Cryptocurrency Predictions

Group 2: Sai Gubba, Seth Trimberger, Darron Li, Ben Metzger, Josh Santiago, Giri Vignesh, Travis Takushi, Jacob Kolk, and Jacob Fox

What is crypto currency?

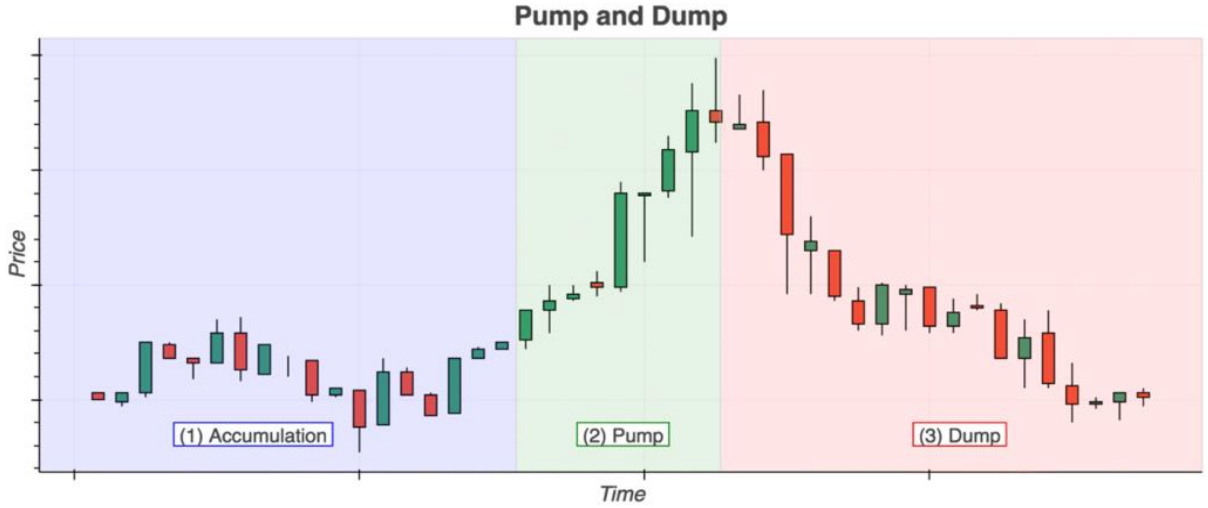
Cryptocurrency is a form of digital currency that uses special techniques to secure transactions and control the creation of new units.

Why is cryptocurrency so volatile?

As opposed to traditional stocks, cryptocurrencies do not have a clear intrinsic value and is primarily driven by speculation and sentiment, not fundamentals. Simply put, ***when people believe in a coin's future, the price can skyrocket. If that faith falls, then the coin can crash hard.***

Other factors can include:

- Lack of regulation / Market manipulation
  - "Pump and Dump" coin manipulation
- Low Liquidity
- Global, 24/7 markets
- Social media
  - memes, Reddit threads, hype



Our Approach

- LSTM
- Random Forests
- Linear Regression

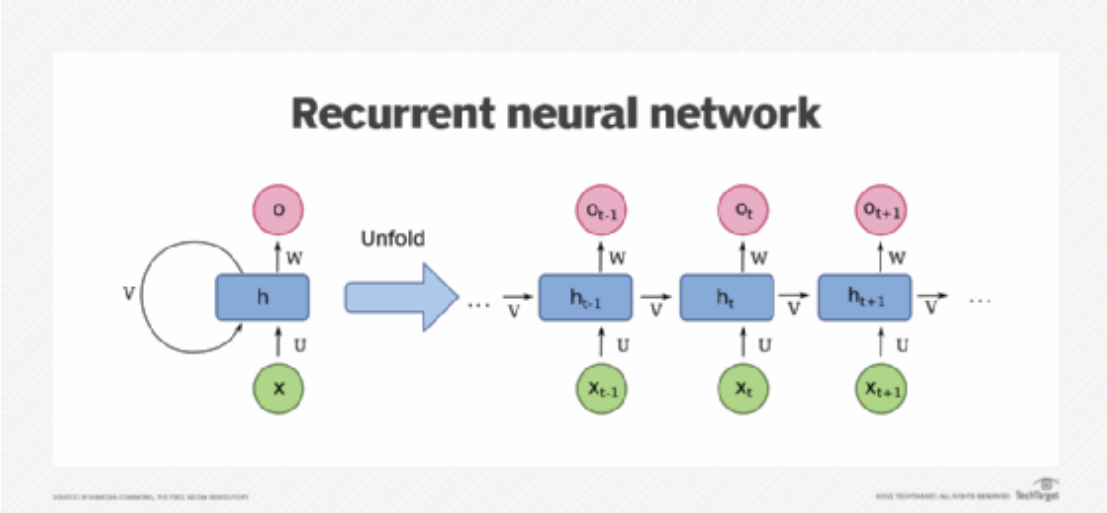
LSTM approach

Darron Li and Jacob Kolk

Recurrent Neural Network (RNN)

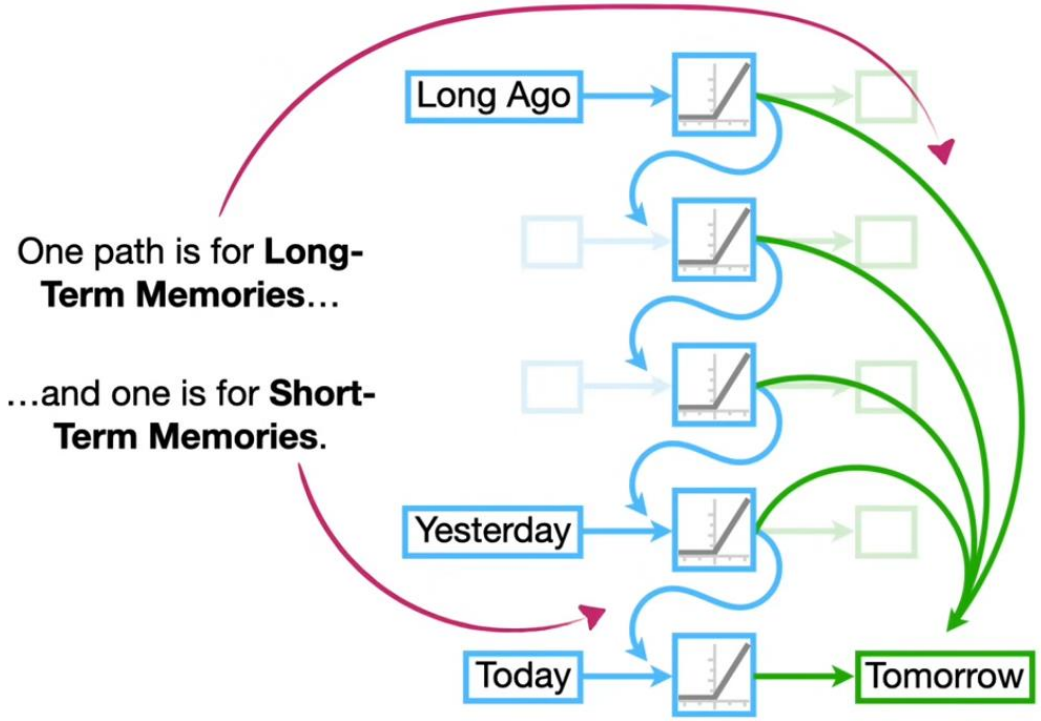
Long
Short
Term
Memory

A recurrent neural network or RNN is a deep neural network trained on sequential or time series data to create a machine learning (ML) model that can make sequential predictions or conclusions based on sequential inputs.

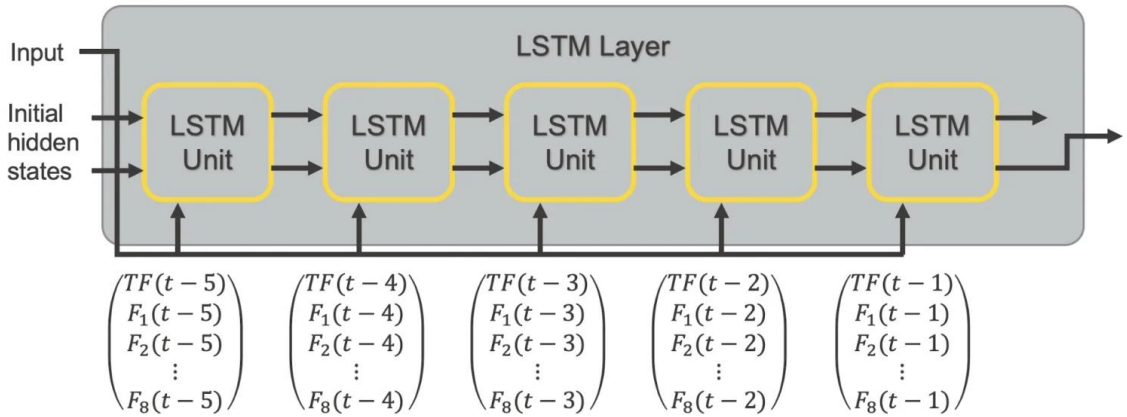


Advantages

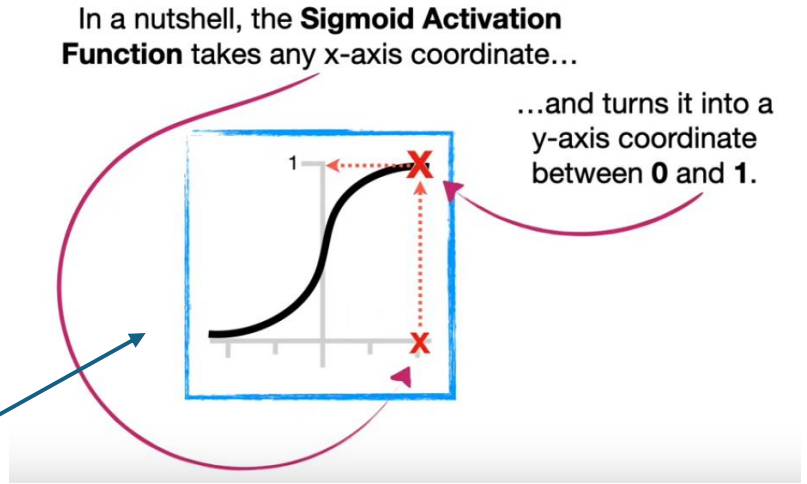
Long term memory, old data will not be forgotten



