

BITCOIN PRICE PREDICTION USING ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

A PROJECT REPORT

Submitted by

ANANYA GUPTA [Reg No: RA1911003030265]

Under the guidance of

Ms. Nidhi Pandey

(Professor, Department of Computer Science & Engineering)

*in partial fulfillment for the award of the degree
of*

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



SRM INSTITUTE OF SCIENCE & TECHNOLOGY, NCR CAMPUS

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled “**BITCOIN PRICE PREDICTION USING ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**” is the bonafide work of “**ANANYA GUPTA [Reg No: RA1911003030265]**”, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Ms. Nidhi Pandey
GUIDE
Professor
Dept. of Computer Science & Engineering

Signature of the Internal Examiner

SIGNATURE

Dr. R.P.Mahapatra
HEAD OF THE DEPARTMENT
Dept. of Computer Science & Engineering

Signature of the External Examiner

ABSTRACT

The objective of this project is to determine the accuracy with which the closing price of the bitcoin can be predicted with the help of classification and linear regression methods. For predicting the future price of bitcoin, we have implemented machine learning algorithms i.e., Dummy Regressor & Support Vector Regression with accuracy of approximately 45% and 68%. The best accuracy was given by Support Vector Machine in comparison to Dummy Regressor. Both models are good as individually for prediction but mostly depends on data set we are containing and driven by same method i.e., regression. Using multiple regression models, we deduced that it has more performance in comparison to each other with different data sets relatively. By Support Vector Machine, Mean of Absolute value of Errors (MAE)(Mean of Absolute value of Errors) & Root of the Mean of the Square of Errors (RMSE)(Root of the Mean of the Square of Errors) values were 1163.568 & 2171.857 and the it increased with increasing accuracy of regression models.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Ms. Nidhi Pandey for her valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

Author

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	viii
ABBREVIATIONS	ix
1 Introduction	1
1.1 Motivation	1
1.1.1 Project Overview	2
1.1.2 Requirement Specification	2
2 Literature Survey	3
3 Methodology	5
3.1 Dummy Regressor	5
3.2 Support Vector Regression	5
3.2.1 Support Vector Machine (SVM)	6
4 Technologies	8
4.1 Machine Learning	8
4.1.1 Types of Machine Learning	9
4.2 Python	10
4.2.1 Libraries in python	11
5 Feature Engineering	14
5.1 Feature Smoothing	14
5.1.1 MACD(Moving Average Convergence/Divergence)	14
5.1.2 SMA(Simple Moving Average)	15

5.1.3	WMA(Weighted Moving Average)	16
5.1.4	EMA(Exponential Moving Average)	16
5.1.5	DEMA(Double Exponential Moving Average)	17
5.1.6	TEMA(Triple Exponential Moving Average)	17
6	Data Processing	19
6.0.1	Data Gathering	19
6.0.2	Data Cleaning	19
6.0.3	Data Normalization	20
6.0.4	Data Training and Splitting	20
7	Implementation & Result	21
7.1	Next Day Modelling Data	21
7.1.1	Steps to implement the model Support Vector Regression (SVR) & Dummy Regressor (DuR)	21
7.1.2	Code with Resultant Output	25
8	Social Impact on Society	31
8.0.1	Sparking Innovation	31
8.0.2	Building Trust	31
9	Conclusion and Future Enhancement	32
10	Reference	33

LIST OF FIGURES

1.1	Comparison between Traditional and Bitcoin Model Systems	1
1.2	Data set with 2923X7 entries.	2
3.1	Graph for Support Vector Regression	6
3.2	Graph for Support Vector Machine	7
4.1	11
4.2	Top libraries of python for Machine Learning (ML) Algorithms . . .	11
5.1	Feature Smoothing	14
5.2	MACD	15
5.3	SMA	15
5.4	Formula for WMA	16
5.5	Graph for WMA	16
5.6	Formula for EMA	17
5.7	Graph for EMA	17
5.8	Formula for DEMA	17
5.9	Graph for DEMA	18
5.10	Formula for TEMA	18
5.11	Graph for TEMA	18
7.1	Code to execute library for Dummy Regressor Support Vector Regression	21
7.2	Data Set	22
7.3	Test Train split	22
7.4	Graph with Temporary window split 0	22
7.5	DuR did test train data with X & Y test_train_split	23
7.6	DuR Model metrices	23
7.7	DuR Resultant Graph	23

7.8	SVR Model metrics	24
7.9	SVR Resultant Graph	24
7.10		25
7.11		26
7.12		26
7.13		27
7.14		27
7.15		28
7.16		28
7.17		29
7.18		29
7.19		30

ABBREVIATIONS

AI	Artificial Intelligence
DuR	Dummy Regressor
MAE	Mean of Absolute value of Errors
RMSE	Root of the Mean of the Square of Errors
SVR	Support Vector Regression
SVM	Support Vector Machine
DL	Deep Learning
ML	Machine Learning

CHAPTER 1

INTRODUCTION

1.1 Motivation

Bitcoin is the world's most valuable cryptocurrency, traded on over 40 exchanges worldwide that accept over 30 different currencies. According to <https://www.blockchain.info/>, it currently has a market capitalization of USD 9 billion and records more than 250,000 transactions per day.

