PREDICTING CRYPTOCURRENCY FUTURE PRICES

Bitcoin, Ethereum, XRP, Litecoin, Shiba Inu, Saitama

ABSTRACT

Can machine learning be used to predict cryptocurrency future prices based on "social media trends" and historical data (Inamdar, et al., "Predicting cryptocurrency value using sentiment analysis" [1])? Random Forest Regression was used to train models for Bitcoin and other coins for comparison. Bitcoin new feature data was used to predict close prices. My conclusion is the same as Inamdar, et al., which is Bitcoin future prices can be predicted.

Tom Va
ITCS-5156 – Applied Machine Learning

https://github.com/tomtongva/itcs5156-051-final-project

Contents

Introduction

A cryptocurrency is a world-wide community-regulated digital currency without ties to any central bank or government. For the purpose of this paper the terms cryptocurrency, coin, and token will be used interchangeably, even though they are technically different in the cryptocurrency space. Since a cryptocurrency is community-regulated, two entities can decide on the price of a coin without going through a third-party. An entity can be a trader or a trading bot, and can belong to multiple cryptocurrency communities. The aforementioned properties along with other complexities of a cryptocurrency, can lead to dramatic price fluctuations. This paper discusses how machine learning is used to **predict cryptocurrency future prices**.

Problem, Motivations, Challenges, Solution Summary

First, the problem I tried to solve in this project was to use machine learning to predict cryptocurrency future prices based on the text in Tweeter and historical data. Second, a practical motivation behind predicting future prices is for investors to have better return on their investments. A personal motivation was to learn how to use a given dataset to predict the outcomes, for example prices for the next few days. As my project evolved I also became interested in comparing different cryptocurrencies. Third, to complete my project I had to overcome many challenges such as re-reading different papers to understand how they used machine learning to predict future prices. The other main challenge was to develop code that can take time-series data to train a machine learning model. Last, I was able to predict future prices for Bitcoin using a Random Forest Regression model.

Background/Related Work

Before venturing into tuning my Random Forest Regression model, I read different papers to get a good survey of the existing approaches. The first or main paper I chose to replicate is "Predicting cryptocurrency value using sentiment analysis" [1] for Bitcoin. My original intent was to include a text/sentiment feature from social media such as Facebook, Twitter, or Reddit. But because the authors concluded that the sentiment feature did not have a huge impact on predicting prices I dropped this feature from my model. The paper concluded with a prediction for future day 1 with a mean absolute error (MAE) of 2.7526 and root mean square error (RMSE) of 13.7033. Inamdar, et al. only went as far as predicting up to future day 2 with a mean absolute error of 3.1885 and a root mean square error of 15.1686. Note that the authors mentioned future day 1 and day 2 prices can vary up to a little over $13,000 and $15,000, respectively.

The second paper, "Cryptocurrency price analysis with artificial intelligence," [2] predicted future prices for Bitcoin, XRP, and Ethereum. The authors, Yiying and Yeze, also used historical data such as opening, high, low, and close prices as features for their predictions. But instead of using Random Forest Regression, they used a vanilla artificial neural network (ANN) and a long short-term memory (LSTM) recurrent neural network (RNN) to compare predictions. In their ANN model they found that increasing the historical dataset does not produce a better model for these three coins. This idea of smaller dataset was extended to their LSTM model. Even though both these models concluded that predicting prices one day into the future were comparable they did note that LSTM model needed less historical data as opposed to ANN.

The third paper, "Forecasting Cryptocurrency Prices using Machine Learning" [3] was interesting because the author, Chaudhari, compared a statistical model, AutoRegressive Integrated Moving Average (ARIMA), with two machine learning models, LSTM and Facebook's Prophet. The paper evaluated Bitcoin, Ethereum, and Litecoin cryptocurrencies based on additional performance metrics such as mean absolute error (MAE), mean absolute percentage error (MAPE), and R2 score. The conclusion is that the LSTM model outperformed the other two models when using historical data to predict future prices.

