G-Research Crypto Forecasting

Sidra Hussain
Information Technology
Frankfurt University of Applied Sciences
Frankfurt, Germany
sidra.hussain2633@gmail.com

Doina Logofatu
Frankfurt University of Applied Sciences
Germany
doina.logofatu@gmail.com

***Abstract*—Cryptocurrency is the current evolving financial market that gives scope for many researchers and machine learning assets to be put in production. Cryptocurrency forecasting is a challenge as following financial assets is difficult due to their volatility and unscalable factors dependencies. This paper puts forward the data of fourteen different types of cryptocurrencies which will be used to build machine learning models to forecast the crypto scores in the next fiscal year. For this, four different models are used, and their performance is evaluated and tested. The models being Linear Regression, Decision Tree, Random Forest, and Gradient Boosting. The results obtained from the models show the perfect good fit and good hyperparameter tuning, giving evidence of good feature engineering and data scraping. Overall, the models are to benefit financial market immensely to forecast and help investments to build in sales and purchasing.**

***Index Terms*—Blockchain, hyperparameter tuning, cryptocurrency, machine learning, loss curves**

I. Introduction

With the innovation and rise of the Distributed Ledger Technology (DLT) in the coming years, blockchain has been reformed and replaced stocks and shares. Cryptocurrency is the current investing asset that uses DLT, and its volatility has been increasing day by day due to the ease of purchase and sell points in the leaderboard across the globe [1]. One of the main quests of ML engineers now is to predict the trend of cryptocurrencies in the future run cycles. As accordance to shares and stocks that depend on intangible dependencies such as the utility and government inclusion, cryptocurrency is quite independent of these factors and so exposed to be forecasted using ML and Data engineering [1].In this paper, machine learning models will be used to forecast the price values for the future fiscal year.

This paper follows the main structure of building the theoretical knowledge at first, with explaining the problem statement and then describing the machine algorithms. The data set is introduced in the next section and data engineering is implemented to maintain the model performance in process. Within the Data section, exploratory analysis is used to find association within the final input data. The last section is dedicated to gauge performance of the models using different evaluations and lastly results and discussions are put forward. Section VII concludes the paper.

II. Material and Methodologies

To achieve the aim of this paper, four models were trained on the historical data of 14 cryptocurrencies to form four cryptocurrency price predictions. The data dates to 2018 and consists of an extensive dataset. To evaluate the performance of the schemes included in this paper, the illustrated project pipeline was followed in Fig. 1. It shows the processing of the dataset from inheritance to evaluation:

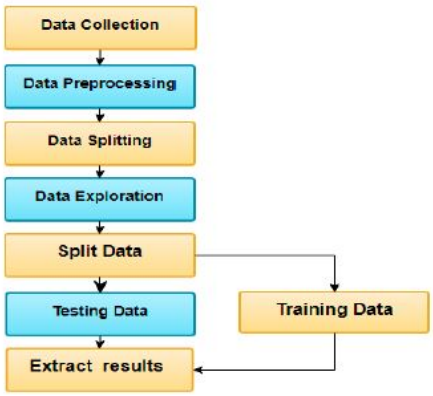

Fig. 1. Methodology of processing data and Model selection [2]

As mentioned in section I, following is a discreet explanation of the machine learning algorithms used.

A. Machine Learning Algorithms

1) Linear Regression:

a) Definition: Linear regression was one of the first algorithms to be studied in detailed for machine learning and forecasting. It models the input to output using linear predictor functions of which primarily conditional mean is used as a response or entity [3]. As the linear one to one to the dependent and independent variables, conditional probability distribution is the main essence of the response calculated rather than joint probability distribution which is a multi-variable analysis [4].

b) Mathematical Implementation: For any dependent variable, a set of independent variables are assigned which are associated with β a predictor function or parameter vector and a noise or error, disturbance term ϵ [3].

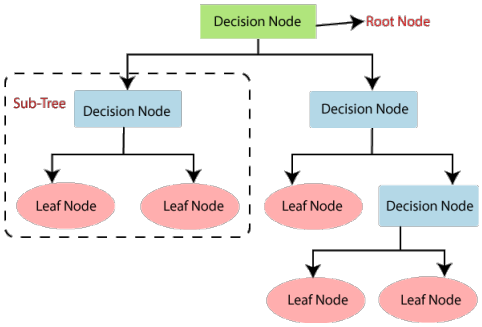

Fig. 2. Decision Tree Representation [5].

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i = \ x_i^T \beta + \varepsilon_i \quad (1)$$

Where, $i = 1, \ldots, \ n$. The term $x_i^T \beta$ is the inner product between the vectors x_i and β. n being the number of equations needed for the solution to y and p being the p-vector regressor of x.

And,

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} x = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} 1 & x_1 1 & \cdots & x_1 p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n 1 & \cdots & x_n p \end{pmatrix} \quad (2)$$

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix} \quad (3)$$

c) Usage: Apart from machine learning, linear regression is widely used in finance and trend lining. It is the activity of subjecting features across parameters and depicting the trends within this time serial data. Business analytics with proof and grounding of trend line is always desired [4]. In finance with the concept of beta coefficient, a particular asset can be analyzed to decrease or increase with overall stock movement.

2) Decision Tree:

a) Definition: A decision tree is a supervised machine learning classification model with a tree structure. It is used to visually and explicitly portray decisions and decision-making so that non-expert users can understand easily and can be effectively derived from a given data set. In the decision tree, the data is continuously split according to a specific attribute. The two main decision tree entities are decision nodes, where the data is split into further decision nodes, or leaves, where we get the outcome [5]. Fig. 2 depicts a decision tree graph.

b) Mathematical Implementation: The Decision Tree algorithm works by selecting the best attribute using Attribute Selection Measures(ASM) to split the records. ASM is a heuristic for choosing the best possible splitting criterion for data partitioning. Since it helps determine breakpoints for tuples on a given node, it is also known as the splitting rules.

