The Application of Random Forest on the Closing Price Prediction of Bitcoin

Cameron Zerrilla

April 2022

1 Introduction

In an environment where NFT's and cryptocurrencies are viable sources for investments, it is important to maintain strong investment strategies. A way to do this is to use machine learning algorithms to predict future prices so investors can make intelligent business decisions. One form of cryptocurrency that has been on the rise is Bitcoin. With my research, I would like to implement the Random Forest machine-learning algorithm to improve previous studies' results. If it is possible to do so, it would be valuable for the strategies I use to be implemented into the future prediction of cryptocurrencies.

2 Hypotheses

I hypothesize that given previous research on Random Forest, it is possible to utilize the algorithm to improve the closing price prediction of stock market values. If it can outperform previously used algorithms, it could be utilized to make more intelligent investment decisions. To test my hypothesis, I will be using Random Forest to predict Bitcoin prices. If I am able to improve on previous results, it could be a good indicator that Random Forest can be used for future stock price prediction. The specific results I will be trying to improve on are the results of research done by Tan and Kashef. They were able to use Arima, SVM, LSTM, and Bayesian Regression [8, p. 4-5]. Their results showed that LSM and SVM performed best, so I will be trying to improve those [8, p. 4-5]. I will still be trying to improve their other results as well, but the mentioned two are of greater importance. Below are ideas and works that have influenced my choice and reasoning behind studying machine learning for Bitcoin prediction as well as my choice for Random Forest.

3 Literature Survey

3.1 Bitcoin and Machine learning

When doing research about a machine learning algorithm that can be utilized to predict stock prices, Bitcoin specifically, it is important to analyze the research that has already been done. Bitcoin is now considered to be a source of investment in the stock market, but it is in the category of highly volatile stocks. In Mangla and colleagues' research, they mention that "bitcoin does not depend on the business events or intervening government authorities, unlike the stock market." [5, p. 318]. That is precisely why they decided to use machine learning and not typical statistical methods for prediction [5, p. 318].

3.2 Work that has been done

Many machine learning algorithms have been used to try to predict Bitcoin closing prices, price change detection, and more. First, Mangla and colleagues used logistic regression, SVM, ARIMA, and RNN to detect Bitcoin price change prediction [5, p. 320]. Mangla and colleagues were able to determine that ARIMA was able to perform with the highest accuracy; it performed well for the next days prediction but it did not perform well looking at week for prediction [5, p. 320]. Li used lasso regression, ridge regression, and XGBoost to predict the rise and fall of Bitcoin [4, p. 49]. Li was able to show that XGBBoost outperformed lasso and ridge regression [4, p. 51-52]. Lastly, Tan and Kashef use Bayesian regression, ARIMA, SVM, and LTSM to predict Bitcoin closing prices [8, p. 4-5]. More of their results will be discussed throughout the paper.

3.3 Random Forest

Random Forest is not mentioned in any of the articles but has been used in previous research to evaluate and predict future values. For example, in Vijh and colleagues' research, Random Forest performed almost as well as an artificial neural network [9, p. 602-603]. The research was done to see if ANN or Random Forest would perform better for stock analysis classification [9, p. 602-603]. Research by Sadorsky has shown that Random Forest is a better prediction algorithm than logit algorithms when predicting stock market value direction [7, p. 17]. Research done by Polamuri and colleagues suggests that Random Forest is better at stock price forecasting than Linear Regression or Multivariate Regression [6, p. 1228]. With research suggesting that stock market direction and stock prediction can be implemented with Random Forest and tends to show good results, it could be used to improve Tan and Kashef's results.

4 Methods

4.1 Plan for My Work

To see if there is room for improvement on Bitcoin price prediction, I will be using Random Forest using their data for stock prediction to the best of my ability. I will use Root Mean Square Error to determine the accuracy because that is what Tan and Kashef used to define accuracy, and I intend to follow their method and results as closely as possible.The other values I will also use used for measurement are, Mean Squared Error and Mean Absolute Error. They did not include the formulas but I will use python to implement each one.

4.2 Figure 4.2

($\bar{y}$ are the observed values and y is the predicted values)
RMSE= $\sqrt{\frac{\sum_{t=1}^{n}(\bar{y_t}-y_t)^2}{n}}$ [8, p. 3]
MAE=$\frac{1}{n}\sum_{t=1}^{n}(|\bar{y_t}-y_t|)$
MAPE=$100*\frac{1}{n}\sum_{t=1}^{n}(\frac{|\bar{y_t}-y_t|}{\bar{y_t}})$
MPE=$100*\frac{1}{n}\sum_{t=1}^{n}(\frac{\bar{y_t}-y_t}{\bar{y_t}})$
ME= $\frac{\sum_{t=1}^{n}(\bar{y_t}-y_t)}{n}$
The formulas above are obtained and or manipulated from http://jrhs.umsha.ac.ir/index.php/JRHS

/rt/printerFriendly/1059/html.

4.3 Methodology

The data, as mentioned before, was taken from CoinMarketCap [1]. The features used for prediction were Open, High, Low, and Volume for Bitcoin for each day. The feature being predicted was the closing price for each day. The proccess, which is shown below in code was to use sckitlearn to get the tools needed for random forest. The first time I ran random forest I used 1 to 300 trees with 100 test for each set. For example the first was one tree tested 100 times, the second was 2 trees tested 100 times and so on. Once I ran this it was noted that most of the test were inefficient because the results did not change much. Therefore I decided to remove 50 sets from the end and 74 from the beggining. Below in Fig 1.1 are Tan and Kashef's RMSE scores I am looking to improve upon and in the appendix Fig M.1 is the code used to implement Random Forest.

As it can be seen this implementation with a 70/30 split ($test-size=0.3$). The RMSEAVG, MAEAVG, and MSEAVG are all arrays that will hold the

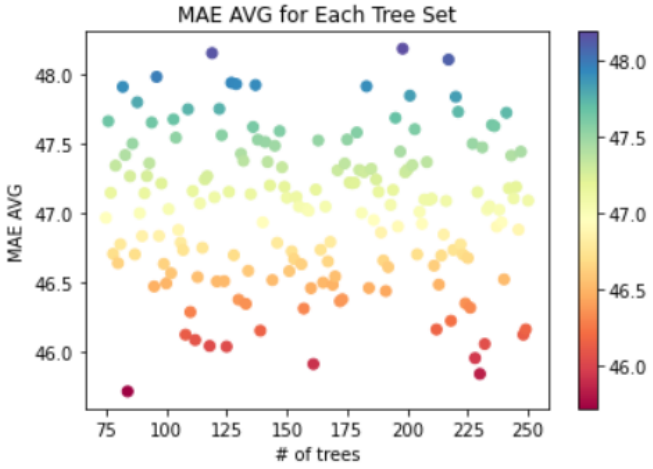
average or each of their respective error measurement's of each set of trees. It is simply calculated by taking the result for each set and adding it to a sum (RMSETotal, MAETotal, MSETotal). Then after each set of trees is done with the 100 test it will divide by 100 and store the result in the previously mentioned arrays. The error results were then plotted each using similar code to what is displayed below in figure 1.2.