Each of these three approaches have pros and cons. In the first paper even though they noted that sentiment feature does not have an impact on predicting prices they still included it in their training. Therefore this just added noise to their data. On the positive side because they used Random Forest Regression it is easier to tune the model as compared to the other two papers. In the second paper they seem to be selecting datasets that performs better with LSTM. This gives better results but introduces their own bias, which may perform worst on new data. In the third paper the pro is that LSTM was used to compare to Facebook's Prophet model and the ARIMA statistical model. This comparison demonstrated how well a standard machine learning methodology performs as compared to other models. The con here is that the dataset may be bias since it selected data samples from a four-years-range without accounting for some of the irregularities during this period. The author alluded to this point by mentioning that the performance for these models could be better.

I used the different cryptocurrencies, from these three papers, in evaluating a model because their prices had a small value to large value range. For example, on the lower end of the spectrum were XRP prices, which were around $1. And on the other end of the spectrum were Bitcoin prices, which were around $40,000. This inspired me to include two other coins, Shiba Inu and Saitama because their prices were even smaller, around $0.00004 and $0.00000005, respectively. The sklearn.preprocessing.StandardScaler was used to scale the features. The results turned out well when using the Random Forest Regression model I chose from the first paper.

Method

Since cryptocurrency data are time-series data, I converted them into a supervised learning problem. To do this I adapted code from *machinelearningmastery.com* [4]. Then after training my model I used new datasets to predict future prices.

Step 1 in Diagram A below, shows analyzing the data from various sources, and downloading of the data into CSV files. I chose *coinmarketcap.com* [7] as my source because the website was easier to navigate. I used the following columns as my features: Open* (opening price), High (daily high price), Low (daily low price), and Volume. The Close** (close price) column was used as the target. I first copied the month of January 2022's data for Bitcoin into Excel and removed all dollar signs and commas. Then I finally saved the data into a CSV file.
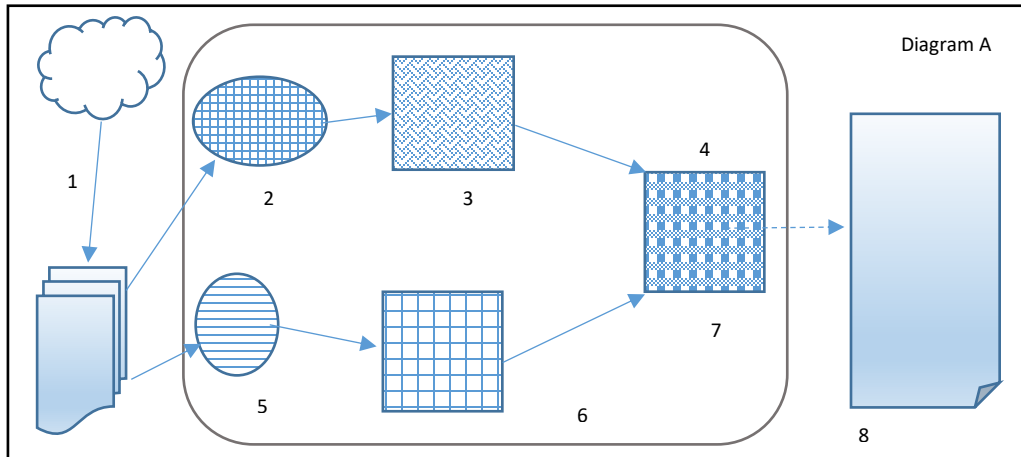
Diagram A

Step 2 in the diagram, I initially modeled my code after a project at *kaggle.com*[5]. I did this because none of the papers I chose provided any source code or description of their algorithms. After I got more comfortable with the machine learning concepts and code, I scoured the internet for alternative solutions. This is how I arrived at Jason Brownlee's "Random Forest for Time Series Forecasting" [4] solution for monthly births. The dataset was different from mines, but the concept was the same. So I adapted his solution for my time-series problem.

The sliding window algorithm was used to convert Bitcoin, and later other coins, time-series data into a supervised machine learning problem. This algorithm drops the date/time column and uses the previous time step's value to predict the next step's value. The algorithm is implemented in the series_to_supervised() function.

After validating the sliding window function, I split the Bitcoin dataset into two months of testing set and 12 months of training set. To do this I modified the walk_forward_validation() function to take in a model object parameter and a boolean parameter to disable/enable printing of the actual versus predicted prices. This method test each day in the testing set with the training set.

