Cryptocurrency Price Prediction using Machine Learning

Abinaya Seshadre (015314013), Hiteshkumar Bajoria (015264470), Kaamya Ravikumar (015400138), Nirvisha Garara (013851539), and Payal Padmanabhan (015319499), Member, Group 4

**Abstract**— With each passing year, Cryptocurrencies have been gaining immense popularity and are being adopted by many governments all over the world.  Cryptocurrencies are digital currencies, also referred to as coins, that use cryptography to secure transaction records that are created on trading coins over the internet. Through this project, we aim to build a machine learning model that would mitigate risk factors associated with Cryptocurrency price prediction. This project shall address the problem of predicting cryptocurrency rates using machine learning algorithms. Three machine learning algorithms have been used in this project, namely, Support Vector Regression (SVR), Linear Regression and Long short-term memory (LSTM). After model evaluation, it could be seen that, long short-term memory (LSTM) has a better accuracy as compared to the other Machine Learning models.

**Index Terms**— Cryptocurrency, Machine Learning, Support Vector Regression (SVR), Linear Regression, long short-term memory (LSTM), Model Evaluation

DATA 245: MACHINE LEARNING TECHNOLOGIES

—————————— ◆ ——————————

# 1 Introduction

W

ith the increasing popularity, public accessibility and availability, and adoption from around the world, Cryptocurrencies are slowly evolving to be even used as a form of online payment in exchange for goods and services. Cryptocurrencies are digital currencies, that use cryptography to secure transaction records that are created on trading coins over the internet. The transaction records are duplicated, stored, and updated in millions of computers, by people all over the world, along with hashing and other forms of cryptography. This makes the transactions secure and slightly less prone to fraud.

Many risks are associated with cryptocurrencies, such asvolatility, irregularity, and susceptibility to errors and hacking. Profits made due to cryptocurrency are based on trading. One should be aware of the risk factors associated with cryptocurrency while trading. The motivation for this project is to try and mitigate these risk factors related to price changes. Cryptocurrency prices constantly fluctuate, and therefore, there is no guarantee that these prices will cross the minimum requirement level during trading in the stock market. Cryptocurrency randomly jumps levels, thereby increasing the likelihood of missing an essential level of trading. This project shall address the problem of predicting cryptocurrency rates using machine learning algorithms. It will only improve the trade market and help users invest more in cryptocurrencies and profit from them.

# 2 Literature Survey

Cryptocurrency has seen a significant rise in recent years, and this form of currency trading has become more popular. Machine Learning methods are used to predict cryptocurrencies, such as regression algorithms, sentiment analysis, and various other models.

Akyildirim et al. [1] discussed four different machine learning algorithms in the paper. They were Logistic Regression, Support Vector Machines (SVMs), Random Forests (RFs), and Artificial Neural Networks (ANNs). The four models were compared to find one with high accuracy and all four of the algorithms had an accuracy of around 55-65%. SVM was slightly higher in the range in terms of predictive accuracy compared to the other three models.

Patel et al. [2] suggested a hybrid model for the prediction of cryptocurrency prices. This hybrid model constituted a Long-Short Term Memory (LTSM) neural network and Gated Recurrent Unit (GRU). In this paper, the two leading cryptocurrencies were Litecoin and Monero. Four parameters were measured for comparison between LTSM and the hybrid LTSM-GRU model. These four parameters were Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). It was observed the accuracy was higher in the hybrid model compared to the stand-alone LTSM model.

Lahmiri et al. [3] implemented deep learning techniques to predict the price of three highly traded cryptocurrencies: bitcoin, Ripple, and Digital Cash. Long-Short Term Memory (LTSM) neural network was modelled and compared with Generalized Regression (GR) neural network based on RMSE score. It was found that although the LTSM is computationally complex in nature as compared to non-linear pattern recognition, it is highly efficient. Using LTSM, the RMSEs were as follows, 2.75 x 103 for Bitcoin, 19.2923 for Digital Cash, and 0.0499 for Ripple.

