CAPSTONE 2:

BITCOIN PRICE PREDICTION

AUTHOR: Akinkunle Akinola

ABSTRACT

There has been a decent flow in people's engagement in the cryptocurrency market over the past years, around the world. Cryptocurrency investments are seen as a platform that yields rewarding returns. Every trader's goal is to be able to accurately predict the stock prices ahead of time or maybe predict the trend of the stocks. Stock market prediction is determining the future value of stock and this can be done using the previous and current price features. Now the question is, how does users' activities like notifications, alerts, messages etc. help in making accurate predictions? The purpose of this project is to build a forecasting model to predict the trend of cryptocurrency using LSTM, a recurrent neural network based on proprietary alert dataset. The business goal is for traders to utilize this forecasting model to maximize their return on investment capital and make accurate trading decisions.

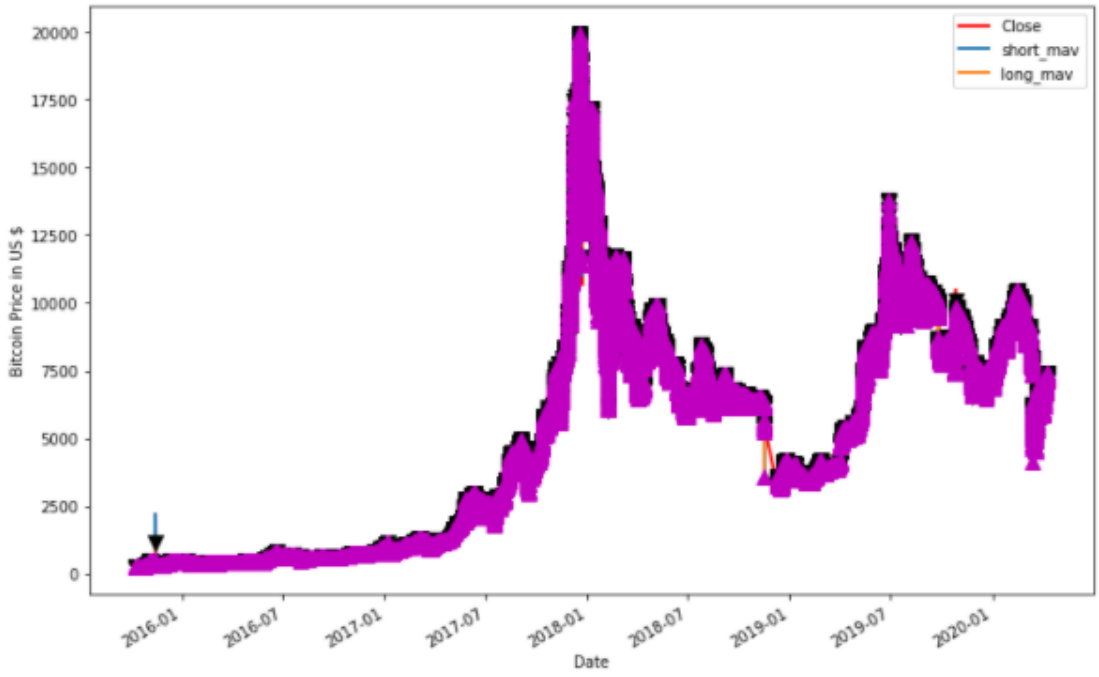
DATASETS

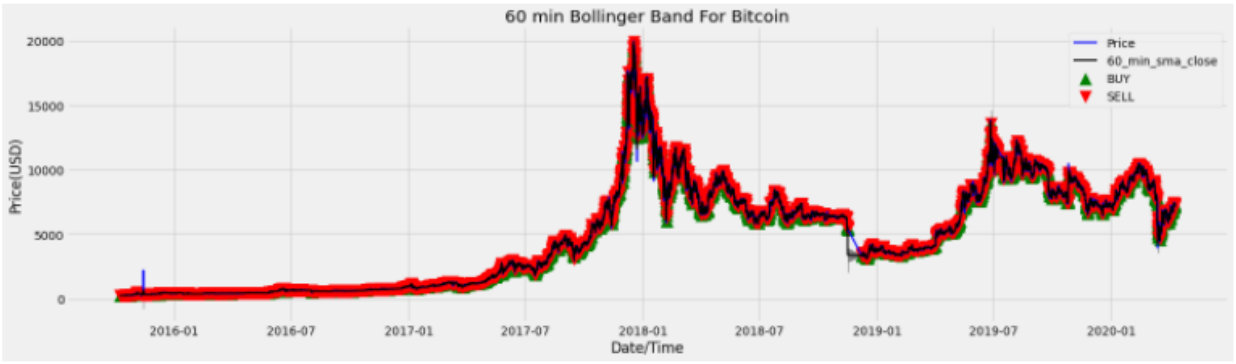
A market dataset and alert dataset will be used in the course of this project. The market dataset which was found on kaggle, has 2283519 rows of observations per minute between 2015/10/08 and 2020/04/09 and 8 features which makes it sufficient for this project to develop a good predictive neural network model.

