FINANCIAL MANAGEMENT (MGT-1029)

TEAM MEMBERS (TEAM-10):
V RAGHAV ANAND 19BCE1415
ANTHRA DEVARAJAN 19BCE1681
S SABHARI GIRISH 19BCE1759
NITHISH J V 19BEE1067
ROHIT DINESH 19BEE1111

FACULTY: PROF SUBHAMITRA.P

J COMPONENT REVIEW-3

PROJECT TITLE: BITCOIN PRICE PREDICTION USING MACHINE LEARNING ALGORITHMS

RESEARCH PROBLEM

- To find an appropriate dataset historical dataset for bitcoin.
- Do the required pre-processings to work further on the data for analytics as well as predictions.
- Predict the close price of bitcoin for the given dataset using multiple machine learning techniques.
- Compare and analyze as to which machine learning algorithm works well for the given bitcoin historic data.

MOTIVATION FOR THE STUDY

- In today's times of automated investing and data- driven decision making, it is vital to explore and formulate a way to make investing more informative.
- Bitcoin has been gaining traction since inception and the bull-run during FY21 has bought in more retail investors into this space. Since cryptocurrencies is a new concept many of these investors are prone to bad investments. Automated investing can solve this problem.
- Bitcoin price changes being volatile by nature, has been difficult to predict. Through this project, we aim to test various training algorithms to achieve a method that will return the least error when predicting the price.

LITERATURE SURVEY

Paper-1:

Bitcoin price prediction using machine learning- An approach to sample dimension engineering:

The paper uses a set of high-dimension features including property and network, trading and market, attention and gold spot price for Bitcoin daily price prediction, while the basic trading features acquired from a cryptocurrency exchange are used for 5-minute interval price prediction. Statistical methods including Logistic Regression and Linear Discriminant Analysis for Bitcoin daily price prediction achieved an accuracy of 66%, outperforming more complicated machine learning algorithms. Compared with benchmark results for daily price prediction, a better performance with highest accuracies of the statistical methods and machine learning algorithms of 66% and 65.3% were achieved. Machine learning models including Random Forest, XGBoost, Quadratic Discriminant Analysis, Support Vector Machine and Long Short-term Memory for Bitcoin 5-minute interval price prediction are superior to statistical methods, with accuracy reaching 67.2%.

Reference: Zheshi Chen, Chunhong Li, Wenjun Sun, Bitcoin price prediction using machine learning: An approach to sample dimension engineering, Journal of Computational and Applied Mathematics, Volume 365, 2020, 112395, ISSN 0377-0427, https://doi.org/10.1016/j.cam.2019.112395.

Paper-2:

Bitcoin price prediction using machine learning:

This paper attempts to predict the Bitcoin price accurately taking into consideration various parameters that affect the Bitcoin value. For the first phase of our investigation, it aims to understand and identify daily trends in the Bitcoin market while gaining insight into optimal features surrounding Bitcoin price. The data set consists of various features relating to the Bitcoin price and payment network over the course of five years, recorded daily. For the second phase of the investigation, using the available information, it predicts the sign of the daily price change with highest possible accuracy. They have acquired bitcoin values from two different databases namely: Quandl and CoinmarketCap. After acquiring this time-series data recorded daily for five years at different time instances, it is normalized and smoothened. For this, they have implemented different normalization techniques like log transformation, z-score normalization, boxcox normalization, and so on. After this, data is smoothened over the complete time period, and the prediction is done.

Reference: S. Velankar, S. Valecha and S. Maji, "Bitcoin price prediction using machine learning," 2018 20th International Conference on Advanced Communication Technology (ICACT), 2018, pp. 144-147, doi: 10.23919/ICACT.2018.8323676.

Paper -3:

A Comparative Study of Bitcoin Price Prediction Using Deep Learning:

In this paper, various state-of-the-art deep learning methods such as a deep neural network (DNN), a long short-term memory (LSTM) model, a convolutional neural network, a deep residual network, and their combinations are used for Bitcoin price prediction. Experimental results showed that although LSTM-based prediction models slightly outperformed the other prediction models for Bitcoin price prediction (regression), DNN-based models performed the best for price ups and downs prediction (classification). In addition, a simple profitability analysis showed that classification models were more effective than regression models for algorithmic trading. Overall, the performances of the proposed deep learning-based prediction models were comparable.

Reference: Ji, S.; Kim, J.; Im, H. A Comparative Study of Bitcoin Price Prediction Using Deep Learning. Mathematics 2019, 7, 898. https://doi.org/10.3390/math7100898

Paper-4:

Machine Learning Models Comparison for Bitcoin Price Prediction:

This research aims to discover the most efficient and highest accuracy model to predict Bitcoin prices from various machine learning algorithms. By using 1-minute interval trading data on the Bitcoin exchange website named bitstamp from January 1, 2012 to January 8, 2018, some different regression models with scikit-Iearn and Keras libraries had experimented. The best results showed that the Mean Squared Error (MSE) was as low as 0.00002 and the R-Square (R 2) was as high as 99.2%. In this research, regression machine learning model was chosen due to continuous values of Bitcoin price. With the scikit-learn library, the best two regression models; Theil-Sen Regression and Huber Regression were selected to compare. For deep learning based regression models, Keras library was used to create LSTM and GRU models.