For the actual prediction, instead of just initializing the RandomForestRegressor() to 1,000 estimators I found that 100 worked well for my dataset. I also used a Pipeline with StandardScalar() since prices are in the tens of thousands of dollars, but volume is in the hundreds of millions of dollars. This is initialized outside of and passed into the prediction function random_forest_forecast().

Step 3 in the diagram is where the model was trained. This is where it is important to identify the correct hyperparameters, otherwise training can take an excessive amount of time but still give the same results; hence, I changed the number of estimators to 100 instead of 1,000. I also played around with different models such as Linear Regression, Lasso, Elastic Net, etc., and found that they produced similar results. Eventually, I settled on Random Forest Regression because this is what my main paper uses to predict future prices.

Step 4 is where I evaluated how well my model did with the training data. In the very first evaluation I only worked with one month's worth of Bitcoin data, but because the model produced such good results I expanded my date range to six months, and then to one year. After verifying Bitcoin I

went back and modified my code to include the rest of the coin. Plot A shows the actual prices versus the predicted prices for my chosen coins.
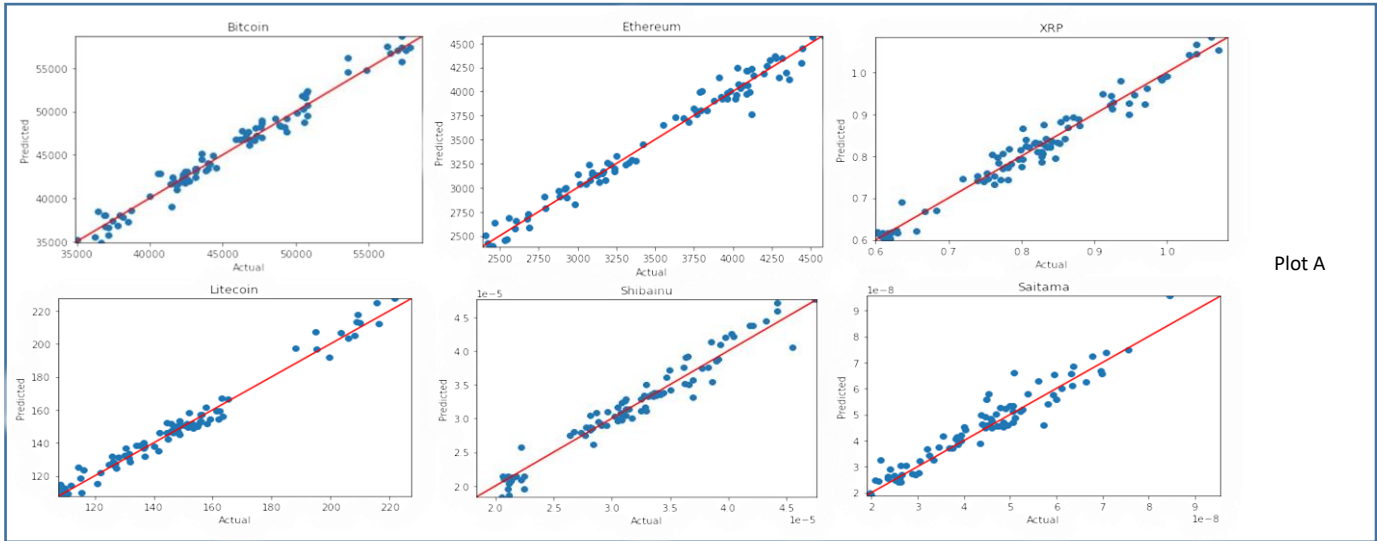

Plot A

Table A, below, shows the performance metrics of each coin.

Coin	MAE	R2
Bitcoin	952.200	0.962
Ethereum	85.143	0.969
XRP	0.018	0.968
Litecoin	4.712	0.956
Shiba Inu	0.000002126935	0.844
Saitama	0.000000006634	0.814

Table A

*Note: price range for Bitcoin is 10^5, Ethereum is 10^3, XRP is 10^1, Litecoin is 10^2, Shiba Inu is 10^{-8}, and Saitama is 10^{-12}

Step 5 is where new feature data is loaded to predict future prices. One year's worth of historical data for Bitcoin, Ethereum, XRP, Litecoin, and Shiba Inu were copied from CoinMarketCap. For Saitama historical data only goes as far back as June 18, 2021 or since inception. All data up until, but not including January 1, 2022 were used for training and testing. All data from January 1, 2022 until February 18, 2022 were used as new data. The new data was formatted in such a way where only the feature columns stayed in the file. This is so I can simplify the code when loading these new features into a two dimensional array to predict future prices. Each row is a consecutive future day, starting with future day 1. For example in Screenshot A, data row 1 is for future day 1, data row 2 is for future day 2, and so on.