FEATURE ENGINEERING

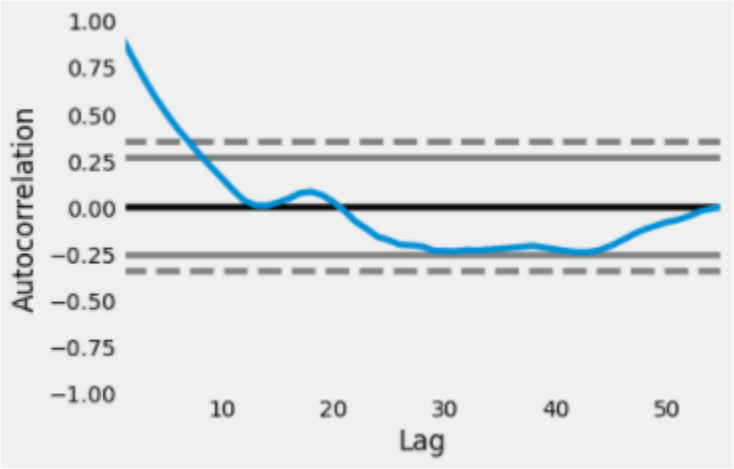
The below dataframe shows the summary of the features as well as the percentage change of each feature. In order to be able to predict the estimated future's percentage change in price, a simple moving average of absolute change columns were generated for each feature.

EXPLORATORY DATA ANALYSIS


Close
short_mav
long_mav
Bitcoin Price in US $
20000
17500
15000
12500
10000
7500
5000
2500
0
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
Date


60 min Bollinger Band For Bitcoin
Price
60_min_sma_close
BUY
SELL
Price(USD)
20000
15000
10000
5000
0
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
Date/Time