Reference: T. Phaladisailoed and T. Numnonda, "Machine Learning Models Comparison for Bitcoin Price Prediction," 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), 2018, pp. 506-511, doi: 10.1109/ICITEED.2018.8534911.

Paper-5:

Next-Day Bitcoin Price Forecast:

This study analyzes forecasts of Bitcoin price using the autoregressive integrated moving average (ARIMA) and neural network autoregression (NNAR) models. Employing the static forecast approach, it forecasts next-day Bitcoin price both with and without re-estimation of the forecast model for each step. For cross-validation of forecast results, it considers two different training and test samples. In the first training-sample, NNAR performs better than ARIMA, while ARIMA outperforms NNAR in the

second training-sample. Additionally, ARIMA with model re-estimation at each step outperforms NNAR in the two test-sample forecast periods. The Diebold Mariano test confirms the superiority of forecast results of ARIMA model over NNAR in the test-sample periods. Forecast performance of ARIMA models with and without re-estimation are identical for the estimated test-sample periods. Despite the sophistication of NNAR, this paper demonstrates ARIMA enduring power of volatile Bitcoin price prediction.
Reference: Munim, Z.H.; Shakil, M.H.; Alon, I. Next-Day Bitcoin Price Forecast. J. Risk Financial Manag. 2019, 12, 103. https://doi.org/10.3390/jrfm12020103

Paper-6:
Bitcoin Price Prediction: An ARIMA Approach:
This paper aims at revealing the usefulness of traditional autoregressive integrative moving average (ARIMA) model in predicting the future value of bitcoin by analyzing the price time series in a 3-years-long time period. Empirical studies reveal that this simple scheme is efficient in sub-periods in which the behavior of the time-series is almost unchanged, especially when it is used for short-term prediction, e.g. 1-day. On the other hand, when the ARIMA model is trained to a 3-years-long period, during which the bitcoin price has experienced different behaviors, or when it is used for a long-term prediction, it is observed that it introduces large prediction errors. Especially, the ARIMA model is unable to capture the sharp fluctuations in the price, e.g. the volatility at the end of 2017. Then, it calls for more features to be extracted and used along with the price for a more accurate prediction of the price. The paper further investigates the bitcoin price prediction using an ARIMA model, trained over a large dataset, and a limited test window of the bitcoin price, with length w, as inputs.
Reference: arXiv:1904.05315 [cs.SI] (or arXiv:1904.05315v1 [cs.SI] for this version) https://doi.org/10.48550/arXiv.1904.05315

Paper-7:
Bitcoin price prediction using ensembles of neural networks:
This paper explores the relationship between the features of Bitcoin and the next day change in the price of Bitcoin using an Artificial Neural Network ensemble approach called Genetic Algorithm based Selective Neural Network Ensemble, constructed using Multi-Layered Perceptron as the base model for each of the neural network in the ensemble. To better understand the practicality and its effectiveness in real-world application, the ensemble was used to predict the next day direction of the price of Bitcoin given a set of approximately 200 features of the cryptocurrency over a span of 2 years. Over a span of 50 days, a trading strategy based on the ensemble was compared against a “previous day trend following” trading strategy through back-testing. The former trading strategy generated almost 85% returns, outperforming the “previous day trend following” trading strategy which produced an approximate 38% returns and a

trading strategy that follows the single, best MLP model in the ensemble that generated approximately 53% in returns.
Reference: E. Sin and L. Wang, "Bitcoin price prediction using ensembles of neural networks," 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2017, pp. 666-671, doi: 10.1109/FSKD.2017.8393351.

Paper-8:
Deep Learning Approach to Determine the Impact of Socio Economic Factors on Bitcoin Price Prediction:
In this paper, a comparative study of the various parameters affecting bitcoin price prediction is done based on Root Mean Square Error (RMSE) using various deep learning models like Convolutional Neural Network (CNN), Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). It studies the effect of Gold price on the price of bitcoin. In their proposed methodology, the paper includes both social and financial factors to predict the bitcoin price to ease out predictions associated with bitcoin over years. It also used a new parameter Gold in predicting the price of bitcoin. Gold is one of the most important financial aspects which has a great influence in determining a country's economy. Therefore, the trends of gold price in the international market can also be seen as a parameter to give a better approach for bitcoin price prediction.
Reference: A. Aggarwal, I. Gupta, N. Garg and A. Goel, "Deep Learning Approach to Determine the Impact of Socio Economic Factors on Bitcoin Price Prediction," 2019 Twelfth International Conference on Contemporary Computing (IC3), 2019, pp. 1-5, doi: 10.1109/IC3.2019.8844928.

Paper-9:
A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial Institutions:
A cryptocurrency is a network-based digital exchange medium, where the records are secured using strong cryptographic algorithms such as Secure Hash Algorithm 2 (SHA-2) and Message Digest 5 (MD5). It uses blockchain technology to make the transactions secure, transparent, traceable, and immutable. Due to these properties, the cryptocurrencies have gained popularity in almost all the sectors especially in financial sectors. In this paper, a LSTM and GRU-based hybrid cryptocurrency prediction scheme is proposed, which focuses on only two cryptocurrencies, namely Litecoin and Monero. The results depict that the proposed scheme accurately predicts the prices with high accuracy, revealing that the scheme can be applicable in various cryptocurrencies price predictions.
Reference: Mohil Maheshkumar Patel, Sudeep Tanwar, Rajesh Gupta, Neeraj Kumar, A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial

Institutions, Journal of Information Security and Applications, Volume 55, 2020, 102583, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2020.102583.