Bitcoin as a currency, due to its relatively young age and resulting volatility, offers new methods of price prediction that are much greater than fiat currencies. It is also unique in its open nature compared to traditional fiat currencies. We do not have complete data on fiat currency cash transactions and currency circulation.

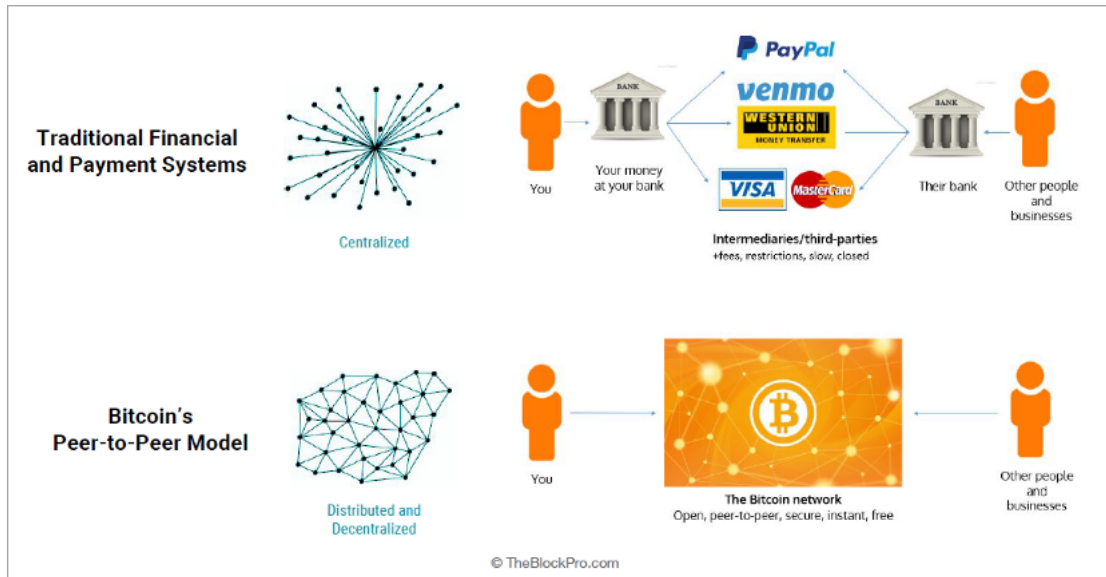


Figure 1.1: Comparison between Traditional and Bitcoin Model Systems

Bitcoin is a decentralized digital currency with no central administrator and can be transmitted directly from user to user on the Bitcoin network. Network nodes verify transactions cryptographically and record them on a publicly distributed ledger called a blockchain. Blockchain also tracks ownership, prevents tampering with transaction records, and prevents double spending.

1.1.1 Project Overview

Bitcoin is a type of digital currency similar to dollars, euros and yen. The difference is that instead of backing national or federal banks, they use online ledgers with strong encryption to secure online transactions. Bitcoin exchanges allow you to buy and sell virtual currencies. . You can also "mining". Bitcoin's popularity in 2021 is the result of exponential growth in market capitalization over several months.

Geopolitical and economic troubles over the past two years have caused global currency values to fall, stock markets to languish, and investors to lose their fortunes. This has reignited interest in digital currencies. Our system supports Bitcoin price prediction using machine learning. Using machine learning, this program helps to predict Bitcoin price. In order to predict Bitcoin price more accurately and quantitatively. The system uses the DUMMY REGRESSOR DuR and SUPPORT VECTOR REGRESSION SVR model to make forecasts. The system is well-suited to assist in price prediction, using machine learning to predict prices, achieving accuracy in all situations of technical trading indications and lowering price predictions during evaluation.

1.1.2 Requirement Specification

The data set which is used is from the last eight years (08.10.2014 – 08.10.2022). Users can view predictions for bitcoin and the admin can view all users using the system.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-10-08	336.115997	354.364014	327.187988	352.940002	352.940002	54736300
1	2014-10-09	352.747986	382.726013	347.687012	365.026001	365.026001	83641104
2	2014-10-10	364.687012	375.066986	352.963013	361.562012	361.562012	43665700
3	2014-10-11	361.362000	367.191010	355.950989	362.299011	362.299011	13345200
4	2014-10-12	362.605988	379.433014	356.144012	378.549011	378.549011	17552800
...
2918	2022-10-04	19623.583984	20380.343750	19523.839844	20336.843750	20336.843750	35887278685
2919	2022-10-05	20335.900391	20343.748047	19801.800781	20160.716797	20160.716797	33223790572
2920	2022-10-06	20161.039063	20408.392578	19900.087891	19955.443359	19955.443359	34711412966
2921	2022-10-07	19957.558594	20041.085938	19395.792969	19546.849609	19546.849609	29227315390
2922	2022-10-08	19537.158203	19599.453125	19460.628906	19496.669922	19496.669922	19980466176

2923 rows x 7 columns

Figure 1.2: Data set with 2923X7 entries.