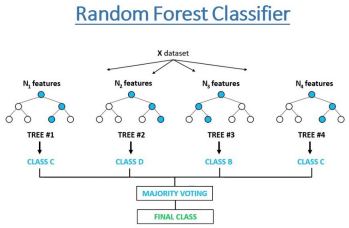

Fig. 3. Random Forest Classification [7].

By describing the given data set, ASM assigns a rank to each function (or attribute). As a splitting attribute, the best score attribute will be chosen. Information Gain, Gain Ratio, and Gini Index are the most commonly used selection criteria [5].

Entropy gives the degree of randomness or impurity contained within a given data set.Information gain calculates the difference between the entropy in advance of the split and the average entropy after the split following attribute values [6].

$$Entropy = -\sum_{i=1}^{m} p_i log_2 p_i \quad (4)$$

$$Information\ Gain = 1 - Entropy \quad (5)$$

$$Gini\,(D) = 1 - \sum_{i=1}^{m} {p_i}^2 \quad (6)$$

Where, p_i is the probability that an arbitrary tuple in data D belongs to class Ci. Variable splits should have a low Gini Index [7].

c) Usage: Data analysts often employ Decision Trees in their predictive analysis, such as to develop operations strategies in businesses. They are also a popular machine learning and artificial intelligence tool used as supervisory learning training algorithms (e.g., categorization of data based on various tests, such as 'yes' and 'no' classifications). Overall, decision-making trees are used to solve a variety of problems in a variety of industries. They are used in technology and health sectors to financial planning because of their flexibility [6].

3) Random Forest:

a) Definition: The Random Forest (RF) classifier is a supervised machine learning classification method that uses decision trees as its foundation. A Random Forest is a set of unbiased, unrelated, and de-correlated decision trees. It is thus called a 'random forest' [7]. It's a type of meta-estimator that uses decision-tree classifiers on random sub-spaces and data set samples and applies to mean averaging tools to improve prediction accuracy over time while avoiding over-fitting [7]. A simple illustration of random forest is presented in Fig. 3.

b) Mathematical Implementation: Random Forest works by first selecting random samples from a given data set using which it constructs a decision tree for each sample. Prediction results from each decision tree is collected, and voting is

performed on them. Prediction result with the highest vote is selected as the final result. The idea of variable importance is an implicit feature selection done by RF with a random subspace methodology, and the Gini impurity criterion index assesses it. The Gini index is a measure of variable predictive power in regression or classification based on the impurity reduction theory. It is non-parametric, which means it does not depend on data from a specific type of distribution [8]. The Gini index of a node n in a binary split is determined as follows:

$$Gini\,(n) = 1 - \sum_{j=1}^{2} (p_j)^2 \tag{7}$$

Where, p_j is the relative frequency of class j in the node n. The Gini index should be improved to the greatest extent possible when splitting a binary node. In another explanation, a low Gini (i.e., a more significant decrease in Gini) indicates that a specific predictor function is more critical in dividing the data into two groups. In a classification problem, the Gini index can be used to rank the importance of features [9].

c) Usage: The Random Forest algorithm can be used for both classification and regression tasks. It can deal with missing values and keep a large proportion of data accurate. Having the ability to work upon an extensive data set with higher dimensionality, RF is used in the banking sector, medicines, stock market, and E-Commerce. For example, to determine whether the customer is loyal or fraudulent, RF analysis can be useful. With RF, it has grown easier to detect and predict the drug sensitivity of a medicine [9].

4) Gradient Boosting:

a) Definition: Gradient boosting is a model used as an alternative approach for weak prediction models as an ensemble, which is to use multiple algorithms to learn and predict better in algorithms. The main objective is to obtain a performance that is better than the constituent weak learning algorithms [10]. It does so by using boosting which is the method that operates by reducing bias and variance. That is to see if weaker algorithms could make a stronger algorithm. A weak algorithm would classify and label good on training set but won't correlate well in testing set.

b) Mathematical Implementation: Gradient boosting works by minimizing the loss function that relates the input to the output. It uses a greedy approach to set the loss function into optimizing in a maximum descent which is represented by γ which is a small amount for the validity of linear approximation [10]. The iterations in the algorithm are over finding the local minimum in each iteration for the value of $Fm(x)$. If the base equation becomes:

$$F\,(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + constant \tag{8}$$

Where the h is the weak algorithmic models that form an ensemble. And constant F_0 is the association of the loss function $L(y_i, \gamma)$ with the training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ (from where the approach starts) that expands the constant in a greedy manner in every iteration of the model.

$$F_0 = \ argmin_\gamma \sum_{i-1}^{n} L(y_i, \gamma) \tag{9}$$

$$F_m\,(x) = F_{m-1}\,(x) + \arg min_{h_m \epsilon H} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}\,(x_i) + h_m(x_i)) \right] \tag{10}$$

Where h is the base learner function in class H or the weak algorithms iterating m times.

c) Usage: The best usage of Gradient boosting algorithm which is generally a gradient descent algorithm was seen in discovering the Higgs Boson in the Deep Neural Networks of the Large Hadron Collider [10]. Apart from practical physics it finds its usage entirely in recommendation systems in web services like Youtube etc. that rank the entities on priorities or any other imperative feature [10].

B. Data Exploration

The data is the information of 14 cryptocurrencies dating back from 2018. It is a public domain data from global stock exchange that equips over million high frequency rows. The features for each cryptocurrency given are Asset ID, which is unique to every crypto, Time stamp, that includes the minute and date the row was caught. The discriminatory features include Count, Open, High, Low, Close, VWAP, Volume and Target. Where, count is the number of trades with Open and close as USD price of opening and closing of the trade. Volume is the volume of traded crypto and VWAP is the per average volume price over minute. Target is the return on investment in time frame of 15 minutes.

III. Data set

The data set has a training set with 24 million records of 14 crypto assets. That is 24236806 rows over 10 columns. This training set will be split into training and testing set.