# 3 methodology

**3.1 Data Collection**

Data collection for this project is in real time using CryptoCompare website. It is a website for trading data for almost 250,000+ crypto per day. This website has all types of crypto data from which in this project, we are using only bitcoin data to do the analysis. CryptoCompare provides free access through their personal type of account in which we are allowed to make 250,000 API calls in lifetime. For our project we are using the given free API key to access the data through their historic data link.

We are accessing the data through the ‘Request’ package of python. It is a HTTP library specifically for python programming. It is one of the most famous python libraries which is used to fetch the HTTP request through API so that users can get a more flexible way of accessing the data through web servers.

The data that is coming in real time through API is in JSON format. This json format data is then fed into the python pandas data frame which will convert the data into the pandas data frame for further processing. Each row represents one day of bitcoin data. It has the bitcoin’s high, low, open, close, volumeTo and volumeFrom columns. The following figure is showing the snippet of the raw data which is in JSON that is being fetched in real time using requests library and CryptoCompare API.

A picture containing text, plaque

Description automatically generated

Fig. 1: A snapshot of the raw dataset

Table

Description automatically generated

Fig. 2: A table describing the variables in the dataset

**3.2 Data Pre-Processing**

*3.2.1 Data Cleaning*

Before implementing any model or proceeding with further steps of Data Transformation and EDA, it is important to handle the anomalies present in the dataset including missing or duplicate values. The data which we have sourced does not have any missing values and this is represented by the heatmap illustration in Fig 5.

Chart

Description automatically generated

Fig. 5: Heatmap for missing data values

*3.2.2 Outliers*

Prior to exploratory data analysis, we need to remove the outliers. Outliers in simple terms can be defined as the data points which are far away from the observation point. Detecting and removing outliers will help us to analyse the data with less error and more accuracy.

For our dataset, we have identified outliers into two parts i.e., Univariate and Multivariate outliers. Univariate outliers are the points which are extreme values in the distribution whereas multivariate are a combination of values in an observation that is unlikely. In our dataset, we have identified outliers using Tukey’s box plot method and Z-score method.

Tukey’s box plot method uses box plots to identify the individual observations as outliers. It helps us to distinguish between possible and probable outliers. When we talk about possible outliers, it is mainly located between the inner and outer fence, whereas probable outliers are located outside the outer fence.

Inner and outer fence are calculated using Interquartile range (IQR)

(1)

where, : 75th quartile and : 25th quartile

(2)

(3)

Considering our dataset, when we plotted our box plot, we got the below image, where we can clearly see that volumeTo column is the one who have maximum points which are out of range/outliers, and which need to be removed.

Graphical user interface, table

Description automatically generated

Fig. 6: Tukey’s box plot for the dataset

Text

Description automatically generated with medium confidence

Fig. 7: Number of probable and possible outliers using Tukey’s method

As we can see in the above image, after applying the IQR formula we get probable outliers to be 72 for volumeTo and for possible outliers it is 192. That means we have 264 outliers in total for volumeTo. After this step, we have plotted histogram for the possible and probable outliers. Below is the image of both the possible and probable outliers (as seen in Fig 8).

Graphical user interface, chart, waterfall chart

Description automatically generated

Fig. 8: Histogram of probable and possible outliers

Internally standardized residuals are one of the most common statistical methods to detect and remove outliers. For each observation, it is measured how many standard deviations () the data point is away from its mean .

Z- score is given by:

(4)

Normal rule for detecting outliers using Z-score is if , where is the threshold limit and is set at three, the observed points are marked as an outlier.

Graphical user interface, text, application, email

Description automatically generated

Fig. 9: Implementation of Z-score using mean and standard deviation

Graphical user interface, text, application

Description automatically generated

Fig. 10: Number of total outliers

As we can see the number of outliers in volumeTo column is 44 when we use Z score test. Thus, we have used two methods Tukey’s and Z- score to remove the outliers from our datasets.