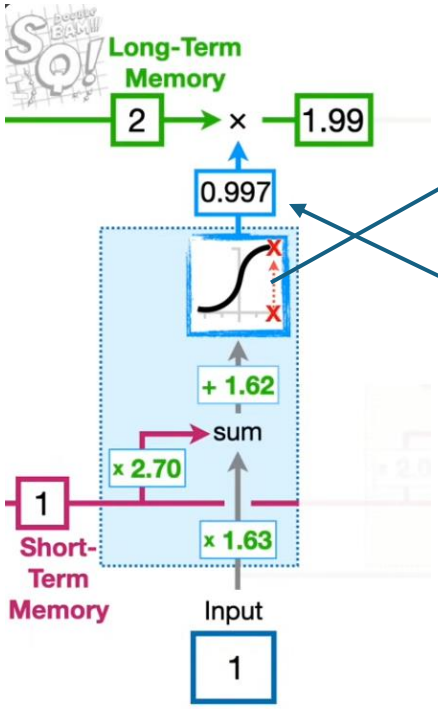
Multi variable inputs. With the LSTM model, there can be multiple input variables



How each module works



How the % of short-term to save is determined



What % of short-term memory is saved to long term

The % of short-term memory that is saved based on backpropagation of the error to calculate the new weights, this is the training and iterative refinement that makes this family of models unique.

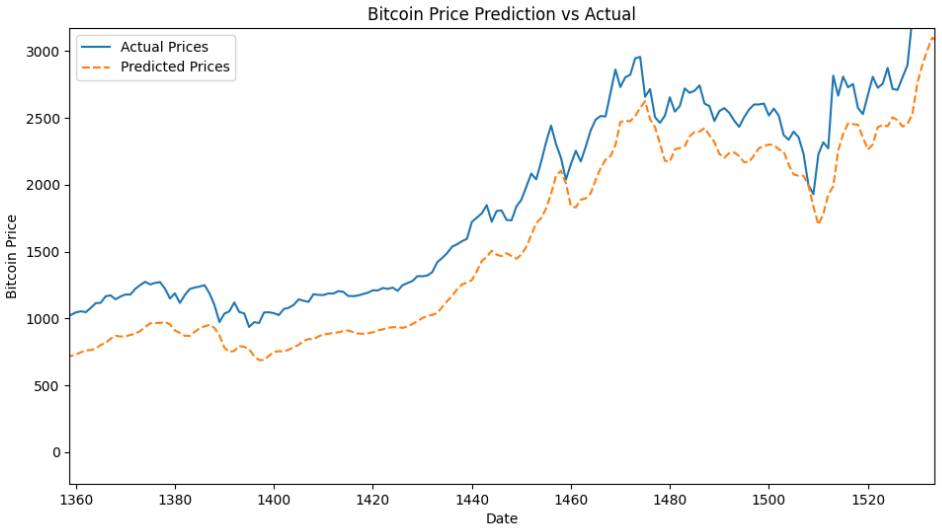
Our implementation: Hyperparameter and configuration breakdown

```
input_size = 6
hidden_size = 64
num_layers = 2
output_size = 1
num_epochs = 50
batch_size = 32
learning_rate = 0.001
```

- Input size: This is the number of features in our data set, this includes high, low, open, close, market cap, and volume values.
- Hidden_size: The number of units in the units hidden state for each step, basically how much is remembered
- Num_layers: This is how many LSTMs are stacked on top of each other, more can mean more comprehension of complex patterns but also runs the risk of overfitting
- Num_epochs: How many times the entire data set is run through the LSTM
- Batch_size: The number of samples passed through at each step
- Learning_rate: This is the step size the optimizer uses to update the model's weights during training. Slower learning rates make the process slow and steady, reducing instability

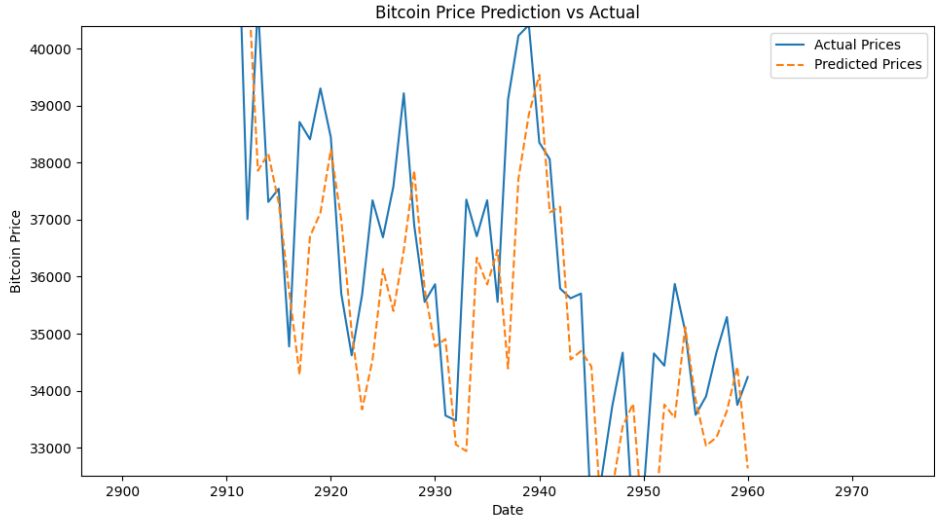
A close look at the data:

Low volatility



Similar shape, higher error

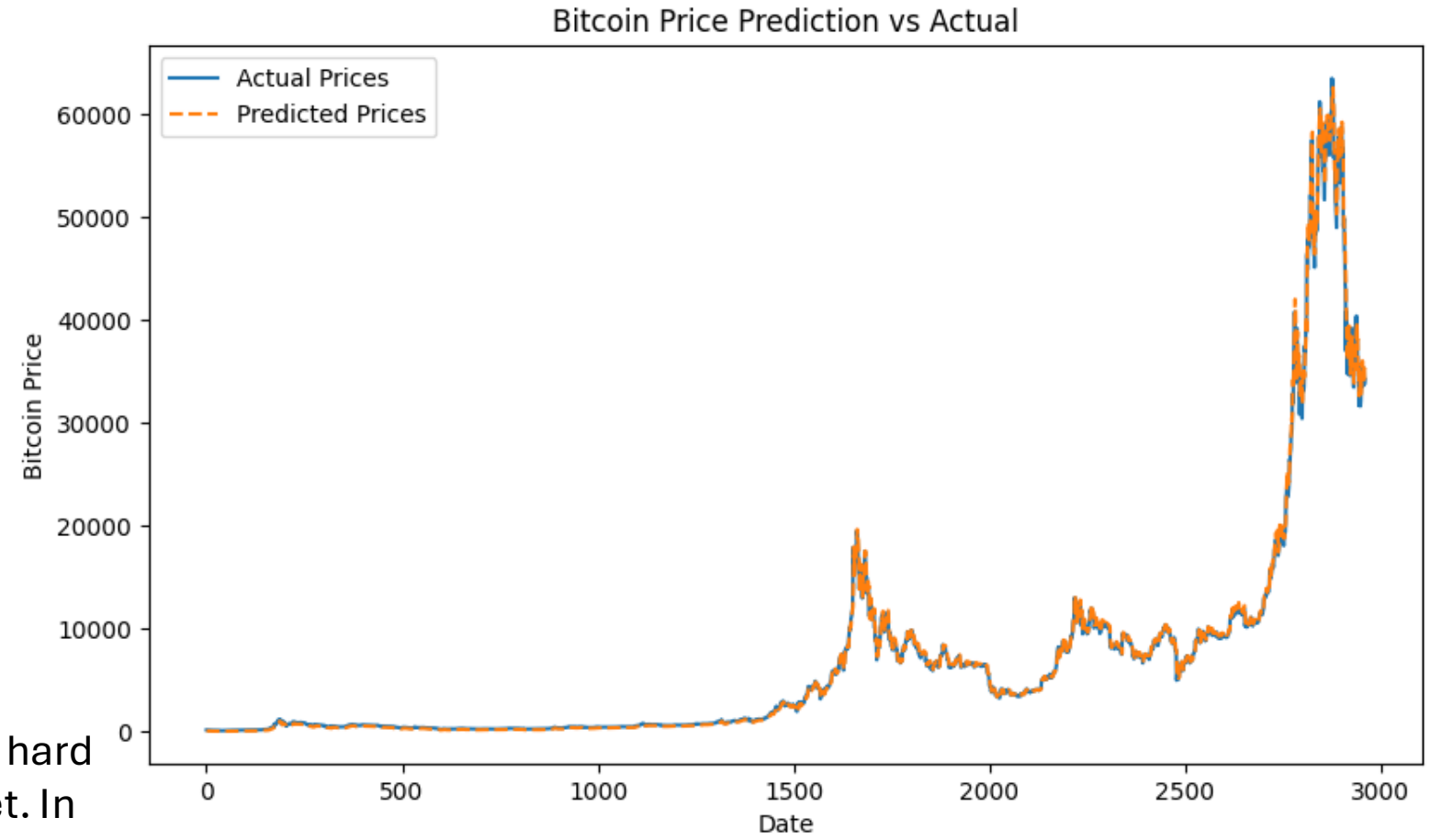
High volatility



Different shape, lower error

Results:

As you can see, our model predicted future values relatively closely. The MAPE was about 15%. This means throughout the graph the prediction was an average of 15% off of the actual value.




```
Mean Absolute Percentage Error: 15.043624
```

Predicting crypto prices is inherently hard because of the volatility of the market. In testing, error percentages would vary from less than 10% to over 70%. Most landing somewhere around 25%. There are many things that could lead to this, including less than optimal hyper parameter, unlucky randomness in optimization and initialization, and overfitting, underfitting.

