Cryptocurrency Price Prediction with Python

Team Members: Gina Zazzi, Ellen Mo, Leah Nabangi

GitHub Repository: https://github.com/lifeisellen/AMS325GroupProject_Cryptocurrency.git

Background

Cryptocurrency is a financial medium that has been growing in popularity over the last five years. Many people, especially those of the older generation, detest cryptocurrency due to the improbable implication that crypto will eventually replace fiat currency. However, many recognize cryptocurrency as a part of a new, advanced economic model in which, through Blockchain technology, global financial inclusion is encouraged. Additionally, most people are reluctant to invest in cryptocurrency due to its volatile, unpredictable fluctuations in price. But what if there were a way to predict future prices?

Objectives

- ➢ Perform an exploratory data analysis, with visualizations, of the dataset
- ➢ Predict the future prices of a cryptocurrency of choice using various machine learning techniques in Python
- ➢ Select the best performing model upon completion of predictions

Techniques and Tools

Software: We chose to use Python for our project due to its user-friendliness, open-source nature, and variety of capabilities. The following libraries were used universally for the project:

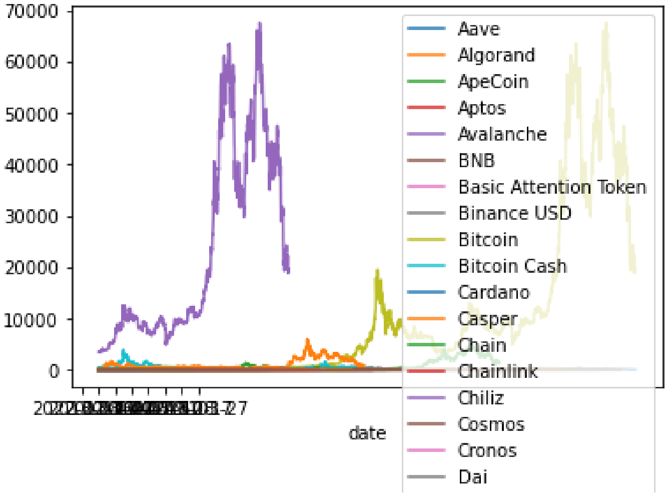
- ➢ pandas (for data preprocessing and analysis)
- ➢ numpy (for mathematical operations)

- ➢ matplotlib (for creating visualizations)
- ➢ sklearn (for predictive analysis)

Additionally, we used GitHub to keep track of our files and easily collaborate on the project. A link to our repository can be found at the top of this document.

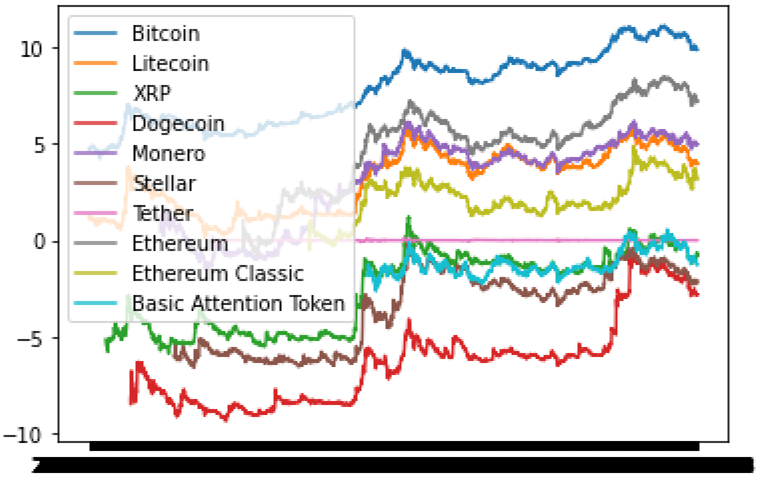
Exploration of Dataset: We retrieved a dataset from Kaggle which contains over 50 cryptocurrencies historical OHLC (Open High Low Close) data ranging from May 3013 to the present.