Paper-10:

Systematic Erudition of Bitcoin Price Prediction using Machine Learning Techniques:

This paper conducts an in-depth study on evolution of Bitcoin and also a systematic review is done on various machine learning algorithms used for predicting the prices. Comparative analysis envisions to select optimal technique to forecast prices more precisely. The models used in the paper are as follows: Auto-Regressive Integrated Moving Average Model, Regression model, Latent Source Model, Binomial Generalized Linear Model, Generalized Autoregressive Conditional Heteroskedasticity Model, Support Vector Machine Model, Long Short Term Memory Network Model and the Non-linear Auto-Regressive with Exogenous Input Model. The solutions provided in many of the existing system gives 60- 70% accuracy. Although some techniques are not considered as the accuracy obtained by them is very less, the overall study is very promising and can help investors to invest accordingly. The accuracy of NARX Model is the best model in predicting the Bitcoin prices, according to the study conducted in this paper.

Reference: P. V. Rane and S. N. Dhage, "Systematic Erudition of Bitcoin Price Prediction using Machine Learning Techniques," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp. 594-598, doi: 10.1109/ICACCS.2019.8728424.

RESEARCH GAP

- Price prediction tool have been around for a while but they have either been inaccessible to the public or are too technical to understand.
- In stock-trading price-predictor models, indicators such as the ichimoku cloud and pattern prediction have been used. But in case of cryptocurrencies, due to its high volatility and assets being concentrated in few individual wallets (whales), it is difficult to come up with a trading strategy that involves data predictions.
- There are price predictors that use time series predictions. But price predictors that use ML algorithms are currently unavaliable or are vastly inaccurate.

OBJECTIVE OF THE STUDY

- As already stated, bitcoin has gained traction for the last few years and the bull run has bought more retail investors to the space. These non-professionalists and newbie buyers have less experience in the field of the bitcoin market.

Therefore prior forecast of the bitcoin price or the prediction of the trend might help them buy and sell them accordingly.

- Use different machine learning algorithm on the bitcoin dataset to find which is efficient and predict the close price of the bitcoin at the end of the day.
- Minimize the error in the dataset using appropriate scaling and optimization techniques.
- Use the live dataset of the bitcoin to predict or forecast the current trend in the close price at the end of the day.
- Compare the results obtained using different prediction or forecast algorithms and differentiate them accordingly.

DATA EXPLANATION AND SOURCE

The dataset has been obtained from the following source:
https://finance.yahoo.com/quote/BTC-USD/history/

A sample screenshot of the dataset:

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	20-04-2021	55681.79297	57062.14844	53448.04688	56473.03125	56473.03125	67849323955
3	21-04-2021	56471.12891	56757.97266	53695.46875	53906.08984	53906.08984	54926612466
4	22-04-2021	53857.10547	55410.23047	50583.8125	51762.27344	51762.27344	74798630778
5	23-04-2021	51739.80859	52120.79297	47714.66406	51093.65234	51093.65234	86668667320
6	24-04-2021	51143.22656	51167.5625	48805.28516	50050.86719	50050.86719	49014494781
7	25-04-2021	50052.83203	50506.01953	47159.48438	49004.25391	49004.25391	46117114240
8	26-04-2021	49077.79297	54288.00391	48852.79688	54021.75391	54021.75391	58284039825
9	27-04-2021	54030.30469	55416.96484	53319.1875	55033.11719	55033.11719	49448222757
10	28-04-2021	55036.63672	56227.20703	53887.91797	54824.70313	54824.70313	48000572955
11	29-04-2021	54858.08984	55115.84375	52418.02734	53555.10938	53555.10938	46088929780
12	30-04-2021	53568.66406	57900.71875	53129.60156	57750.17578	57750.17578	52395931985
13	01-05-2021	57714.66406	58448.33984	57052.27344	57828.05078	57828.05078	42836427360
14	02-05-2021	57825.86328	57902.59375	56141.90625	56631.07813	56631.07813	38177405335
15	03-05-2021	56620.27344	58973.30859	56590.87109	57200.29297	57200.29297	51713139031

Description of the attributes:

Date - The date at which the bitcoin prices are registered.
Open - When an exchange starts for the day, the opening price is the price at which a securities trades for the first time.
High - The highest price of bitcoin registered on that day.
Low - The lowest price of bitcoin registered on that day.
Close - The price of a security in a financial market at the end of the day's trading.

Adj Close - The adjusted closing price adjusts a stock's closing price to reflect its value after any corporate actions have been taken into account.
Volume - It's computed by adding up all of the dollars traded in each transaction (price multiplied by the number of shares traded) and then dividing by the total number of shares traded.