Screenshot A

Step 6 and 7 is where the new feature data is read into a 2-dimensional array. This is done in my own wrapper function called predict_next_days_prices(). The implementation basically loops through the given 2-dimensional array and passes the features to predict_next_day_price() function, adapted from Jason Brownlee's[4] code, to predict the next day's price. Each prediction is then added back to the original data frame through another wrapper function, add_predicted_val_to_coin(). For example, the data frame in Screenshot B shows the predicted rows added back to the original Bitcoin data frame. The Close prices for all the rows with January dates were predicted from their respective features using the same Random Forest Regression model setup as with the training data.

Date	High	Low	Open*	Volume	Close
2021-12-31	48472.53	45819.95	47169.37	3.697417e+10	46306.450000
2022-01-01	40026.02	40418.88	39713.06	1.373656e+10	40513.436600
2022-01-02	40026.02	40418.88	39713.06	1.373656e+10	40556.937418

Screenshot B

Step 8 is where the predicted close prices along with their performance scores were generated and written to file. This extra step is so I can validate my predictions with the original close prices. Screenshot C is an example of the results for Bitcoin.

predictedpricesBitcoin.txt - Notepad

File Edit Format View Help

```
future day 1 40842.69848778429 for Bitcoin
MAE: 916.5474606644427, R2: 0.9718229261334549
future day 2 44629.20474304157 for Bitcoin
MAE: 897.9571317744131, R2: 0.9736977341419334
future day 3 44947.619490899575 for Bitcoin
MAE: 890.1799184653436, R2: 0.9738419614661392
future day 4 43525.631930895644 for Bitcoin
MAE: 873.2746270473015, R2: 0.9753898141670522
future day 5 43175.23511785236 for Bitcoin
```

Screenshot C

Project Experiment – Deep Dive

The core of the project setup is to get the feature data, from multiple cryptocurrencies, into input for the sklearn.ensemble.RandomForestRegressor so it can produce close prices. The dictionary data structure maps a coin name to its data. This dictionary along with a "convenience" array which contains the name of each of the coin are initialized at the beginning. The dictionary is then updated with time series data from CSV files. The time series data is then converted into a supervised machine learning problem. Finally the data for each coin is fed into Random Forest Regressor model for training and prediction.

The Journey

Before going into the details of the project setup, I would like to share my journey through this project. I began this journey with an unrelated topic, and wanted to implement a machine that can navigate an area while avoiding obstacles. But as I dove deeper into the details it seemed too complex to implement within the time frame of this course. While searching for example projects on *kaggle.com*, as suggested, by Professor Lee, I came across some cryptocurrency machine learning projects. Because I am somewhat familiar with cryptocurrency, this motivated me to dive into some of these code instead.

The idea of predicting future cryptocurrency prices using machine learning came to me while looking through some of the kaggle projects. As luck would have it, when I did a search for "predicting cryptocurrency future prices" in Google Scholar I found many results on this topic. Overjoyed, I started reading through some of the papers. It was a bit overwhelming because I understood only 40% or less of the technical details in these articles. Luckily, I came across a paper called "Predicting cryptocurrency value using sentiment analysis" [1]. I immediately knew that this was going to be my choice of topic.

I wanted to find what impact Facebook, Twitter, Reddit, and in general social media posts had on the price of cryptocurrencies, and how I can leverage this feature in predicting future prices. Unfortunately, Inamdar, Abid, et al. [1], the authors of "Predicting cryptocurrency value using sentiment analysis" [1], came to the conclusion that this feature did not have much of an impact on future prices of Bitcoin. Under pressure from a deadline to submit a summary of the three papers I have chosen for my project, I chose this as my main paper, and the other two papers were the ones I recently read.