CHAPTER 2

LITERATURE SURVEY

Bitcoin Price Prediction Using Artificial Intelligence (AI)&ML. Here, we used an artificial intelligence method i.e., machine learning, contains many algorithms like supervised, unsupervised, reinforcement, semi-supervised algorithms. To study the relationship between Bitcoin characteristics and next-day Bitcoin price changes. We used Dummy Regressor and Support Vector Regression Model to build this project.

We've all wondered what the cost of Bitcoin will be in 1, 2, 5, or even 10 years. It's really hard to predict, but we all love to do it. If you do it carefully, you can make big profits from buying and selling bitcoins. It has proven to be an asset to many in the past and still makes a lot of money today. You can also lose a lot of money if you don't think and calculate correctly. You should have a very good understanding of how and why Bitcoin prices change (organic market, guidelines, news, etc.). Considering these things (supply and demand, regulation, news, etc.), we should also think about Bitcoin technology and its advancements. Apart from that, the technical part needs to be addressed with various algorithms and technologies that can predict the exact Bitcoin price. Although I've come across various models that currently exist, such as the Dummy Regressor & Support Vector Regressions.

To better understand its practicality and its effectiveness in real-world applications, we use Entities to predict the direction of next-day bitcoin prices based on a series of approximately 2000 data entries and cryptocurrency characteristics over a eight-year period. We predicted. Over a period of 50 days, the ensemble-based trading strategy was compared by backtesting to the "previous day's trend-following" trading strategy. The previous trading strategy produced a return of around 45% & 68%, outperforming the follow the adjacent next days trendtrading strategy which produced a return of around 42% & 63% and the trading strategy following the single best Regression model in the ensemble. We were. About approximately 55% & 38% reduction.

In the second period of our examination we are just focusing in on the bitcoin price

information alone and utilized information at 10 days time frame. This is due to the fact that we saw an incredible opportunity to precisely evaluate price predictions at various levels of granularity and noisiness are modelling. This resulted in incredible results which had 4568% accuracy in precisely predicting the future bitcoin price changes using 10 minute time intervals.

CHAPTER 3

METHODOLOGY

3.1 Dummy Regressor

DummyRegressor is a regressor that makes predictions based on simple rules. This regressor serves as a simple baseline for comparison with other (real) regressors. A dummy regressor is a type of regressor that makes predictions based on a simple strategy without considering the input data. Similar to dummy classifiers, the sklearn library also provides dummy regressors that are used to create a baseline to compare other existing regressors (Poisson regressor, linear regression, ridge regression, etc.).

In short, there are four types of strategies used by dummy regressors.

- a) Mean: This is the default strategy used by dummy regressors. Always predict the mean of the training target values.
- b) Median: Predicts the median training target value.
- c) Quantile: Used to predict the specific quantile of the training target value when the quantile parameters are used together.
- d) Constant: This is typically used to predict a specific custom value provided and requires a constant parameter to be specified.

3.2 Support Vector Regression

Support vector machines SVM are commonly and frequently used for machine learning classification problems. Support Vector Regression SVR uses the same principles as Support Vector Machine, but for regression problems.

The regression problem is to find a function that approximates the mapping from the input range to the real numbers based on the training samples.

Consider these two red lines as decision boundaries and the green line as a hyperplane. Continuing with SVR, our goal is basically to consider points that lie within the decision boundary. The best fit straight line is the hyperplane with the maximum number of points.

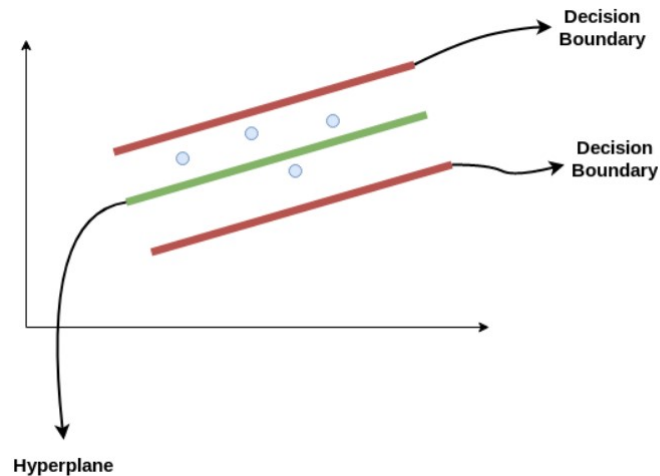


Figure 3.1: Graph for Support Vector Regression

3.2.1 SVM

Support vectors are data points near a hyperplane that affect the position and orientation of the hyperplane. Use these support vectors to maximize the classifier margin. Deleting support vectors changes the position of the hyperplane. These are the points that help build the SVM.