- ➢ First, we visualized the graphs for the closing prices of the cryptocurrencies (legend has been cropped for visual purposes).



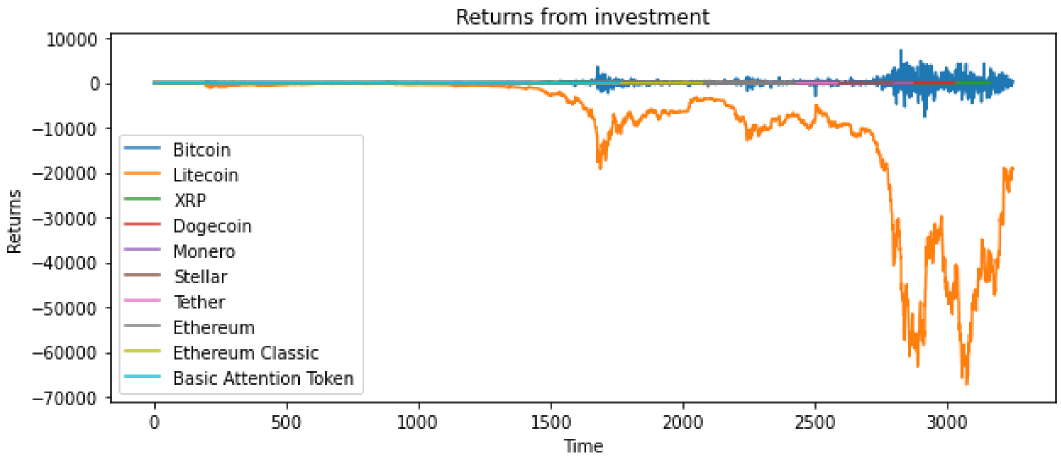
This plot shows that at the moment, Bitcoin has the highest closing prices. However, the rest of the observations were mostly clumped together in the bottom half of the plot.

- ➢ To make our observations clearer, we limited our plot to the 10 cryptocurrencies with the highest number of samples, which resulted in the figure below:



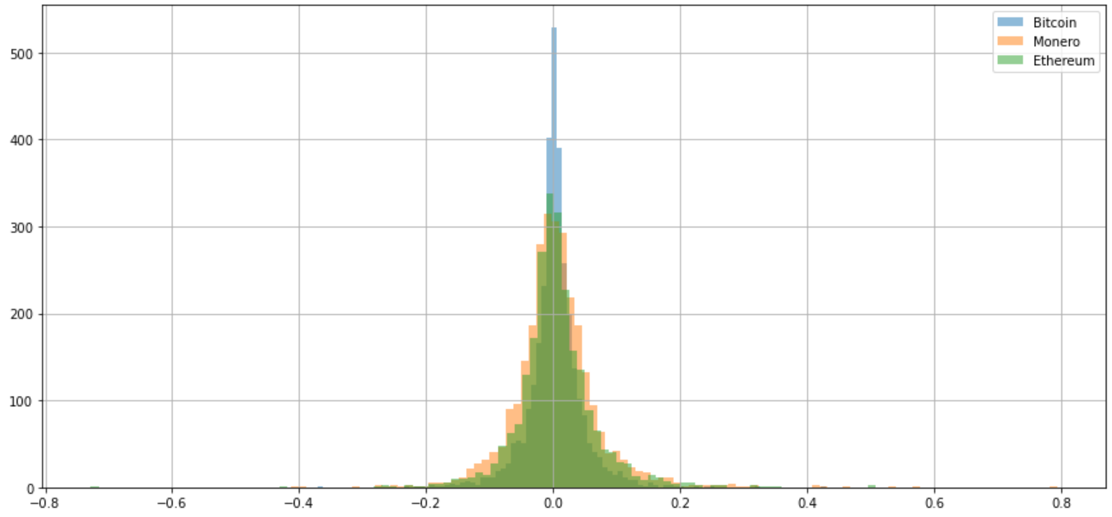
As shown in the plot, Bitcoin has the highest closing prices, then Ethereum, then Monero, then Litecoin, then Ethereum Classic. Thus, the stocks are valued in terms of their prices in that particular order.