The other two papers were "Cryptocurrency price analysis with artificial intelligence" [2] and "Forecasting Cryptocurrency Prices using Machine Learning" [3]. In the former paper, Yiying and Yeze compared a vanilla artificial neural network (ANN) with a Long Short-Term Memory (LSTM) recurrent neural network, and concluded that their models were able to predict future prices for Bitcoin, Ethereum, and XRP. In the latter paper, Chaudhari compare a statistical model called AutoRegressive Integrated Moving Average (ARIMA) with LSTM and Facebook's Prophet machine learning models. The conclusion was that LSTM did a much better job at predicting future prices for Bithcoin, Ethereum, and Litecoin.

As mentioned earlier in this paper, I started with code from Renuka Devi at *https://www.kaggle.com/renukadevird/crypto-corr* [5] because I misunderstood the project assignment as trying to produce a fresh solution versus replicating my main paper's solution. However, this was not all for naught because this helped me understand the trend of Bitcoin, Shiba Inu, and Saitama. In fact a

valuable technique I learned from this solution was to convert the Date column values in my dataset to row indexes. I also improved on lessening duplicate code with multiple coins by mapping the name of these coins to their datasets in a dictionary. For example, instead of copying and pasting the same code to multiple lines and just changing the coin name or dataset, I just loop through my map and execute one line of code multiple times.

Once it became clear to me that I'm supposed to replicate the solution from my main paper, I scoured the internet for existing code because the authors did not provide source code nor a description of their algorithm. In fact the second and third paper also did not provide their source code nor algorithm. Luckily, I came upon Jason Brownlee's "Random Forest for Time Series Forecasting" [4] tutorial. It was in this tutorial that I first learned about time-series data (taught in class a couple of weeks later) and how to convert them into a supervised learning problem. Also, by this time I had learned most of the machine learning concepts in my papers. Re-reading the papers gave me clarity on the next steps in my project.

Project Setup

There are three main parts in my project setup. The first part is copying the data for each cryptocurrency coin from *coinmarketcap.com* and preparing them for processing. The second part is the actual code to train a machine learning model for future price prediction. The third part is predicting future prices on new data.

Project Setup – Retrieving Datasets

Cryptocurrency historical data are available on many exchange and finance websites. In fact some exchanges actually provide APIs to pull historical data. Even though there are a plethora of places data can be retrieved from, they all show slightly different numbers. For example, the close price for one day is reported for Bitcoin on CoinMarketCap as $42,412.43, Coindesk (another cryptocurrency data source) shows $42859.15, and Wall Street Journal shows $44502. I chose CoinMarketCap because it had a more user friendly interface.

Since my papers worked with Bitcoin, XRP (Ripple), Ethereum, and Litecoin, I included them in the project. While looking through the data for those coins, I realized that there were two new coins with extremely low price values, Shiba Inu and Saitama; therefore, these were also included in the project. One of the reasons I chose so many coins is because I wanted to verify the predictions of my chosen papers. The other reason is to compare how these coins stacked up against each other with their wide range in prices, from the very small to very large. Table B shows the different close prices for each coin on a particular day.

Coin	Close Price
Bitcoin	$37,296.57
Ethereum	$2,590.36
XRP	$0.6987 (usually around $1)
Litecoin	$105.97
Shiba Inu	$0.00002387
Saitama	$0.00000001581

Table B

As mentioned earlier, I started with data for just Bitcoin because that was the coin my main paper predicted prices on. First, one month's worth of data was copied; the dollar sign ($) and commas from each historical number was removed by using Microsoft Excel. Then the file was saved as a CSV file. After a working machine learning model was found, six months' worth of data for Bitcoin and the other five coins were copied, repeating the manual data preparation on each coin and saving them to their own CSV file. Later on some of the data needed to be split into model training and testing, and some into new data for predictions; therefore I copied one year's worth of data, splitting data from February 18, 2021 to December 31, 2021 as testing and training data. The data from January 1, 2022 to February 18, 2022 were used as new data.