There are some important Support Vector Machine's(SVM) parameters to know.

a) Kernel

Kernels help find hyperplanes in high-dimensional space without increasing computational cost. Generally, the amount of computation increases as the dimensionality of the data increases. This dimensionality increase is necessary when the separating hyperplane is not found in a particular dimension and we need to move to a higher dimension.

b) Hyperplane

This is basically the boundary between his two classes of data in SVM. But in support vector regression this is the line used to predict the continuous output.

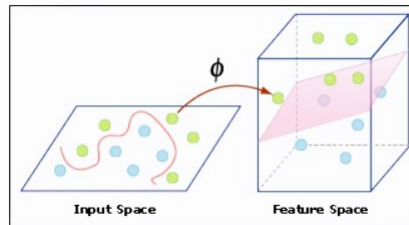


Figure 3.2: Graph for Support Vector Machine

c) Decision Boundary

A decision boundary can be thought of (for simplicity) as a boundary with positive examples on one side and negative examples on the other. This line allows you to classify examples as positive or negative. The same SVM concepts apply to support vector regression.

CHAPTER 4

TECHNOLOGIES

There are many technologies which are being used for bitcoin price prediction like deep learning Deep Learning (DL), machine learning ML, artificial intelligence AI, generic algorithms, etc with different different platforms like tensorflow, pytorch, google colab, jupyter, etc.

Regarding these technologies available, machine learning with python is more impactful and result oriented as both technologies are also used these in this project.

4.1 Machine Learning

Artificial intelligence applies machine learning, deep learning, and other techniques to solve real-world problems. Artificial intelligence (AI) brings real human-machine interaction.

Machine learning is a simple algorithm that enables computers to learn from data and make decisions and predictions. AI refers to the idea that machines can perform tasks intelligently. This is a faster process for learning risk factors and profit opportunities.

They have the ability to learn from their mistakes and experiences. Combining machine learning with artificial intelligence has the potential to become a vast field of gathering vast amounts of information, correcting mistakes and learning from further experience to develop smarter, faster and more accurate handling techniques. there is. The main difference between machine learning and artificial intelligence is probably machine learning if written in Python and artificial intelligence if written in PowerPoint.

4.1.1 Types of Machine Learning

Based on the method and type of learning, machine learning is mainly classified into four types:

a) Supervised Machine Learning

The main goal of supervised learning techniques is to map an input variable (x) to an output variable (y). Practical applications of supervised learning include risk assessment, fraud detection, and spam filtering. Two methods in this type are-

(i) Classification - Random Forest Algorithm, Decision Tree Algorithm, Logistic Regression Algorithm, Support Vector Machine Algorithm, Support Vector Regression.

(ii) Regression - Simple Linear, Regression Algorithm, Multivariate Regression Algorithm, Decision Tree Algorithm, Lasso Regression, Dummy Regressor.

b) Unsupervised Machine Learning

The main purpose of unsupervised learning algorithms is to group or classify unsorted data sets based on similarities, patterns, and differences. Two methods in this type are-

(i) Clustering - K-Mean Clustering algorithm, Mean-shift algorithm, DBSCAN Algorithm.

(ii) Association - Apriori Algorithm, Eclat, FP-growth algorithm.

c) Semi-supervised Machine Learning

Semi-supervised learning is a type of machine learning algorithm that falls between supervised and unsupervised machine learning.

To overcome the shortcomings of supervised and unsupervised learning algorithms, the concept of semi-supervised learning was introduced. The main goal of semi-supervised learning is to effectively use all available data, not just labeled data as in supervised learning.

d) Reinforcement Learning

Reinforcement learning uses a feedback-based process in which AI agents (software components) automatically explore their environment through hit and trial, take action, learn from experience, and improve their performance. Agents are rewarded for every good action and punished for every bad action. Therefore, the goal of reinforcement learning agents is to maximize the reward.

Reinforcement learning is mainly divided into two types of methods/algorithms.

(i) Positive Reinforcement Learning: Positive Reinforcement Learning claims that adding something increases the likelihood that the requested behavior will occur again. Amplifies the strength of an agent's actions and has a positive impact.

(ii) Negative Reinforcement Learning: Negative reinforcement learning works in exactly the opposite way to positive RL. Avoiding negativity increases the tendency for certain behaviors to occur again.

