Prediction of Bitcoin Price Using Machine Learning

Shwedha Srinivasan
Computer Science
Illinois Institute of Technology
ssrinivasan2@hawk.iit.edu

Narendra Kumar Srivastava
Computer Science
Illinois Institute of Technology
nsrivastava1@hawk.iit.edu

Varun Anavatti
Computer Science
Illinois Institute of Technology
vanavatti@hawk.iit.edu

***Abstract*— The volatility of the stock market and the unique transaction structure of cryptocurrencies, particularly Bitcoin, make accurate price forecasting challenging. However, the availability of vast amounts of data and sophisticated machine learning algorithms offer an opportunity to develop models that can generate reasonably accurate predictions. This report highlights the importance of Bitcoin as the most traded cryptocurrency and discusses how machine learning models can be trained using historical data to forecast future Bitcoin prices by identifying patterns. Despite the significant fluctuations in Bitcoin prices, ranging from a high of $20,000 to a low of $900 in the past two years, our machine learning models can provide reliable predictions.**

I. INTRODUCTION

Because of its volatility, the stock market is a difficult dataset to forecast. Making accurate predictions could result in substantial financial rewards for specific people. Due to their unique transaction structure, cryptocurrencies, especially Bitcoin, have recently grown in popularity among investors all over the world. However, it might be challenging to forecast bitcoin prices due to rumors and social media activity. Bitcoin continues to be the biggest and most traded cryptocurrency, with a market share of over 55%, followed by Ethereum at 8.57%, despite the development of thousands of new cryptocurrencies. Fortunately, there is much data available and numerous machine learning methods that enable the development of models that can generate accurate predictions. Machine learning models can be trained to predict future Bitcoin prices by analyzing historical data and identifying patterns. Due to the significant fluctuations in Bitcoin prices, ranging from a high of $20,000 to a low of $900 in the past two years, it has become an attractive and important stock to predict. The availability of vast amounts of data and sophisticated machine learning algorithms have made it possible to create models that can provide reasonably accurate predictions. By utilizing historical data and recognizing patterns, machine learning models can effectively forecast future Bitcoin prices.

II. PROBLEM STATEMENT

A. *The problem*

The main problem addressed in Bitcoin price prediction using machine learning is how to accurately predict the future price of Bitcoin. It is a highly volatile and unpredictable cryptocurrency, and its price movements are influenced by a wide range of factors, including supply and demand, market sentiment, news events, and regulatory changes. The challenge in building a Bitcoin price prediction model is to identify the most relevant factors that impact Bitcoin prices and to develop a model that can accurately forecast future price movements. This involves collecting and preprocessing a large amount of historical data, selecting appropriate features and algorithms, training and validating the model. Accurate predictions can help investors and traders make informed decisions about buying and selling Bitcoin, which can potentially lead to higher profits and reduced risks. Additionally, Bitcoin price prediction models can provide valuable insights into market trends and help to inform broader discussions about the future of cryptocurrencies. .

B. Related Work

The Bitcoin prediction using machine learning is an active research area that has seen many studies and approaches in recent years. Some of the techniques and models used for bitcoin prediction include:

I. Neural networks like CNN, RNN, GAN.

II. Time-series forecasting models like ARIMA, LSTM, And Prophet.

III. Regression models like linear, lasso, ridge, bayesian regression models have been already used.

IV classification models like decision, random forests, xgboost, SVM

III. PROPOSED MODEL

The proposed model for forecasting Bitcoin prices includes the use of different techniques.

I. Time-series forecasting models like AR, MA, ARIMA, LSTM.

II. Regression models like Linear, Lasso, Ridge, Multi-Perceptron .

III. Classification models like Random forests.

These models will be trained and tested using Bitcoin price data, and the best performing model will be used to predict future prices.

The ARIMA model will be used to identify the autoregressive and moving average components of the data and integrate them. This will allow the model to predict future Bitcoin prices based on past values.

The LSTM model will be used to learn long-term dependencies and remember past values of the time series. This makes it useful for forecasting Bitcoin prices, which can be affected by both short-term and long-term trends.

The Linear Regression model will be used to model the relationship between historical Bitcoin prices and predict future prices using a linear equation.

The Random Forest model will combine multiple decision trees to make predictions, which can help to reduce overfitting and improve the accuracy of the predictions.

After training and testing each model, the best performing model will be selected based on its accuracy in predicting future Bitcoin prices. The selected model will then be used to forecast future Bitcoin prices, based on the available data.

IV. Data Preprocessing and Visualization

Data used in this project is a real time data that we get from the cmcscraper. cmcscraper is a Python package that is primarily used for scraping cryptocurrency market data from the CoinMarketCap (CMC) API. The package provides a simple and easy-to-use interface for querying various endpoints of the CMC API, such as cryptocurrency prices, market capitalization, trading volume, and more. The obtained data from cmcscraper is preprocessed by removing null values and noise. The data is visualized by plotting a graph for time against the price. The below visualization shows the opening and closing price of the bitcoin on that day.

```
from cryptocmd import CmcScraper
#IMPORTING THE DATA OF TEN YEARS
scraper = CmcScraper('BTC', '01-04-2013', '01-05-2023')
#CONVERTING INTO DATAFRAMES
data = scraper.get_dataframe()
data.sort_values(by='Date', ascending=True, inplace=True)

pd.set_option('display.max_columns', None)
display(data)
pd.reset_option('display.max_columns')
```

- The cmcscraper package in Python is a third-party library that allows users to easily retrieve cryptocurrency market data from the CoinMarketCap (CMC) API. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- It provides a simple interface for querying various endpoints of the CMC API, including cryptocurrency prices, market capitalization, trading volume, and more. Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersted.