Random Forests

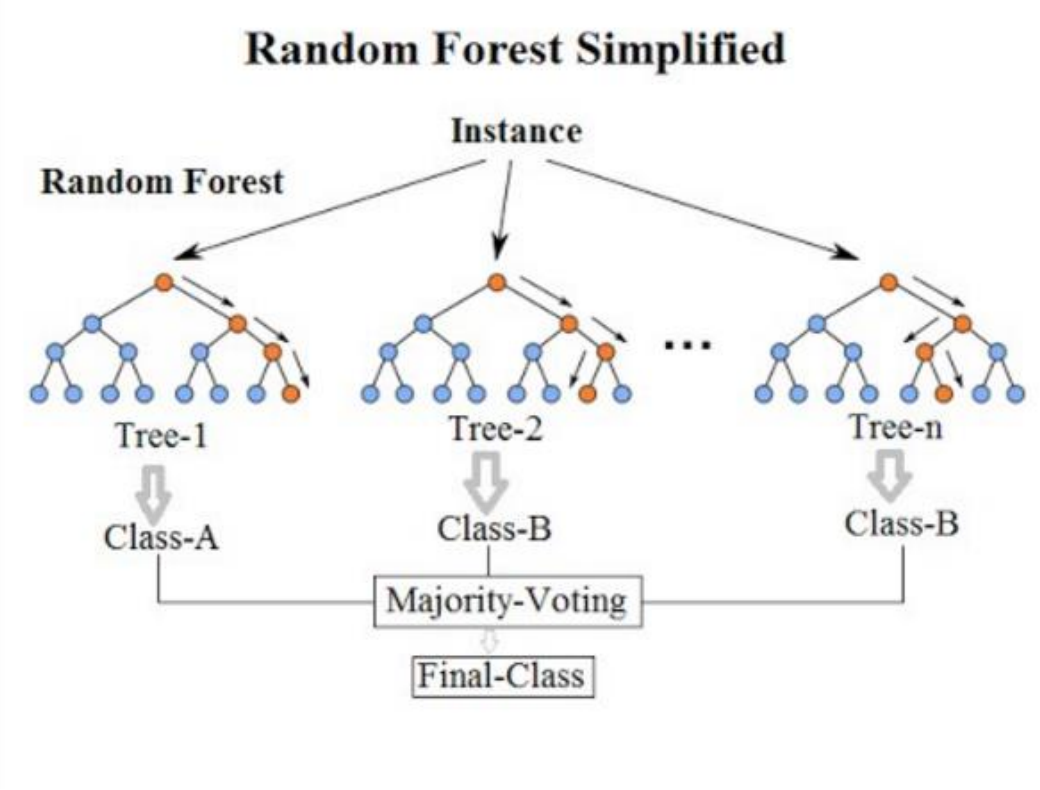
Sai Gubba, Seth Trimberger,

Why Random Forests?

- Random Forests is a method used when processing large amounts of data to simplify and return a more accurate prediction of a statistic

- For our project we used the Random Forest Method to parse through our dataset to calculate patterns at random times and valuations of Crypto to return a more accurate prediction of whether a stock would go up or down in price

- Random Forests are useful since they gather data from a range of patterns that are random throughout a period rather than smaller sections, this is useful for crypto prices since its very volatile



Processing Data

- The data in Bitcoin and Ethereum is sorted by Date
- We want our prediction models to predict the Close price based on the Open, Close, Low, and High prices one day and two days after the start date
- To split the data into training and testing data, we split the data based on an index and place the data into the training and testing data
- For our x training data and x test data, we get the columns of the Open, Close, Low, and High prices one day and two days after the start date
- Our y training data and y test data are in the Close column
- After creating our model using the x and y training data, we predict it on the x testing data
- To calculate accuracy, we use the prediction model and the y testing data to calculate the mean based on absolute differences and check if the prediction is within 10% of the allowed error
- The R^2 score is also calculated to indicate the fit of the prediction model on the y testing data

```
fileInput['Open1'] = fileInput['Open'].shift(1)
fileInput['High1'] = fileInput['High'].shift(1)
fileInput['Low1'] = fileInput['Low'].shift(1)
fileInput['Close1'] = fileInput['Close'].shift(1)
fileInput['High2'] = fileInput['High'].shift(2)
fileInput['Low2'] = fileInput['Low'].shift(2)
fileInput['Close2'] = fileInput['Close'].shift(2)
fileInput['Open2'] = fileInput['Open'].shift(2)
```

```
fileInput.dropna(inplace=True)
split_index = int(len(fileInput) * 0.8)
train_data = fileInput.iloc[:split_index]
test_data = fileInput.iloc[split_index:]
```

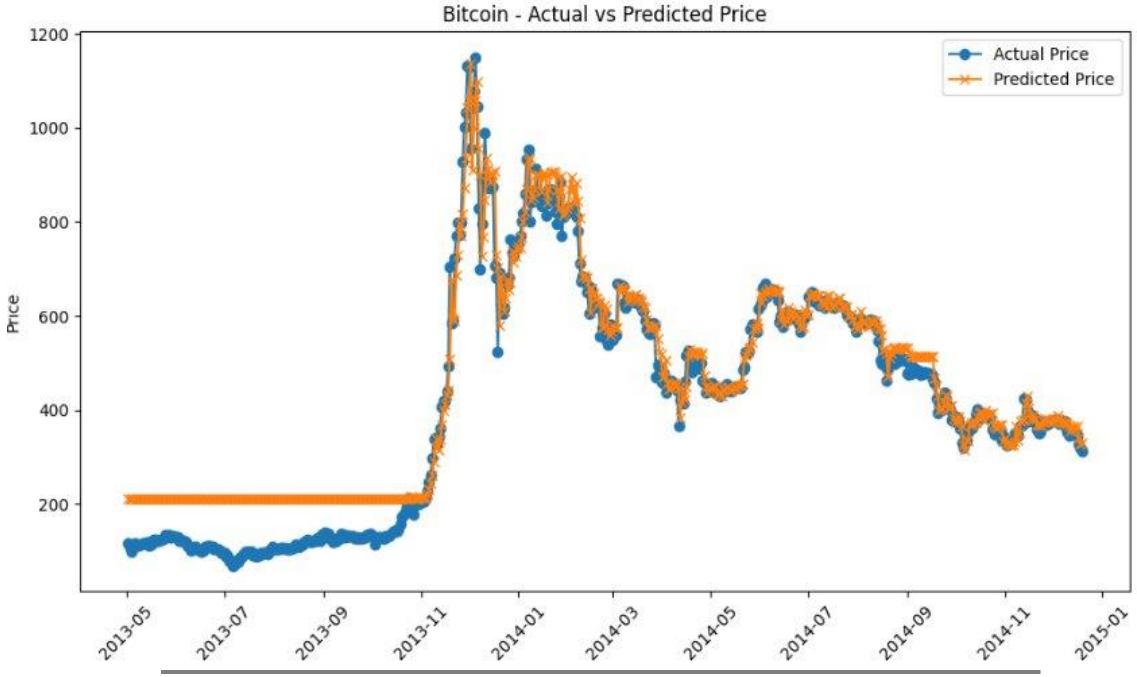
i.e. placing the first 20% of data into test_data and the remaining 80% into train_data