The format for new data file is different from training/testing data file. This is because only the features are needed to predict future prices. Therefore only Open*, High, Low, and Volume columns are in the file. Screenshot D shows the difference in training/testing data format, on the left, and the new data (features only) format, on the right.

```
1   Date,Open*,High,Low,Close**,Volume,Market Cap              1   Open*,High,Low,Volume
2   31-Dec-21,4.742E-08,4.964E-08,4.71E-08,4.85E-08,13962909,0  2   1.88E-08,2.02E-08,1.68E-08,14450435
3   30-Dec-21,4.892E-08,4.908E-08,4.539E-08,4.741E-08,12751490,0 3   1.85E-08,1.89E-08,1.65E-08,10809249
4   29-Dec-21,4.721E-08,5.357E-08,4.667E-08,4.892E-08,14126982,0 4   1.97E-08,1.97E-08,1.80E-08,8889964
5   28-Dec-21,5.055E-08,5.064E-08,4.552E-08,4.721E-08,11122305,0 5   1.91E-08,2.02E-08,1.86E-08,12862550
6   27-Dec-21,0.00000005,5.099E-08,4.868E-08,5.055E-08,11947084,0 6  1.80E-08,2.12E-08,1.68E-08,12263913
7   26-Dec-21,5.725E-08,5.879E-08,4.993E-08,4.995E-08,24591674,0 7   2.00E-08,2.02E-08,1.72E-08,11615385
8   25-Dec-21,4.476E-08,5.734E-08,4.305E-08,5.734E-08,17957294,0 8   1.72E-08,2.12E-08,1.62E-08,21724979
9   24-Dec-21,4.602E-08,4.611E-08,4.42E-08,4.474E-08,12152768,0  9   2.09E-08,2.18E-08,1.72E-08,26406683
10  23-Dec-21,4.506E-08,4.724E-08,4.416E-08,4.602E-08,11880693,0 10  2.88E-08,2.92E-08,2.05E-08,48598653
11  22-Dec-21,4.954E-08,5.001E-08,4.472E-08,4.500E-08,18258601,0 11  3.02E-08,3.11E-08,2.78E-08,21517492
```
Screenshot D

Project Setup - Code and Execution

My Python code is in a Jupyter Notebook file called ProjectCryptoPricePrediction-Final.ipynb. The code file and the data files all reside in one directory. In fact the predictions for the new data are also printed to the same directory. At the beginning of the code there is an array with each coin's name so that they can be easily identified in the code. Then the training dataset is loaded individually and wrapped into a dictionary, with the name of the coin as the key; see Code Snippet A. Later on the code just loops through the dictionary to train and predict new prices for each coin. The code is executed by sequentially running each Jupyter cell.

```python
# use these as the keys to map, add name of coin here so don't need to do it everywhere else
crypto_names = ['Bitcoin', 'Shibainu', 'Saitama', 'XRP', 'Ethereum', 'Litecoin']

#Load coin data
c_bitcoin = pd.read_csv('bitcoin-coinmarketcap-year2021-only.csv')
c_shibainu = pd.read_csv('shibainu-coinmarketcap-year2021-only.csv')
c_saitama = pd.read_csv('saitama-coinmarketcap-year2021-only.csv')
c_xrp = pd.read_csv('xrp-coinmarketcap-year2021-only.csv') #this is the Ripple coin
c_ethereum = pd.read_csv('ethereum-coinmarketcap-year2021-only.csv')
c_litecoin = pd.read_csv('litecoin-coinmarketcap-year2021-only.csv')

coin_dict = {crypto_names[0]:c_bitcoin, crypto_names[1]:c_shibainu, crypto_names[2]:c_saitama \
            ,crypto_names[3]:c_xrp,crypto_names[4]:c_ethereum,crypto_names[5]:c_litecoin}
```
Code Snippet A

Project Setup - Predicting Future Prices On New Data

After getting an acceptable Random Forest Regression model, the new data, divided from the original dataset, was used to predict consecutive future day close prices. In the new dataset only feature columns used in training the model was kept in the file. Each row corresponds with a consecutive future day. For example, in Screenshot D the first data row is future day 1, the second data row is future day 2, and so on. The future day does not correspond with the original data's day. For example the first row for future day 1 is not January 1 (the day after the last training day, December 31). Instead it's February 18, or the last day of the original dataset. You can think of it as the new dataset is in the reverse order of original dataset.