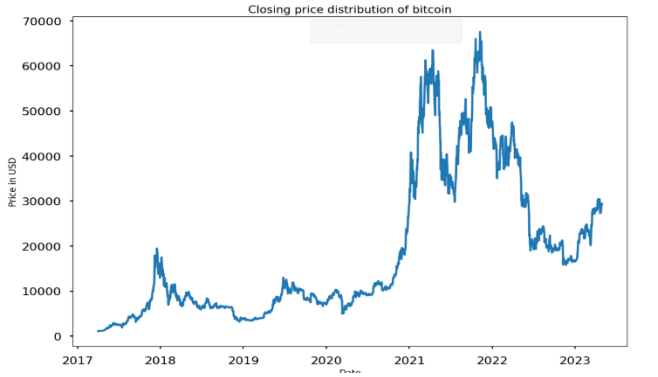
A. *Description of the data*

- The Gather relevant data on Bitcoin price and related variables such as trading volume, market capitalization, and sentiment indicators.
- Preprocessing the data to remove noise and missing values, and prepare it for use in machine learning models.

Below is a brief description of the data being used in the project.

1. Date: Date of the record.
2. Open: The opening price, the price at which Bitcoin trades at the beginning of the day. (USD).
3. High: The maximum price of the day, the highest price reached by Bitcoin on that day, (USD).
4. Low: The minimum price of the day, the lowest price reached by the Bitcoin on that day, (USD).
5. Close: The closing price, the price at which Bitcoin trades at the end of the day, (USD).

B. Data Visualization.



V. Model Training

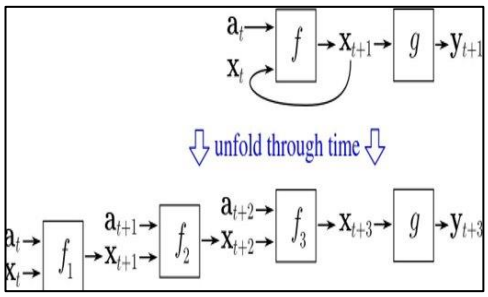
The preprocessed data is then trained using different Machine Learning Models and the results and snippets of the accuracy are given below for each model used in the project. .

A. *Deep RNN (Recurrent Neural Networks) with LSTM*

We use RNN because RNNs allow using the output from the model as a new input for the same model and The process can be repeated indefinitely.

One serious limitation of RNNs is the inability of capturing long-term dependencies in a sequence (e.g., Is there a dependency between today`s price and that 2 weeks ago?).

One way to handle the situation is by using a Long short-term memory (LSTM) variant of RNN. The default LSTM behavior is remembering information for prolonged periods of time.



1) ***2 Layers with 230 LSTM***.

2) *Our model seems to do well on the test data.*.

B. *ARIMA*

AR*: Autoregression dependent relationship between some number of lagged observations.*

I: *Integrated*. To make the data stationary

MA: *Moving Average*. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

ARIMA, short for "AutoRegressive Integrated Moving Average," is a statistical model used for time series forecasting. ARIMA is a powerful tool for analyzing time series data because it can capture both the short-term and long-term patterns in the data, as well as any trend or seasonality. It also captures both linear and non-linear relationships in the data, making it a powerful tool for modeling complex time series.

The model is also relatively easy to implement. ARIMA models are based on the idea of decomposing a time series into three components: autoregression (AR), integration (I), and moving average (MA). By combining these three components, an ARIMA model can accurately forecast the future values of a time series.

ARIMA is particularly useful for analyzing and forecasting stationary time series data (where the mean and variance of the data do not change over time) that exhibit patterns such as trend and seasonality.

For example, it can be used to forecast sales, stock prices, weather patterns, and many other types of time series data. It can also be used in conjunction with other models, such as exponential smoothing, to improve the accuracy of forecasts.

1. Terminology

Stationarity: A time series is said to be stationary if its statistical properties, such as the mean and variance, do not change over time.

Stationarity is an important assumption in many time series models, including ARIMA models, because it allows the model to make accurate forecasts by assuming that the future will be like the past.

A trend : is a long-term upward or downward movement in the data.

Seasonality: is a pattern that repeats over a fixed period, such as daily, weekly, or yearly. Identifying and removing trends and seasonality from the data is an important step in time series.

2. AUGMENTED DICKY FULLER TEST:

The Augmented Dickey-Fuller (ADF) test is a statistical test used to determine whether a time series has a unit root, which is a characteristic of non-stationary time series. A nonstationary time series is one where the statistical properties of the series, such as the mean or variance, change over time.
The ADF test is an extension of the Dickey-Fuller test, which tests for the presence of a unit root in a time series. The ADF test improves upon the Dickey-Fuller test by including additional terms in the regression equation to account for the possibility of serial correlation in the errors and higher-order differences in the time series.