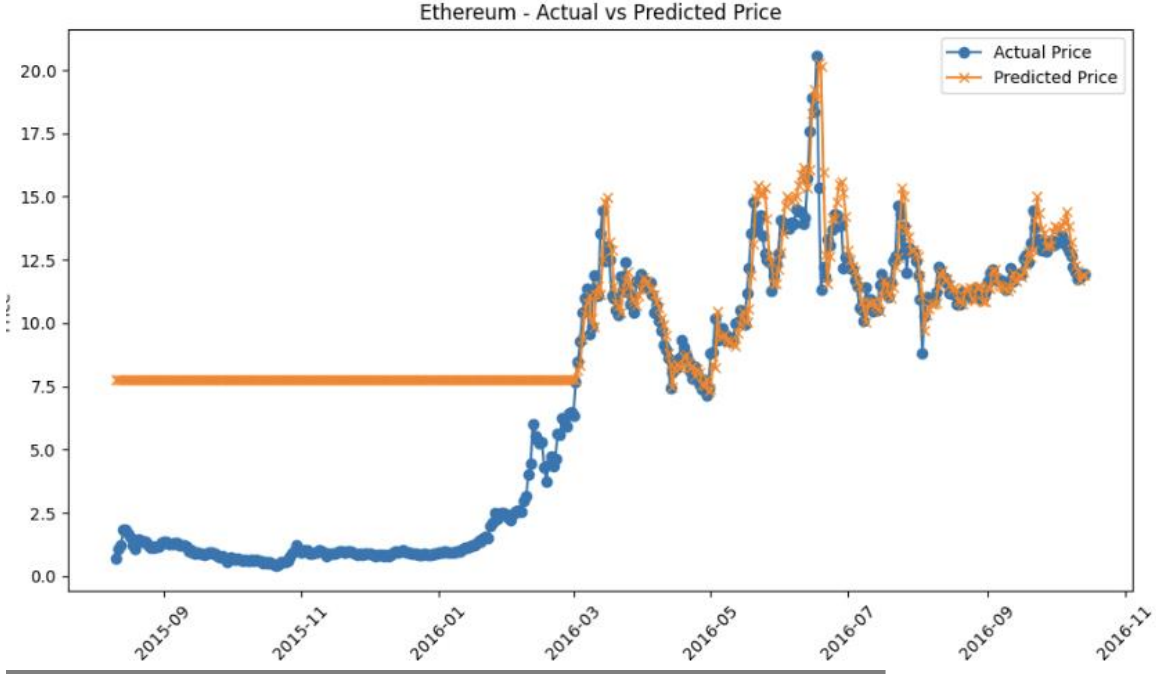
```
features = ['Open1', 'High1', 'Low1', 'Close1', 'High2', 'Low2', 'Close2', 'Open2']

x_train = train_data[features]
y_train = train_data[target]
x_test = test_data[features]
y_test = test_data[target]
```

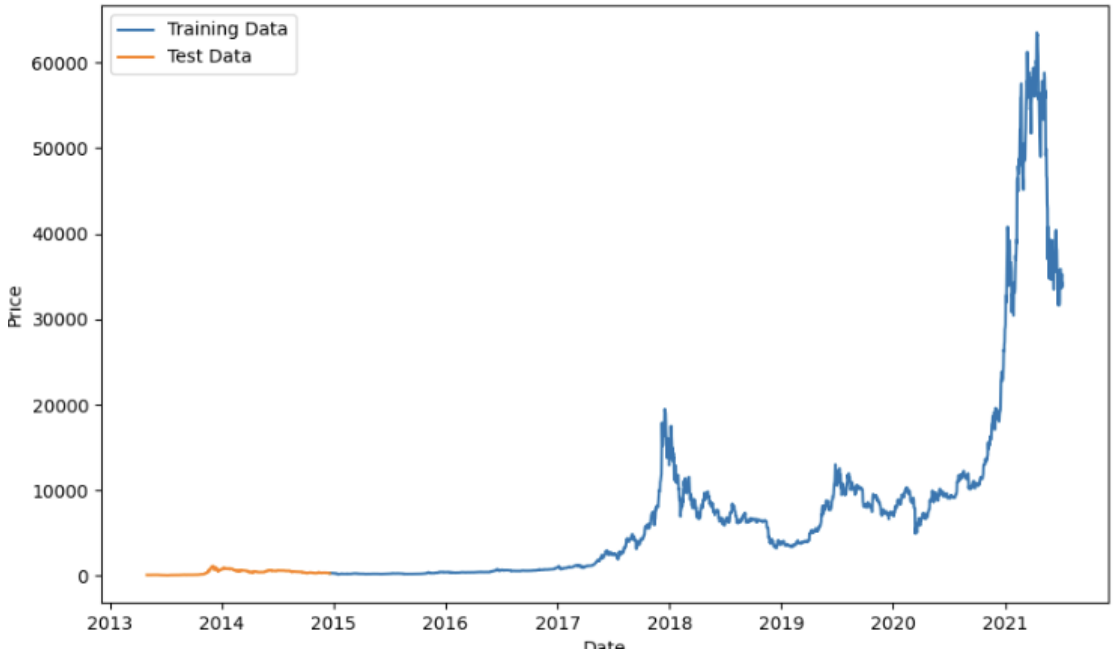
```
def accuracy(predictions, y_test):
    allowed_error = 0.10
    correct_predictions = np.mean((np.abs(predictions - y_test) / y_test) <= allowed_error)
    return correct_predictions * 100
```

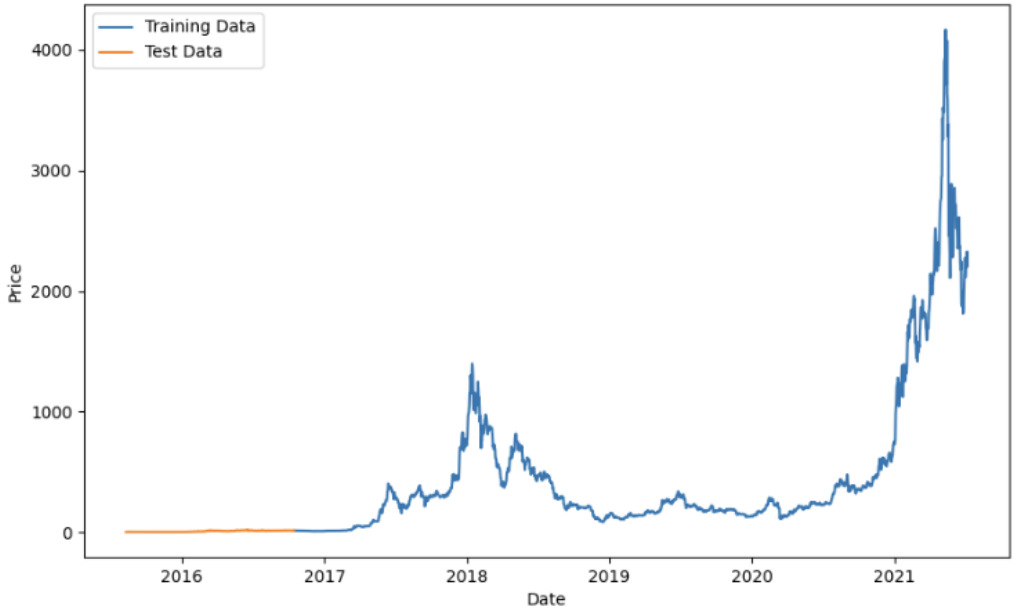
Testing The First 20% of the Data

- Our first random forest model is training the Bitcoin and Ethereum data on the last 80% of their data and testing the prediction model on the first 20% of their data
- For Bitcoin, the model has an accuracy of 63.82% and an R^2 score of 0.9380
- For Ethereum, the model has an accuracy of 42.92% and an R^2 score of 0.3044

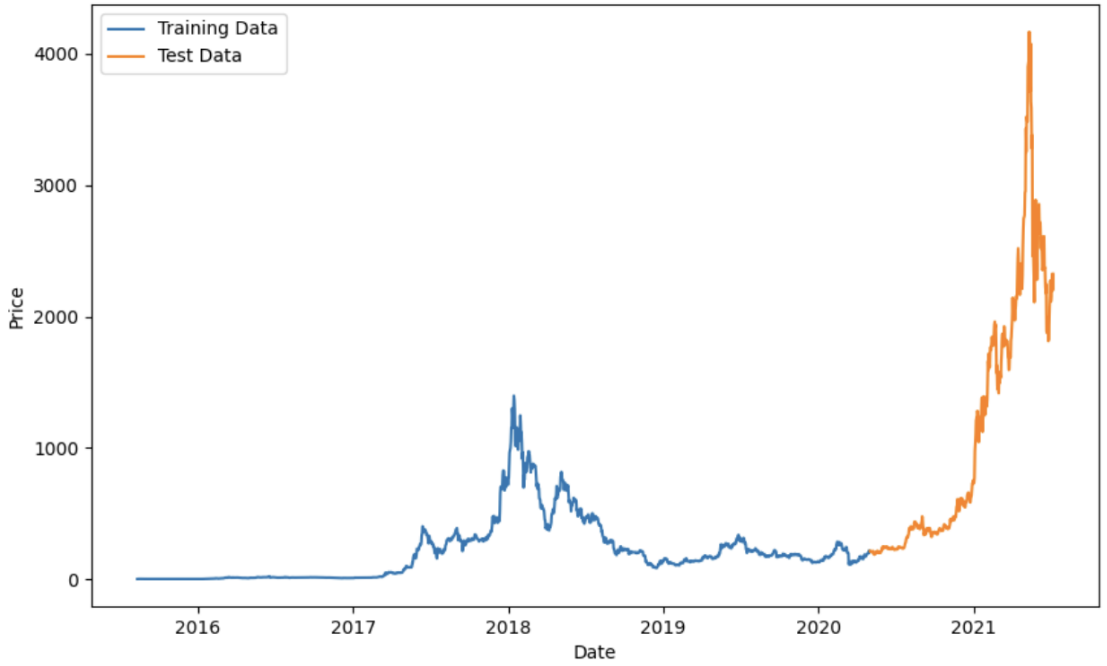

Bitcoin - Actual vs Predicted Price
Actual Price
Predicted Price
Price
1200
1000
800
600
400
200
2013-05
2013-07
2013-09
2013-11
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01