The Results

Table A, from above, shows the training results. The R2 score for Bitcoin is 0.962 and MAE is 952.2. Ethereum's R2 score is 0.969 and MAE is 85.143. XRP's R2 score is 0.968 and MAE is 0.018. Litecoin's R2 score came in at 0.956 and its MAE score came in at 4.712. Shiba Inu's R2 score is 0.844 and MAE is 0.000002126935. Finally, Saitama's R2 score is 0.814 and MAE is 0.000000006634.

The R2 scores for the coins mentioned in my chosen paper are better than the two other coins I picked. This indicates the more history a coin has the more the features are able to explain the predicted close prices. The MAEs are larger for Bitcoin and Ethereum because their historical values are in the tens of thousands and thousands, respectively. As the values dropped in magnitude the MAEs also dropped.

The linear regression line plots in Plot A, from above, also indicate that the data points for Bitcoin, Ethereum, XRP, and Litecoin fit better than Shiba Inu and Saitama. This is also supporting evidence that the model needs more historical data to predict prices for the latter coins. The coins can also be separated to execute on their own to find better hyperparameters to increase the number of data points that fits closer to the linear regression line.

Future Day	MAE	R2 Score	Predicted Close Price									
1	546.6876	0.986113	40871.21	16	364.8007084	0.984157252	36444.05		33	234.7082	0.99054	43954.89
2	541.973	0.985444	44598.91	17	383.6037443	0.98293303	38466.22		34	240.2459	0.989986	44041.42
3	512.5285	0.985584	44758.97	18	372.9509905	0.981860895	38460.73		35	228.4044	0.991729	44036.60
4	515.6148	0.985607	43800.43	19	386.2976426	0.981095435	38186.85		36	214.0108	0.991655	43792.35
5	516.8244	0.984554	43399.57	20	351.317939	0.982995854	38083.08		37	210.6369	0.991699	44612.32
6	482.6229	0.985224	43206.18	21	347.0971355	0.982230894	38069.69		38	219.467	0.99109	43942.51
7	472.1468	0.984953	43760.94	22	332.9789085	0.984388593	36765.58		39	193.2132	0.992603	43798.09
8	456.7396	0.985056	44439.35	23	330.8363155	0.985086967	35867.64		40	182.2736	0.992654	42122.39
9	455.1366	0.983611	44815.11	24	314.5785244	0.986436661	37025.92		41	183.5619	0.992319	43230.39
10	439.63	0.983512	44722.85	25	302.824739	0.987391312	35834.59		42	162.8815	0.992969	42334.12
11	429.2274	0.984028	44786.45	26	313.8522254	0.986372148	35908.17		43	150.7625	0.995029	43985.23
12	408.0768	0.985476	43875.48	27	275.6187153	0.989468362	34932.03		44	149.9572	0.993898	44439.70
13	407.6707	0.985294	42576.99	28	270.0380593	0.989624298	35906.30		45	150.336	0.994141	44999.38
14	382.3159	0.985512	42294.99	29	278.4166053	0.989479151	41137.35		46	135.9151	0.995203	45716.86
15	378.6582	0.984938	40632.78	30	252.058485	0.990740722	43755.03		47	143.8402	0.996022	47178.77
				31	246.5563945	0.991022584	42560.83		48	132.4612	0.996757	48174.00
				32	229.5176227	0.991222836	42660.25					

Table C – Bitcoin Future

Table C, above, shows the predicted prices and the model's performance on Bitcoin's new data. As more days are predicted the R2 score improves to 99%, starting on day 30. The MAE also improves over time. Figure B, below, shows how the predicted close prices compare to actual prices.
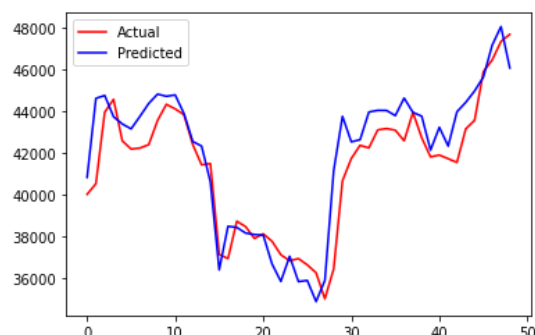
Conclusion

The Random Forest Regression model demonstrates the ability to predict prices for cryptocurrencies such as Bitcoin, Ethereum, XRP, Litecoin, Shiba Inu, and Saitama. The model outputs better predictions for Bitcoin, Ethereum, XRP, and Litecoin. Since Shiba Inu and Saitama are newer coins their predictions were a little worst, but I believe it will have better predictions as more data becomes available. Overall, my conclusion is that Random Forest Regression can be a model to predict close prices for any cryptocurrencies given enough historical data and tuning of the model.