SAMPLE TIME-PERIODS AND VARIABLES TO BE CONSIDERED IN STUDY

- Sample time periods: Each row of the data contains a single day's bitcoin price, its high, low value, etc. So the sample's time period is one day.
- Variables to be considered: Variables to be considered for the study are open price, close price, volume, the high value of the bitcoin, low value of the bitcoin, and change in the percentage of the value of the bitcoin in comparison to the previous day.

CODE AND METHODOLOGY

The coding is implemented using the Jupyter notebook interface where all the data preprocessing work and the Machine Learning models are developed.

Initially, the libraries are imported to get started with the work:

Importing the required libraries for the analysis

```
from functools import reduce
import pandas as pd
import numpy as np
import pmdarima as pmd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn import svm
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sb
from statsmodels.tsa.arima.model import ARIMA
import statsmodels.api as sm
```

The csv file of the bitcoin price data is read into the Jupyter notebook, and the datatypes of the various attributes present in the dataset are observed:

```
data
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2021-04-20	55681.792969	57062.148438	53448.046875	56473.031250	56473.031250	67849323955
1	2021-04-21	56471.128906	56757.972656	53695.468750	53906.089844	53906.089844	54926612466
2	2021-04-22	53857.105469	55410.230469	50583.812500	51762.273438	51762.273438	74798630778
3	2021-04-23	51739.808594	52120.792969	47714.664063	51093.652344	51093.652344	86668667320
4	2021-04-24	51143.226563	51167.562500	48805.285156	50050.867188	50050.867188	49014494781
...	...	...	...	...	...	...	...
361	2022-04-16	40552.316406	40633.679688	40078.425781	40424.484375	40424.484375	16833150693
362	2022-04-17	40417.777344	40570.726563	39620.894531	39716.953125	39716.953125	19087633042
363	2022-04-18	39721.203125	40986.320313	38696.191406	40826.214844	40826.214844	33705182072
364	2022-04-19	40828.175781	41672.960938	40618.632813	41502.750000	41502.750000	25303206547
365	2022-04-20	41453.355469	41510.558594	41282.078125	41282.078125	41282.078125	24289802240

366 rows × 7 columns

```
data.dtypes
```

```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

<u>Checking using the correlation map to find out how the columns in the dataset are dependent on each other (A value close to 1 indicating strong relationship and a value close to 0 indicating weak relationship):</u>

Plotting the heatmap for the correlation of the dataset

```
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	Open	High	Low	Close	Adj Close	Volume
Open	1.000000	0.992602	0.985523	0.979704	0.979704	0.239840
High	0.992602	1.000000	0.984882	0.990655	0.990655	0.270608
Low	0.985523	0.984882	1.000000	0.990554	0.990554	0.145477
Close	0.979704	0.990655	0.990554	1.000000	1.000000	0.208262
Adj Close	0.979704	0.990655	0.990554	1.000000	1.000000	0.208262
Volume	0.239840	0.270608	0.145477	0.208262	0.208262	1.000000

The closing price of bitcoin is the dependent variable, which will be predicted using the various other independent variables present in the dataset. And the date column has been removed from the dataset as it does not contribute towards building the ML model for predicting the price of a bitcoin:

```
y=data["Close"]
del data["Date"]
```

The dataset is then split into training and testing data. The ML models will be trained using the training dataset and its performance will be measured using the test data.

Dividing the test and train split

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1,shuffle = False, stratify = None)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(329, 5) (37, 5) (329, 1) (37, 1)
```

MODEL BUILDING AND PREDICTING THE PRICE OF THE BITCOIN:

An object for linear regression is created and the training data is fit into the object.

1. Building the linear regression model

```
model=LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression()
```

After the object model is created, the test data is used to predict the prices of the bitcoin, and the predicted value is compared with the actual prices of the test data.

Predicting and analysing the difference with the test data

```
y_pred=model.predict(X_test)
val={
    "Predicted": reduce(lambda z, y :z + y, y_pred.tolist()),
    "Actual": reduce(lambda z, y :z + y, y_test.values.tolist())
}
ans=pd.DataFrame(val)
ans
```

	Predicted	Actual
0	39338.785156	39338.785156
1	41143.929688	41143.929688
2	40951.378906	40951.378906
3	41801.156250	41801.156250
4	42190.652344	42190.652344
5	41247.824219	41247.824219
6	41077.996094	41077.996094
7	42358.808594	42358.808594
8	42892.957031	42892.957031
9	43960.933594	43960.933594
10	44348.730469	44348.730469

Next, the decision tree algorithm is used to predict the prices, and the actual vs predicted prices are observed:

2. Building the decision tree regression model

```
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(random_state=0)
```

Predicting and analysing the difference with the test data

```
y_pred=regressor.predict(X_test)
val={
    "Predicted": y_pred.tolist(),
    "Actual": reduce(lambda z, y :z + y, y_test.values.tolist())
}
ans=pd.DataFrame(val)
ans
```

	Predicted	Actual
0	39400.585938	39338.785156
1	40869.554688	41143.929688
2	41034.542969	40951.378906
3	41821.261719	41801.156250
4	42197.515625	42190.652344
5	41626.195313	41247.824219
6	41034.542969	41077.996094
7	42375.632813	42358.808594
8	42909.402344	42892.957031
9	43949.101563	43960.933594
10	44338.796875	44348.730469