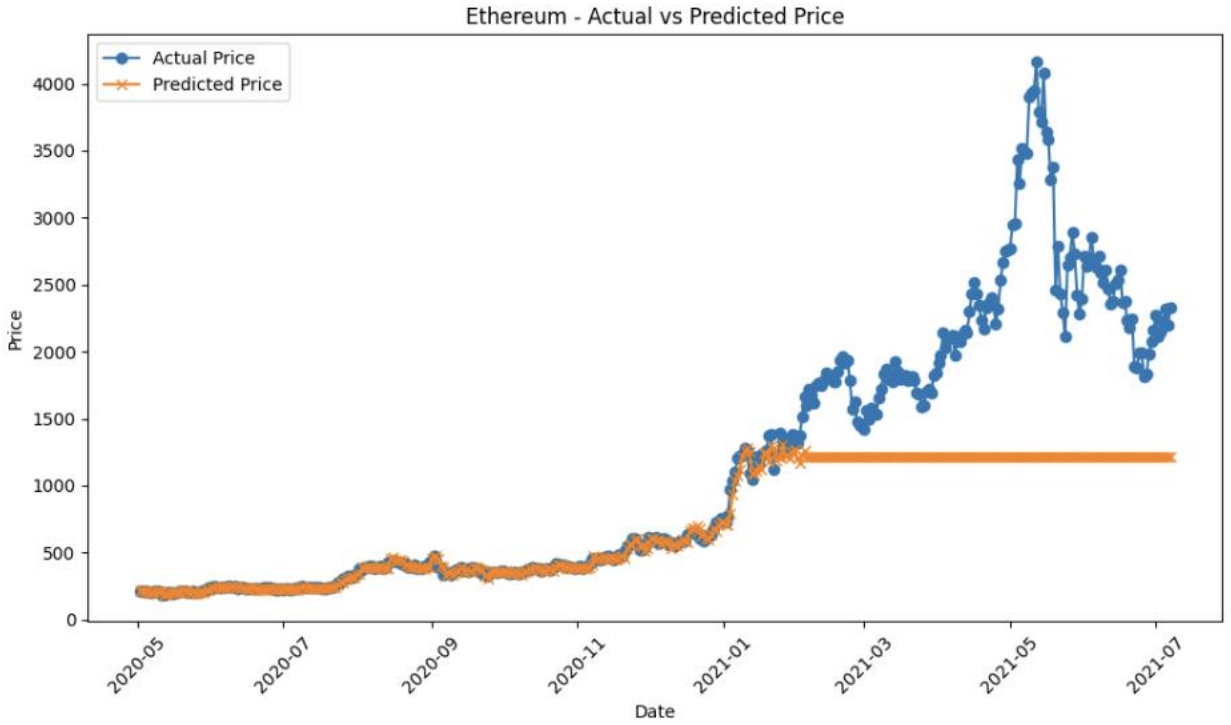

Ethereum - Actual vs Predicted Price
Actual Price
Predicted Price
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2015-09
2015-11
2016-01
2016-03
2016-05
2016-07
2016-09
2016-11


Training Data
Test Data
Price
60000
50000
40000
30000
20000
10000
0
2013
2014
2015
2016
2017
2018
2019
2020
2021
Date

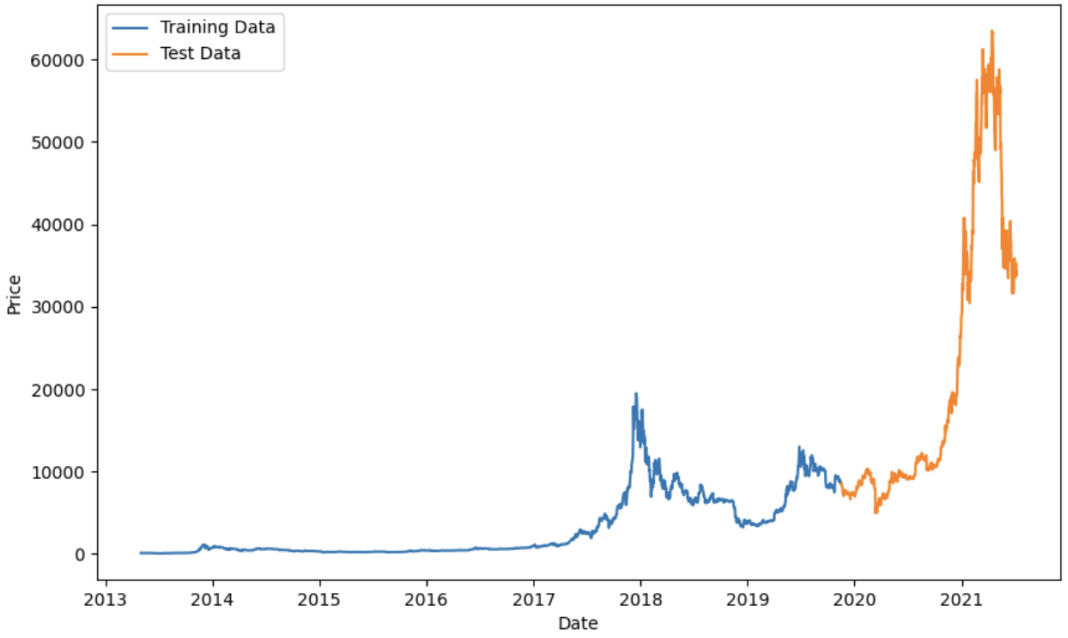

Training Data
Test Data
Price
4000
3000
2000
1000
0
2016
2017
2018
2019
2020
2021
Date