- *The Augmented Dicky Fuller test is a type of statistical test called a unit root test.*
- *The intuition behind a unit root test is that it determines how strongly a time series is defined by a trend.*
- *There are numbers of unit root tests available and ADF is one of the most widely used.*

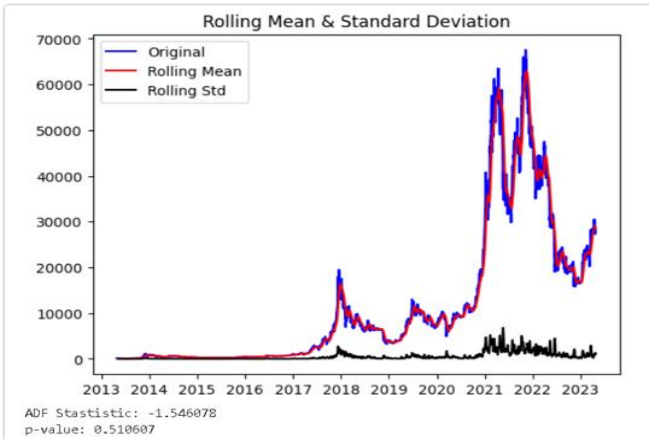
Null Hypothesis (H0): Null hypothesis of the test is that the time series can be represented by a unit root that is not stationary.

Alternative Hypothesis (H1): Alternative Hypothesis of the test is that the time series is stationary.

Interpretation of p value:

1. **p value** > **0.05:** Accepts the Null Hypothesis (H0), the data has a unit root and is non-stationary.

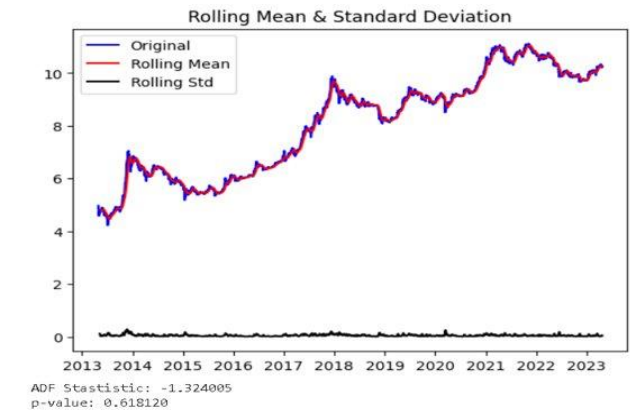
2. **p value** < = **0.05:** Rejects the Null Hypothesis (H0), the data is stationary.



Since the p value is greater than 0.05.the time series is non stationary. there is some work that needs to be done here. So now we use transformations to make the series stationary.

3. *Log Transforming the series*

Log transformation is used to unscrew highly skewed data. Thus, helping in forecasting process.

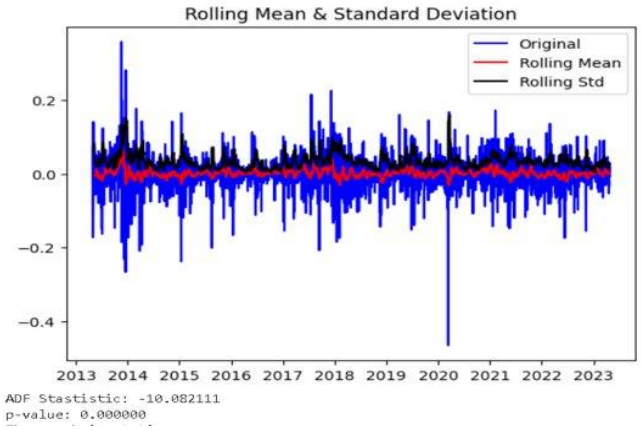


The series is still non-stationary as p value is still greater than 0.05 so we need to make further transformations. So, let's go ahead with differencing.

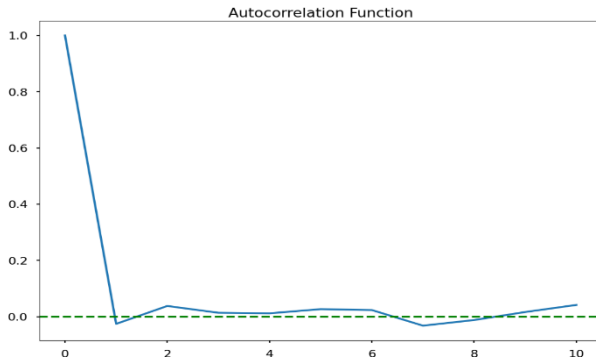
4. **Differencing**

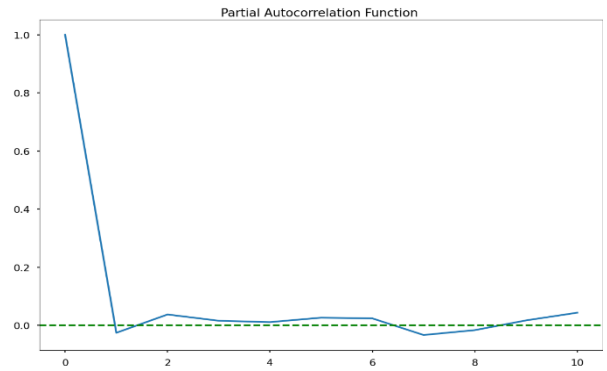
: Differencing is the process of transforming a nonstationary time series into a stationary time series by subtracting the value of the observation at the current time step from the value of the observation at the previous time step.

Differencing is often used in time series analysis to remove trends and seasonality from the data. It is also an important step in ARIMA modeling, where it's used to make the time series stationary.



In case of differencing to make the time series stationary the current value is subtracted with the previous values. Due to this the mean is stabilized and hence the chances of stationarity of time series are increased.