<u>K Nearest Neighbour algorithm:</u>

3. Building the KNN regression model

```
neigh = KNeighborsRegressor(n_neighbors=30)
neigh.fit(X_train, y_train)
```

```
KNeighborsRegressor(n_neighbors=30)
```

Predicting and analysing the difference with the test data

```
y_pred=neigh.predict(X_test)
val={
    "Predicted": reduce(lambda z, y :z + y, y_pred.tolist()),
    "Actual": reduce(lambda z, y :z + y, y_test.values.tolist())
}
ans=pd.DataFrame(val)
ans
```

	Predicted	Actual
0	41868.330078	39338.785156
1	47189.765560	41143.929688
2	38589.714714	40951.378906
3	45900.532682	41801.156250
4	38831.392513	42190.652344
5	38787.365300	41247.824219
6	42804.990755	41077.996094
7	47622.347005	42358.808594
8	43788.262630	42892.957031
9	47412.395833	43960.933594
10	47066.029427	44348.730469

Support Vector Machine algorithm:

4. Building a support vector model for regression

```
support=svm.SVR()
support.fit(X_train, y_train)
```

Predicting and analysing the difference with the test data

```
y_pred=support.predict(X_test)
val={
    "Predicted": y_pred.tolist(),
    "Actual": reduce(lambda z, y :z + y, y_test.values.tolist())
}
ans=pd.DataFrame(val)
ans
```

	Predicted	Actual
0	44683.846114	39338.785156
1	44702.728639	41143.929688
2	44681.813646	40951.378906
3	44696.567129	41801.156250
4	44679.570821	42190.652344
5	44679.991502	41247.824219
6	44684.601740	41077.996094
7	44693.546465	42358.808594
8	44685.312033	42892.957031
9	44692.340031	43960.933594
10	44691.753427	44348.730469

Random Forest algorithm:

5. Building random forest regressor model

```
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor.fit(X_train, y_train)
```

Predicting and analysing the difference with the test data

```
y_pred=regressor.predict(X_test)
val={
    "Predicted": y_pred.tolist(),
    "Actual": reduce(lambda z, y :z + y, y_test.values.tolist())
}
ans=pd.DataFrame(val)
ans
```

	Predicted	Actual
0	39333.161524	39338.785156
1	40961.264688	41143.929688
2	40825.590352	40951.378906
3	41735.312188	41801.156250
4	42210.755977	42190.652344
5	41413.453203	41247.824219
6	40926.951211	41077.996094
7	42378.569688	42358.808594
8	42848.758906	42892.957031
9	43932.695196	43960.933594
10	44397.558242	44348.730469

TIME SERIES FORECASTING:

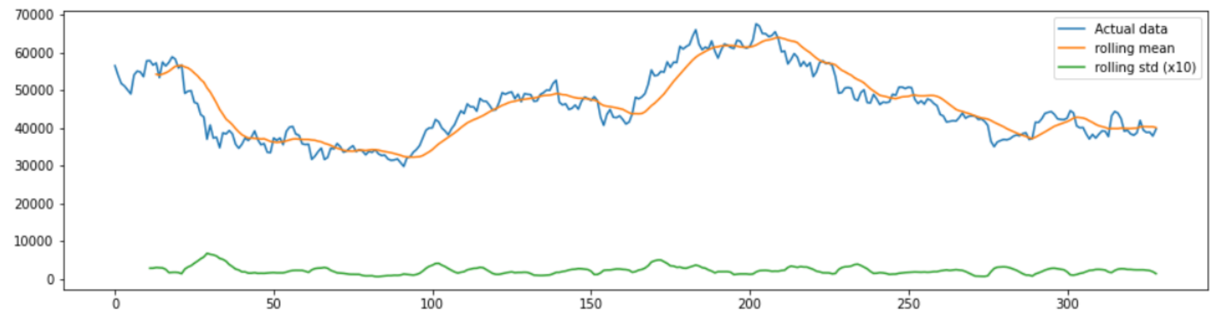
Using time series forecasting

Plotting the rolling mean and rolling standard deviation to check if the data set is stationary

```
rolmean = y_train.rolling(window=14).mean()
rolstd = y_train.rolling(window=12).std()

fig, ax = plt.subplots(figsize=(16, 4))
ax.plot(y_train, label= "Actual data")
ax.plot(rolmean, label='rolling mean');
ax.plot(rolstd, label='rolling std (x10)');
ax.legend()
```

```
<matplotlib.legend.Legend at 0x2a9017e28e0>
```



The above graph shows that the dataset is not stationary

Conducting mathematical tests and converting the dataset into a stationary dataset:

Conducting a mathematical test to see if the dataset is stationary

```
from statsmodels.tsa.stattools import adfuller
dftest = adfuller(y_train.dropna(), autolag='AIC')
print('Test statistic = {:.3f}'.format(dftest[0]))
print('P-value = {:.3f}'.format(dftest[1]))
print('Critical values :')
for k, v in dftest[4].items():
    print('\t{}: {} - The data is {} stationary with {}% confidence'.format(k, v, 'not' if v<dftest[0] else '', 100-int(k[:-1]))
```