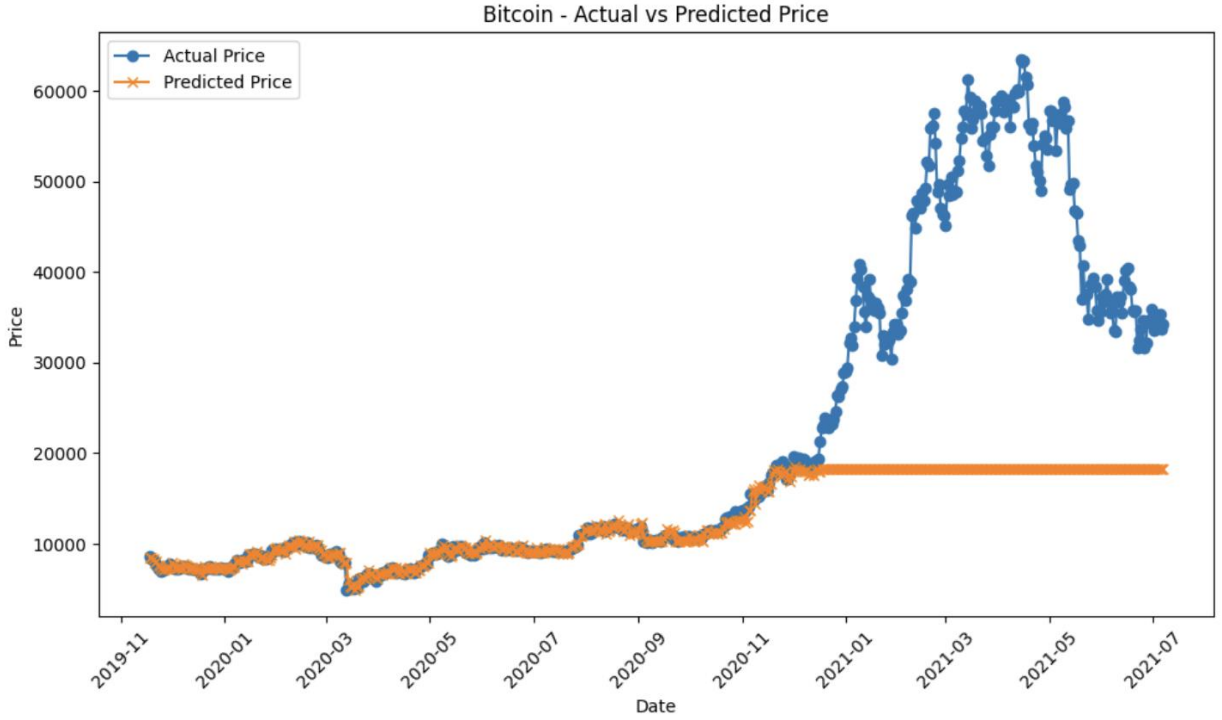
Testing The Last 20% of the Data



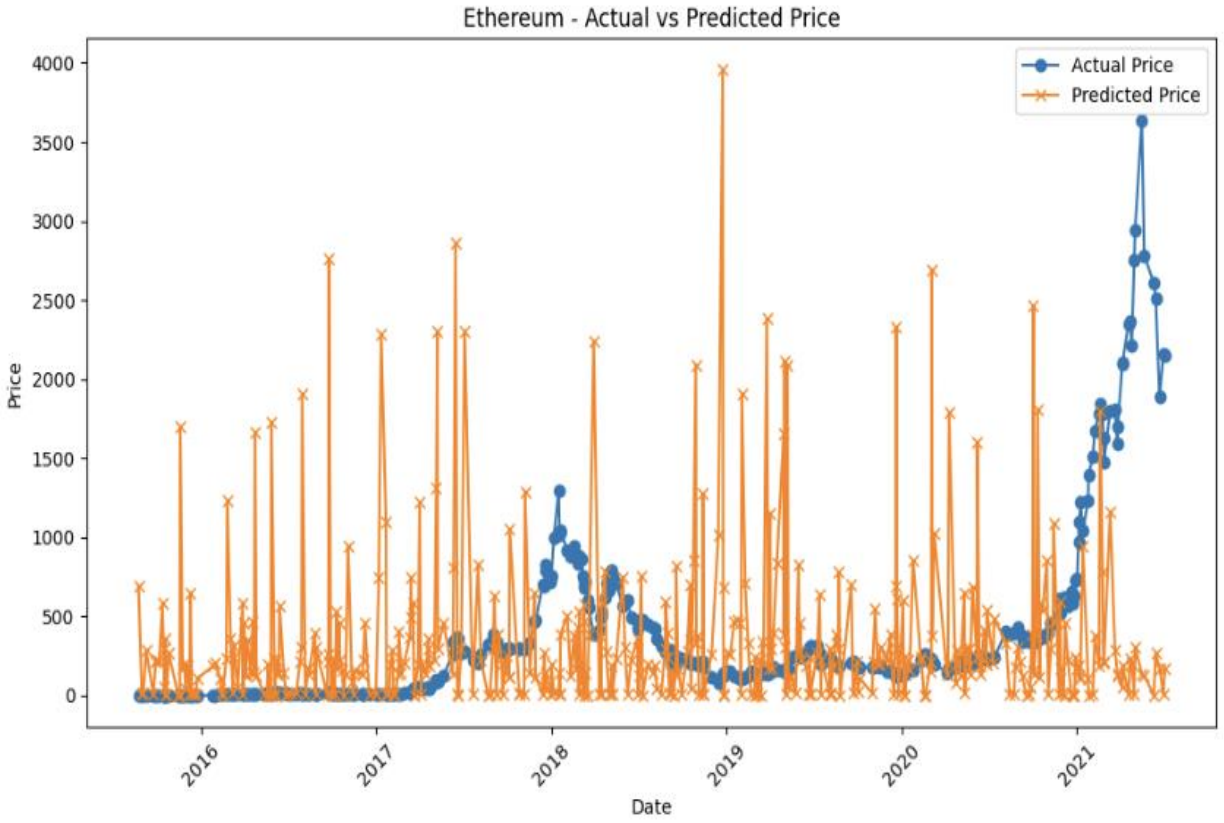


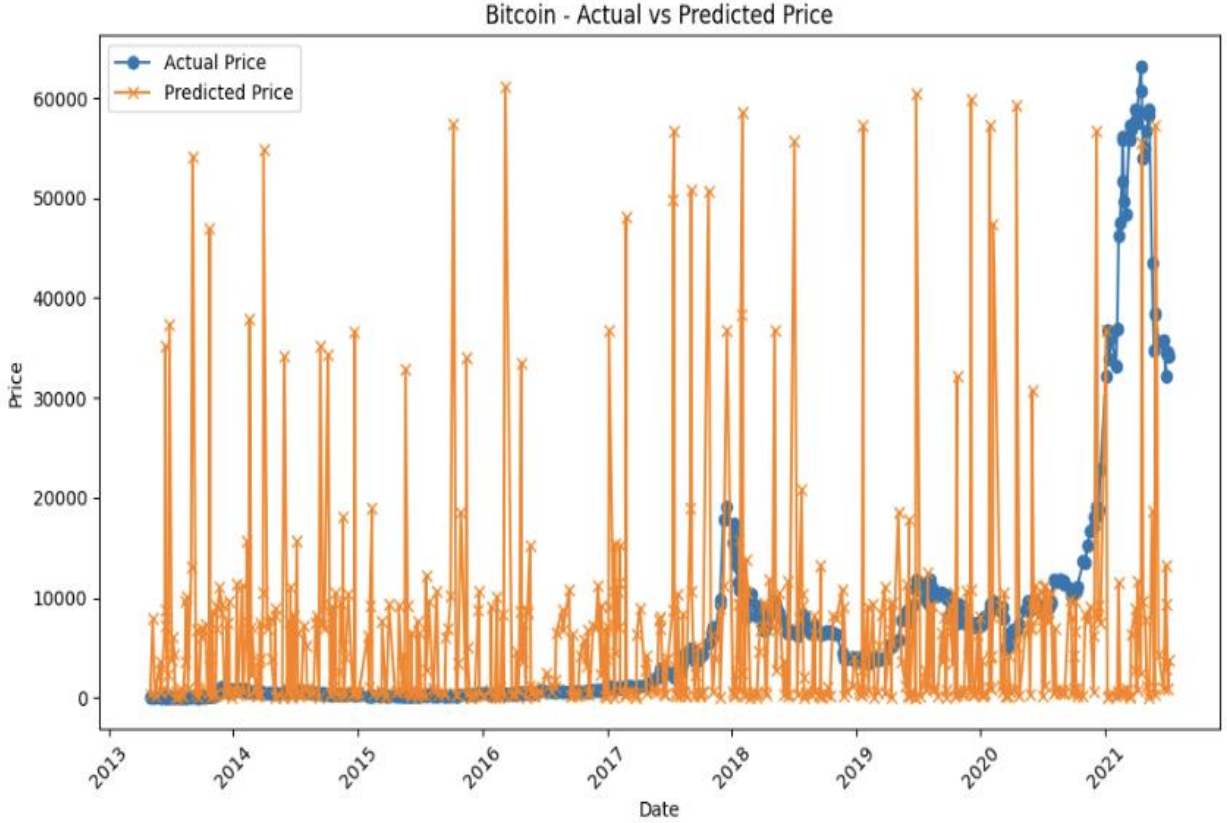
Testing The Last 20% of the Data





Testing: Random 20% of The Data





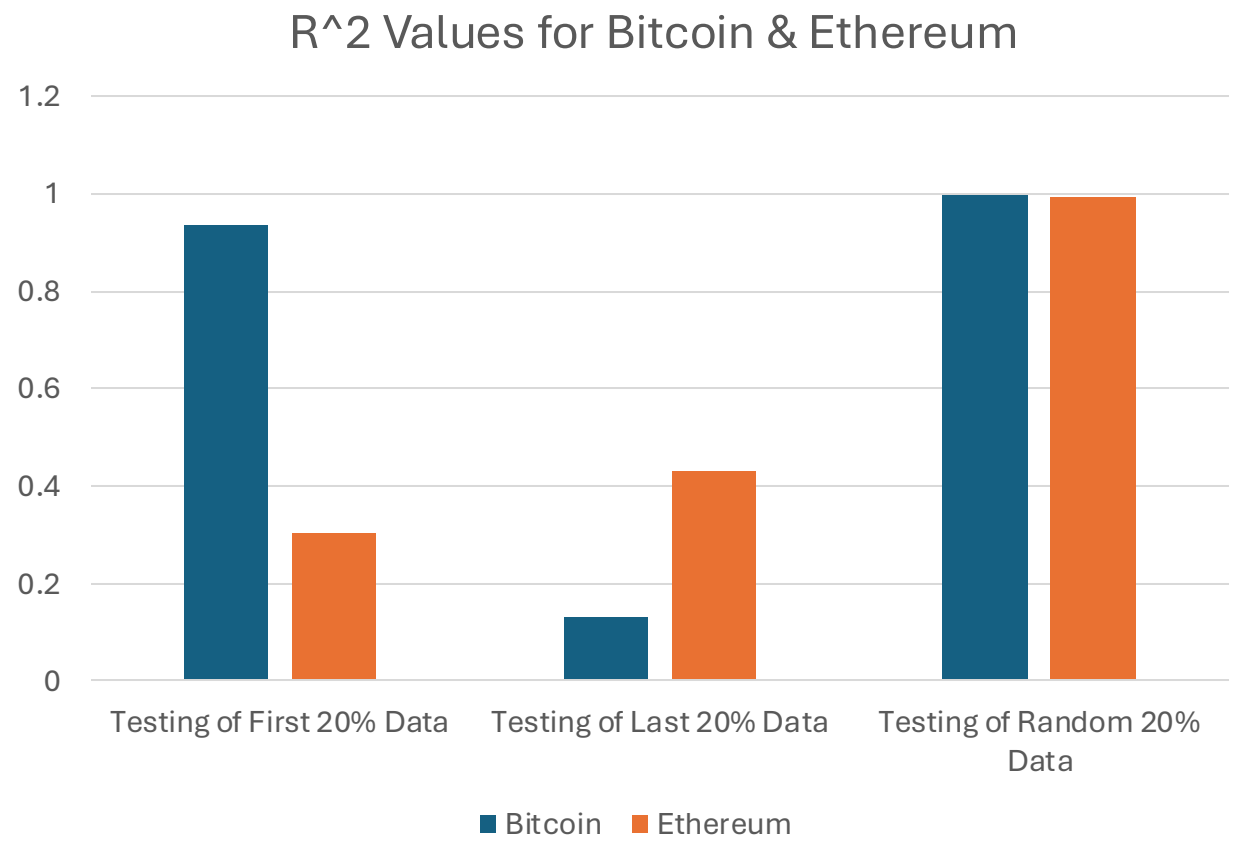
Ethereum Accuracy: 66.44%

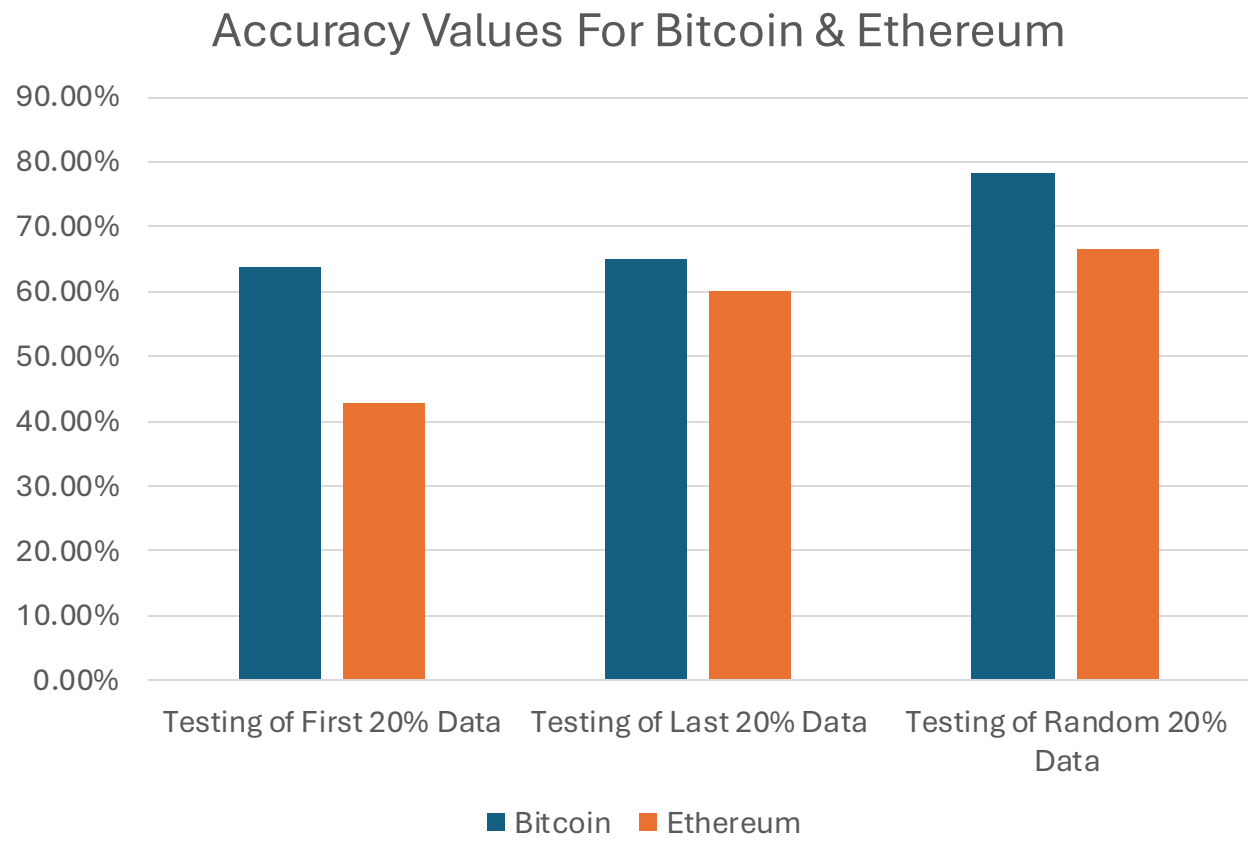
R^2 score of .9932

Bitcoin Accuracy: 78.43%

R^2 score of .9965

Our Preferred Random Forest Model Is: A Random 20% of Data For Testing





- On average it has combined value if R^2 of .995 vs .621 (First 20%) and .282 (Last 20%)
- On average it has a combined accuracy of 72.4% vs 53.4% (First 20%) and 62.4 (Last 20%)
- Captures trends most accurately by exposing the model to the full patterns in the dataset

Improvements

- We need more data as right now our model is limited by the amount of available data, which is limiting how accurate are prediction can be

- We also aim to improve the models performance by optimizing model arameters such as max depth, and minium samples per split This will help reduce the risk of overfitting and give a high accuracy

- In the future we want to introduce fresh data, that are model has never seen so we can see how the model performs on truly unseen data, which is needed to predict future real world prices.

- Go back to 80/20 split of the data as its best pracitce to test the most recent data on your model.

Linear Regression

Ben, Jacob, Travis

What is Linear Regression?

