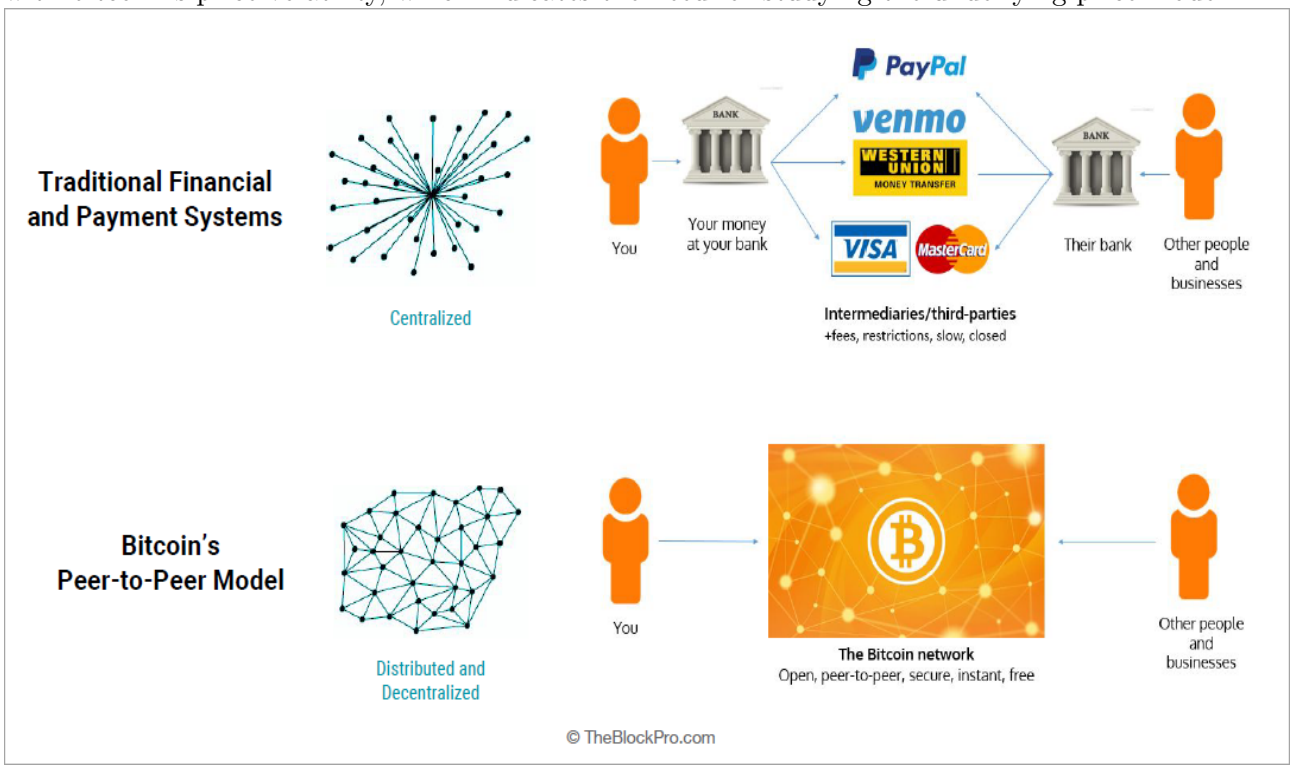
Bitcoin closing price prediction based on the opening price on a particular day

Name:	**Yash Saxena**
Registration No./Roll No.:	19338
Institute/University Name:	IISER Bhopal
Program/Stream:	Economic Sciences
Problem Release date:	February 02, 2022
Date of Submission:	April 24, 2022

1 Introduction

We are provided with a problem statement where we need to predict the closing price of bitcoin on a particular day based on the set of input features. Bitcoin is a decentralized digital currency without any central administrator and can be sent directly from user to user on the bitcoin network. Bitcoin gains value based on the community involvement. i.e. Demand and Supply. One of the main problems with bitcoin is price volatility, which indicates the need for studying the underlying price model.



It's a regression problem as the desired output is a continuous variable. We have used supervised machine learning techniques as we are provided with the class labels along with the training data set.

1.1 Dataset and Features

We are a provided with a training dataset, training label dataset and test dataset. The training dataset consists of 749 instances of all the features, namely, Date, Timestamp, Weighted Price, Highest Price hit during the day, Lowest Price hit during the day and Opening Price of Bitcoin on that particular day.

date:	Date
ts:	Timestamp
wp:	Weighted Price
hi:	Highest price hit on the day
lo:	Lowest price hit on the day
open:	Opening price on the particular day

All the features are numerical and not categorical.