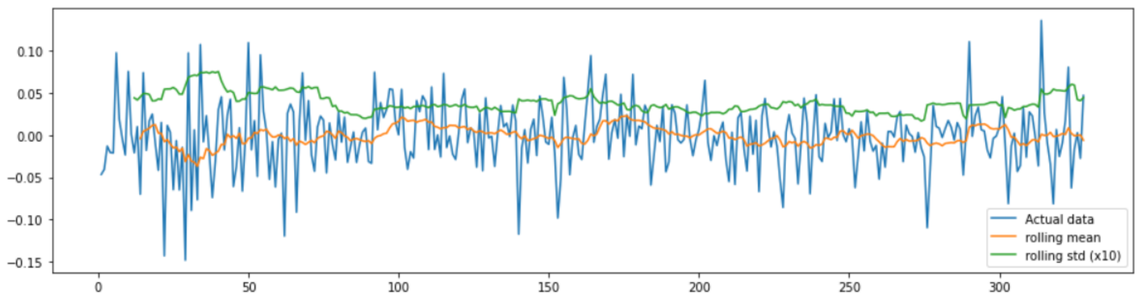
```
Test statistic = -1.935
P-value = 0.316
Critical values :
	1%: -3.4504451681828194 - The data is not stationary with 99% confidence
	5%: -2.870392380216117 - The data is not stationary with 95% confidence
	10%: -2.571486353732897 - The data is not stationary with 90% confidence
```

Using log transformation to make the dataset stationary

```
y_detrend= np.log(y_train)
y_detrend = y_detrend - y_detrend.shift(1)
y_detrend.dropna()
y_detest= np.log(y_test)
y_detest = y_detest - y_detest.shift(1)
y_detest.dropna()
rolmean = y_detrend.rolling(window=14).mean()
rolstd = y_detrend.rolling(window=12).std()

fig, ax = plt.subplots(figsize=(16, 4))
ax.plot(y_detrend, label= "Actual data")
ax.plot(rolmean, label='rolling mean');
ax.plot(rolstd, label='rolling std (x10)');
ax.legend()
```

```
<matplotlib.legend.Legend at 0x2a901877f40>
```



Proving that the dataset is stationary:

Proving that the dataset is stationary

```
dftest = adfuller(y_detrend.dropna(), autolag='AIC')
print('Test statistic = {:.3f}'.format(dftest[0]))
print('P-value = {:.3f}'.format(dftest[1]))
print('Critical values :')
for k, v in dftest[4].items():
    print('\t{}: {} - The data is {} stationary with {}% confidence'.format(k, v, 'not' if v<dftest[0] else '', 100-int(k[:-1]))
```

```
Test statistic = -19.051
P-value = 0.000
Critical values :
	1%: -3.45050711373316 - The data is  stationary with 99% confidence
	5%: -2.8704195794076743 - The data is  stationary with 95% confidence
	10%: -2.571500856923753 - The data is  stationary with 90% confidence
```

Finding the best AR(p) and IM(q) value

```
autoarima_model = pmd.auto_arima(y_train.dropna(), start_p=1, start_q=1,test="adf",trace=True)
autoarima_model
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=5846.592, Time=0.06 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=5847.006, Time=0.01 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=5846.311, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=5846.041, Time=0.02 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=5845.277, Time=0.01 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 0.127 seconds
```

```
ARIMA(order=(0, 1, 0), scoring_args={}, suppress_warnings=True,
      with_intercept=False)
```

ARIMA model:

Building a forecast model for the actual close price value(Using ARIMA model)

```
from statsmodels.tsa.arima.model import ARIMA
ARIMAmodel = ARIMA(y_train, order = (0,1,0))
ARIMAmodel = ARIMAmodel.fit()
```

Predicting for the built model

```
y_pred = ARIMAmodel.get_forecast(len(y_test.index))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = ARIMAmodel.predict(start = y_pred_df.index[0], end = y_pred_df.index[-1])
y_pred_df.index = y_test.index
y_pred_out = y_pred_df["Predictions"]
y_pred_out = y_pred_df["Predictions"]
val={
    "Predicted": y_pred_out.tolist(),
    "Actual": reduce(lambda z, y :z + y, y_test.values.tolist())
}
ans=pd.DataFrame(val)
ans
```

	Predicted	Actual
0	39666.753906	39338.785156
1	39666.753906	41143.929688
2	39666.753906	40951.378906
3	39666.753906	41801.156250
4	39666.753906	42190.652344
5	39666.753906	41247.824219
6	39666.753906	41077.996094
7	39666.753906	42358.808594
8	39666.753906	42892.957031
9	39666.753906	43960.933594
10	39666.753906	44348.730469

Building ARIMA model using the log transformed values:

Building the ARIMA model for the log transformed value(Finding the best p and q value)

```
autoarima_model = pmd.auto_arima(y_detrend.dropna(), start_p=1, start_q=1,test="adf",trace=True)
autoarima_model
```

```
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=-1177.114, Time=0.11 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=-1180.174, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=-1179.179, Time=0.04 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=-1179.251, Time=0.06 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=-1181.934, Time=0.02 sec

Best model:  ARIMA(0,0,0)(0,0,0)[0]
Total fit time: 0.280 seconds

ARIMA(order=(0, 0, 0), scoring_args={}, suppress_warnings=True,
      with_intercept=False)
```