C. Auto Regressive model

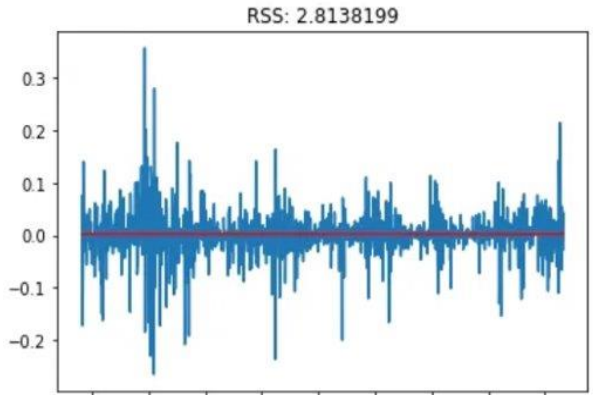
Auto regressive model is a time series forecasting model where current values are dependent on past values.

SARIMAX Results

Dep. Variable:	Close	No. Observations:	2219
Model:	ARIMA(1, 1, 0)	Log Likelihood	3547.393
Date:	Sun, 30 Apr 2023	AIC	-7090.786
Time:	22:15:21	BIC	-7079.377
Sample:	04-02-2017 - 04-29-2023	HQIC	-7086.619
Covariance Type:	opg		

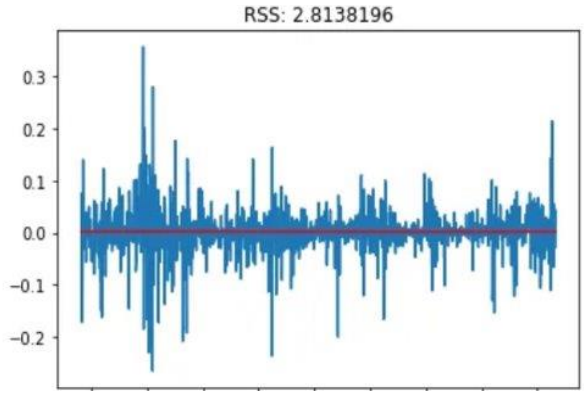
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.5311	0.010	-51.486	0.000	-0.551	-0.511
sigma2	0.0024	3.48e-05	68.619	0.000	0.002	0.002

Ljung-Box (L1) (Q):	69.47	Jarque-Bera (JB):	4474.19
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	0.64	Skew:	-0.04
Prob(H) (two-sided):	0.00	Kurtosis:	9.96



D. **Moving Average Model**

In moving average model, the series is dependent on past error terms.



SARIMAX Results

Dep. Variable:	Close	No. Observations:	2219
Model:	ARIMA(0, 1, 1)	Log Likelihood	3973.689
Date:	Sun, 30 Apr 2023	AIC	-7943.379
Time:	22:15:22	BIC	-7931.970
Sample:	04-02-2017 - 04-29-2023	HQIC	-7939.212
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.9935	0.002	-415.662	0.000	-0.998	-0.989
sigma2	0.0016	1.94e-05	83.833	0.000	0.002	0.002

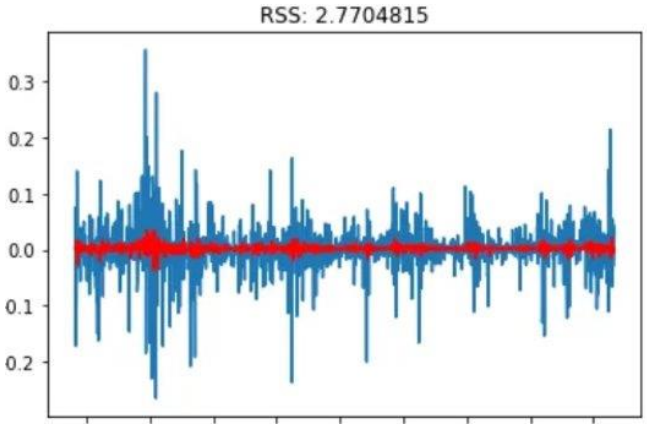
Ljung-Box (L1) (Q):	2.22	Jarque-Bera (JB):	10903.49
Prob(Q):	0.14	Prob(JB):	0.00
Heteroskedasticity (H):	0.63	Skew:	-0.76
Prob(H) (two-sided):	0.00	Kurtosis:	13.75