**3.3 Data Transformation**

We have limited the amount of data to 2000 rows for building our model. The first step of our exploratory data analysis was to understand the data which we had collected. The data collected had 9 features and 2000 rows of data. We have used pandas and NumPy which are basic libraries used to import data, perform manipulations and numerical calculations on it. Matplotlib and seaborn are libraries that we will use to visualize the data. The dataset which we have is a time series data and we would be performing time series analysis on it. We call this time series as we have the closing price of bitcoin, which is also our target variable recorded until the previous day by time. The initial data which had 9 features has been transformed into a dataset with 7 features. We have dropped the time, conversion type and conversion symbol from our dataset. We decided that these features did not have major significance towards the target variable and hence were dropped. Fig 11 shows the data after the columns were dropped.

Table

Description automatically generated

Fig. 11: A snapshot of the dataset after dropping two columns

We have loaded the real time data into a Pandas data frame using python libraries for further processing. Fig 12 explains the data in terms of its statistics. We can see that there are a total of 2000 rows in the entire dataset. The mean closing price of bitcoin is around 13000 which indicates the skewness of the entire dataset as we are now at almost 67000.  The standard deviation or variance can be used to understand the volatility of the asset. The variance of bitcoin is 16000 which is a very high value and in financial terms we can conclude that bitcoin is indeed a risky asset. The minimum closing price is just around 500 compared to today’s value. The 25th, 50th and 75th quartile values are again an indication of the skewness as only the 75th quartile value is coming close to the mean value. This skewness could possibly be attributed to the growth bitcoin and crypto currency have seen over the past few years.

Graphical user interface, application, table

Description automatically generated

Fig. 12: Statistical description of the data

In finance or investments, the return value is an important parameter. Return refers to the percentage change that we have calculated using the closing price of bitcoin. We have created a new column called Return\_Value using the pct\_change function in python to calculate the return value. The first value of this column will be NA as the percentage change is basically the difference between the current and the previous value. Although the price of an asset can never be negative, returns can be either positive or negative. Thus, it is very important to analyse data on the return series as well as price series.

Table

Description automatically generated

Fig. 13: Dataset after the addition of the Return\_Value column

**3.4 Exploratory Data Analysis**

Fig 14 shows a histogram plot to understand the distribution of the features in the dataset. The plots clearly show that the values are all skewed and are not normally distributed.

Chart

Description automatically generated

Fig. 14: Visualization of the distribution of each feature

This is a simple time series plot of the variations in the closing price of bitcoin which has been created using the pyplot function of matplot libraries. We can see the increase and the crash of the bitcoin price in this visualization.

Chart, histogram

Description automatically generated

Fig. 15: Visualization of the closing price of the Bitcoin

Fig 16 shows how plotting the return looks like. Return is an interesting aspect of time series as it can be both positive and negative return. We can see huge drops and increases for the first few years and then we can see a huge drop in recent times.  Again, the bottom line is its volatility indicating it can give huge income as well as huge losses in the longer run.

A picture containing chart

Description automatically generated

Fig. 16: Visualization of the percentage change of Bitcoin Price

Histogram is a plot used to understand the distribution of the variables and Fig 17 and Fig 18 shows the distribution of closing price and return value of bitcoin respectively. The closing price does not follow a normal distribution and is heavily skewed towards the left whereas the return value follows a normal distribution.  For financial return it is always expected for the return to be log normally distributed which is the logarithmic of the normal distribution. Also, the return value is evenly distributed which implies the change in the bitcoin price daily is evenly distributed indicating that it has no constant price.

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 17: Distribution of closing price of Bitcoin

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 18: Distribution of Bitcoin returns

Fig 19 and Fig 20 have been created using the kernel density plot which is a smoothening form of the histogram plot to visualize it better. It is a non-parametric way of estimating data.

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 19: Kernel Density Distribution of The Bitcoin Closing Price

A picture containing chart

Description automatically generated

Fig 20. Kernel Density Distribution of the Return Value

The next visualization is an exponential smoothing which takes historical data from this time series data and adds more weights to the recent data and less weights to earlier data and smooths it. For this purpose, we have used holtwinters from time series analysis in the statsmodel package and import exponential smoothing for the same. Red colour has been used to indicate actual data and green has been used to indicate the fitted data. We can see that exponential smoothing has fit the data very well up to a certain extent where there was a pattern but later the sudden boom has not been fitted very well and this cannot be predicted very easily using any technique.