Building the model

```
ARIMAmodel = ARIMA(y_detrend, order = (0,0,0))
ARIMAmodel = ARIMAmodel.fit()
```

Predicting for the above built model

```
y_pred = ARIMAmodel.get_forecast(len(y_detest.index))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = ARIMAmodel.predict(start = y_pred_df.index[0], end = y_pred_df.index[-1])
y_pred_df.index = y_detest.index
y_pred_out = y_pred_df["Predictions"]
y_pred_out = y_pred_df["Predictions"]
val={
    "Predicted": y_pred_out.tolist(),
    "Actual": reduce(lambda z, y :z + y, y_detest.values.tolist())
}
ans=pd.DataFrame(val)
ans
```

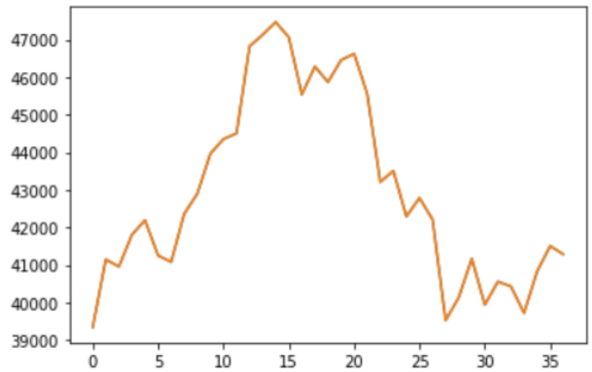
	Predicted	Actual
0	-0.001082	NaN
1	-0.001082	0.044865
2	-0.001082	-0.004691
3	-0.001082	0.020539
4	-0.001082	0.009275
5	-0.001082	-0.022600
6	-0.001082	-0.004126
7	-0.001082	0.030704
8	-0.001082	0.012531
9	-0.001082	0.024594
10	-0.001082	0.008783

RESULT ANALYSIS

Linear Regression:

Plotting the predicted and actual data

```
plt.plot(ans["Predicted"])
plt.plot(ans["Actual"])
plt.show()
```



Mean absolute error is low

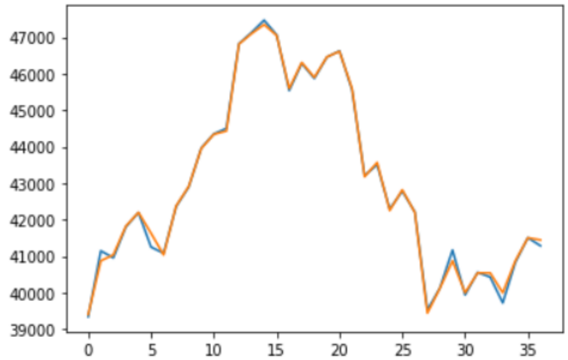
```
mean_absolute_error(ans["Predicted"].values.tolist(), ans["Actual"].values.tolist())
```

```
3.3948634990264436e-08
```

Decision tree algorithm:

Plotting the predicted and actual data

```
plt.plot(ans["Actual"])
plt.plot(ans["Predicted"])
plt.show()
```



There is a minimal mean absolute error

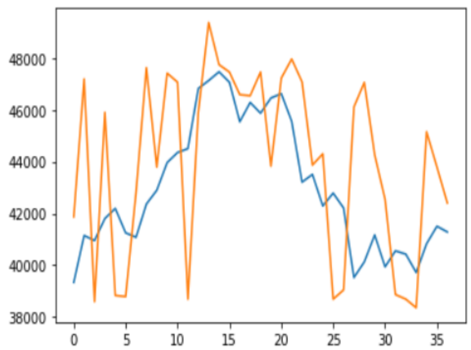
```
mean_absolute_error(ans["Predicted"].values.tolist(), ans["Actual"].values.tolist())
```

```
69.30901616216265
```

K Nearest Neighbour algorithm:

Plotting the predicted and the actual graph

```
plt.plot(ans["Actual"])
plt.plot(ans["Predicted"])
plt.show()
```



Mean absolute error is high but reasonable since the close price values are arguably large

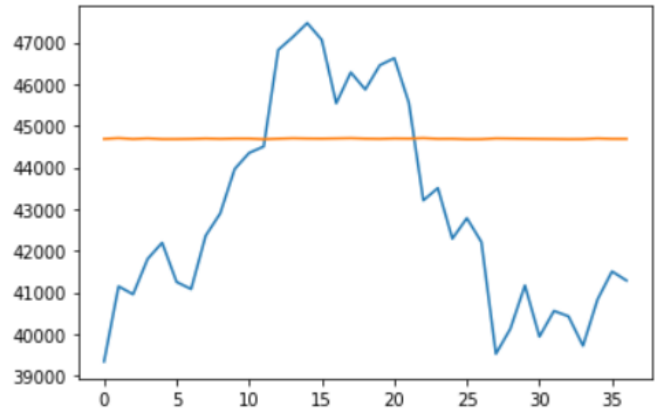
```
mean_absolute_error(ans["Predicted"].values.tolist(), ans["Actual"].values.tolist())
```

```
2657.564931725223
```