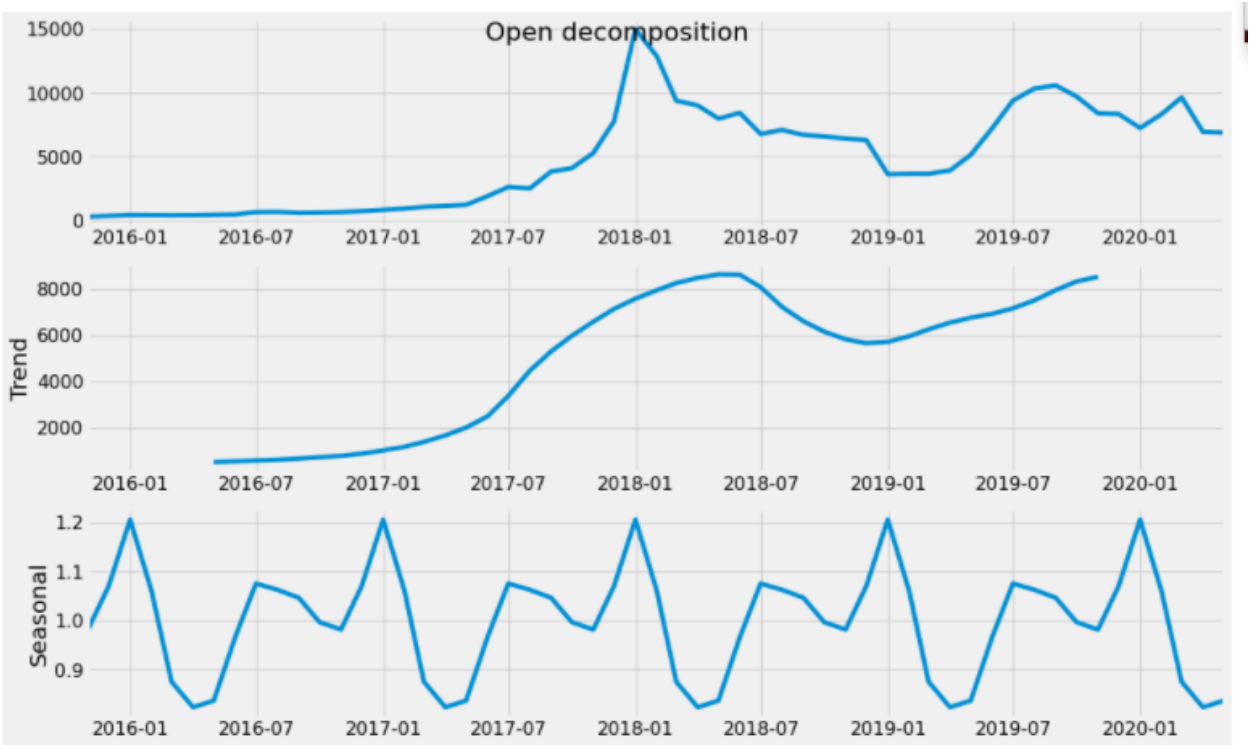
AUTOCORRELATION PLOT

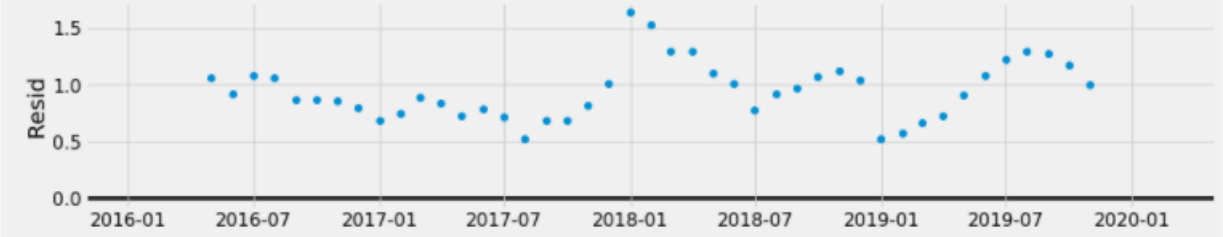


We can see from the plot that about more than 50% of the data line shows a significant correlation with time. So in this case, a Recurrent Neural Network output will be efficient in forecasting the prices.

SEASONAL PLOT

From the plot, we can see a high bitcoin price after the month of june then a slight drop. Similar to other years except for 2020 and 2015. In 2017, there was an upward trend in price.



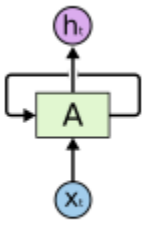


Trend, Seasonal, and Residual plots to show the recurring variations i.e. the ups and downs of time-series data, pattern across the season and the residual.

MODELING

A recurrent neural network (RNN)model called Long Short Term Memory (LSTM) was used for price prediction. RNN models allow us to operate over sequences of vectors, that is, Sequences in the input, the output, or both.

LSTMs are explicitly designed to avoid the problem of long-term dependency. Their behavior is basically the ability to remember information for long periods of time which they don't have to struggle to learn which is why they are good for predicting stock prices.



Recurrent Neural Networks have loops.

An LSTM model was built to see how well it will perform on the candle data (Open, Close, Low, High, and Volume) using the mean square error metric in the prediction of price. The simple moving average SMA of absolute change of the features was generated.

LSTM MODEL WITH 60 MINUTES SIMPLE MOVING AVERAGE OF ABSOLUTE CHANGE

ADDED FEATURES

```
df_features['60_min_sma_volume_change'] = df_features['abs_volume'].rolling(window=60).mean()
df_features['60_min_sma_open_change'] = df_features['abs_open_price'].rolling(window=60).mean()
df_features['60_min_sma_high_change'] = df_features['abs_high_price'].rolling(window=60).mean()
df_features['60_min_sma_low_change'] = df_features['abs_low_price'].rolling(window=60).mean()
df_features['60_min_sma_close_change'] = df_features['abs_close_price'].rolling(window=60).mean()
```