A. Data Preliminary Analysis

After the inheritance of data, to make the data set able to fit the algorithm it is processed beforehand. This could also be called data mining and data cleaning. It is an important and integral part of the project pipeline. This section investigates the preprocessing of data for the topic sentence using two sequential methods.

1) Data set Framing: In our case the crypto information ranges till 24 million records. This type of data would need an extensive processing time and CPU power. To reduce the processing time, the record of information is reduced. It must be noted that the data set can not be reduced that important trends are lost. This means that data set framing can only be done when data set is known, and exploration is done right. As discussed in section 4, the crypto assets are non-stationary and used for forecasting the current value. This gives us the liberty and flexibility to reduce the data as it is a non-stationary randomized time series data. Which means if i denotes the

	timestamp	Asset_ID	Count	Open	High	Low	Close	Volume	VWAP	Target
17714600	2020-11-01 00:01:00	3	61.0	0.092705	0.092764	0.092566	0.092626	1.658528e+05	0.092651	0.001047
17714601	2020-11-01 00:01:00	2	163.0	261.896000	262.160000	261.590000	261.904000	1.985741e+02	261.854820	-0.002429
17714602	2020-11-01 00:01:00	0	150.0	28.347800	28.362300	28.303400	28.342000	5.011629e+03	28.328415	-0.005357
17714603	2020-11-01 00:01:00	1	2445.0	13734.228920	13754.300000	13720.000000	13741.110730	2.135172e+02	13737.261551	0.000794
17714604	2020-11-01 00:01:00	4	10.0	0.002560	0.002562	0.002558	0.002561	1.174202e+05	0.002561	0.004920
...	...	...	...	...	...	...	...	...	...	...
18313364	2020-11-30 23:59:00	9	281.0	87.689714	87.880000	87.590000	87.671286	3.960266e+03	87.733833	-0.000098
18313365	2020-11-30 23:59:00	10	49.0	568.198400	569.230000	567.805800	568.227433	2.044260e+01	568.515584	-0.006150
18313366	2020-11-30 23:59:00	13	149.0	0.032355	0.032400	0.032340	0.032361	3.776827e+06	0.032367	0.000044
18313367	2020-11-30 23:59:00	12	365.0	0.202874	0.203569	0.202651	0.202946	9.302105e+05	0.203051	-0.006227
18313368	2020-11-30 23:59:00	11	340.0	130.010000	130.780000	130.000000	130.480000	9.174512e+02	130.290738	-0.000804

598769 rows × 10 columns

Fig. 4. Framed Data set.

```
timestamp        0
Asset_ID         0
Count            0
Open             0
High             0
Low              0
Close            0
Volume           0
VWAP             0
Target        8814
dtype: int64
```

Fig. 5. 8814 missing values in data set.

	Asset_ID	Count	Open	High	Low	Close	Volume	VWAP	Target
count	589955.000000	589955.000000	589955.000000	589955.000000	589955.000000	589955.000000	5.899550e+05	589955.000000	589955.000000
mean	6.473456	285.892887	1320.409986	1322.450504	1318.366282	1320.418393	3.544014e+05	1320.401154	0.000027
std	4.068315	628.253640	4301.293082	4307.891268	4294.558528	4301.324037	1.416194e+06	4301.260318	0.004511
min	0.000000	1.000000	0.002451	0.002453	0.002450	0.002452	6.000000e-05	0.002452	-0.126551
25%	3.000000	28.000000	0.128017	0.128417	0.127590	0.128001	1.872239e+02	0.127994	-0.001631
50%	6.000000	80.000000	27.651600	27.671300	27.629000	27.651700	2.062042e+03	27.651692	-0.000049
75%	10.000000	262.000000	262.734500	263.210000	262.340000	262.732917	6.408410e+04	262.729967	0.001547
max	13.000000	38146.000000	19842.913750	19873.230000	19762.400000	19840.008750	1.539141e+08	19826.506346	0.089697

Fig. 6. Statistical information of the data set.

	Asset_ID	timestamp
17714600	3	2020-11-01 00:01:00
17714601	2	2020-11-01 00:01:00
17714602	0	2020-11-01 00:01:00
17714603	1	2020-11-01 00:01:00
17714604	4	2020-11-01 00:01:00
...	...	...
18313364	9	2020-11-30 23:59:00
18313365	10	2020-11-30 23:59:00
18313366	13	2020-11-30 23:59:00
18313367	12	2020-11-30 23:59:00
18313368	11	2020-11-30 23:59:00

589955 rows × 2 columns

Fig. 7. Categorical Variables.

initial instant value of a particular asset, the value at instant i-1 which is the future next instant, would be a disjointed event. But it is to be noted, this disjoint parametric does not influence the overall forecasting which is done on a comparative large and sufficient data set.

For that matter the following 24 million records are reduced to 598k records by reducing the data set till November 2020. Below is the image Fig. 4 of the data frame of the reduced training data set with the crypto assets features for modeling.

2) Enriching Data set: Data preliminary analysis is the pre-processing of data that ensures the data is clean and will not stump the model to be made. One of the most important steps is to ensure that the data is populated completely i.e., it does not have NAN, NA, or null values. These values make a sparse vector and run the danger of underwhelmed training and a biased model. It can also lead to a less precise or inaccurate modeling. Also, many machine learning algorithms do not support missing values. The best way to avoid this is by filling/imputing these missing values or deleting the specific rows with missing values if the records are sufficient [11].

Deletion can be done via row or column of the data set. Whereas, filling can be done via many aspects, for example filling the missing locations by an arbitrary value, mean, mode or median, forward, and backward fill. For categorical values, filling can be done with the most frequent variable, or the missing values can have their own category [11].

To check for missing values, any values that are not in finite category are checked and following results in Fig. 5 were received on 598k records.