Fig 1.1

Method	RMSE
Bayesian Regression	461.9379
ARIMA	439.9801
SVM	288.0618
LSTM	33.7091

Fig 1.2

```
numtrees=[]
for i in range(75,251):
    numtrees.append(i)
plt.scatter(numtrees, MAEAVG, c=MAEAVG, cmap='Spectral')
plt.colorbar()
plt.title('MAE AVG for Each Tree Set')
plt.xlabel('# of trees')
plt.ylabel('MAE AVG')
plt.show()
```



The scatter plot was made using tools by matplotlib. Line 4 uses the array for datapoints. numtrees is the array that stores the number of trees in each set

and MAEAVG stores the MAE average for each set. This process was similarly followed for each time data needed graphics.

5 Results

5.1 70/30 split with only Bitcoin data

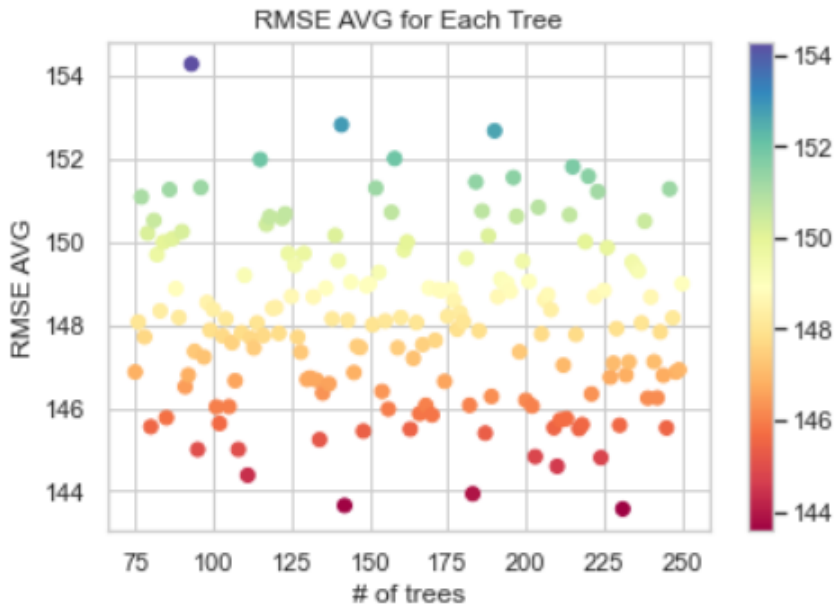
Below are the results for a 70/30 split using random forest with only the data presented by Tan and Kashef. Each numerical row is rounded at the 100th's place.

Fig 2.1

ErrorStat	Average	Lowest
RMSE	148.127	143.569
MSE	22378.691	21061.861
MAE	46.390	45.233

The lowest RMSE was located at tree set 231. Below the figure 2.2 shows the RMSE scores with the corresponding number of trees that were used. The MSE and MAE scores are shown in figures A.1 and A.2 in the appendix.

Fig 2.2



5.2 80/20 split with only Bitcoin data

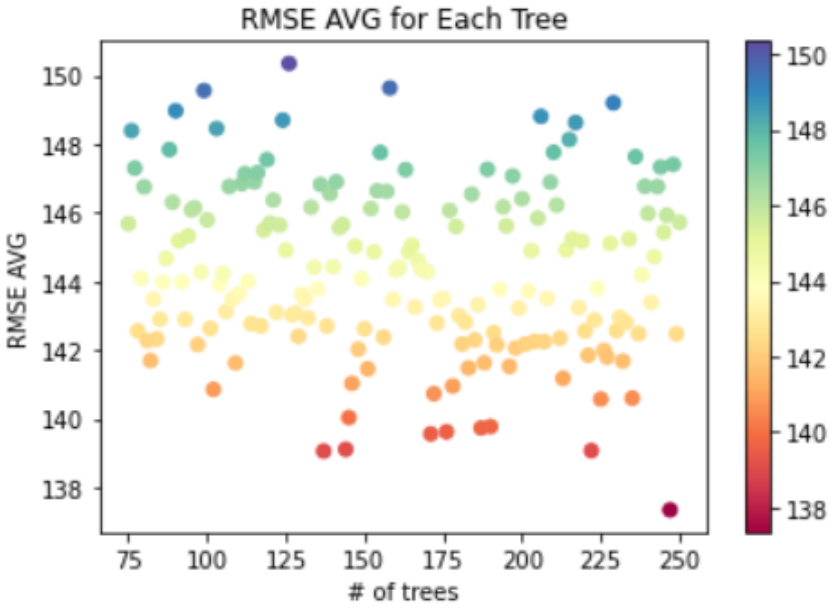
Below are the results for a 80/20 split using random forest with only the data presented by Tan and Kashef. Each numerical row is rounded at the 100th's place.

Fig 2.3

ErrorStat	Average	Lowest
RMSE	144.289	137.343
MSE	21367.887	19272.241
MAE	45.610	43.896

The lowest RMSE was located at tree 247. Below the figure 2.4 shows the RMSE scores with the corresponding number of trees that were used. The MSE and MAE scores are shown in figures A.1 and A.2 in the appendix.

Fig 2.4



As it can be seen from the 70/30 and 80/20 results, Random Forest does not perform terribly, but it does not perform the best either. With an 80/20 result sitting at an RMSE score of 137.343. Figures 2.2 and 2.4 show that there was not much of a difference between the number of trees used for each test. Therefore it would be acceptable to decrease the number of trees used but for the rest of the research, I will stick to 75 to 250 trees. This is an improvement

upon Tan and Kashefs' Bayesian Regression, ARIMA, and SVM, but it does not improve among LSTM's results.

5.3 Room for Improvement on the Data

It is a possibility that the data given might not be suitable for random forest. Therefore I added Ethereum data because another form of cryptocurrecy could help in comparison for closing price prediction. Below, figure 3.1 shows the data used. The Ethereum data comes from a website called Coincodex. [2]

Fig 3.1

A	B	C	D	E	F	G	H	I	J	K	L	M
Date	Open	High	Low	Volume	MarketCa	ETHOpen	ETHHigh	ETHLow	ETHClose	ETHVolum	ETHMarke	Close
28-Apr-13	135.3	135.98	132.1	0	1.49E+09	0	0	0	0	0	0	134.21
29-Apr-13	134.44	147.49	134	0	1.6E+09	0	0	0	0	0	0	144.54
30-Apr-13	144	146.93	134.05	0	1.54E+09	0	0	0	0	0	0	139
1-May-13	139	139.89	107.72	0	1.3E+09	0	0	0	0	0	0	116.99
2-May-13	116.38	125.6	92.28	0	1.17E+09	0	0	0	0	0	0	105.21
3-May-13	106.25	108.13	79.1	0	1.09E+09	0	0	0	0	0	0	97.75
4-May-13	98.1	115	92.5	0	1.25E+09	0	0	0	0	0	0	112.5
5-May-13	112.9	118.8	107.14	0	1.29E+09	0	0	0	0	0	0	115.91
6-May-13	115.98	124.66	106.64	0	1.25E+09	0	0	0	0	0	0	112.3

The new features added are Ethereum opening price, Ethereum low price, Ethereum low price, Ethereum high price, Ethereum closing prices, and Ethereum market cap. The largest issue with this data set is that Ethereum has not been around as long as Bitcoin so a lot of the values are zeros.