E. ARIMA

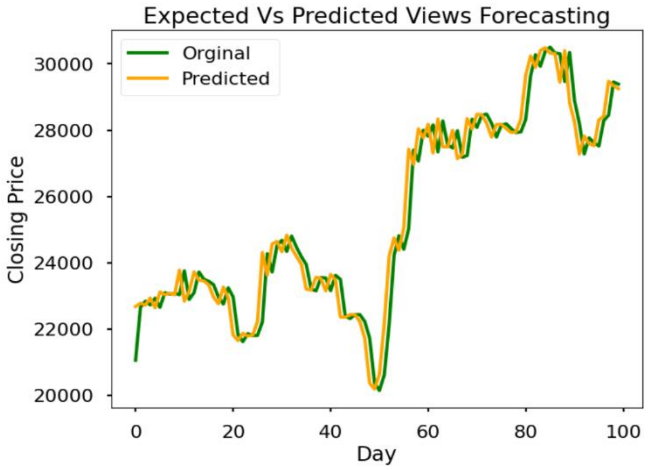
It is a combination of both AR and MA models. It makes the time series stationary by itself through the process of differencing. Therefore, differencing need not be done explicitly for ARIMA model.

```
predicted = 27942.779587,   expected = 28333.050048,   error = 1.377439  %
predicted = 28324.888582,   expected = 29652.980097,   error = 4.478779  %
predicted = 29637.339511,   expected = 30235.058939,   error = 1.976908  %
predicted = 30274.306178,   expected = 29892.988700,   error = 1.275608  %
predicted = 29922.900659,   expected = 30399.066385,   error = 1.566383  %
predicted = 30374.185544,   expected = 30485.700029,   error = 0.365793  %
predicted = 30503.234773,   expected = 30318.496953,   error = 0.609324  %
predicted = 30325.605505,   expected = 30315.355456,   error = 0.033811  %
predicted = 30309.074897,   expected = 29445.044905,   error = 2.934382  %
predicted = 29464.581849,   expected = 30397.552736,   error = 3.069230  %
predicted = 30341.419049,   expected = 28822.680473,   error = 5.269248  %
predicted = 28894.334954,   expected = 28245.987310,   error = 2.295362  %
predicted = 28201.673055,   expected = 27276.909688,   error = 3.390279  %
predicted = 27277.462916,   expected = 27817.500917,   error = 1.941361  %
predicted = 27767.224774,   expected = 27591.384632,   error = 0.637301  %
predicted = 27617.652864,   expected = 27525.339500,   error = 0.335376  %
predicted = 27518.164647,   expected = 28307.598091,   error = 2.788769  %
predicted = 28286.485600,   expected = 28422.700878,   error = 0.479248  %
predicted = 28450.865756,   expected = 29473.786184,   error = 3.470611  %
predicted = 29453.593978,   expected = 29340.262498,   error = 0.386266  %
predicted = 29384.931858,   expected = 29248.488373,   error = 0.466498  %

Means Error in Predicting Test Case Articles : 1.956850  %
```

Thus, we see that the RSS (Residual Sum of Squares) error is minimum for ARIMA model. Therefore, ARIMA model is the best among the three models because of use of dependence on both lagged values and error terms.



For every value in the test, we apply an ARIMA model and then the error is calculated and then after iterating over all values in the test set the mean error between predicted and expected value is calculated.



SARIMAX Results

Dep. Variable:	Close	No. Observations:	2219
Model:	ARIMA(8, 1, 0)	Log Likelihood	3870.541
Date:	Sun, 30 Apr 2023	AIC	-7723.081
Time:	22:15:24	BIC	-7671.742
Sample:	04-02-2017	HQIC	-7704.329
	- 04-29-2023		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.9179	0.014	-66.718	0.000	-0.945	-0.891
ar.L2	-0.7703	0.021	-37.142	0.000	-0.811	-0.730
ar.L3	-0.6546	0.024	-27.731	0.000	-0.701	-0.608
ar.L4	-0.5416	0.027	-20.412	0.000	-0.594	-0.490
ar.L5	-0.4096	0.028	-14.823	0.000	-0.464	-0.355
ar.L6	-0.2843	0.026	-10.756	0.000	-0.336	-0.232
ar.L7	-0.2175	0.025	-8.707	0.000	-0.266	-0.169
ar.L8	-0.1247	0.019	-6.573	0.000	-0.162	-0.088
sigma2	0.0018	2.52e-05	70.765	0.000	0.002	0.002

Ljung-Box (L1) (Q):	0.65	Jarque-Bera (JB):	6356.91
Prob(Q):	0.42	Prob(JB):	0.00
Heteroskedasticity (H):	0.62	Skew:	-0.39
Prob(H) (two-sided):	0.00	Kurtosis:	11.26

F. REGRESSION MODELS

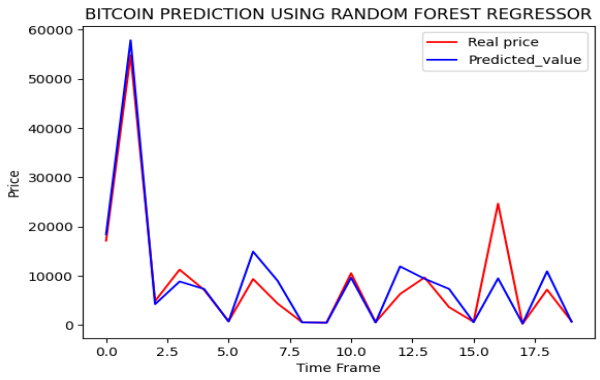
Various regressors models are used, but all of them are more or less of the same accuracy.

- For the predicting the bitcoin prices for the next 100 days, we put it as a separate dataset.
- Today, the bitcoin price is approximately 28,400$, the regression models have predicted around 21,000$.
- The next upcoming graphs explain the various comparisons of regressors, on the basis of metrics and graphs.

G. RANDOM FOREST

Using a Random Forest Regressor to predict the prices of Bitcoin can be an effective approach since it is a powerful machine learning algorithm that can handle non-linear relationships and interactions between features.

Import the required libraries and initialize a Random Forest Regressor model with hyperparameters such as the number of trees, maximum depth, and minimum samples required to split a node. Train the model on the training data and tune the hyperparameters using techniques such as cross-validation and grid search.

