**Cryptocurrency Price Forecasting Using Machine Learning**

The main goal of Cryptocurrency prices forecasting is enabling the businesses make accurate decisions by observing the possible closing price of the currency over the certain time interval, thus making businesses efficient in the performance.

Hence, we need to predict the closing price and forecast it for the next week, month, quarter or year. Here, Close i.e. closing price is the continuous variable which varies from moment to moment.

Now, as a Data Scientist, it is our responsibility to develop the prediction models which produce the continuous variable as an output.

Hence, we need to develop Regression Prediction Models.

1. **Linear Regression** is a type of supervised machine-learning algorithm that learns from the labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets. It assumes that there is a linear relationship between the input and output, meaning the output changes at a constant rate as the input changes. This relationship is represented by a straight line.

This algorithm is used to predict the outcome of a continuous variable based on other variables by using the equation

Where y is the output variable.

Performance in the project:

**Mean absolute percentage error: 0.7934279346709617**

**Root mean square error (RMSE): 14.047045679351324**

**R2 score: 0.9993079400794427**

1. **Support Vector Machine (SVM)** is a supervised machine learning algorithm that can be used for classification and regression tasks. The main idea behind SVM is to find the best boundary (or hyperplane) that separates the data into different classes.

SVMs can also be used for regression tasks by allowing for some of the data points to be within the margin, rather than on the boundary. This allows for a more flexible boundary and can lead to better predictions.

Performance in the project:

**Mean absolute percentage error: 20.195947021760304**

**Root mean square error (RMSE): 545.410273160165**

**R2 score: -0.0433267368128365**

1. **Decision Tree** is a supervised learning algorithm used for both classification and regression tasks. It has a hierarchical tree structure which consists of a root node, branches, internal nodes and leaf nodes. It It works like a flowchart help to make decisions step by step where: Internal nodes represent attribute tests, Branches represent attribute values and Leaf nodes represent final decisions or predictions.

Decision trees are widely used due to their interpretability, flexibility and low preprocessing needs.

They use criterion like Gini Index and Entropy to split the dataset for making more accurate decisions and deduce the patterns in the data.

Performance in the project:

**Mean absolute percentage error: 0.023876658822981518**

**Root mean square error (RMSE): 17.140638614096073**

**R2 score: 0.9989695481429972**

1. **Random Forest** is a machine learning algorithm that uses many decision trees to make better predictions. Each tree looks at different random parts of the data and their results are combined by voting for classification or averaging for regression. This helps in improving accuracy and reducing errors.

Bootstrap Aggregating (Bagging):

Random Forest uses bagging, where multiple subsets of the training data are randomly sampled (with replacement) to train individual decision trees.

Random Feature Selection:

When building each tree, a random subset of the features is considered for splitting nodes, which helps reduce correlation between the trees and improves generalization.

Majority Voting (Classification) or Averaging (Regression):

For classification tasks, the final prediction is determined by the majority vote of the individual trees. For regression tasks, the predictions are averaged.

Performance in the project:

**Mean absolute percentage error: 0.020963280980523726**

**Root mean square error (RMSE): 15.634507698125104**

**R2 score: 0.9991426816282051**

**Boosting** is an ensemble learning technique that sequentially combines multiple weak classifiers to create a strong classifier. It is done by training a model using training data and is then evaluated. Next model is built on that which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or predefined number of iterations is reached.

1. **Adaptive Boosting** is a boosting technique that assigns equal weights to all training samples initially and iteratively adjusts these weights by focusing more on misclassified datapoints for next model. It effectively reduces bias and variance making it useful for classification tasks but it can be sensitive to noisy data and outliers.

Performance in the project:

**Mean absolute percentage error: 1.834572285340577**

**Root mean square error (RMSE): 73.86765313684302**

**R2 score: 0.9808626274449461**

1. **Gradient Boosting** is the powerful ensemble learning method used for classification and regression tasks. It is a boosting algorithm which combine multiple weak learners to create a strong predictive model.

Here, each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration the algorithm computes the gradient of the loss function with respect to predictions and then trains a new weak model to minimize this gradient. Predictions of the new model are then added to the ensemble (all models prediction) and the process is repeated until a stopping criterion is met.

Performance in the project:

**Mean absolute percentage error: 42.59619856470977**

**Root mean square error (RMSE): 529.8177636367058**

**R2 score: 0.015475020769962655**

1. **XGBoost** is an optimized implementation of Gradient Boosting and is a type of ensemble learning method that combines multiple weak models to form a stronger model.

XGBoost uses decision trees as its base learners and combines them sequentially to improve the model’s performance. Each new tree is trained to correct the errors made by the previous tree and this process is called boosting.

It has built-in parallel processing to train models on large datasets quickly. XGBoost also supports customizations allowing users to adjust model parameters to optimize performance based on the specific problem.

XGBoost extends traditional gradient boosting by including regularization elements in the objective function, XGBoost improves generalization and prevents overfitting.

Performance in the project:

**Mean absolute percentage error: 38.97930240407099**

**Root mean square error (RMSE): 484.66901774945813**

**R2 score: 0.17611949557208173**

There are several other regression algorithms but these help in providing accurate results efficiently as 3/4th of the algorithms support parallel-processing of data. The performance of the models is evaluated with metrics like mean absolute percentage error, root mean squared error, R^2 score etc. to showcase the scope of improvement of the project.