- Supervised Machine Learning algorithm
  - Looks at labels
  - Learns and labels own data
  - Time aware
  - Predicts continuous outputs
- Very Simple/Easy to interpret
  - Look at difference between actual and predicted

Linear Regression Equation

$$Y = a + bx$$

$$a = \frac{[(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma xy)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

$$b = \frac{[n(\Sigma xy) - (\Sigma x)(\Sigma y)]}{[n(\Sigma x^2) - (\Sigma x)^2]}$$

How Linear Regression Worked In Our Project

Using Regression Model

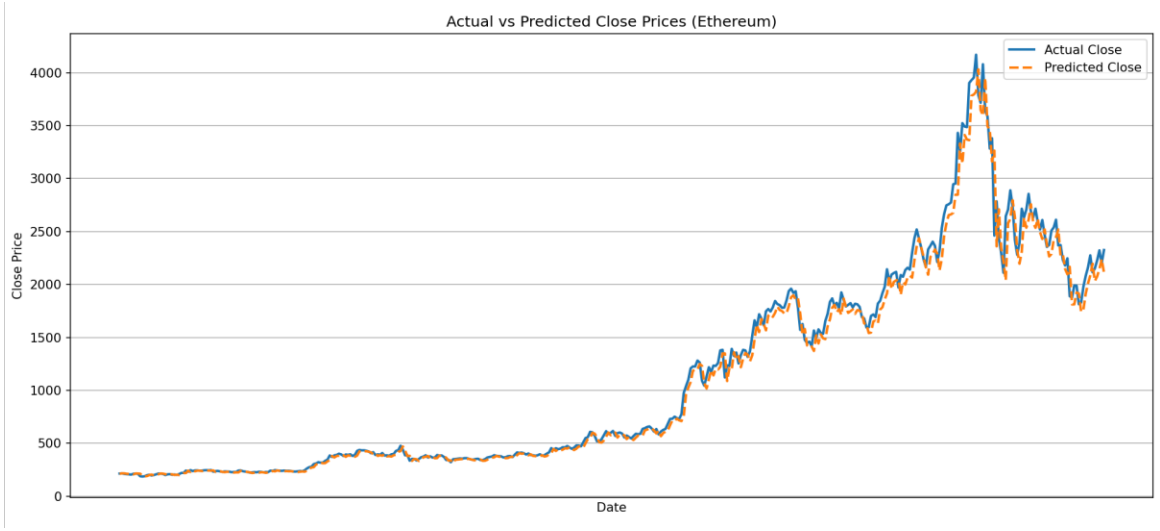
- Used ScikitLearn Linear Regression Model
  - Separated a portion of data to be "lag"
  - Shifted close data back 1 to 2 days
  - Allows us to compare and predict later
  - Using double lag on close allows for higher accuracy
  - Then used 20% of data as test data

- Uses weights on each variable to see what effects the next closing price the most

$$\text{Predicted Close} = \beta_0 + \beta_1 \cdot \text{Close}_{-1} + \beta_2 \cdot \text{Close}_{-2} + \beta_3 \cdot \text{Volume}_{-1} + \beta_4 \cdot \text{Marketcap}_{-1}$$

Using Regression Model

- The Model would "predict" what the next close would be
  - Compare to actual close
  - Readjust weights
  - Then repeat process
- Very Simple model, but highly accurate
  - Looks for R2 Score
  - 1 - (Sum of Squared Errors / Total Sum of Squares)
  - How far predictions from actual vs How far actual are from mean