4.2 Python

In today's world where technology plays an increasingly important role in all aspects of our lives, it's important to choose a programming language that can efficiently solve real-world problems. Python is one such programming language. Python's popularity has skyrocketed in recent years thanks to its use in various industries such as software development, machine learning, and data science. The variety of libraries Python provides is the reason for its popularity. For this reason, many new talents today are drawn to Python as their primary programming language of choice. Therefore, through this article, we aim to provide readers with knowledge of the most popular Python libraries and how they are used today. modern world.

A Python library is a collection of related modules. Contains code bundles that can be used repeatedly in various programs. This makes Python programming easier and more convenient for programmers. Because you don't have to keep writing the same code for different programs. Python libraries play a very important role in areas such



Figure 4.1

as machine learning, data science, and data visualization.

4.2.1 Libraries in python

Some main libraries in python are -



Figure 4.2: Top libraries of python for ML Algorithms

a) TensorFlow

This library was developed by Google in collaboration with the Brain Team. This is an open source library used for high level computations. It is also used in machine learning and deep learning algorithms. It contains a large number of tensor operations. Researchers also use this his Python library to solve complex calculations in mathematics and physics.

b) Matplotlib

This library is responsible for plotting numerical data. As such, it is used for data analysis. It is also an open-source library that plots high-resolution numerical values

such as pie charts, histograms, scatterplots, and graphs.

c) Pandas

Pandas is an essential library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and various analytical tools. Facilitates data analysis, data manipulation and data cleansing. Pandas supports operations such as sorting, reindexing, iterating, concatenating, transforming data, visualizing, and aggregating.

d) Numpy

The name "Numpy" stands for "Numerical Python". It's a popular library. This is a popular machine learning library that supports large matrices and multidimensional data. It consists of built-in math functions for simple calculations. Even libraries like TensorFlow use Numpy under the hood to perform multiple operations on tensors. Array Interface is one of the main features of this library.

e) SciPy

The name "SciPy" stands for "Scientific Python". This is an open source library used for high-level scientific computing. This library is based on the Numpy extension. Works with Numpy to handle complex calculations. Numpy allows sorting and indexing of array data, while SciPy stores numeric data code. It is also commonly used by application developers and engineers.

f) Scrappy

An open source library used to extract data from websites. It offers very fast web crawling and high level screen scraping. It can also be used for data mining and automated data testing.

g) scikit-learn

A popular Python library for working with complex data. scikit-learn is an open source library that supports machine learning. It supports various supervised and unsupervised algorithms such as linear regression, classification and clustering. This library works in conjunction with Numpy and SciPy.

h) PyTorch

PyTorch is the largest machine learning library for optimizing tensor computations. It has an extensive API for performing tensor computations with powerful GPU acceleration. It also helps solve application problems related to neural networks.

CHAPTER 5

FEATURE ENGINEERING

5.1 Feature Smoothing

We explain the crucial idea of smoothing before moving on to the next section of the lesson on machine learning algorithms. A very effective method utilised in all types of data analysis is smoothing. Linear regression and noise reducer filtering are other names for this technique. When the pattern of the trend is uncertain, it is intended to detect trends even in the face of noisy data. The term "smoothing" refers to the assumption that the pattern is smooth, like a smooth surface, in order to achieve this feat. The noise, or variation from the trend, on the other hand, is unpredictable and erratic.

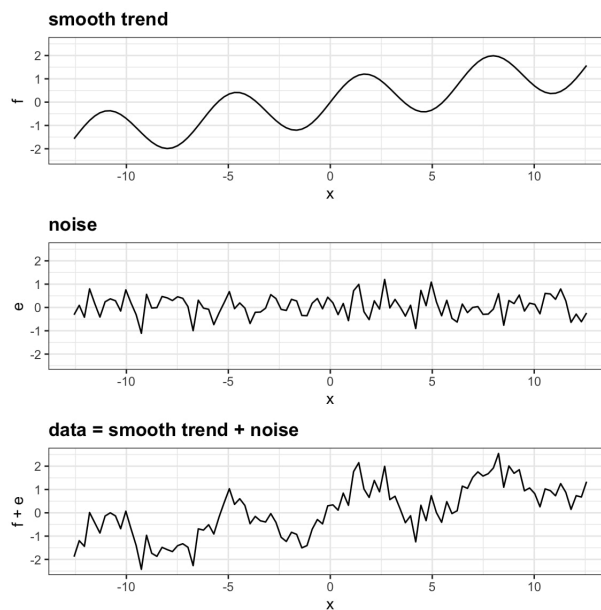


Figure 5.1: Feature Smoothing

5.1.1 MACD(Moving Average Convergence/Divergence)

An oscillator of momentum, the Moving Average Convergence/Divergence indicator is usually employed in trend trading. Despite being an indicator, it is rarely employed

to spot overbought or oversold positions. It shows as two lines that oscillate without boundaries on the graph. Similar to a two rolling average technique, trading signals are generated by the crossing of the parallel pairs.