- ➢ However, higher prices are not equivalent to higher returns. Since more people are investing in cryptocurrency, it is important to determine which cryptocurrency coins have the best returns. We calculated returns for the 10 cryptocurrencies with the highest number of samples by subtracting their opening prices from their closing prices. The returns are plotted in the below figure:



As shown in the plot, Bitcoin has the highest returns on investment so it would be wiser to store your assets in the form of Bitcoin.

- ➢ Lastly, we checked the volatility of the top 3 cryptocurrencies with the highest returns by plotting a comparison of the percentage increase in their values.



The widest margins indicate the most volatile coins, while those with the narrowest margins are the least volatile. Thus, from the graph, the percentage increase in closing price for Monero has the widest margins, indicating that Monero is the most volatile of the three coins, while Bitcoin has the most stable prices.

- ➢ Since Bitcoin has the most stable prices among the best performing cryptocurrencies, we concluded it would be best to predict its prices using various predictive models, namely Scikit-Learn regression methods, Long-Short-Term-Memory networks and the ARIMA model.

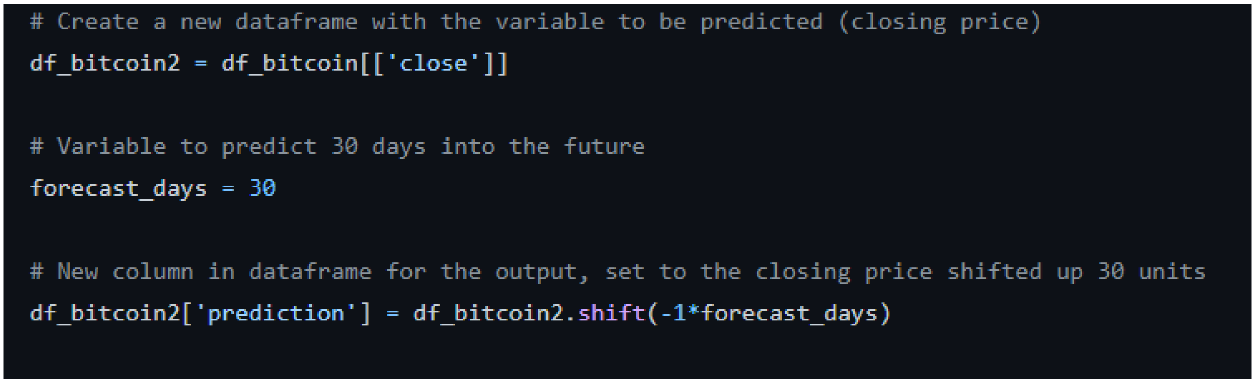
Predictive Models:

Regression Methods: These methods heavily rely on the machine learning library Scikit-Learn (sklearn) with the following modules: linear_model, preprocessing, model_selection, svm

- ➢ Data Preparation:

1. Create a new dataframe which only contains closing bitcoin prices

2. Create a variable to hold the number of days into the future to predict
3. Create a new column in the dataframe for the predicted output, which contains the closing prices shifted up 30 units