This illustrates around 1.47 percentage or 8814 of target values are missing. To tackle this, these rows are dropped as 598769 records are more than enough for training of our algorithms. Following is the resultant data set with now 589955 records and a general statistical description of the features content for each crypto asset.

B. Feature Engineering

In this step of implementation of the forecast, features that influence the forecast are dealt with. In feature engineering normally features are extracted and a transformed data frame is made from the features. In scope of this project feature extraction is omitted and instead features are transformed for increasing the accuracy of the algorithm. This is important as raw features could lead to problems with the algorithms learns from itself in the model. This step aids the model to refine itself to improve performance. These features become the important factors that affect the business problem [12].

C. Categorical Variables

In the data set in question there are two categorical variables that need to be aligned with the numeric trade features. The feature names are Asset ID which is unique for all fourteen crypto asset and timestamp, which is the minute at which trade started.

1) One Hot Encoding: The above illustrated categorical variables are turned to numeric using one hot encoding. It is a method to better express categorical variables. First the feature timestamp is divided into weekday and hour associated with each variable.

For all variable instances an additional column is added in the data set. That is, fourteen Asset IDs with range (Asset ID0: AssetID13), twenty-four hours with range (hour0: hour23) and seven weekdays with range (weekday0: weekday6). For all asset value for a particular row, only the specific column would be 1 of which the asset belongs to, and others would be 0. This saturates the data set with a precise value over the whole range of categorical variables where it is on only at the given AssetID, hour and weekday. Following Fig. 9 is the snippet of the one hot encoded data set for categorical variables.

	Asset_ID	hour	weekday
17714600	3	0	6
17714601	2	0	6
17714602	0	0	6
17714603	1	0	6
17714604	4	0	6
...	...	...	...
18313364	9	23	0
18313365	10	23	0
18313366	13	23	0
18313367	12	23	0
18313368	11	23	0

589955 rows × 3 columns

Fig. 8. One Hot Encoded Categorical Variables.

	Asset_ID_0	Asset_ID_1	Asset_ID_2	Asset_ID_3	Asset_ID_4	Asset_ID_5	Asset_ID_6	Asset_ID_7	Asset_ID_8	Asset_ID_9	...	hour_21	hour_22
17714600	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
17714601	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
17714602	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
17714603	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
17714604	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18313364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0
18313365	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
18313366	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
18313367	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
18313368	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

589955 rows × 45 columns

Fig. 9. Encoded Data set.

D. Numeric Variables

For numeric variables there are extensive trends in the data. In order to scale the data for each specific feature and in accordance to the specific entity MinMax scaler approach is used.

1) Min Max Scaler: The min max scaler takes the variables and scales the entire feature information according to the minimum and maximum value of the feature. This is the most efficient alternative to zero mean and unit variance scaling. Without disturbing the shape of the original distribution, the data is transformed and scaled. Following is the unscaled (Fig. 10) and scaled data set (Fig. 11).

E. Data Exploratory Analysis

Data set exploration is important and useful to see the distribution and behavior of data. This section explores the association of all features together. There are two methods by which this could be achieved.

	Count	Open	High	Low	Close	Volume	VWAP	Target
17714600	61.0	0.092705	0.092764	0.092566	0.092626	1.658528e+05	0.092651	0.001047
17714601	163.0	261.896000	262.160000	261.590000	261.904000	1.985741e+02	261.854820	-0.002429
17714602	150.0	28.347800	28.362300	28.303400	28.342000	5.011629e+03	28.328415	-0.005357
17714603	2445.0	13734.228920	13754.300000	13720.000000	13741.110730	2.135172e+02	13737.261551	0.000794
17714604	10.0	0.002560	0.002562	0.002558	0.002561	1.174202e+05	0.002561	0.004920
...	...	...	...	...	...	...	...	...
18313364	281.0	87.689714	87.880000	87.590000	87.671286	3.960266e+03	87.733833	-0.000098
18313365	49.0	568.198400	569.230000	567.805800	568.227433	2.044260e+01	568.515584	-0.006150
18313366	149.0	0.032355	0.032400	0.032340	0.032361	3.776827e+06	0.032367	0.000044
18313367	365.0	0.202874	0.203569	0.202651	0.202946	9.302105e+05	0.203051	-0.006227
18313368	340.0	130.010000	130.780000	130.000000	130.480000	9.174512e+02	130.290738	-0.000804

589955 rows × 8 columns

Fig. 10. Unscaled Data set.

	Count	Open	High	Low	Close	Volume	VWAP	Target
17714600	0.001573	4.548390e-06	4.544355e-06	4.559973e-06	4.545024e-06	1.077568e-03	4.549415e-06	0.001047
17714601	0.004247	1.319834e-02	1.319149e-02	1.323663e-02	1.320068e-02	1.290162e-06	1.320719e-02	-0.002429
17714602	0.003906	1.428487e-03	1.427038e-03	1.432061e-03	1.428404e-03	3.256122e-05	1.428692e-03	-0.005357
17714603	0.064071	6.921478e-01	6.921018e-01	6.942476e-01	6.925960e-01	1.387249e-06	6.928735e-01	0.000794
17714604	0.000236	5.478027e-09	5.499862e-09	5.464924e-09	5.478829e-09	7.628944e-04	5.475302e-09	0.004920
...	...	...	...	...	...	...	...	...
18313364	0.007340	4.419072e-03	4.421906e-03	4.432031e-03	4.418791e-03	2.573038e-05	4.424955e-03	-0.000098
18313365	0.001258	2.863471e-02	2.864293e-02	2.873150e-02	2.864036e-02	1.328179e-07	2.867440e-02	-0.006150
18313366	0.003880	1.506997e-06	1.506902e-06	1.512468e-06	1.507470e-06	2.453854e-02	1.508825e-06	0.000044
18313367	0.009543	1.010048e-05	1.011995e-05	1.013040e-05	1.010554e-05	6.043701e-03	1.011770e-05	-0.006227
18313368	0.008887	6.551838e-03	6.580589e-03	6.578025e-03	6.576487e-03	5.960801e-06	6.571420e-03	-0.000804