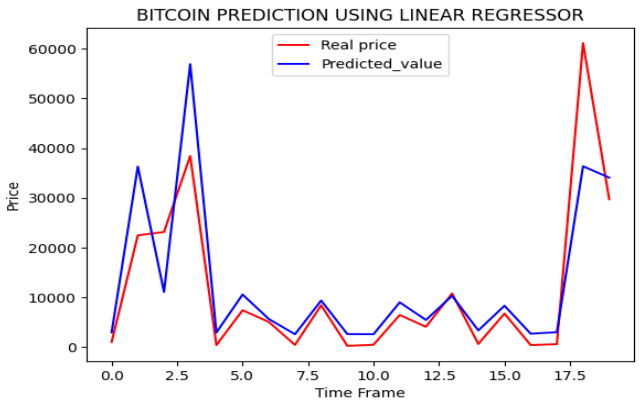


H. LINEAR REGRESSOR

The methodology to predict Bitcoin prices using a Linear Regressor model involves the following steps: Firstly, collect historical data of Bitcoin prices and other relevant features, preprocess the data, and split it into training and test sets. Next, import the required libraries, initialize a Linear Regressor model, and train it on the training data using the fit() method.

```
LINEAR REGRESSOR

R^2
9.8131236428093442

MSE
39794158.270627104

MAE
4236.65532944987

PREDICTION FOR THE NEXT 100 DAYS
[20365.14366303 20358.24793191 19956.4665675  20294.73823099
 21645.87704068 21731.78007305 21683.1570291  21865.04950515
 21611.81089023 22020.94550488 21948.63407648 21987.68731007
 21947.19959235 22579.08141985 21784.91007713 22039.15399803
 22535.90957108 22321.81993949 22302.6556536  22202.81386026
 21883.09679224 21716.89272659 22145.39792715 21869.27116899
 20917.07605242 20774.41479873 20961.12999699 20890.86853669
 20907.77970494 21258.53625394 23032.31322644 22450.66850827
 23251.38451433 23315.70080994 23049.14157724 23475.37241747
 23141.53680837 22931.17629524 22726.05207423 22089.16526968
 22069.82842627 22397.75227863 22365.16619103 22046.01226481
 22470.28035131 22324.8764388  21379.11625304 21371.18729848
 21441.01874685 21436.12614721 21257.65654359 20831.26987993
 19679.6028329  19530.20922344 19908.5566593  21210.2157655
 22938.5619611  23404.76858835 23090.2085722  23665.44631832
 25680.68360276 25291.3852693  26203.15599272 25972.45978465
 26319.71315128 25581.67523922 26454.13013861 25739.62813567
 25740.83647081 26165.46672314 25439.27566517 25548.26960044
 26466.42790658 26198.81146702 26576.95107934 26563.74299769
 26339.67866402 25991.99437552 26313.14594275 26321.55465213
 26207.80054384 26107.27384671 26125.91656605 26453.34596489
 27575.15708742 28069.86703002 27779.14086714 28209.2572786
 28282.88738135 28140.7811609  28138.11119652 27398.43256222
 28207.97082454 26869.48381797 26379.35117485 25555.7302137
 26015.17975396 25823.00308203 25766.8711953  26431.71429704]
```

Then, evaluate the performance of the model by predicting Bitcoin prices on the test data and computing metrics such as MSE, RMSE, and R-squared using the score() method. If the model performance is not satisfactory, improve it by adding more features, tuning hyperparameters, or using a different algorithm. Finally, use the trained model to predict the prices of Bitcoin on new data using the predict() method.

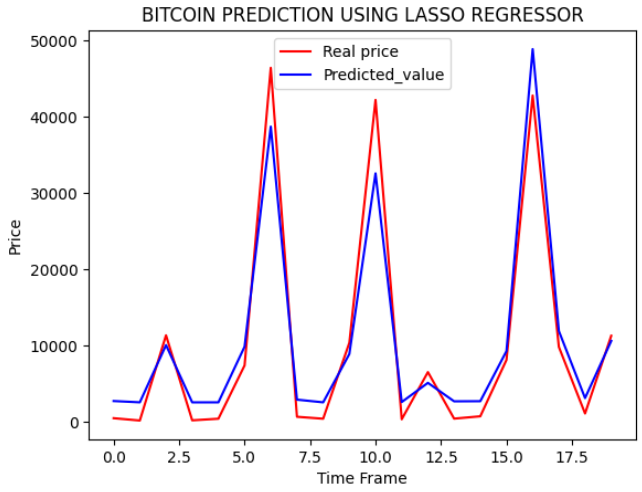


I. LASSO REGRESSOR

The methodology to predict Bitcoin prices using Lasso Regressor involves the following steps: Firstly, collect historical data of Bitcoin prices and other relevant features, preprocess the data, and split it into training and test sets. Next, import the required libraries, initialize a Lasso Regressor model, and train it on the training data using the fit() method.

Then, evaluate the performance of the model by predicting Bitcoin prices on the test data and computing metrics such as MSE, RMSE, and R-squared using the score() method. If the model performance is not satisfactory, improve it by adding more features, tuning hyperparameters such as the

regularization parameter alpha, or using a different algorithm. The Lasso Regressor model uses L1 regularization to impose sparsity on the coefficients, which can help in feature selection and reduce overfitting.



Therefore, it is important to tune the regularization parameter to find the optimal balance between model complexity and performance. Finally, use the trained model to predict the prices of Bitcoin on new data using the predict() method.