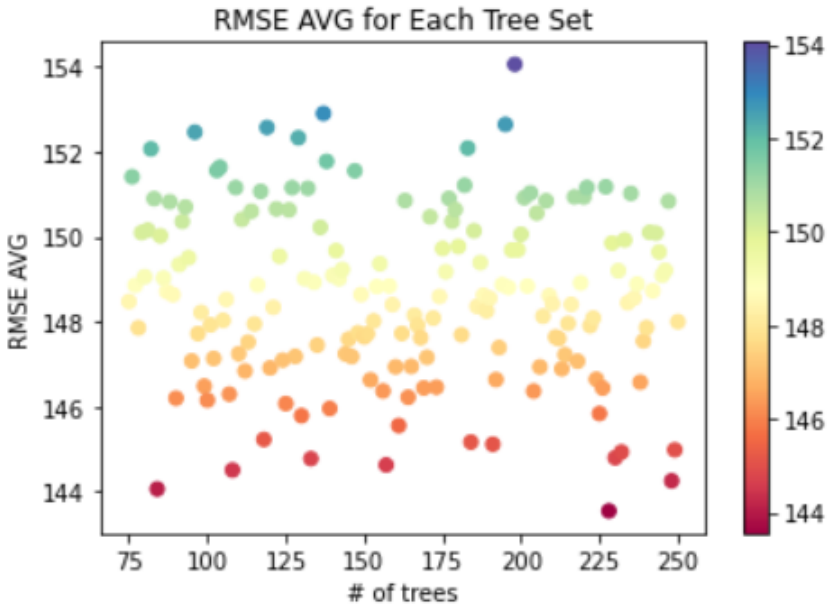
5.4 70/30 split with Bitcoin and Ethereum data

Fig 3.2

ErrorStat	Average	Lowest
RMSE	22516.938	20888.006
MSE	47.0177	45.716
MAE	45.610	43.896

The lowest RMSE was located at tree set 229. Below the figure 3.3 shows the RMSE scores with the corresponding number of trees that were used. The MSE and MAE scores are shown in figures B.1 and B.2 in the appendix.

Fig 3.3



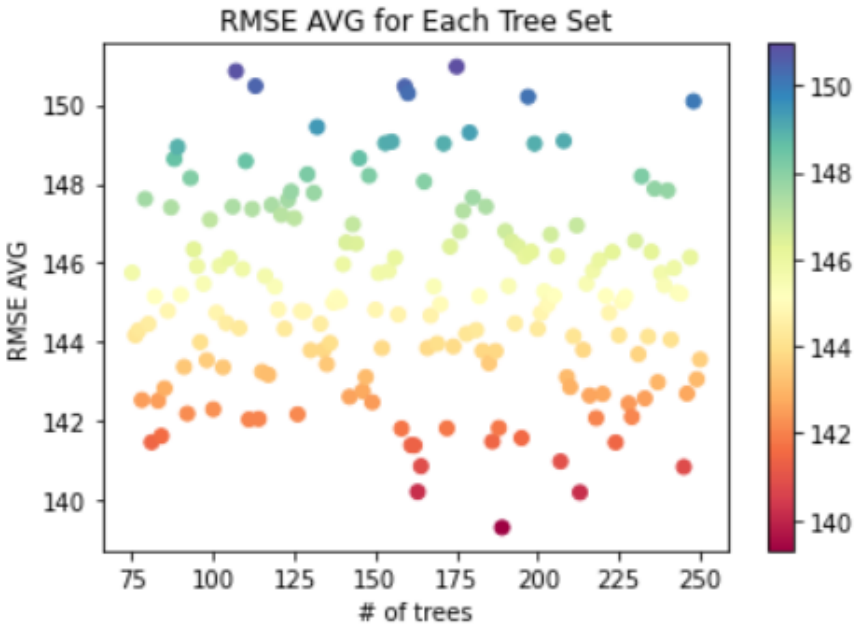
5.5 80/20 split with Bitcoin and Ethereum data

Fig 3.4

ErrorStat	Average	Lowest
RMSE	145.133	139.301
MSE	22645.606	20888.006
MAE	46.316	44.551

The lowest RMSE was located at tree set 190. Below the figure 3.5 shows the RMSE scores with the corresponding number of trees that were used. The MSE and MAE scores are shown in figures B.3 and B.4 in the appendix.

Fig 3.5

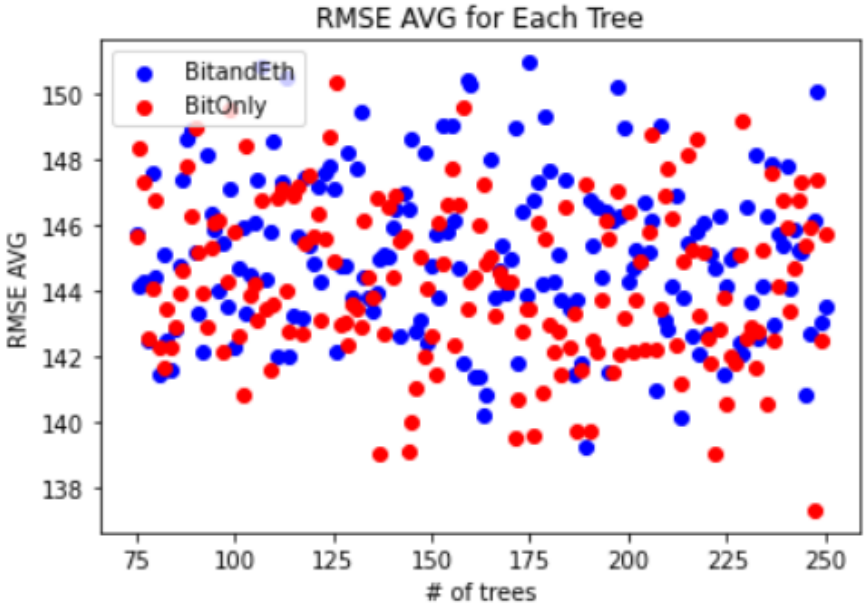


As it can be seen, the lowest RMSE achieved was a value of 139.301. This does not come close to the value of around 33 from LSTM by Tan and Kashef. The values graphed are not much different than the results from the 70/30 and 80/20 split shown from figure 2.2 and 2.4. A direct comparison of the 80/20 results are shown in the next section. It is not even an improvement, but most likely, the Random Forest Algorithm got confused with the amount of 0 values.

5.6 Comparison between Bitcoin and Ethereum

Below figure E.1 has the RMSE Average scores for Ethereum and Bitcoin data versus just Bitcoin data at an 80 20 split. The results are each of the averages from 100 runs for each set of trees.