Figure 5.2: MACD

5.1.2 SMA(Simple Moving Average)

A specified range of attribute values for a variety of times in that range are averaged by SMA. It establishes whether a price will move in a bullish or bearish trend.



Figure 5.3: SMA

5.1.3 WMA(Weighted Moving Average)

WMA gives more significance to data points that were collected recently and less weight to data points collected in the deep past. The weights must add up to 1 when totaled (or 100 percent).

$$\text{WMA} = \frac{\text{Price}_1 \times n + \text{Price}_2 \times (n - 1) + \dots + \text{Price}_n}{\frac{n(n + 1)}{2}}$$

Figure 5.4: Formula for WMA



Figure 5.5: Graph for WMA

5.1.4 EMA(Exponential Moving Average)

Like the WMA, the EMA similarly gives a higher weight to the most recent data points, although the pace of decline among one price and its predecessor varies. The rate of reduction varies exponentially.

$$EMA_{Today} = (Value_{Today} \times (\frac{Smoothing}{1 + Days})) + EMA_{Yesterday} \times (1 - (\frac{Smoothing}{1 + Days}))$$

Figure 5.6: Formula for EMA



Figure 5.7: Graph for EMA

5.1.5 DEMA(Double Exponential Moving Average)

DEMA reacts to short-term price volatility more faster than a typical exponential smoothing (EMA). It aids in noise filtering.

$$DEMA = 2 \times EMA - EMA(EMA)$$

Figure 5.8: Formula for DEMA

5.1.6 TEMA(Triple Exponential Moving Average)

It produces a technical analysis indicator that responds swiftly to price movements by combining several EMA calculations and subtracting the lag.



Figure 5.9: Graph for DEMA

$$TEMA = 3 \times EMA - 3 \times EMA(EMA) + EMA(EMA(EMA))$$

Figure 5.10: Formula for TEMA



Figure 5.11: Graph for TEMA

CHAPTER 6

DATA PROCESSING

6.0.1 Data Gathering

Since 2013, daily data for four channels has been monitored. First, the price history of the original bitcoin. Coin Market is leading the market with open API. Second, it contains data from the blockchain. I especially like the standard block size, number of user addresses, production volume, and number of miners. Given the infinite measurement problem, it turns out that having blockchain data should be condemned. On the other hand, by definition, the number of accounts depends on price movements, and more accounts can mean more transactions (perhaps not just transferring bitcoins to another address) trade different parties by signaling more users to join the network. Third, in emotional detail, the term Bitcoin has been used in the PyTrends SkLearn library. Please be careful. Finally, two indices are considered: the S&P 500 and the Dow and Jones. Both are refundable Yahoo Finance APIs. In total, this creates 12 features. Pearson interactions between symptoms. Some attributes are not closely related.

For example, financial indicators that are well suited to each other but not to any of the attributes associated with Bitcoin. We also see that Google Trends is related to Bitcoin transactions.

6.0.2 Data Cleaning

From exchange data, we only look at relevant volumes, deals, activation's, highs and market caps. In all data sets, NaN values present are replaced with the corresponding attribute description. Then all the records are merged into one depending on the amount of time. Looking at the behavior of the Bitcoin price over the period 2013-2014, we found it appropriate to remove data points prior to 2014. As such, the details fed into the neural network lie dormant from 2014 to September 2018.

In cleaning of data here we had used 20% Testing data and 80% Trained data to make the data normalised and to remove all data duplication and integration.

6.0.3 Data Normalization

Deciding how to get used to schedules, especially finances, is never easy. As a sixth rule, neural networks should be loaded with data containing a large amount of heterogeneous data (related to different time series scales, such as exchange rates or Google Trends). This allows us to create larger gradient updates that keep the network from changing. To be readable on the network, data should have the following characteristics:

Use small values - Generally, most values should be between 0 and 1. Homogeneous - that is, all features should have approximately the same range of values. Min-Max Scaling: $x' = \frac{x - \min(X)}{\max(X) - \min(X)}$ if the data input is mapped to numbers between 0 and 1.

6.0.4 Data Training and Splitting

I wanted to forecast next year first, but this could mean that data from 08.10.2014 to 08.10.2022 will be used for testing. The downside to this is that he actually has a big tilt in 2017 and the neural network may repeat this pattern and eventually learn. input, and predicting the year 2022 did not make much sense. Therefore, training on data from 08.10.2014 to 08.10.2022 would take 2 months to predict, but the data We expect the set to split and run out of space two months early: 08.08.2022. Each training and test set is based on an output function for fresh installations.