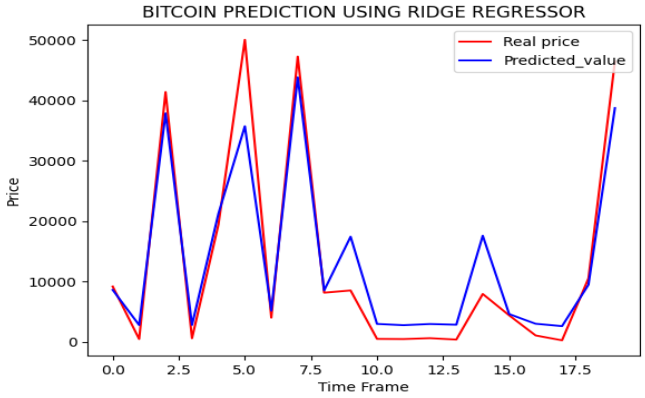
```
 LASSO REGRESSOR

 R^2
0.8284245411347072

 MSE
33135063.40278955

 MAE
4150.534760352805

 PREDICTION FOR THE NEXT 100 DAYS
[20355.23796956 20348.35019601 19947.03248183 20284.91378442
 21634.49339832 21720.29729976 21671.73036614 21853.41294075
 21600.46655974 22009.12903862 21936.90105663 21975.90922331
 21935.46822787 22566.62087234 21773.36599254 22027.31651942
 22523.4988433  22309.65626823 22290.51409767 22190.7875204
 21871.43940152 21705.4271331  22133.43784449 21857.62973285
 20906.53343648 20764.03681193 20950.53654339 20880.35616386
 20897.24781686 21247.59959758 23019.3296556  22438.35614752
 23238.14813821 23302.39021369 23036.13858671 23461.87756235
 23128.427195   22918.30943504 22713.42192459 22077.27007874
 22057.95554978 22385.50098248 22352.95249878 22034.16687185
 22457.94535876 22312.70924038 21368.040449   21360.12064434
 21429.87150817 21424.98455452 21246.7209024  20820.82628314
 19670.48824421 19521.26703294 19899.17786101 21199.33487032
 22925.68657795 23391.3552091  23077.15819092 23651.73212051
 25664.64384856 25275.79476008 26186.5133125  25956.08332471
 26302.93596568 25565.74973933 26437.19783765 25723.52036028
 25724.72730102 26148.86753582 25423.51449244 25532.3826502
 26449.48141416 26182.17380032 26559.87704468 26546.68420498
 26322.8784385  25975.59537293 26296.37633562 26304.77534147
 26191.15250388 26090.74181318 26109.36301908 26436.41456885
 27556.93113662 28051.07019068 27760.67952183 28190.29958482
 28263.84471943 28121.90248762 28119.23560434 27380.41054912
 28189.01461531 26852.07220452 26362.50516779 25539.83465401
 25998.75399576 25806.79909309 25750.7319818  26414.80786365]
```

J. RIDGE REGRESSOR

Ridge regression is a machine learning algorithm that can be used to predict Bitcoin prices. To use ridge regression, you will need to gather historical data on Bitcoin prices, clean the data, choose features, split the data into training and test sets, train the model, evaluate the model, and make predictions.

```
 RIDGE REGRESSOR

 R^2
0.8137479364898424

 MSE
26976648.374747522

 MAE
3664.589016139368

 PREDICTION FOR THE NEXT 100 DAYS
[20376.05055967 20369.15225118 19967.22071591 20305.61881271
 21657.26262759 21743.1977673  21694.55654988 21876.51701054
 21623.18374448 22032.47127836 21960.13282266 21999.20065288
 21958.69780237 22590.81580368 21796.34762939 22050.68657717
 22547.62781889 22333.4581686  22314.28671982 22214.40760934
 21894.57104303 21728.30485651 22156.9702163  21880.74025227
 20928.18924076 20785.47466562 20972.25965104 20901.97192963
 20918.88941865 21269.77706734 23044.21701139 22462.35489616
 23263.37017994 23327.71051457 23061.05165199 23487.44180138
 23153.48141701 22943.04227898 22737.8413902  22100.71654116
 22081.37247036 22409.41888841 22376.82062134 22057.54740732
 22481.97406937 22336.51581031 21390.40213474 21382.47021663
 21452.32776538 21447.43333707 21268.89702819 20842.35099712
 19690.2535     19540.80405278 19919.29290078 21221.4385185
 22950.43070533 23416.81158319 23102.13399626 23677.58674475
 25693.57724958 25304.13341095 26216.24492019 25985.4624865
 26332.84564346 25594.53188042 26467.31287084 25752.54381375
 25753.75260052 26178.54156377 25452.07908272 25561.11375586
 26479.61523526 26211.89877067 26590.17971745 26576.96669912
 26352.81861857 26005.00437867 26326.27598035 26334.6878326
 26220.89120728 26120.32693702 26138.97662432 26466.52840402
 27588.75881764 28083.65366432 27792.81883887 28223.09601177
 28296.7536347  28154.59430025 28151.92333794 27411.96823941
 28221.80907688 26882.82179389 26392.5059575  25568.57715761
 26028.19842294 25835.94992255 25779.79705582 26444.88865107]
```

It is important to note that predicting Bitcoin prices is a challenging task, and it is important to use caution when making predictions.



K. MULTI PERCEPTRON REGRESSOR

Import the required libraries and initialize an MLP regressor model with appropriate parameters such as the number of hidden layers, number of neurons per layer, activation function, and learning rate.