Fig E.1



As it can be seen, there is not a clear difference when including Ethereum data and when not using Ethereum data. With the zeros mentioned earlier, it is not a great idea to include Ethereum for price prediction. A possible improvement could be to limit the dates to the Ethereum dates that do have values. For now, I will maintain my focus to include the dates to keep it as similar to Tan and Kashefs' work as possible.

5.7 Final attempt at room for improvement

The first and second attempts at predicting the closing price of Bitcoin have not been very successful. I attribute the first attempt's lack of great results to be due to the lack of data and the second's to be the lack of the right data. With the availability of more time, I have decided to include NASDAQ stock data into the mix. This was not mentioned in the guideline because it was not originally planned. The data comes from Yahoo Finance [3]. This form of stock includes many high-level tech companies. This means it might have some relation to the Bitcoin price prediction and can possibly lead to better predictions. Below, Figure 4.1 shows the columns used.

Fig 4.1

A	B	C	D	E	F	G	H	I	J	K	L
NasDaqop	NasDaqCl	NasdaqLo	NasdaqHig	NasdaqVo	Date	Open	High	Low	Close	Volume	MarketCa
0	0	0	0	0	########	135.3	135.98	132.1	134.21	0	1.49E+09
3290.31	3315.33	3289.42	3307.02	1.6E+09	########	134.44	147.49	134	144.54	0	1.6E+09
3308.05	3328.79	3298.58	3328.79	2E+09	########	144	146.93	134.05	139	0	1.54E+09
3325.35	3330.02	3296.5	3299.13	1.9E+09	5/1/2013	139	139.89	107.72	116.99	0	1.3E+09
3306.15	3344.9	3305.81	3340.62	1.8E+09	5/2/2013	116.38	125.6	92.28	105.21	0	1.17E+09
3371.41	3388.12	3370.3	3378.63	1.7E+09	5/3/2013	106.25	108.13	79.1	97.75	0	1.09E+09
3382.33	3396.21	3381.44	3392.97	1.5E+09	5/4/2013	98.1	115	92.5	112.5	0	1.25E+09
3382.33	3396.21	3381.44	3392.97	1.5E+09	5/5/2013	112.9	118.8	107.14	115.91	0	1.29E+09

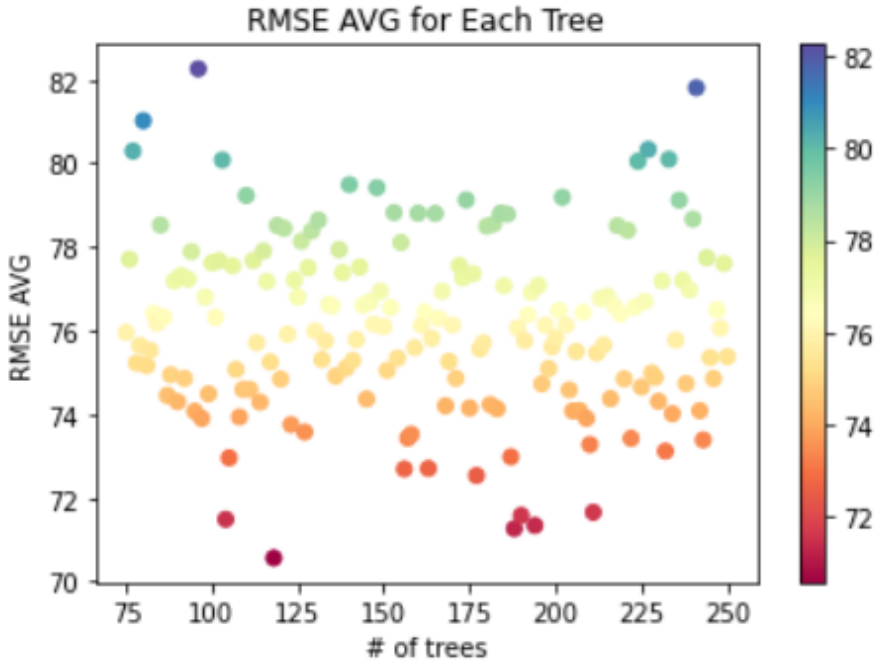
5.8 70/30 split with Bitcoin and NASDAQ data

Fig 4.2

ErrorStat	Average	Lowest
RMSE	76.123	70.566
MSE	6193.721	5286.410
MAE	23.262	22.154

The lowest RMSE was located at tree set 118. Below the figure 4.3 shows the RMSE scores with the corresponding number of trees that were used. The MSE and MAE scores are shown in figures C.1 and C.2 in the appendix.

Fig 4.3



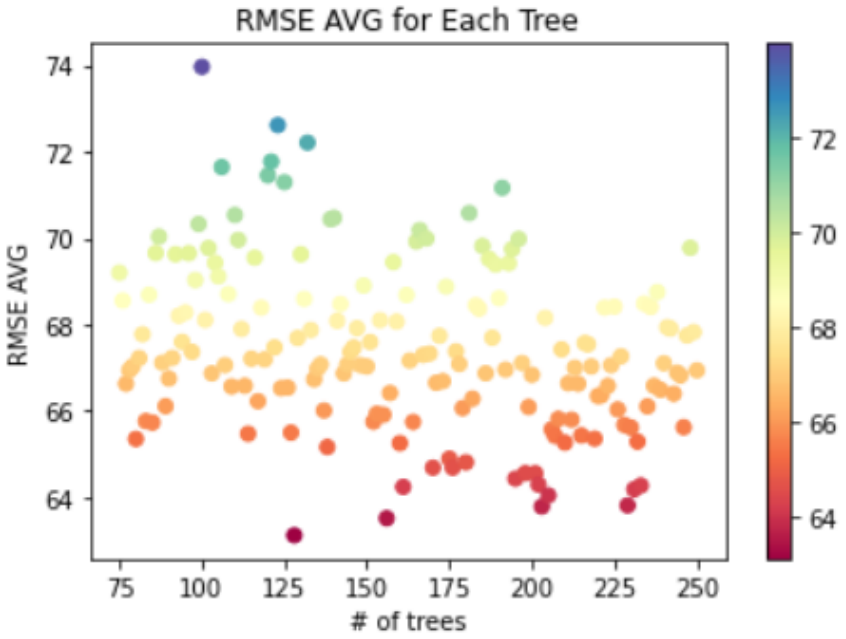
5.9 80/20 split with Bitcoin and NASDAQ data

Fig 4.4

ErrorStat	Average	Lowest
RMSE	67.444	63.123
MSE	4916.142	4203.321
MAE	20.972	20.135

The lowest RMSE was located at tree set 128. Below the figure 4.5 shows the RMSE scores with the corresponding number of trees that were used. The MSE and MAE scores are shown in figures C.3 and C.4 in the appendix.