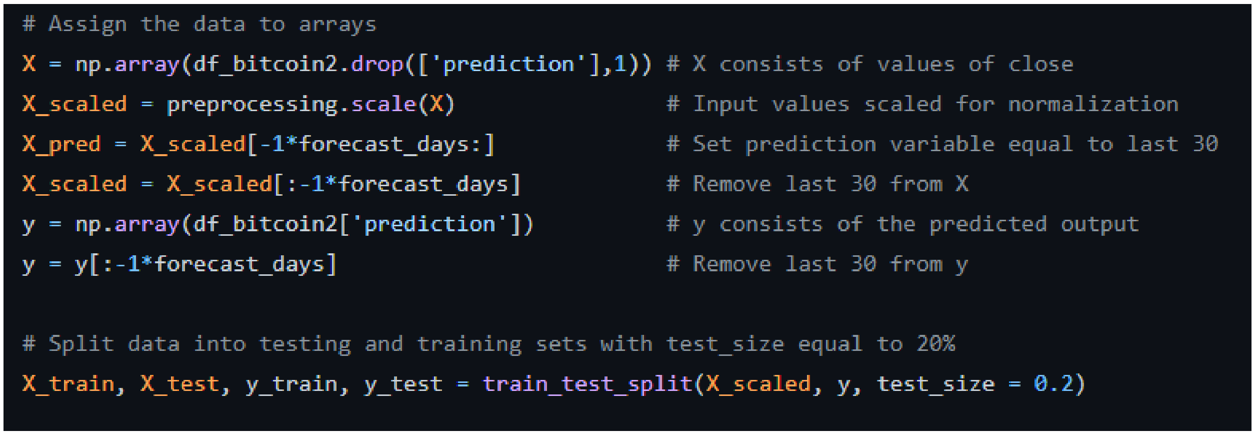


```
# Create a new dataframe with the variable to be predicted (closing price)
df_bitcoin2 = df_bitcoin[['close']]

# Variable to predict 30 days into the future
forecast_days = 30

# New column in dataframe for the output, set to the closing price shifted up 30 units
df_bitcoin2['prediction'] = df_bitcoin2.shift(-1*forecast_days)
```

4. Assign the data to numpy arrays X and y
   a. Use `scale` from the preprocessing package to normalize the data
5. Split the data into testing and training sets with test size equal to 20%

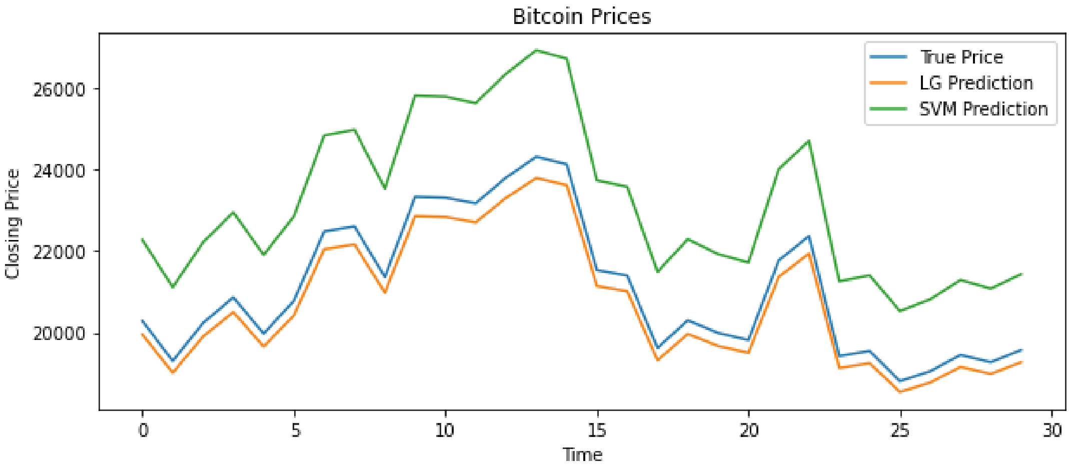


```
# Assign the data to arrays
X = np.array(df_bitcoin2.drop(['prediction'],1)) # X consists of values of close
X_scaled = preprocessing.scale(X)                # Input values scaled for normalization
X_pred = X_scaled[-1*forecast_days:]             # Set prediction variable equal to last 30
X_scaled = X_scaled[:-1*forecast_days]           # Remove last 30 from X
y = np.array(df_bitcoin2['prediction'])          # y consists of the predicted output
y = y[:-1*forecast_days]                         # Remove last 30 from y

# Split data into testing and training sets with test_size equal to 20%
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.2)
```

➢ Linear Regression: `LinearRegression()` fits a linear model to minimize the residual sum of squares between the observed targets and the targets predicted by the linear approximation.
   - After training and fitting the model, an accuracy score of 91.28% was achieved.