589955 rows × 8 columns

Fig. 11. Scaled Data set.

	timestamp	
	amin	amax
Asset_ID		
0	2020-11-01 00:01:00	2020-11-30 23:59:00
1	2020-11-01 00:01:00	2020-11-30 23:59:00
2	2020-11-01 00:01:00	2020-11-30 23:59:00
3	2020-11-01 00:01:00	2020-11-30 23:59:00
4	2020-11-01 00:01:00	2020-11-30 23:59:00
5	2020-11-01 00:01:00	2020-11-30 23:59:00
6	2020-11-01 00:01:00	2020-11-30 23:59:00
7	2020-11-01 00:01:00	2020-11-30 23:59:00
8	2020-11-01 00:01:00	2020-11-30 23:59:00
9	2020-11-01 00:01:00	2020-11-30 23:59:00
10	2020-11-01 00:01:00	2020-11-30 23:59:00
11	2020-11-01 00:01:00	2020-11-30 23:59:00
12	2020-11-01 00:01:00	2020-11-30 23:59:00
13	2020-11-01 00:01:00	2020-11-30 23:59:00

Fig. 12. Time Range of Assets

1) Time Series Plot: A time series plot would exhibit the changes of the features for each crypto asset. This shows if the data is co-dependent or not. Our case is a non-stationary time series business problem. There is no dependability and influence of one crypto asset to another. A crypto currency can only be influenced by external factors other than the currencies.

For the following approach first the time range for each asset is caught.

Fig. 12 shows that all asset ID target variable would be associated with the same time range. Fig. 13 is the time series plot against the Target values and time stamp of the asset IDs.

The following time plot does not provide any information on the association of variables to the Target. It shows all the assets have a unique pattern with respect to the return of investment.

2) Correlation Matrix: A correlation matrix shows association between variables in a matrix form. It shows correlation between all set of variable pair possible. This helps to see patterns and trends in complex and large data sets easily. Following is the correlation matrix for all the crypto assets features.

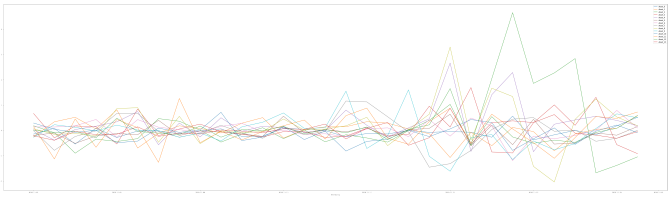

Fig. 13. Time Series Plot

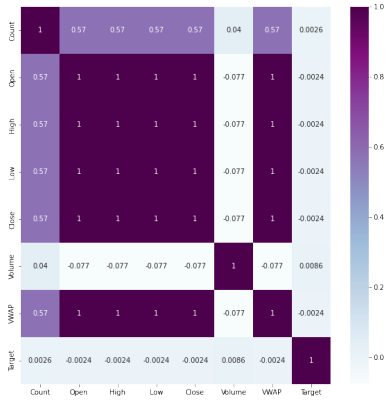

Fig. 14. Correlation Matrix

The following depicts that the features are highly correlated to each other which was information that was not showcased in the time series plot. The features of count, volume and target are disjointed independent variable. But the features of Open, Close, Low and High show a trend of correlation that also logically are understood to be linked between the crypto assets.

Overall, two conclusions are drawn from the correlation matrix. First, the features are deemed to be fit for model training due to high correlation. No further features need to be engineered in. And second, the positive correlation depicts that the crypto assets will decrease and increase together, their behavior in the market would be proportional.

IV. Evaluation

The evaluation of the model is the entity that shows if the model performed well. In this case, a forecast result is subjected to a future value. Thus, the evaluations are mathematical formulas implemented and tested for errors. Following are the few ways used to evaluate the models in use [13].

A. Train Error

Train error is a metric to see how well the model is trained on training data. The lower the training error, the better the model is trained. It uses Root Mean Squared Error (RMSE) on predicted and target values of train data set [13].

B. Validation Error

Validation error is a metric to see how well the model performs on new or test data. The lower the validation error the higher the accuracy considered of the model. It uses RMSE on predicted and target values of test data set [13].

C. GridSearchCV

To determine the best values for modeling on all algorithms, hyperparameter tuning is used. The hyperparameter tuning used in the current case is called Grid Search. It involves taking a parameter dictionary that holds all the parameters which would be tested in the grid search for optimal performance, an estimator that takes the algorithm model that would be hyper-tuned, a cross validation value that is the number of times the hyperparameters will be cross validated, a scoring method which could be the accuracy score and the number of jobs wished to be used in parallel [15].

```
DT_GS = DecisionTreeRegressor()

gs_DT = GridSearchCV(DT_GS,
                     param_grid = {'max_depth': range(1, 11),
                                   'min_samples_split': range(10, 60, 10)},
                     cv=5,
                     n_jobs=1,
                     scoring='neg_root_mean_squared_error')

gs_DT.fit(train_set[input_cols], train_set[target_col])

print(gs_DT.best_params_)
print(-gs_DT.best_score_)
```

```
{'max_depth': 1, 'min_samples_split': 10}
0.00418323335733631
```

Fig. 15. Hyperparameter Tuning by GridSearchCV

The one downside of using grid search is the processing time. In the current project hyperparameter tuning was used three times which took one hour of collective processing time. The main advantage of hyperparameter tuning is it aids in reforming the regressors to lead to an optimal prediction. Following is the example of hyperparameter tuning of Decision Tree estimator [15].

As illustrated in Fig. 15, the Grid search takes hyperparameters of max depth and min sample splits for a decision tree estimator and cross validates these parameters 5 times for the scoring of negative RMSE. And only one parallel job is allowed for this case. The low output error shows the listed output parameters are optimal.