Fig 4.5

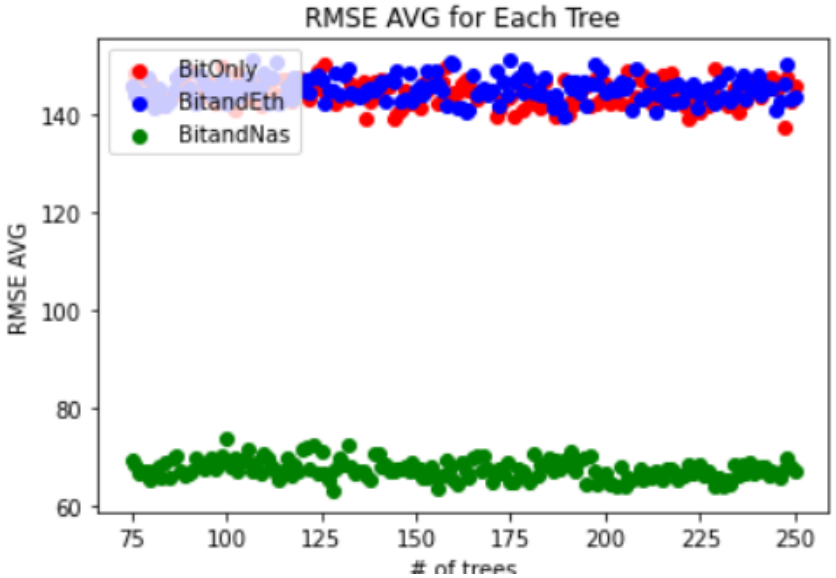


As seen above the RMSE values have improved dramatically compared to the previous results. The lowest being 63.123 for an 80/20 split this comes close to Tan and Kashef's LSTM result but not quite. The reason I believe NASDAQ data with bitcoin data influences the results so dramatically is because. The influence of Tech stocks would seem to have influence on the impact and outlook of bitcoin prices.

5.10 Comparison between Bitcoin only, Bitcoin and Ethereum Added, Bitcoin and NASDAQ

Below, figure E.2 has the RMSE Average scores for Bitcoin only, Bitcoin and Ethereum, Bitcoin and NASDAQ data at an 80 20 split. The results are each of the averages from 100 runs for each set of trees.

Fig E.2



As it can be seen Bitcoin and NASDAQ data have a large decrease in RMSE scores. The data inclusion decreases the error by almost half. NASDAQ data included shows a much needed improvement for Bitcoin closing price prediction.

6 Conclusion

With my results, I was not able to bring my Random Forest Model RMSE lower than the results that Tan and Kashef had with LSTM. I believe this can be due to 3 main issues.

1. The change of data: Tan and Kashef only provided a location where they got their data; they did not give a location where a file could have been used to download data. This means it is possible the data changed over time with people updating the page if prices were skewed or they did not have the most accurate results during a previous time.

2. Availability of Data It is possible Random Forest would need more data for its decision trees to predict a forecasting problem like this. I was able to show that with access to more data from NASDAQ, Random Forest RMSE was able to be cut in half. This means with even more data, it could be possible to improve results even further.

3. Problem Analysis It could be possible that Random Forest is not the best suited to solve this problem. It does not mean it is bad at it necessarily, but another algorithm could possibly perform better. In this case, LTSM was able to provide better results than I could with Random Forest.

Concluding thoughts I was not able to improve the results of Tan and Kashefs' research, but I believe including the NASDAQ data in direct use with Random Forest was a step in the right direction. All of the code, data and results can be found on Github at:

https://github.com/Czerrilla/SeniorProjectBitcoinPricePrediction.

I believe my results could be further improved upon by taking various steps mentioned in the next section.

7 Future Research

1. Different data: I would like to see different data used, whether that be trying to predict a different cryptocurrency or using other stock data to improve results further. I think it would also be interesting to use the Bitcoin data and some other data that would not be related to stocks to see if there can be any sort of other correlations that can be derived between the Bitcoin data and the other data features.

2. Different Algorithm: I chose Random Forest because I liked the concept of it and how it worked as well as the fact I never used it for prediction before and was interested in its capabilities but I think it might be good to try a different approach and choose another machine-learning algorithm to predict this problem to see if that could improve results even further.

3. Re-attempt Ethereum: As it was seen the zeros most likely influenced results heavily and therefore should be discarded from a possible way to improve Bitcoin price prediction. With this in mind it would be a good idea to decrease the amount of dates used to include only dates that have values. I believe with 2 cryptocurrencies that are fairly high in popularity, it could produce good results.

8 Appendix

8.1 MSE and MAE with just Bitcoin Data

Fig A.1 (70/30)

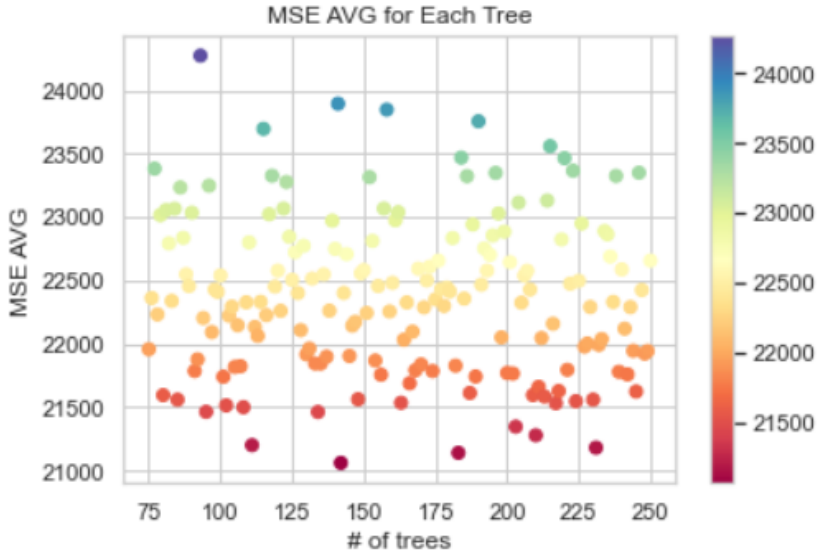


Fig A.2 (70/30)

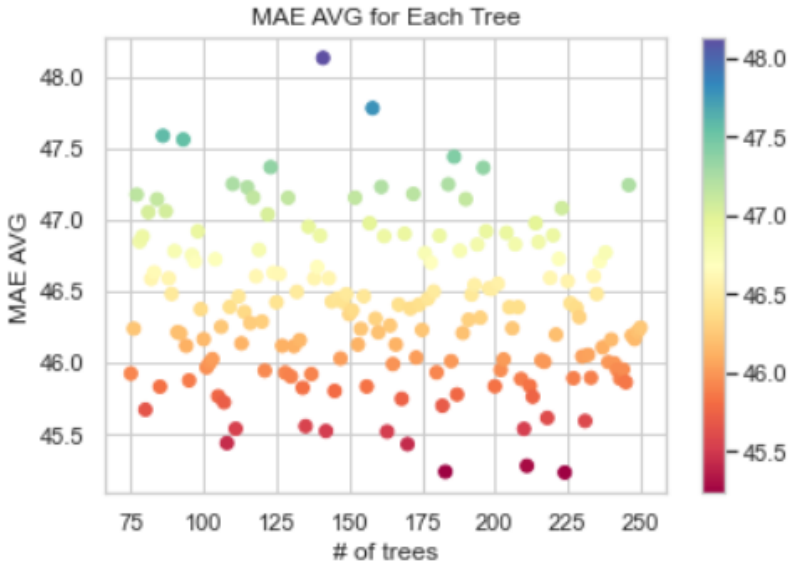


Fig A.3 (80/20)