CHAPTER 7

IMPLEMENTATION & RESULT

7.1 Next Day Modelling Data

7.1.1 Steps to implement the model SVR & DuR

a) Importing Libraries of python in model :-

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import numpy as np
```

```
import random
```

```
import warnings
```

```
import missingno as msno
```

```
from google.colab import data_table
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.dummy import DummyRegressor
from sklearn.model_selection import TimeSeriesSplit
import sklearn.metrics as metrics
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
from google.colab import data_table
```

Figure 7.1: Code to execute library for Dummy Regressor Support Vector Regression

b) Importing Data Sets :-

Used Data Sets currently has 2923 rows \times 7 columns and is from 08.10.2014 to 08.10.2022. The given figure contains all column names specifying the data entries.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-10-08	336.115997	354.364014	327.187988	352.940002	352.940002	54736300
1	2014-10-09	352.747986	382.726013	347.687012	365.026001	365.026001	83641104
2	2014-10-10	364.687012	375.066986	352.963013	361.562012	361.562012	43665700
3	2014-10-11	361.362000	367.191010	355.950989	362.299011	362.299011	13345200
4	2014-10-12	362.605988	379.433014	356.144012	378.549011	378.549011	17552800
...
2918	2022-10-04	19623.583984	20380.343750	19523.839844	20336.843750	20336.843750	35887278685
2919	2022-10-05	20335.900391	20343.748047	19801.800781	20160.716797	20160.716797	33223790572
2920	2022-10-06	20161.039063	20408.392578	19900.087891	19955.443359	19955.443359	34711412966
2921	2022-10-07	19957.558594	20041.085938	19395.792969	19546.849609	19546.849609	29227315390
2922	2022-10-08	19537.158203	19599.453125	19460.628906	19496.669922	19496.669922	19980466176

2923 rows \times 7 columns

Figure 7.2: Data Set

c) Train Test Split :- The data has been test and train by test train split method and visualize split method for better accuracy and performance metrices.

```

train_window = 500
test_window = 100
train_splits = []
test_splits = []
for i in tqdm(range(train_window, len(final_df), test_window)):
    train_split = final_df[i-train_window:i]
    test_split = final_df[i:i+test_window]
    train_splits.append(train_split)
    test_splits.append(test_split)

```

100% |██████████| 26/26 [00:00<00:00, 10477.70it/s]

Figure 7.3: Test Train split

RESULTANT GRAPH -



Figure 7.4: Graph with Temporary window split 0

d) DuR :- Dummy Regressor performs well on small datasets.

We test & train data by using MAE&RMSE values and approximately was able to give the prediction to the accuracy of 45% which is much appropriate as the data includes the pandemic time period also.

```
Xtrain_split = train_splits[i].drop(['Close','Date'],axis=1).values
Xtest_split = test_splits[i].drop(['Close','Date'],axis=1).values