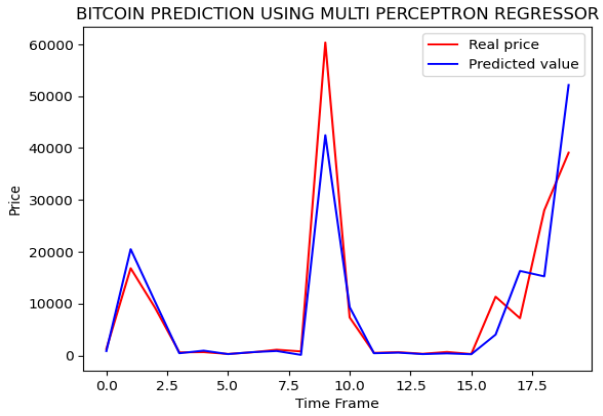
Train the model: Train the MLP regressor model on the training data using the fit() method.

Evaluate the model: Use the trained model to predict Bitcoin prices on the test data and evaluate its performance using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. You can use the score() method to compute the R-squared value.

```
 MULTI PERCEPTRON REGRESSOR

 R^2
0.7788743009768363

 MSE
70068608.99570094

 MAE
4502.898763424813

 PREDICTION FOR THE NEXT 100 DAYS
[19376.67760843 19369.2520013  18936.59728899 19300.86214101
 20755.8240302  20848.3279516  20795.96865634 20991.83796206
 20719.14019708 21159.71319194 21081.84526919 21123.89939872
 21080.30055762 21760.73692055 20905.54052707 21179.32084692
 21714.24769666 21483.70716848 21463.07027654 21355.55652354
 21011.27202371 20832.29664411 21293.72868326 20996.38402346
 19971.02112533 19817.39761478 20018.4602263  19942.79979357
 19961.01043569 20338.71952636 22248.79558695 21622.45661132
 22484.7005677  22553.95900235 22266.91704782 22725.89996214
 22366.4120361  22139.88717618 21919.00097116 21233.17504291
 21212.35233394 21565.47423697 21530.38419645 21186.7061109
 21643.57545122 21486.99853275 20468.565033   20460.02680834
 20535.22418652 20529.95563382 20337.77221804 19878.62150662
 18638.45902262 18477.58583424 18885.00592765 20286.68603532
 22147.84036521 22649.87085258 22311.13967849 22930.57936287
 25100.66982694 24681.45735621 25663.28960375 25414.86643041
 25788.8031501  24994.05354546 25933.54889527 25164.14372639
 25165.44491141 25622.70424681 24840.71182231 24958.08097849
 25946.79163804 25658.61123963 26065.8075401  26051.58453408
 25810.30283609 25435.90208199 25781.73130962 25790.78615402
 25668.29104759 25560.03976199 25580.1150096  25932.70446472
 27140.71686023 27673.4408909  27360.37499112 27823.5420485
 27902.82997408 27749.80414616 27746.92901861 26950.41261672
 27822.15674181 26380.81883502 25853.02382885 24966.11487396
 25460.86905191 25253.92530072 25193.48017446 25909.41059433]
```

If the model performance is not satisfactory, try improving it by adding more hidden layers, increasing the number of neurons per layer, changing the activation function or learning rate, or using a different algorithm.

Once the model is trained and evaluated, use it to predict the prices of Bitcoin on new data using the predict() method.



Note that MLP regressor model can handle non-linear relationships between the target variable and the features, and can capture complex patterns in the data. However, it is important to avoid overfitting by tuning hyperparameters such as the learning rate, regularization parameter, and number of epochs.

CONCLUSION

The trials' findings show that the ARIMA model performed better than the ALL models, achieving the highest R-squared and lowest Mean Squared Error (MSE) values on the test data. This shows that the ARIMA model can handle complicated patterns in the data and capture non-linear correlations between the target variable and the characteristics. Note that the selection of features, the size of the dataset, and the model hyperparameters can all affect how well the models perform. To enhance the performance of the models, it is advised to modify the hyperparameters and experiment with various feature sets.

Overall, the study shows how machine learning algorithms may be used to forecast Bitcoin prices and emphasizes the significance of data pretreatment, feature engineering, and model selection in creating precise and trustworthy prediction models.

VI. FUTURE WORK

- Feature Engineering and data augmentation can be added to improve the accuracy.
- Sentiment based data can also be added.
- More models can be used (including regressor, classifier and neural networks).

VII. GITHUB REPOSITORY

The training dataset, testing dataset, code repository and results of

the project can be found on the below GitHub repository –

https://github.com/Srinarendra/PREDICTION-OF-BITCOIN-PRICES-USING-MACHINE-LEARNING.git

REFERENCES

[1] Siddhi Velankar, Sakshi Valecha, and Shreya Maleji. Bitcoin Price Prediction using Machine Learning Part 1 (Only Part 1 is published which proposes techniques to clean data using data mining). Department of Electronics & Telecommunications, Pune Institute of Computer Technology, Pune, Maharashtra, India, 409–415.

[2] I F. Pedregosa, et al. Bitcoin Price Prediction Using Machine Learning: An Approach to Sample Weighting in Ensemble Models. IEEE Access, vol. 9, pp. 5818–5830, 2021.

[3] M. M. Rahman, et al. Bitcoin Price Prediction with Machine Learning for Next Week. In 2020 IEEE 17th India Council International Conference (INDICON), pp. 1–6, 2020.

[4] K. Salah, et al. Bitcoin Price Prediction using Machine Learning Algorithms: An Experimental Study. In 2018 IEEE 5th International Conference on Industrial Engineering and Applications (ICIEA), pp. 115–120, 2018..

[5] Z. Huang, et al. Predicting Bitcoin Price with Deep Neural Networks. In 2018 IEEE International Conference on Big Data (Big Data), pp. 3902– 3905, 2018.