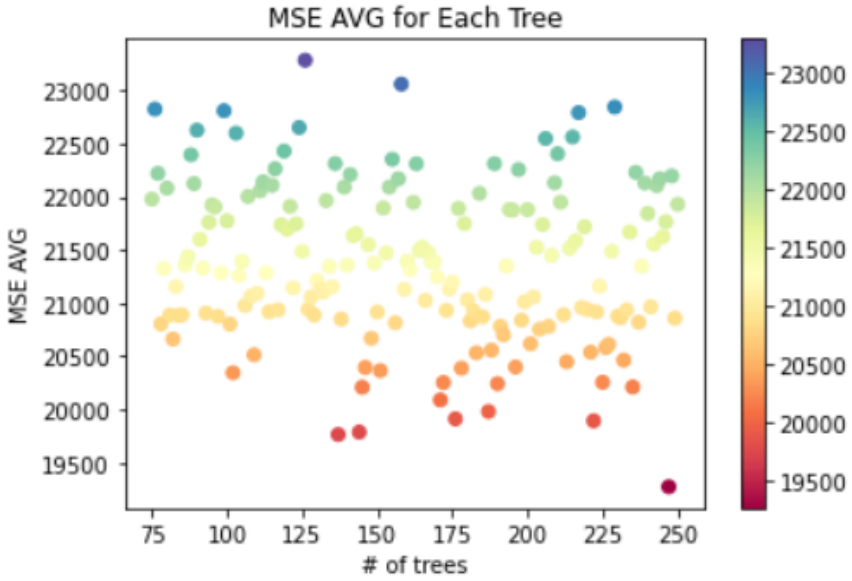
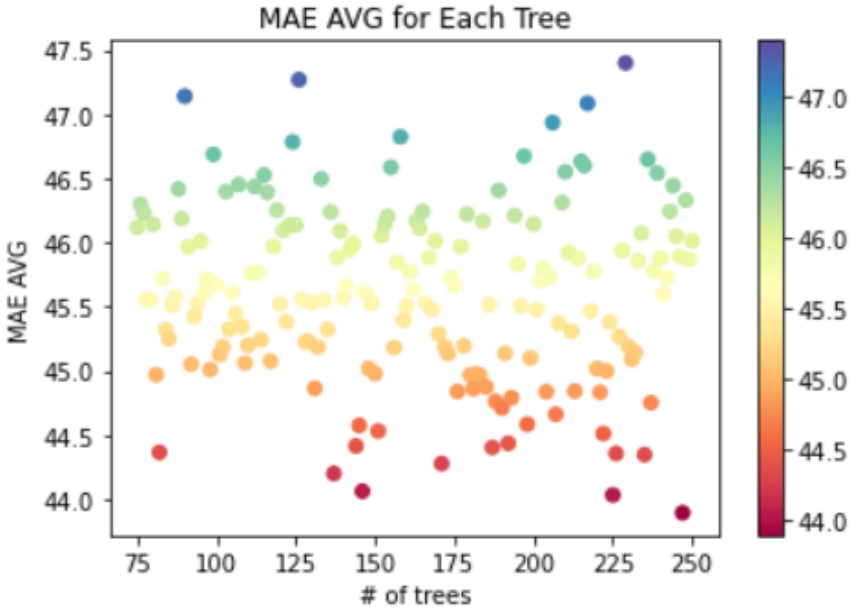

MSE AVG for Each Tree
MSE AVG
# of trees
23000
22500
22000
21500
21000
20500
20000
19500
75
100
125
150
175
200
225
250

Fig A.4 (80/20)


MAE AVG for Each Tree
MAE AVG
# of trees
47.5
47.0
46.5
46.0
45.5
45.0
44.5
44.0
75
100
125
150
175
200
225
250

8.2 MSE and MAE with Bitcoin and Ethereum Data

Fig B.1 (70/30)

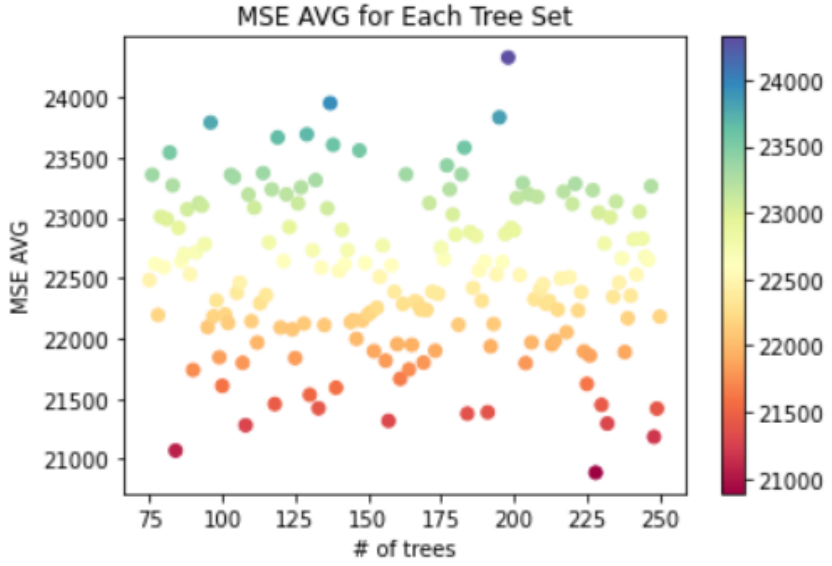


Fig B.2 (70/30)

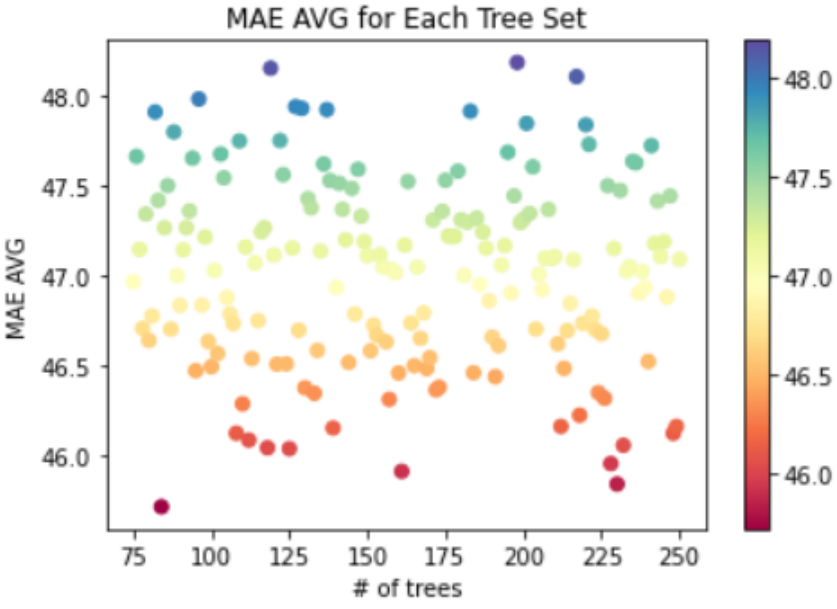


Fig B.3 (80/20)

