was 48%. So I do believe that its possible to get an accuracy score of 92% which would greatly improve the algorithm.

When I was researching an algorithm to use for neural network, I did not know there were multiple algorithms that I can use to help me out, but I had decided to on LSTM Neural Network, which is also an RNN (Recurrent Neural Network) algorithm. If I had to do it again I would try do go a Neural Network with CNN(Convolutional Neural Network).

I do believe that my final solution is the new benchmark. I do believe a better solution exist because once you spend more time create an LSTM model and playing around with it you can score a accuracy percentage. So yes I do believe that there exist a better solution for this problem.