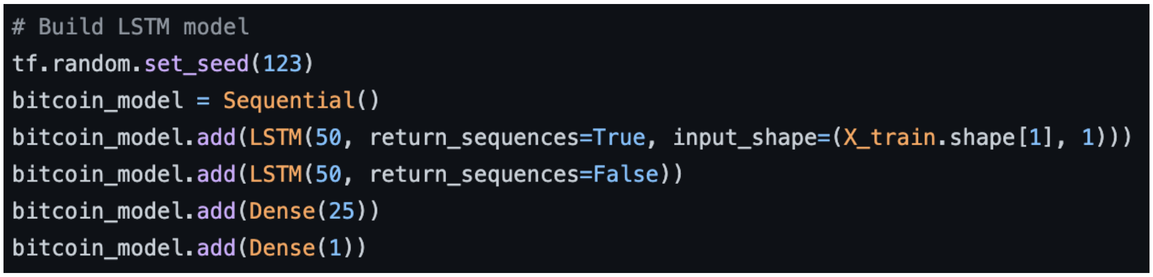
- Support Vector Machines: Support Vector Regression (SVR) uses the same principles as SVM classification (SVC) and similarly depends on a subset of the training data. However, SVM allows non-linear classifiers, which can be specified in the kernel type.
    - After training and fitting the model, an accuracy score of 92.6% was achieved.
- Visualized prediction: The following plot depicts the last 30 days of Bitcoin prices in our dataset, along with both models' predictions:



<u>LSTM(Long Short-term Memory):</u> LSTM stands for long short-term memory networks. It is a variety of recurrent neural networks (RNNs) that are capable of learning order dependence in sequence prediction problems. This technique requires terminal installation of tensorflow with pip.

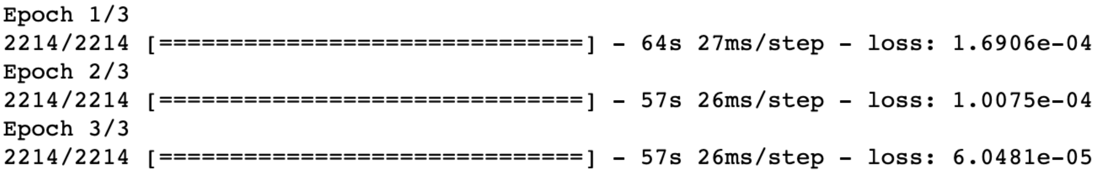
- Data Preparation
    - Remove null values from the new dataframe of closing bitcoin prices
    - Train the model, create train and test sets (calculate 70% of the dataset)
    - Use the MinMaxScalar to normalize all data ranging from 0 to 1

    - Create an empty list for feature data as X_train and label data as y_train (60-day window of historical prices and 60-day window of following prices)
    - Convert the X_train and y_train into numpy array
    - Reshape the data due to LSTM needs 3D
- LSTM model
    - Build an LSTM network and define a sequence model consisting of linearly stacked layers. Add a LSTM layer by giving it 50 network units. Returning sequences set True for another sequence of the same length, set False to only return the last output. Densely connected neutral layer



```
# Build LSTM model
tf.random.set_seed(123)
bitcoin_model = Sequential()
bitcoin_model.add(LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
bitcoin_model.add(LSTM(50, return_sequences=False))
bitcoin_model.add(Dense(25))
bitcoin_model.add(Dense(1))
```

    - Compile the model
        - bitcoin_model.compile(optimizer='adam', loss='mse')
    - Train the model (with each epoch, the dataset's internal model parameters are updated)
        - callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=2)
        - history = bitcoin_model.fit(X_train, y_train, batch_size=1, epochs=3)