Chart, line chart

Description automatically generated

Fig 21. Exponential Smoothing of Bitcoin Closing Price

We have used a lag plot from pandas to check for randomness in the data. In a lag plot if the points are in the straight line, then that would indicate a pattern, but we can see that there are deviations in the plots indicating that identifying a pattern in the data is a challenge in some areas.

Chart, scatter chart

Description automatically generated

Fig 22. Visualization of the Randomness of Data

Checking for autocorrelation is an important aspect of time series analysis. This has been implemented using an autocorrelation plot from pandas library. We have checked for the autocorrelation of the closing price. If the plot lies within the boundary, then there exists no pattern at all, but our graph shows that there exists a pattern in most parts of the data and there exists no pattern in certain places.

A screenshot of a computer

Description automatically generated with medium confidence

Fig 23. Graph to identify any patterns in the data

This graph has been plotted to determine the average value of the closing price of bitcoin. The graph indicates the average value of bitcoin is around 15000.

Chart, histogram

Description automatically generated

Fig 24. Average Closing Value of Bitcoin

**Diagram, engineering drawing

Description automatically generated**

Fig 25. Correlation between the features of the dataset

**3.5 Splitting the Dataset**

*3.5.1 Target and Dependent Variable*

For this project our collected dataset has multiple features like date, highest bitcoin price of the day, low price of the day, opening price of the day, closing price of the day, volume of end of the day and volume of start of the day. All these features basically represent the prices of bitcoin in one day and the change of volume in one day. So, to predict the price of bitcoin we are taking the ‘Close’ price column as the target variable and other columns ‘high’, ‘low’, ‘open’, ‘volumeTo’, ‘volumeFrom’ as the dependent variable.

*3.5.2 Training, Validation, and Testing Dataset*

Dataset splitting is an important technique that is being used in machine learning data analysis. For this project we are splitting our dataset into three parts. The explanation for each of them is given below.

* **Training Data:** We are taking 60% of data for training the model in this project. The reason we are taking a huge portion of data for training the model is that as more data you give to the model while training it can get a more accurate pattern and it helps while predicting the values. The model will observe the data and then learn from this data to optimize the parameters of the model. While in case of using neural network training data will help in estimating the weights and the bias.
* **Validation Data:** Validation dataset is also called cross-validation data in some cases as using this data, we are validating the model while also performing the hyperparameter tuning. During the cross validation we are minimizing the error [5] for this project we are taking 20% of data as the validation set.
* **Test Data:** The purpose of using the test data is to get the evaluation of the trained model. When the model is ready training with train data and validated using validation data, then only the test data is being passed so that the accuracy of the trained model can be mapped. Here we are using 20% data as a test set. Generally, the test set is used to get the unbiased prediction accuracy to evaluate the model’s performance in real world data.

Graphical user interface, chart, line chart

Description automatically generated

Fig 25. Dataset split into training and testing data

**3.6 Feature Scaling**

Feature scaling is a very important process on the data. Using feature scaling we can scale all the numerical columns of the dataset. The main reason behind the need for feature scaling is the higher range of the data. Here the prices of bitcoin over the time have fluctuated in a very huge difference so the overall range for the columns like High, Low, Open, and close is very elevated. Similarly, the volume of the total trade of the data has also an inflated value. So that can affect the algorithms that we are planning to implement.

These types of feature values affect the algorithms that are using the Gradient Descent technique. For this project we are planning to implement machine learning algorithms like linear regression and neural networks which uses the Gradient Descent technique for optimization purposes. So, scaling the data using normalization or standardization becomes crucial. The gradient descent formula is shown in Fig 26 to understand the effect of higher range data.

Text

Description automatically generated with medium confidence

Fig. 26. A mathematical representation of gradient descent

In the above equation the is representing the features. We can see that the value of will affect each step size of the gradient descent [4]. So, changes in will give each step size of gradient descent to be different. To make sure that gradient descent reaches the minima smoothly and each step size is updated at the same time for all the features we need to scale the data. Moreover, the gradient descent will converge more rapidly toward the minima if all the features are in the same scale.