This was a great project to apply some of the machine learning materials presented in this course. The main challenge was understanding the dataset and how to transform it into the input for a machine learning model. For example the data I worked with is considered time-series data so I had to research and figure out how to convert it to a supervised machine learning problem. Other notable challenges are understanding the papers I chose, scouring the internet for the algorithm since the papers I chose lacked implementation details, and reading through the sklearn documentation. But from these challenges and this project, I learned the full process of how to develop a machine learning model with a given dataset.

My Contributions

I appreciate the explanation from the papers I chose, but most of all I appreciate the tutorial and code provided by Jason Brownlee [4]. The papers gave me the idea to download historical datasets for various cryptocurrencies. But I modeled my implementation after the code provided in the tutorial. The original code is predicting daily births with time-series data. So the original implementation of the series_to_supervised() and train_test_split() functions were kept intact.

Changes were made to walk_forward_validation(data, n_test, *model, print_values=True*) function by adding two new parameters, model and print_values. The parameter *model* allows different models to be passed in besides the hard coded Random Forest Regression model in the original implementation. The *print_value*s parameter turns on/off printing of the actual versus predicted values to decrease processing time. The call to series_to_supervised(data, n_in=1, n_out=1, dropnan=True)

function was also changed to pass in the number of features for my dataset, four instead of six from the tutorial. Finally, the original plot was changed from a line plot to scatter plot.

A dictionary data structure was added to the implementation so that multiple coins can be processed without having to change or duplicate code to run each coin. As a side effect anywhere the original code used an array was changed into the dictionary. Not only was the existing implementation modified, but new code was also created. The notable new code are add_predicted_val_to_coin(coin_name, prediction_features, predicted_val) and predict_next_days_prices(coin_name, prediction_features). The former function is to help the latter function by adding each new prediction to the dataset. The latter function is the entry function to predict consecutive future day close prices.

Glossary

*Note: Some of the cryptocurrency definitions used in this paper is meant as a very general or high-level description and not necessary the technical definition.

cryptocurrency. A digital currency with public support, but not necessarily regulated by any government or central bank. It is synonymous with coin and token.

coin. A digital currency. It is synonymous with cryptocurrency and token.

token. A digital currency. It is synonymous with cryptocurrency and coin.

trader. An entity that trades in cryptocurrency.

Bitcoin. The original digital currency that started the cryptocurrency phenomenon.

Ethereum. Another popular coin.

XRP. A coin, presumably, backed by some traditional banks. The project is called Ripple.

Litecoin. A coin modeled after Bitcoin.

Shiba Inu. A meme coin. The project started in August of 2020.

Saitama. A meme coin. The project started in May of 2021.

References

[1]   Inamdar, Abid, et al. "Predicting cryptocurrency value using sentiment analysis." *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. IEEE, 2019.

[2]   Yiying, Wang, and Zang Yeze. "Cryptocurrency price analysis with artificial intelligence." *2019 5th International Conference on Information Management (ICIM)*. IEEE, 2019.

[3]   Chaudhari, Ashwini. *Forecasting Cryptocurrency Prices using Machine Learning*. Diss. Dublin, National College of Ireland, 2020.

[4]   Brownlee, Jason. "Random Forest for Time Series Forecasting." *Machine Learning Mastery*, 31 Oct. 2020, https://machinelearningmastery.com/random-forest-for-time-series-forecasting/.

[5]   Devi, Renuka. "Crypto_corr." Kaggle, Kaggle, 21 Dec. 2021, https://www.kaggle.com/renukadevird/crypto-corr.

[6]   "Free Blue Cryptocurrency Powerpoint Template." Free PowerPoint Templates, 26 Oct. 2020, https://www.free-power-point-templates.com/free-blue-cryptocurrency-powerpoint-template/.

[7]   "Cryptocurrency Prices, Charts and Market Capitalizations." *CoinMarketCap*, https://coinmarketcap.com/.