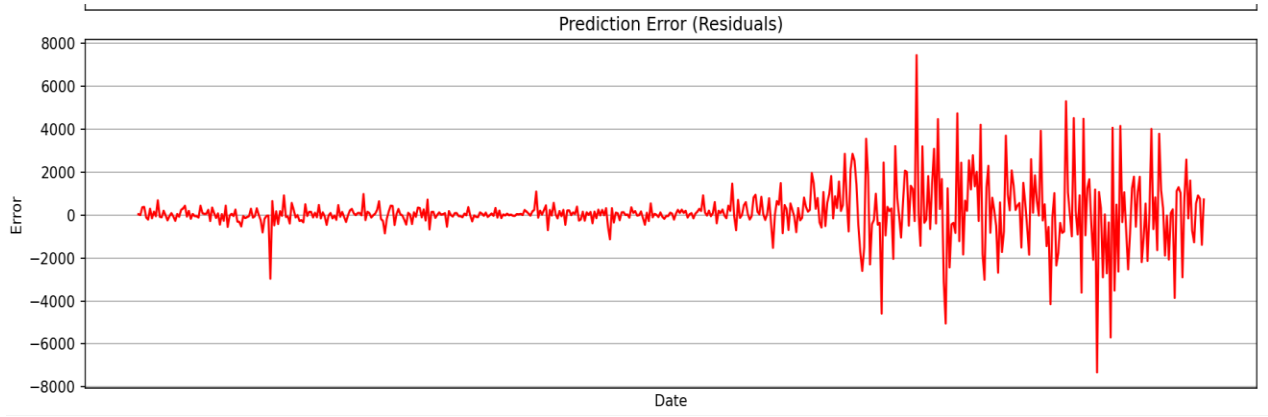
0.9947712380258783

0.9862930922477388

R2 Values of Bitcoin and Etherium Respectively

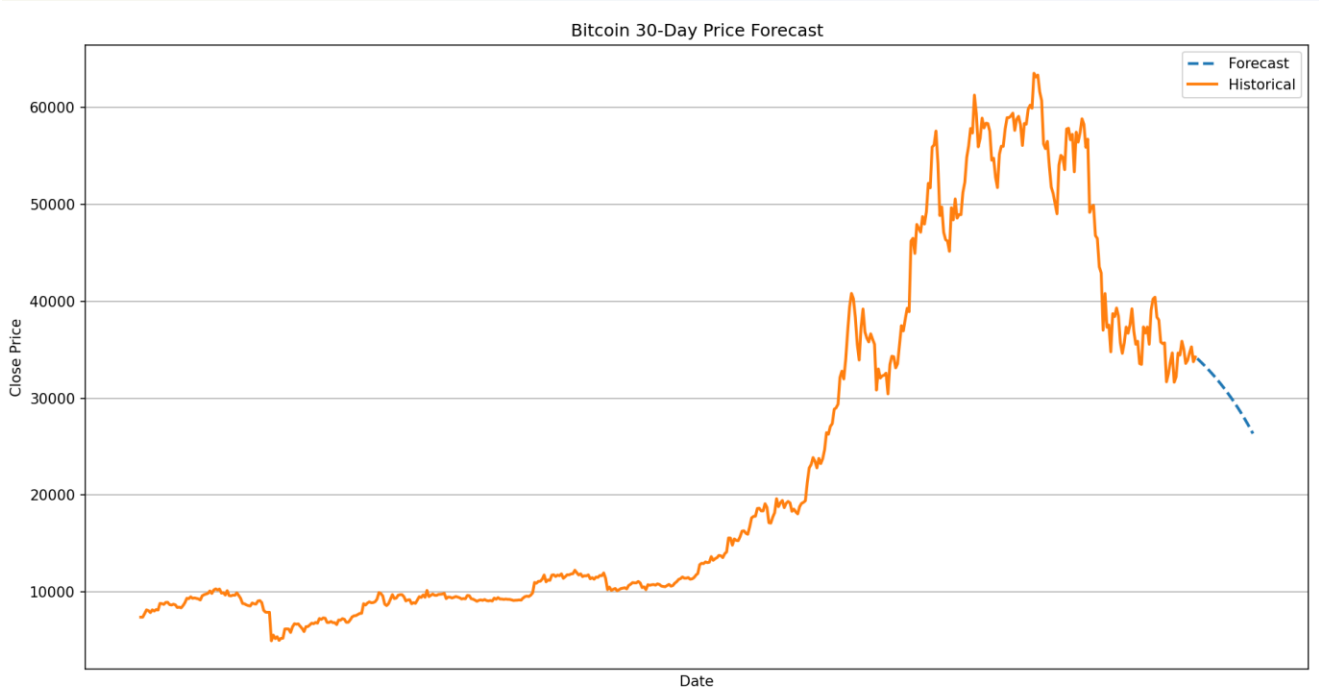
Results

- We can easily interpret through residuals
  - Actual – Predicted
  - Shows errors
  - Further from 0, larger error
- Relatively low error
  - Spikes the less stable the market is



Future Application

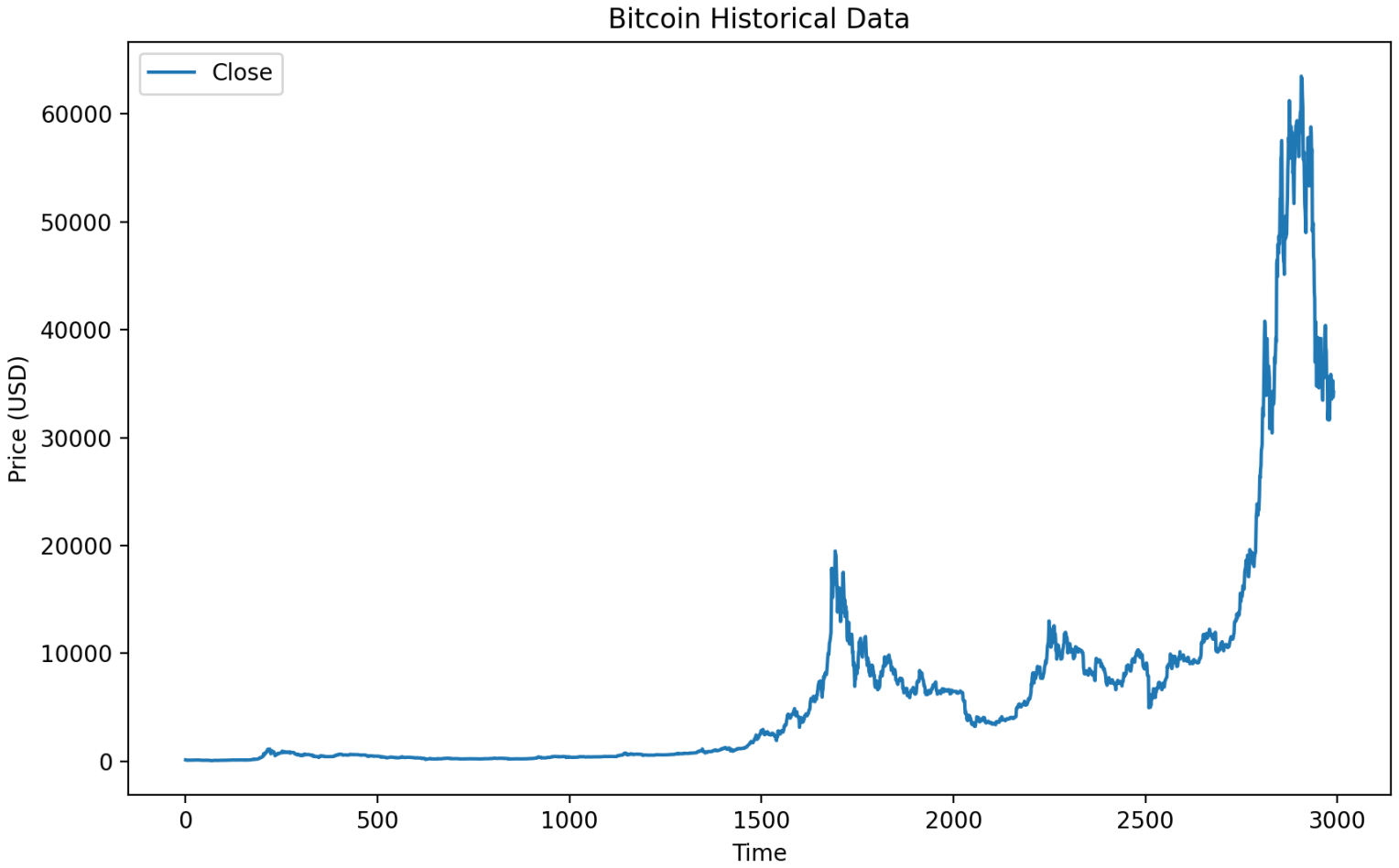
- Predict Future Closes
  - Model can extrapolate off given data
  - Uses own output as next days input(autoregressive)
  - Update lag
  - Once data comes out, readjust



Problems addressed

- Cryptocurrencies are an evolving market -> Using data from a couple years ago to adjust weights will not be accurate
  - Addressed by using a 30-day window -> More accurate local weights

- Using 30 days windows makes the model only able to predict based on short term changes
  - Market regimes last much longer than 30 days
  - Using a short window will not train on past similar trends which would be very beneficial to use



- Linear regression is a very naïve approach to this problem which oversimplifies market shifts and regimes
  - Model only determines bullish/bearish trends, but typically this is an over-generalization

Future Improvements

- Needs to account for external factors
  - Most volatile jumps in market are due to external sources ->Recent tariffs caused 5% drop in Bitcoin
- Add more labels than Bear/Bull
  - Extreme Bear/Bull
  - Sideways
  - Neutral
- Use a more recent dataset
- Linear regression makes too many assumptions -> opt for a better model
  - Works well for small changes but will be unable to learn from previous trends

Conclusion

LTSM

Able to recall previous trends

More configurable through hyperparameters

Will be best suited for more data

Random Forests

Able to recall previous trends

Testing a random sample prevents overfitting

Linear Regression

Unable to recall previous trends

Assumes linearity between attributes

Simple and good at predicting short term changes

Next steps

- Moving forward, we would choose to make improvements to the LTSM model due to its flexibility and deep learning capabilities
  - Better equipped to predict large market shifts
  - More customizable

Thank you for listening

Questions?

Group 2

References

LTSM Approach and RNN

- What is a RNN? https://www.ibm.com/think/topics/recurrent-neural-networks
- Machine Learning Algorithms: A Comparative Analysis. https://www.diva-portal.org/smash/get/diva2:1778251/FULLTEXT03.pdf
- Cryptocurrency Price Prediction using LSTM and Recurrent Neural Networks: https://ieeexplore.ieee.org/document/10141048

Random Forests:

- https://link.springer.com/article/10.1007/s10479-020-03575-y
- https://ieeexplore.ieee.org/abstract/document/10738906

Linear Regression

- https://www.geeksforgeeks.org/linear-regression-formula/
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- https://www.coinbase.com/learn/crypto-basics/what-is-a-bull-or-bear-market