MODEL BUILDING

The model was trained on a 60 minute SMA of absolute change of absolute change . A relu activation function was used and an input shape of 5.

```
model_0 = Sequential()

model_0.add(LSTM(units = 60, activation = 'relu', return_sequences = True, input_shape = (X_train_0.shape[1], 5)))
model_0.add(Dropout(0.2))

model_0.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
model_0.add(Dropout(0.2))

model_0.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
model_0.add(Dropout(0.2))

model_0.add(LSTM(units = 120, activation = 'relu'))
model_0.add(Dropout(0.2))

model_0.add(Dense(units = 1))
```

The model was compiled using adam optimizer to modify the weight in the network to minimize the loss function while the mean squared error was used as the loss function.

```
model_0.compile(optimizer='adam', loss = 'mean_squared_error')
```

MODEL FITTING

The model was trained on 5 epochs with a batch size of 32.

```
model_0.fit(X_train_0, y_train_0, epochs=5, batch_size=32)
```

```
Epoch 1/5
46383/46383 [==============================] - 6468s 139ms/step - loss: 1.0320e-04
Epoch 2/5
46383/46383 [==============================] - 6204s 134ms/step - loss: 3.2335e-06
Epoch 3/5
46383/46383 [==============================] - 4766s 103ms/step - loss: 3.2340e-06
Epoch 4/5
46383/46383 [==============================] - 4823s 104ms/step - loss: 3.2307e-06
Epoch 5/5
46383/46383 [==============================] - 4801s 104ms/step - loss: 3.2347e-06

<keras.callbacks.History at 0x1dd0b840c10>
```

MODEL EVALUATION

The model evaluation shows a loss of 18837.38.

```
model_0.evaluate(X_test_0, y_test_0, batch_size = 32)
```

```
24976/24976 [==============================] - 760s 30ms/step - loss: 18837.3809

18837.380859375
```

```
print('MSE', metrics.mean_squared_error(y_test_0, y_pred_0))
```

```
MSE 0.26026684414236334
```

```
from sklearn import metrics
print('MAE in %', (metrics.mean_absolute_error(y_test_0, y_pred_0)/y_test_0.mean())*100)
```

```
MAE in % 0.051538468966995106
```

The mean absolute error indicates that we are deviating from the mean at about 0.05% when predicting the price at 60 minutes moving average of absolute change

LSTM MODEL WITH 30 MINUTES SIMPLE MOVING AVERAGE OF ABSOLUTE CHANGE

MODEL BUILDING

```
model_1 = Sequential()

model_1.add(LSTM(units = 60, activation = 'relu', return_sequences = True, input_shape = (X_train_1.shape[1], 5)))
model_1.add(Dropout(0.2))

model_1.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
model_1.add(Dropout(0.2))

model_1.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
model_1.add(Dropout(0.2))

model_1.add(LSTM(units = 120, activation = 'relu'))
model_1.add(Dropout(0.2))

model_1.add(Dense(units = 1))
```

The model was compiled using adam optimizer to modify the weight in the network to minimize the loss function while the mean squared error was used as the loss function.

```
model_1.compile(optimizer='adam', loss = 'mean_squared_error')
```

MODEL FITTING

The model was trained on 5 epochs with a batch size of 32.

```
model_1.fit(X_train_1, y_train_1, epochs=5, batch_size=32)
```

```
Epoch 1/5
46383/46383 [==============================] - 8467s 182ms/step - loss: 9.6318e-05
Epoch 2/5
46383/46383 [==============================] - 5914s 128ms/step - loss: 3.1784e-06
Epoch 3/5
46383/46383 [==============================] - 4857s 105ms/step - loss: 3.1793e-06
Epoch 4/5
46383/46383 [==============================] - 4905s 106ms/step - loss: 3.1739e-06
Epoch 5/5
46383/46383 [==============================] - 5544s 120ms/step - loss: 3.1748e-06

<keras.callbacks.History at 0x1dfd670bdf0>
```

MODEL EVALUATION

```
print('MSE', metrics.mean_squared_error(y_test_1, y_pred_1))
print('MAE in %', (metrics.mean_absolute_error(y_test_1, y_pred_1)/y_test_1.mean())*100)
```

```
MSE 3.413735659550736e-06
MAE in % 0.032710496910332204
```

The mean absolute error indicates that we are deviating from the mean at about 0.03% when predicting the price at 30 minutes moving average of absolute change

CONCLUSION

The results above show the mean square error and mean absolute error of 30 minutes and 60 minutes moving average of absolute change. Even though the LSTM performs better, there are lots of factors that might have affected the model such as the number of layers in the model, neurons in each layer, and the number of epochs trained.