2 Methods

I've used ML Regression algorithms, namely, Linear Regression, Decision Tree Regression, Random Forest Regression(RF) and K-Neighbours Regression to predict the closing price.
The features used as the input variables in all the algorithms are timestamp(ts), weighted price(wp), highest price(hi), lowest price(lo), opening price(open).
For hyper-parameter tuning I have used Grid Search with cross-validation with 10 partitions on all the above 4 algorithms to find the best parameters which minimizes the error. During Tuning I have first randomly given the grid of parameters to each model and then got the best out of them , then, based on the results I again modify the values of the parameters and then run the grid search to obtain new set of best values of parameters which can minimize the error further. I have iterated this process until there is no further decrease in the error.
As the target variable is highly volatile, we want to penalize the model when the error is higher. Root Mean Squared Error (RMSE) squares the errors before averaging, and thus RMSE gives a relatively high weight to large errors.

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}$$

n : number of datapoints.
yi : True value of Closing Price.
xi : Predicted Value of Closing Price.

GitHub Repo Link

3 Experimental Analysis

Observations from different results:

3.1 Linear Regression

We've passed various linear regression parameters, namely, fit-intercept, normalize, copy-X and n-jobs while training the model to obtain the best possible results. The following table summarizes the training phases in terms of the best parameters with their values and the optimal score obtained, that is, the value of root mean squared error.

S.No.	Best Set of Parameters	Error
1	'normalize': False, 'copy-X': True	82.47132038220482
2	'fit-intercept': False, 'n-jobs': 1	82.30026558070303
2	'normalize': True, 'copy-X': True, 'fit-intercept': False, 'n-jobs': 1	82.30026558070303

3.2 Decision Tree Regression

We've passed various decision-tree parameters, namely, splitter, max-features, max-depth and ccp-alpha and changed their values while training the model multiple times to obtain the best possible results. The following table summarizes the training phases in terms of the best set of the parameters

and their values, and the optimal score obtained, that is, the value of root mean squared error.

S.No.	Best Set of Parameters	Error
1	'ccp-alpha': 0.1, 'max-depth': 60, 'max-features': 'auto', 'splitter': 'random'	168.8466975506378
2	'ccp-alpha': 0.011, 'max-depth': 70, 'max-features': 'sqrt', 'splitter': 'best'	156.13218634550233
3	'ccp-alpha': 0.0115, 'max-depth': 90, 'max-features': 'auto', 'splitter': 'random'	157.2419547010282
4	'ccp-alpha': 0.0098, 'max-depth': 80, 'max-features': 'log2', 'splitter': 'random'	153.94773130141624
5	'ccp-alpha': 0.0115, 'max-depth': 80, 'splitter': 'random'	162.38749176708188

3.3 Random Forest Regression

We've passed various Random Forest parameters, namely, n-estimators, max-depth and ccp-alpha and changed their values while training the model multiple times to obtain the best possible results. The following table summarizes the training phases in terms of the best set of the parameters and their values, and the optimal score obtained, that is, the value of root mean squared error.

S.No.	Best Set of Parameters	Error
1	'ccp-alpha': 0.05, 'max-depth': 20, 'n-estimators': 30	126.90070986056386
2	'ccp-alpha': 0.03, 'max-depth': 15, 'n-estimators': 40	127.14198722313422
3	'ccp-alpha': 0.07, 'max-depth': 20, 'n-estimators': 29	122.14698024337267
4	'ccp-alpha': 0.065, 'max-depth': 10, 'n-estimators': 19	117.34445116309652

3.4 K-Neighbours Regression

We've passed various K-neighbours parameters, namely, n-neighbours, algorithm and leaf-size and changed their values while training the model multiple times to obtain the best possible results. The following table summarizes the training phases in terms of the best set of the parameters and their values, and the optimal score obtained, that is, the value of root mean squared error.

S.No.	Best Set of Parameters	Error
1	'algorithm': 'auto', 'leaf-size': 20, 'n-neighbors': 3	226.69262575485283
2	'algorithm': 'auto', 'leaf-size': 1, 'n-neighbors': 2	194.29206488725336
3	'algorithm': 'auto', 'leaf-size': 3, 'n-neighbors': 2	194.29206488725336

3.5 Analysis

The maximum fluctuation in the root mean squared error could be up to 100. Above this number, every computation and every algorithm is discarded. We've chosen 100 as an upper bound for RMSE of predicted Bitcoin price because beyond this value any further fluctuation won't add much value to the trader/investor.
In our computation, only Linear Regression satisfies this criterion and hence, it is selected as the only algorithm to predict the closing price of the Bitcoin on a particular day.

4 Discussions

Based on the calculations, I've performed and the results obtained; Linear Regression has emerged as the best algorithm for the given regression problem with the existing number of features used to predict the closing price of the Bitcoin on a particular day.
We can use date as a feature and do the time series analysis to predict the price of the bitcoin. So, we can use Recurrent Neural Networks and Long Short-Term Memory(LSTM) networks to predict the closing price of bitcoin. Although, LSTM has a high computation time but it is competitive to use along with Linear Regression.