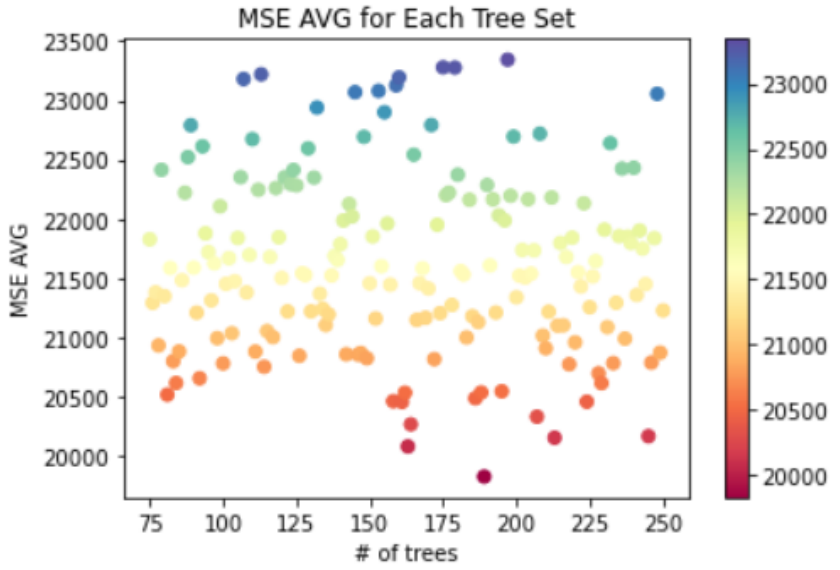
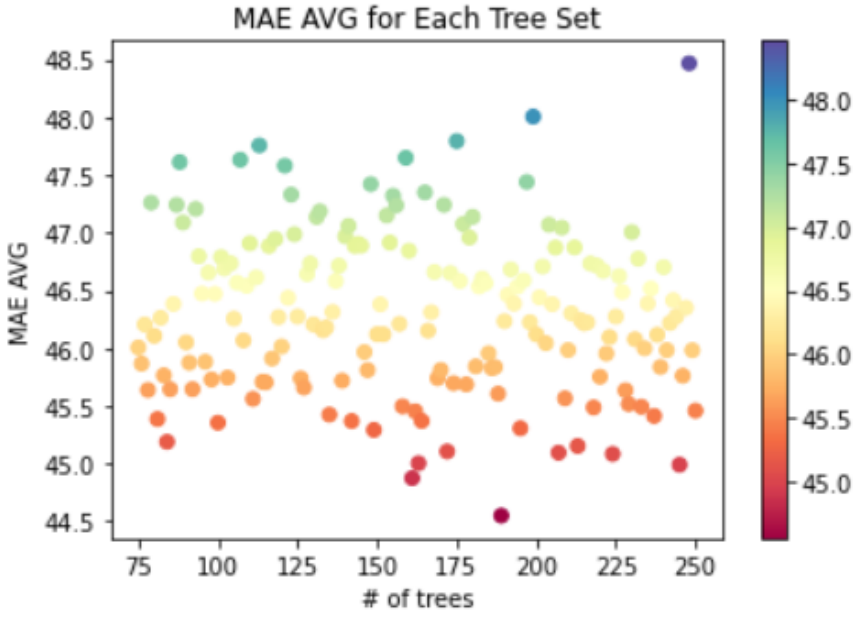

MSE AVG for Each Tree Set
MSE AVG
# of trees
23500
23000
22500
22000
21500
21000
20500
20000
75
100
125
150
175
200
225
250

Fig B.4 (80/20)


MAE AVG for Each Tree Set
MAE AVG
# of trees
48.5
48.0
47.5
47.0
46.5
46.0
45.5
45.0
44.5
75
100
125
150
175
200
225
250

8.3 MSE and MAE with Bitcoin and Nasdaq Data

Fig C.1 (70/30)

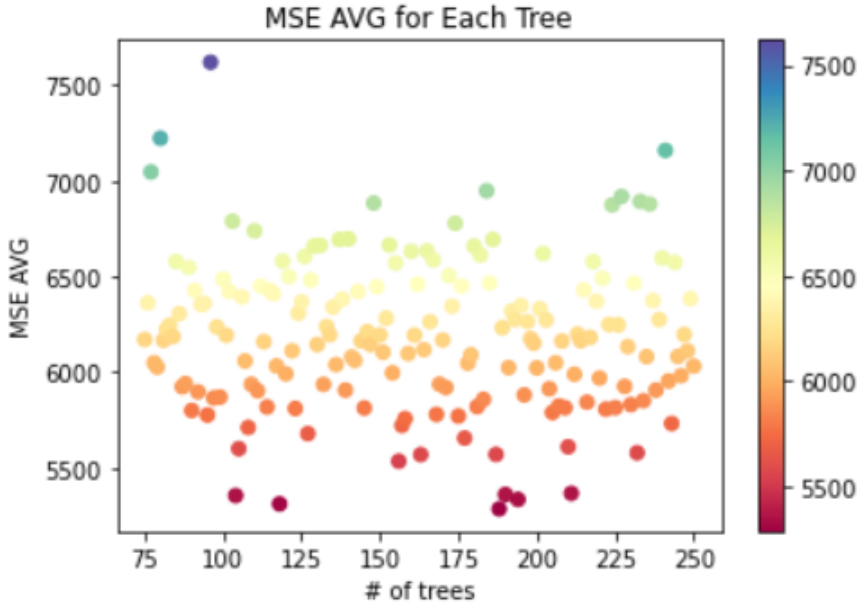


Fig C.2 (70/30)

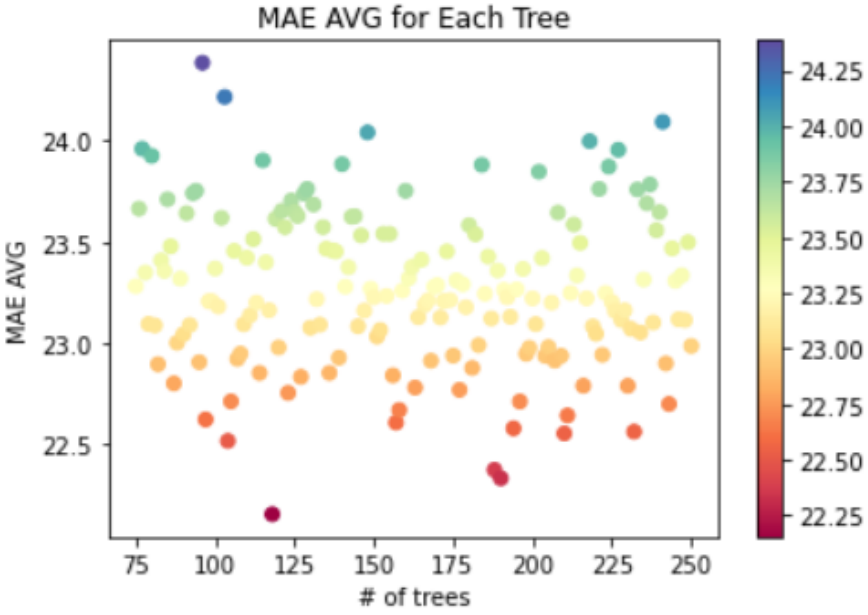


Fig C.3 (80/20)