D. Loss Curves

To better understand the implication of validation and training error, both errors are visualized together via a graph.

1) Over-fitting Curves: A curve that shows validation error more than training error over general period of train time or epoch. An over-fit is a situation when validation error is greater than training error. This indicates the model performs better on training data and bad on new data. To avoid this, models can be halted to not be trained for longer period or made less complex with lesser features so that it can generalize on new data [14]. The following Fig 16 depicts this behavior.

2) Under-fitting Curves: A curve that shows training error more than validation error over general period or epoch. An under fit is a situation when training error is greater than validation error. This indicates that the model did not accurately model the training data. To avoid this, models need to be trained further i.e., features might be added, or data set can be increased or cleaned better [14].The following Fig. 17 depicts this behavior.

V. Results

This section explores the results of the models. To intercept the results we gauge the overall performance of the models as the future value of the cryptocurrencies and their behavior in the market will unfold in the next stock market turn. Following

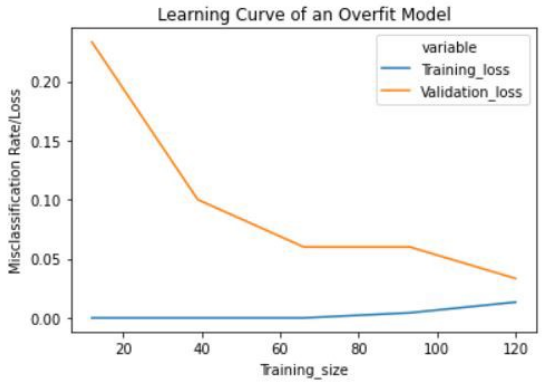

Fig. 16. Over-fitting model loss curve [14]

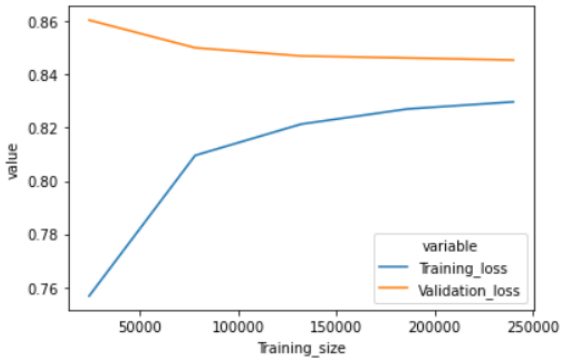

Fig. 17. Under-fitting model loss curve [14]

is the Train and Validation Errors of the models with (in Table. I) and without hyperparamter tuning (in Table. II).

Following are the graphical representations of the predictions scores over test data of June 13th, 2021, on minute 00, 01, 02 and 03 provided as test sample.

1) Linear Regression Evaluation: Fig. 18 shows prediction scores of all cryptocurrencies by the model Linear Regression. There is no particular trend and all assets have a unique

TABLE I
TRAINING AND VALIDATION ERRORS OF MODELS WITHOUT HYPERPARAMETER TUNING.

	Train Error	Validation Error
Linear Regression	0.45164954425267917	0.44541345363574125
Decision Tree	0.4514970554993003	0.44588271774088134
Random Forest	0.14292529508022003	0.524622341082601
Gradient Boosting	0.44461427028706973	0.44894542017408745

TABLE II
TRAINING AND VALIDATION ERRORS OF MODELS WITH HYPERPARAMETER TUNING.

	Train Error	Validation Error
Decision Tree	0.45186397002173306	0.4454573828646934
Random Forest	0.4471405725445213	0.44645373581115466
Gradient Boosting	0.4424690561744056	0.45001584531105976

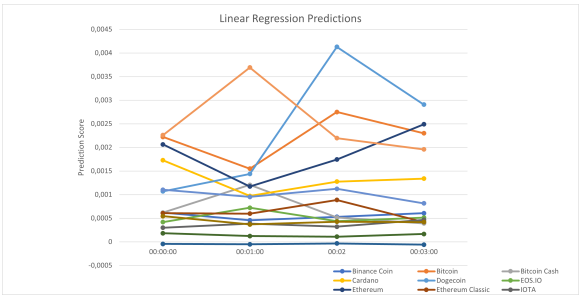

Fig. 18. Linear Regression Prediction Scores of 14 Cryptocurrencies.

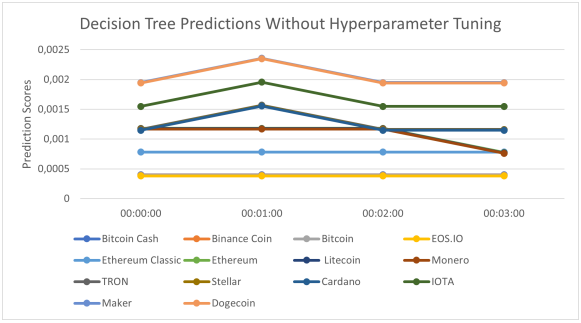

Fig. 19. Decision Tree Prediction Scores of 14 Cryptocurrencies without Hyperparamter tuning.

behavior.

2) Decision Tree Evaluation: In the prediction scores of Decision Tree after hyperparameter tuning in Fig. 20 are finely segregated. The max_depth was given on the range of 1 to 11. It can be seen that predictions are all positive after hyperparameter tuning the model.

3) Random Forest Evaluation: According to the loss curves of Random Forest for max_depth and n_estimator in Fig. 22 and Fig. 23, the values used were 5 and 10 respectively. As from that point the losses increase in accordance to the curves. The predictions have overall increased in value after hyperparameter tuning and have become less negative.