Similarly for our project we are also using a distance-based algorithm Support Vector Machine(SVM). There is a chance that higher weighted features get huge weight assigned during the training while the lower range valued features get assigned smaller weight value and that affects the results of the prediction.

For this project’s RNN-LSTM implementation, we are using the normalization method of data scaling. In this we are using the Min-Max Scaler to scale the dataset. In this technique the data is scaled between the range of 1 and 0 so that all the values are shifted to the smaller numerical value. The equation for MinMax scaler is shown below.

(5)

In Equation 1, is the features. When the value of is the minimum value of all, the numerator is going to be 0 so is going to be 0. On the other hand, when is the maximum value of all, the numerator and denominator is going to be same making value of as 1. Similarly for all data points the range of is going to be between 1 and 0.

For the Linear regression implementation for this project, we are using the standard scaling technique for our dataset. In this technique the values are centred around the mean that is derived through standard deviation [4]. Basically, mean becomes zero and other data is distributed through standard deviation around it. The equation for standard deviation is as seen in Equation 6.

(6)

In Equation 6, is the features, is the mean of features and is the standard deviation. The snapshot of scaled data is shown below.

A screenshot of a computer

Description automatically generated with low confidence

Fig. 27. A snapshot of the scaled data

# 4 Modeling

**4.1 Implementing model using SVR**

*4.1.1 Development of the Algorithm*

Support Vector Machine is one of the highly used ML algorithms that can applied on almost all datasets, whether classification or regression is required. There are variations of SVM algorithm which makes it applicable to both linear and non-linear data. There are variations to specifying whether a soft or hard margin is required too, which can be easily specified using Python code. We have taken the linear and RBF kernels and applied SVM to our data.

Linear SVM works by

**4.2 Implementing model using Linear Regression**

*4.2.1 Development of the Algorithm*

Linear Regression is one of the easiest ML algorithms that can be applied for both classification and regression problems. It assumes linear relationship between the dependent and independent variables and tries to fit a linear equation such as that of a line in the case of one input value or a plane when there is more than one input feature. In our case, multiple linear regression is applied as there are more than one explanatory variable. Linear regression gives continuous values and is of the form,



Fig. 28. A mathematical representation of linear regression

where Yi is the ith observation of the dependent variable, β0 is the intercept and βj is the slope of the variable Xj. β0 is given as the average value of Y if all the X’s are zero and βj is given as the average increase in Y when Xj increases by 1 and all other X’s are held constant. Ɛ is the error term and gives the difference between the observed and predicted values for each data point. The parameter values are estimated such that the error terms are minimized.

**4.3 Implementing model using LSTM method of RNN**

*4.3.1 Development of the Algorithm*

Recurrent Neural Network(RNN) is one of the very useful Deep Learning algorithms in real life applications. LSTM is one type of RNN algorithm. This algorithm is one of the very useful algorithms because of its ability to remember the output from the previous nodes. LSTM is mostly used when the data is either sequential or it has the sentiment involved.

LSTM has three gates named input gate, forget gate and the output gate. In the methodology of the LSTM the long-term memory is initialized from current input data while short-term memory initialized from the previous timestamp. The input layer discards the useless information and gives the output in the sigma function. The sigma function works with two values 0 and 1. 0 indicates that the value is unimportant while 1 indicates the important value. The input layer’s output is saved on long-term memory but the forget gate is very important in the LSTM network as it multiplies the forget vector values by the current input values. The output of the forget gate is then given to the next cell from the long-term memory.

Text, letter

Description automatically generated

Fig. 28. Mathematical representation of LSTM algorithm

In above equations, it is input layer’s output, is weights, is the bias, is the activation function that is used to send the valued information in next cell, is forget gate’s output, represents output gate, represents the cellular cell, represents the input information, while represents the output information [6].