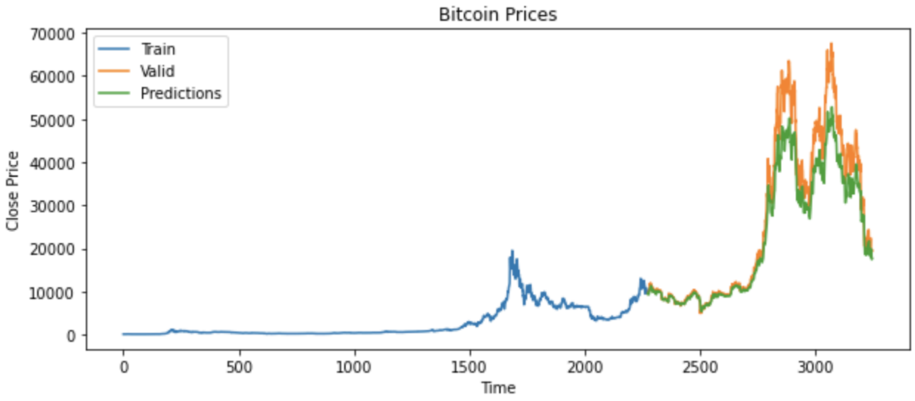


```
Epoch 1/3
2214/2214 [==============================] - 64s 27ms/step - loss: 1.6906e-04
Epoch 2/3
2214/2214 [==============================] - 57s 26ms/step - loss: 1.0075e-04
Epoch 3/3
2214/2214 [==============================] - 57s 26ms/step - loss: 6.0481e-05
```

    - Root mean squared error

- Accuracy
    - tf.keras.metrics.Accuracy( ) - 0.5

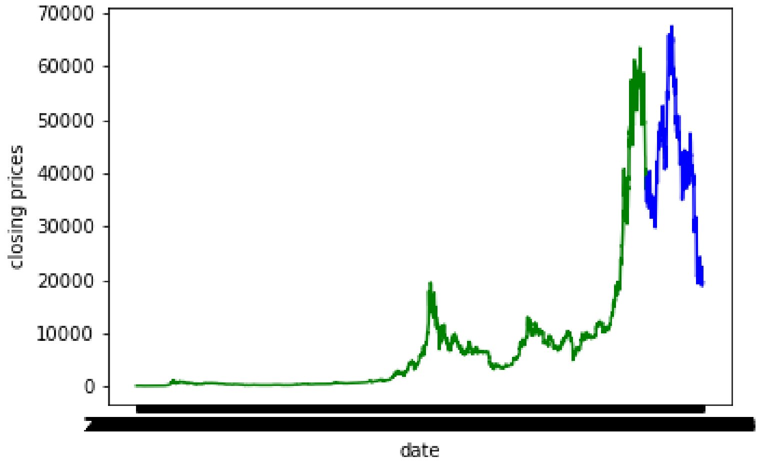
- Predict price value and visualize the data



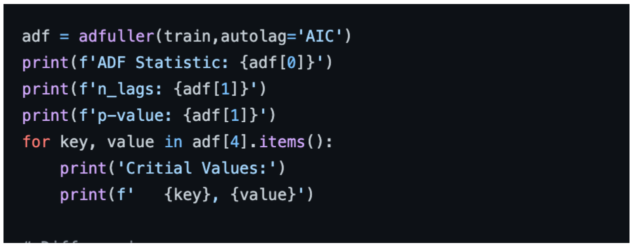
ARIMA(Autoregressive Integrated Moving Average): ARIMA is a statistical analysis model that utilizes time series data to understand data and predict future trends (special form of regression). An autoregressive model predicts future values based on past values, which means they assume the future resembles the past. Therefore, they may be inaccurate under certain conditions. Bitcoin price analysis uses time series data hence we use the ARIMA model to predict future prices. This technique requires terminal installation of pmdarima with pip.