4) Gradient Boosting Evaluation: The predictions of Gradiant Boosting have become significantly less negative and a similar trend can be seen to be followed in a few dominant

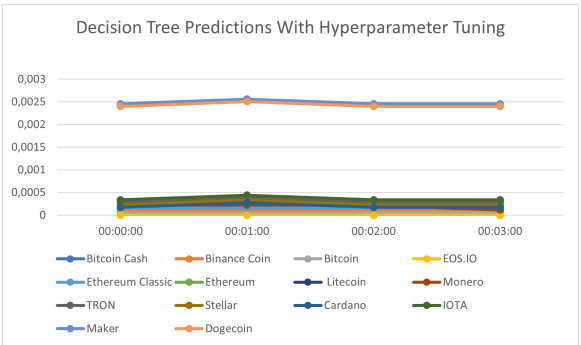

Fig. 20. Decision Tree Prediction Scores of 14 Cryptocurrencies with Hyperparamter tuning.

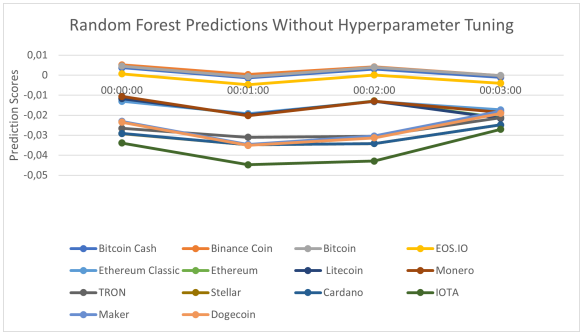

Fig. 21. Random Forest Prediction Scores of 14 Cryptocurrencies without Hyperparamter tuning.

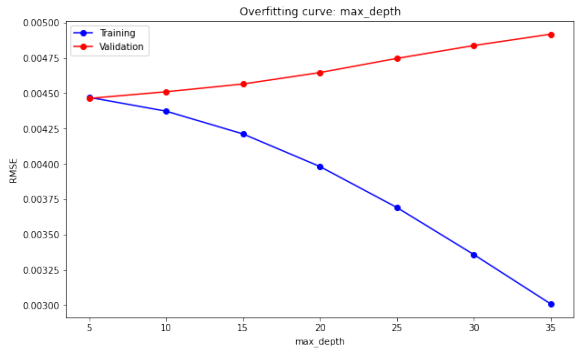

Fig. 22. Over-fitting curve of hyperparameter tuning for max_depth.

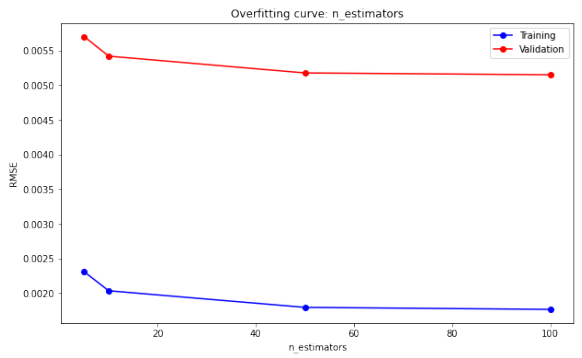

Fig. 23. Over-fitting curve of hyperparameter tuning for n_estimator.

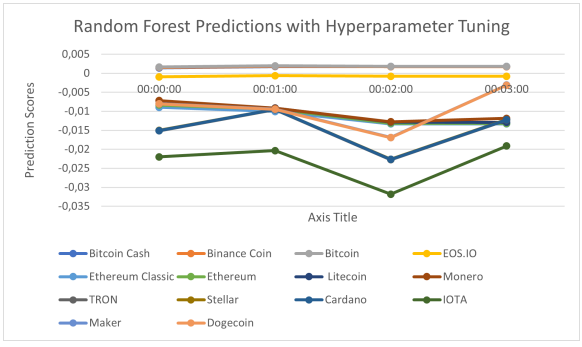

Fig. 24. Random Forest Prediction Scores of 14 Cryptocurrencies with Hyperparamter tuning of max_depth 5 and n_estimator 10.

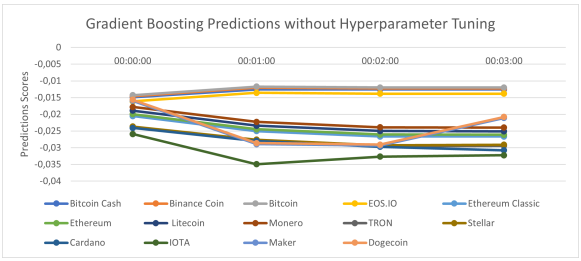

Fig. 25. Gradient Boosting Prediction Scores of 14 Cryptocurrencies without Hyperparamter tuning.

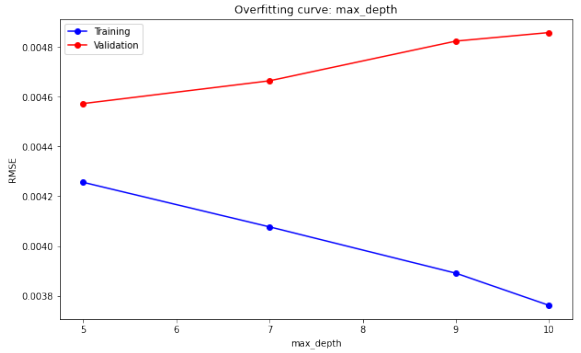

Fig. 26. Over-fitting curve of hyperparameter tuning for max_depth.

cryptoassets such as Dogecoin, IOTA, Stellar, Litecoin etc in Fig. 28. The hyperparameters used according to the loss curves were, n_estimator 20 and maxdepth 5. Because at 20 the curve gets stable for n_estimator as seen in Fig. 27 and at 5 the curve was at the lowest loss for max_depth as seen in Fig. 26.

VI. Discussion

In this section we will discuss the findings of this experiment. Three things can be observed.

1. Almost all the models have the same train and validation error which is approx. 0.44% and 0.45%. Which is a very low value. That shows all the models are trained well and can predict with an acceptable accuracy. The model selection passes and discreetly no model outperformed the other. For the use case at hand, each model has the same behavior.