**Diagram

Description automatically generated**

Fig. 29. Inner architecture of LSTM

*4.3.2 Model Implementation*

The implementation of RNN’s LSTM method for the project purpose was done using keras. Keras is a python’s deep learning library that provides the inbuilt API of using different Deep Learning algorithms directly. The different parameters that have been used for the development of LSTM for cryptocurrency price prediction is shown in Table 1.

|  |  |
| --- | --- |
| **Parameters** | **Values** |
| window\_len | 5 |
| test\_size | 0.2 |
| zero\_base | TRUE |
| lstm\_neurons | 100 |
| epochs | 20 |
| batch\_size | 32 |
| loss | Mean Squared Error(MSE) |
| dropout | 0.2 |
| optimizer | Adam |

Table 1. Parameters and Values used in the model implementation

The epochs are given 20 for the model while the loss function is Mean Squared Error(MSE). Window length is given 5 for each epoch. ‘Adam’ optimizer has been used for the RNN-LSTM model to predict cryptocurrency.

**4.3 Evaluation of the ML Models**

After building, training, and testing several ML models, evaluation metrics can be used to find which model works well for the chosen data and the best model can be picked. We came up with calculating several metrics such as the Mean Absolute Error useful in comparing and evaluating the models. The results of the metrics have been summarized in the table below.

# 5 technical Difficulties

* Lot of datasets that are available on cryptocurrency were quite outdated and no latest detail is available. After great difficulty, we were able to only find an updated and recent dataset for Bitcoin alone. Hence, we are still working on finding updated datasets for various other cryptocurrencies.
* Values found in datasets with regards to cryptocurrency, has a very large range and therefore issues in scaling the data arose. We have made use of MinMax scalar.
* One of the difficulties was to collect the related features. The cryptocurrency prices are not related to a few factors in the industry but there are numerical features that affect the prices of bitcoin and to identify and collect all that feature value was practically not possible. Bitcoin does not work like stock prices which have market sentiments involved.

# 6 Lessons Learnt

* We had the opportunity to understand the importance of understanding our dataset and the role it must play in the development of the final model. Data pre-processing and visualizations helped us to choose the features that had significant contribution toward predicting our target variable.

# 7 Member Roles

|  |  |
| --- | --- |
| **Student Name** | **Contributions** |
| Abinaya Seshadre |  |
| Hitesh Bajoria |  |
| Kaamya Ravikumar |  |
| Nirvisha Garara |  |
| Payal Padmanabhan |  |

# 8 conclusions

**Appendix 1 – List of Figures**

[fig 1.Flowchart describing the workflow of the technical part](#Figure1)

[Fig 2. Source to Target Mapping](#Figure2)

[Fig 3. Initial ER Diagram](#Figure3)

[Fig 4. Reverse engineered EER Diagram](#Figure4)

[Fig 5. Drop-down tab for recipes based on special occasion](#Figure5)

[Fig 6. Drop-down tab for recipes based on mealtimes](#Figure6)

[Fig 7. Recipes displayed in the descending order of their ratings](#Figure7)

[Fig 8. Ingredients and Directions tab pop up on selecting a recipe](#Figure8)

[Fig 9. Recipe Histogram based on Ratings and Calories](#Figure9)

**Acknowledgement**

We would like to thank our professor, Mr. Vishnu Pendyala for giving us this opportunity to learn more about database management systems and its uses in the real world in a practical setting. We would also like to thank our teaching assistant, Mr. Varun Vala for providing constant assistance and clarifying our queries.

**References**

[1] S. Kalajdziski, G. Radevski, I. Ivanoska, K. Trivodaliev and B. R. Stojkoska, "*Cuisine classification using recipe's ingredients,*" 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2018, pp. 1074-1079, doi: 10.23919/MIPRO.2018.8400196.

[2] c. Ooi, c. Iiba and c. Takano, "*Ingredient substitute recommendation for allergy-safe cooking based on food context,*" 2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Canada, 2015, pp. 444-449, doi: 10.1109/PACRIM.2015.7334878.

[3] Hartmann, C., Dohle, S., & Siegrist, M. (2013). “*Importance of cooking skills for balanced food choices.*” Appetite, 65, 125-131.