Steps:

- Data Preparation
    - Set the date as the index in bitcoin closing prices dataframe
    - Split the data into training and testing sets



- ➢ Create the predictive model:
  - ○ Check if the data is stationary: use the Augmented Dickey-Fuller (ADF) test.
    - ■ Null hypothesis: The data is stationary

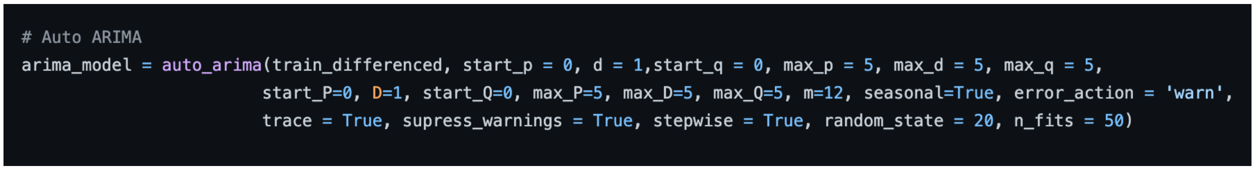


```
adf = adfuller(train,autolag='AIC')
print(f'ADF Statistic: {adf[0]}')
print(f'n_lags: {adf[1]}')
print(f'p-value: {adf[1]}')
for key, value in adf[4].items():
    print('Critial Values:')
    print(f'   {key}, {value}')
```

We reject the null hypothesis

  - ○ Difference the data to make it stationary then perform the ADF test again.
- ➢ Auto arima function from the pmdarima library automatically discovers the optimal order of parameters(p,d,q) for an ARIMA model:
  - ○ p- number of autoregressive terms
  - ○ d-number of nonseasonal differences performed for stationarity
  - ○ q-number of lagged forecast errors in prediction equation

The auto arima function tests the time series with different combinations to find the optimum combination that has the lowest Akaike Information Criterion(AIC).



```
# Auto ARIMA
arima_model = auto_arima(train_differenced, start_p = 0, d = 1,start_q = 0, max_p = 5, max_d = 5, max_q = 5,
                         start_P=0, D=1, start_Q=0, max_P=5, max_D=5, max_Q=5, m=12, seasonal=True, error_action = 'warn',
                         trace = True, supress_warnings = True, stepwise = True, random_state = 20, n_fits = 50)
```

(5,1,0) is the chosen optimum order.

- ➢ Use of the ARIMA function and predict function to predict future bitcoin prices.

Observations and Conclusions

- ➢ The Linear Regression and SVM models produced fairly accurate results, with accuracy scores above 90 percent.
- ➢ The LSTM model had an accuracy score of 50 percent.
- ➢ The ARIMA model predicted negative values which indicates that something went wrong.

Chosen Model: The Support Vector Machine (SVM) model best fulfills our objective of predicting future prices of Bitcoin.

Code Efficiency: The LSTM and ARIMA models have quite a long running time in comparison to the Scikit-Learn regression methods, with ARIMA having the highest computational cost.

Beyond this project: A research into the issue with the ARIMA model suggested that we employ the use of a log scale to avoid negative values. Therefore, in the future, we plan to try this difference in scaling in an attempt to build a better model.

References

Brownlee, Jason. “Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras.” MachineLearningMastery.com, 21 July 2016, https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

Lim, Y. H. (2019). *Autoregressive Integrated Moving Average (ARIMA) Models*. SAAS Berkeley. Retrieved December 7, 2022, from https://saas.berkeley.edu/rp/arima

Nagadia, Meet. “Bitcoin Price Prediction Using LSTM | Deep-Learning Project #DeepLearning #Machine Learning #Python.” YouTube, YouTube, 19 Feb. 2022, https://www.youtube.com/watch?v=p-QY7JNGD60.

Pandya, Maharshi. “Cryptocurrency Prices Data.” *Www.kaggle.com*, 2022, www.kaggle.com/datasets/maharshipandya/-cryptocurrency-historical-prices-dataset?resource=download.

Scikit-Learn Developers. “Sklearn.linear_model.LinearRegression — Scikit-Learn 0.22 Documentation.” *Scikit-Learn.org*, 2019, scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

Scikit-Learn Developers. “1.4. Support Vector Machines — Scikit-Learn 0.20.3 Documentation.” *Scikit-Learn.org*, 2018, scikit-learn.org/stable/modules/svm.html.