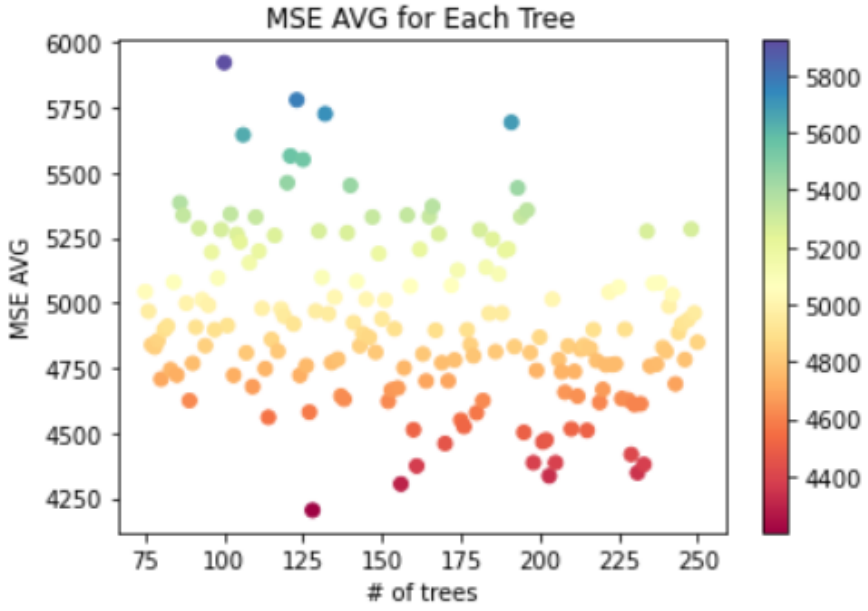


Fig C.4 (80/20)

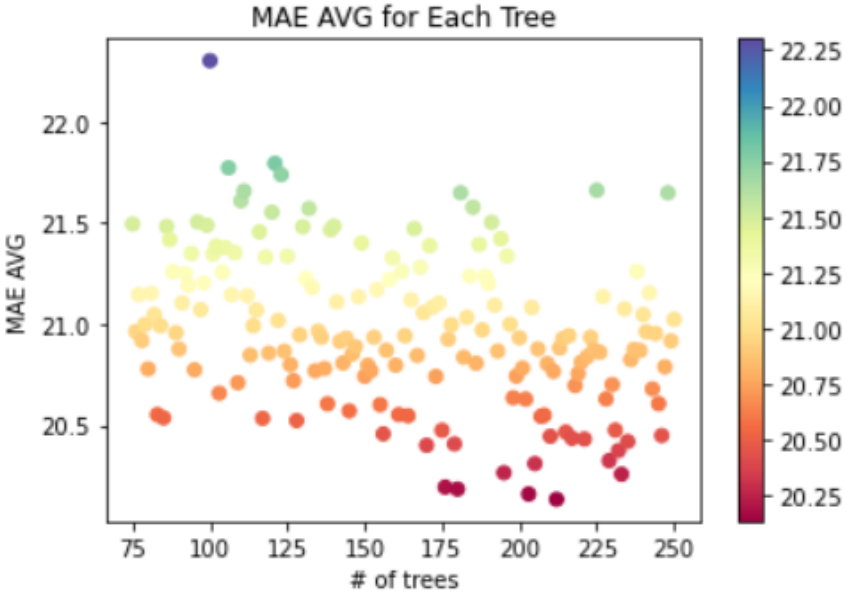


Fig M.1

```
Bitcoin_data = pd.read_csv('Bitcoindata.csv')
#no catagorical data
Bitcoin_data=Bitcoin_data[['Open','High','Low','Volume','Close']]
Bitcoin_data
#split the columns we use open, high, low, volume for X
X = Bitcoin_data.iloc[:,0:4].values
#split the columns we use close for Y
Y = Bitcoin_data.iloc[:, 4].values
# 3 emtpy arrays to hold the average RMSE, MAE, and MSE for each tree
RMSEAVG=[]
MAEAVG=[]
MSEAVG=[]
# run on a number of trees 1 to 300
for i in range(1,301):
    print(i)# keep track
    #initialize all the sums to 0
    RMSETotal=0
    MAETotal=0
    MSETotal=0
    #run each tree set a 100 times for better randomness
    for j in range (1,101):
        #split the data into training and split sets using random seed none to keep getting random splits test size is 30%
        X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,random_state=np.random.seed(None))
        #sets the number of trees
        regressor = RandomForestRegressor(n_estimators=i)
        regressor.fit(X_train, Y_train)
        Y_pred = regressor.predict(X_test)
        # add each of the RMSE, MAE, and MSE totals respectively
        RMSETotal=RMSETotal+np.sqrt(metrics.mean_squared_error(Y_test, Y_pred))
        MAETotal=MAETotal+metrics.mean_absolute_error(Y_test, Y_pred)
        MSETotal=MSETotal+metrics.mean_squared_error(Y_test, Y_pred)
    #when random results are added to their sums divided each by the number of times ran which was 100
    RMSEAVG.append(RMSETotal/100)
    MAEAVG.append(MAETotal/100)
    MSEAVG.append(MSETotal/100)
    print(RMSETotal/100)
```

References

[1] Coinmarketcap bitcoin. `https://coinmarketcap.com/currencies/bitcoin/historical-data/`, May 2022.

[2] Ethereum data. `https://coincodex.com/crypto/ethereum-classic/historical-data/`, May 2022.

[3] Nasdaq composite (ixic). `https://finance.yahoo.com/quote/`, May 2022.

[4] Qinghe Li. Predicting trends of bitcoin prices based on machine learning methods. *2020 The 4th International Conference on Software and e-Business*, 2020.

[5] et al Mangla, Neha. Bitcoin price prediction using machine learning. 2019.

[6] et al Polamuri, Subba Rao. Stock market prices prediction using random forest and extra tree regression. *International Journal of Recent Technology and Engineering*, 2020.

[7] Perry Sadorsky. A random forests approach to predicting clean energy stock prices. *Journal of Risk and Financial Management*, 14(2):48, 2021.

[8] Xue Tan and Rasha Kashef. Predicting the closing price of cryptocurrencies. *Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems - DATA '19*, 2019.

[9] Mehar Vijh, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar. Stock closing price prediction using machine learning techniques. *Procedia Computer Science*, 167:599–606, 2020.