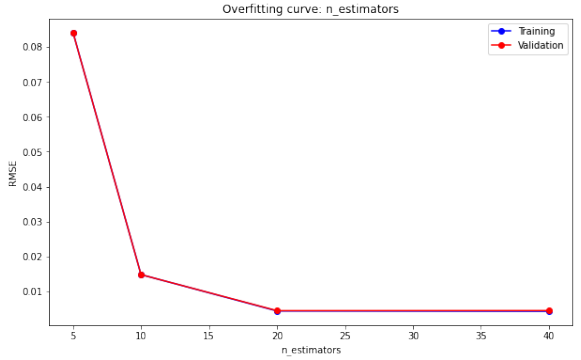

Fig. 27. Over-fitting curve of hyperparameter tuning for n_estimator.

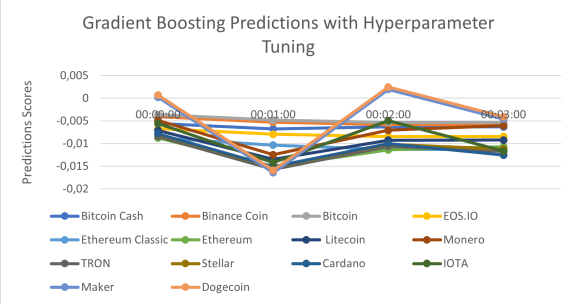

Fig. 28. Gradient Boosting Prediction Scores of 14 Cryptocurrencies with Hyperparamter tuning of max_depth 5 and n_estimator 20.

2. Both the errors, train and validation are low and close to each other. This explains that the much-required middle ground is achieved. This represents a good fit model. Overall, if the errors were calculated at different learning stages of the model (for e.g., at increasing training set sizes or increasing features), and the training and validation errors were drawn at each stage, a good fit model would have been when both the values would decrease and then become constant or flatlined on a graph [14].

3. Hyperparameter tuning worked significantly for Random Forest than the other models. By default, parameters of Random Forest, the validation error was higher that training error. Which means the model did not do well with predicting on data other than training set. After hyperparameter tuning and finding the additional parameters of depth of (max_depth) and number of trees (n_estimator) at which the model could be better, the training and validation error was brought down to the same level as other models.

VII. Conclusion

In this paper four machine learning models were used to predict and forecast for fourteen different types of cryptocurrencies. The evaluation metrics, and the scores show all the models are a good fit with very low losses. The plots convey that with hyperparameter tuning models can better the performance and learning. Also, data engineering is seen in this paper to be the significant factor in efficient learning of the models.

It is to be noted that the dynamism and volatility of the financial crypto market makes forecasting difficult and so factors such as social media and policies need also to be scaled for a better model learning. These are unscalable factors, which could be a future challenge for the successor of this project.

Acknowledgement

I would like to thank Prof. Dr. Doina Logofatu for providing me the opportunity to work on this topic under her supervision and for providing me the proper guidance to complete the assigned project and this report page.

References

[1] M.J.Hammayel, A. Y. Owda, “A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms”, 13 10 2021 [Online]. Available: https://www.mdpi.com/2673-2688/2/4/30 [Accessed 10 01 2022]

[2] A. Chaudhari, “Forecasting Cryptocurrency Prices using Machine Learning”, 12 12 19 [Online]. Available: http://norma.ncirl.ie/4272/1/ashwinichaudhari.pdf [Accessed 10 01 2022]

[3] V. Derbentsev, N. Datsenko, V. Babenko, O. Pushko and O. Pursky, ”Forecasting Cryptocurrency Prices Using Ensembles-Based Machine Learning Approach,” 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC ST), 2020, pp. 707-712, doi: 10.1109/PICST51311.2020.9468090.

[4] Y.-y. Song and Y. Lu, ”Decision tree methods: applications for classification and prediction,” Shanghai Arch Psychiatry, vol. 27, no. 2, pp. 130-135, 2015.

[5] L. Breiman, ”Random Forests,” in Machine Learning 45, https://doi.org/10.1023/A:1010933404324, 2001, p. 5–32.

[6] C. Lui and K. Pareek, ”Random Forest-based Voice Activity Detector,” Unpublished, New York University, 2018.

[7] L. Ceriani and P. Verme, ”The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini. J Econ Inequal,” vol. 10, p. 421–443, 2012.

[8] A. Sarica, A. Cerasa and A. Quattrone, ”Random Forest Algorithm for the Classification of Neuroimaging Data in Alzheimer’s Disease: A Systematic Review,” 06 10 2017. [Online]. Available: https://doi.org/10.3389/fnagi.2017.00329. [Accessed 09 04 2021].

[9] Anurag, ”Random Forest Analysis in ML and when to use it,” 17 08 2018. [Online]. Available: https://www.newgenapps.com/blog/random-forest-analysis-in-ml-and-when-to-use-it/. [Accessed 09 04 2021].

[10] S. Narkhede, ”Understanding Confusion Matrix,” 09 05 2018. [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Accessed 04 10 2021].

[11] Nasima Tamboli, ”All you need to know about different types of missing data values and how to handle them,” 29 10 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/. [Accessed 10 01 2022]

[12] H.Patel, “What is Feature Engineering — Importance, Tools and Techniques for Machine Learning”, 30 08 2021 [Online]. Available: https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10: :text=Feature

[13] Baeldung, ”Training and Validation Loss in deep learning,” 10 03 2022. [Online]. Available: https://www.baeldung.com/cs/training-validation-loss-deep-learning [Accessed 10 01 2022]

[14] K. Muralidhar, ”Learning curve to identify Overfitting and Underfitting in Machine Learning,” 09 01 2021. [Online]. Available: https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5 [Accessed 10 01 2022]

[15] Great Learning Team, ”Hyperparamter Tuning with GridSearchCV,” 29 09 2020. [Online] https://www.mygreatlearning.com/blog/gridsearchcv/. Available: [Accessed 10 01 2022]