Support Vector Machine:

Plotting the predicted and the actual graph

```
plt.plot(ans["Actual"])
plt.plot(ans["Predicted"])
plt.show()
```



A high mean absolute error is obtained

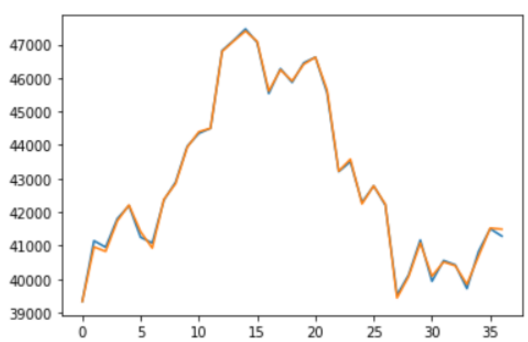
```
mean_absolute_error(ans["Predicted"].values.tolist(), ans["Actual"].values.tolist())
```

2690.701666260613

Random Forest algorithm:

Plotting the predicted and the actual graph

```
: plt.plot(ans["Actual"])
plt.plot(ans["Predicted"])
plt.show()
```



A minimal yet a decent mean absolute error is obtained

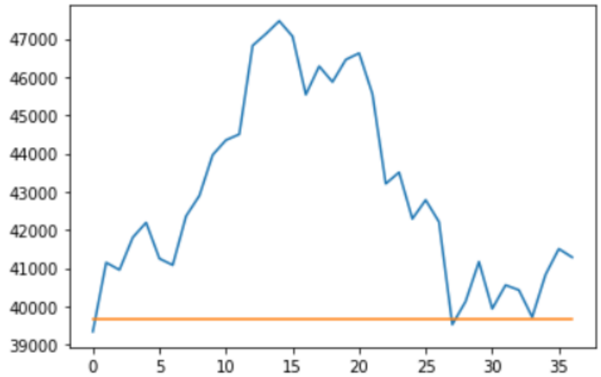
```
: mean_absolute_error(ans["Predicted"].values.tolist(), ans["Actual"].values.tolist())
```

: 63.884005436481246

ARIMA model:

Plotting the graph for predicted and actual

```
plt.plot(ans["Actual"])
plt.plot(ans["Predicted"])
plt.show()
```



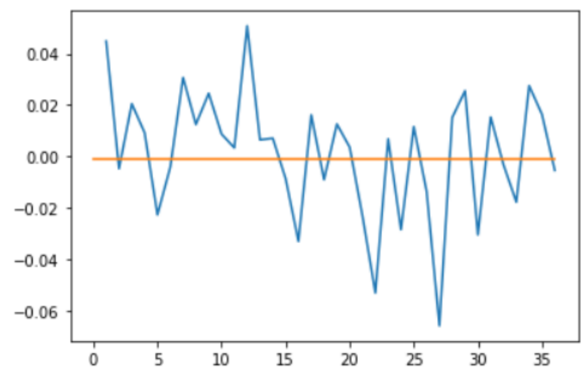
A pretty high mean absolute error is obtained

```
mean_absolute_error(ans["Predicted"].values.tolist(), ans["Actual"].values.tolist())
```

```
3322.5258660270288
```

ARIMA model with log-transformed values:

```
: plt.plot(ans["Actual"])
plt.plot(ans["Predicted"])
plt.show()
```



An extremely low mean absolute error is obtained

```
: ans=ans.iloc[1: , :]
mean_absolute_error(ans["Predicted"].values.tolist(), ans["Actual"].values.tolist())
```

```
: 0.019412426429847418
```

CONTRIBUTION OF THE STUDY

- This study will prove to be a step ahead in predictions models that can be used to predict price models/prices of cryptocurrencies. Although this category of assets are generally regarded as “unpredictable”, our study will lessen the gap between certainty and being unpredictable.
- The code used can be further developed on used for other class of assets such as NFTs (Non- Fungible Tokens) or ICOs (Initial Coin Offering).
- Investing in bitcoin (cryptocurrency) is a form of passive income for many due to its high-risk high-reward nature. Making it automated can make it more accessible.

IMPLICATIONS OF THE STUDY

- Predictions can help financial institutes better manage their clients portfolios and make informed decisions to maximize their profits.
- The advantage of predicting the bit coins is it helps you to invest wisely to make good profits.
- The proposed method predicted fluctuations in the price of cryptocurrencies at low cost.
- Cryptocurrency markets are highly volatile and your investments are at risk is what the most forwarded statement about cryptocurrencies in web
- So our model can effectively reduce the volatility of crypto and brings a security to users investments.

Future scope of the study

The price predictor model has given predicted outputs that come under the standard error rates. This inference leads us to the conclusion that the model can be used to predict other highly volatile asset classes such as other crypto-currencies and DeFi (Decentralised Finance) assets such as NFTs, Liquidity Pool Tokens, etc. The model can also be used in automated investing. Robo-investing which has gained immense interest over the past few years can use similar models for their functioning. Although the asset class is deemed to be risky, investors willing to take the risk and automate their investments can choose to use such features.