ytrain_split = train_splits[i]['Close'].reset_index(drop=True).values
ytest_split = test_splits[i]['Close'].reset_index(drop=True).values
```

Figure 7.5: DuR did test train data with X & Y test_train_split

	0
mae_train	7988.212007
rmse_train	9186.565480
mae_test	8731.168390
rmse_test	9190.741437

Figure 7.6: DuR Model metrices

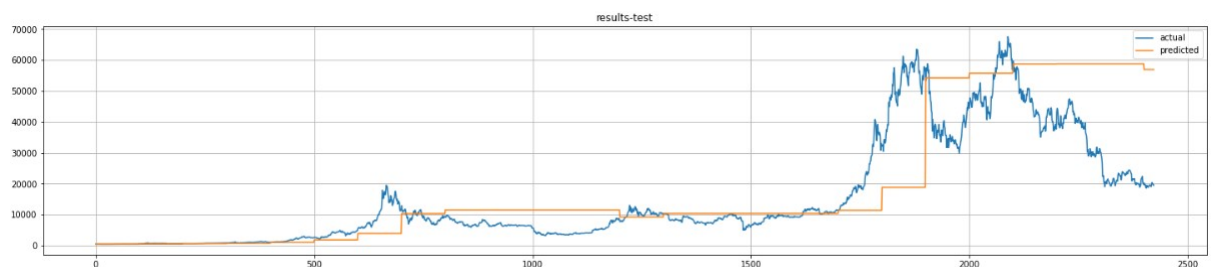


Figure 7.7: DuR Resultant Graph

e) SVR :- Unlike dummy regressor models, which try to minimize the error between actual and predicted values, SVR tries to fit a best-fit line within the threshold. The threshold is the distance between the hyperplane and the boundary. SVR is suitable for small data sets.

Here we used SVR with a regularization parameter on the RBF kernel with the kernel coefficient gamma set to $\text{auto}=1/n_{\text{features}}$.

0	
mae_train	1163.586468
rmse_train	2171.857820
mae_test	8297.390695
rmse_test	8782.841304

Figure 7.8: SVR Model metrics

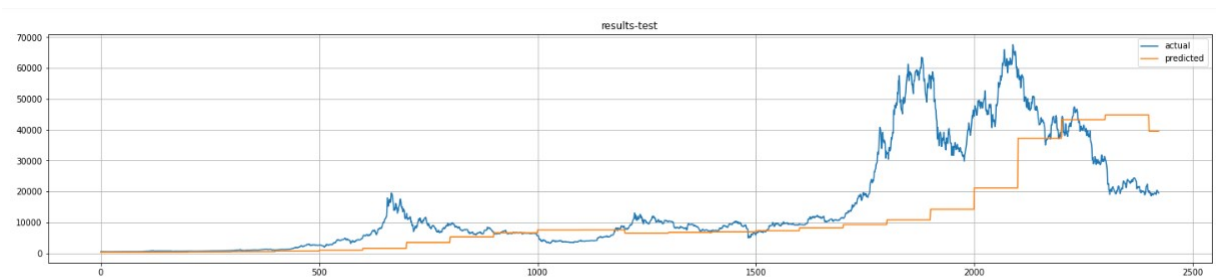
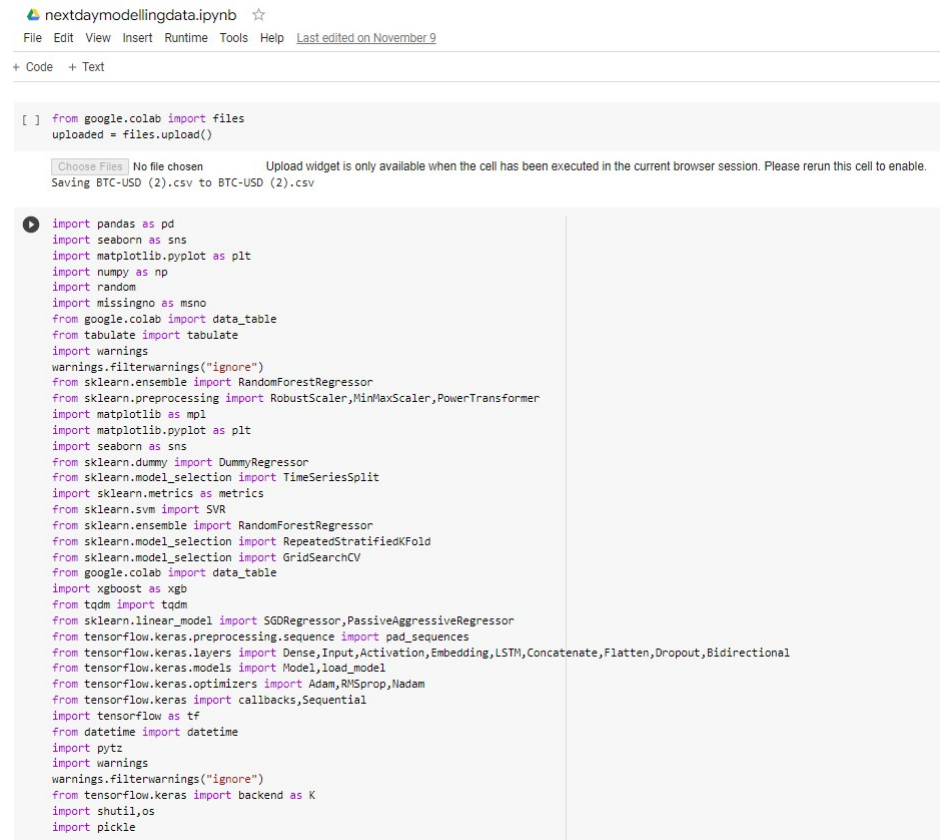


Figure 7.9: SVR Resultant Graph

7.1.2 Code with Resultant Output



The screenshot displays a Jupyter Notebook window titled 'nextdaymodellingdata.ipynb'. The interface includes a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', along with a 'Last edited on November 9' timestamp. Below the menu is a toolbar with '+ Code' and '+ Text' buttons. The main area shows a code cell with the following content:

```
[ ] from google.colab import files
    uploaded = files.upload()
```

Below the code cell, there is a file upload interface. It shows a 'Choose Files' button, a 'No file chosen' status, and a message: 'Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.' Below this, it says 'Saving BTC-USD (2).csv to BTC-USD (2).csv'.

The code cell is followed by a large block of import statements for various Python libraries:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import random
import missingno as msno
from google.colab import data_table
from tabulate import tabulate
import warnings
warnings.filterwarnings("ignore")
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import RobustScaler, MinMaxScaler, PowerTransformer
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.dummy import DummyRegressor
from sklearn.model_selection import TimeSeriesSplit
import sklearn.metrics as metrics
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
from google.colab import data_table
import xgboost as xgb
from tqdm import tqdm
from sklearn.linear_model import SGDRegressor, PassiveAggressiveRegressor
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Input, Activation, Embedding, LSTM, Concatenate, Flatten, Dropout, Bidirectional
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.optimizers import Adam, RMSprop, Nadam
from tensorflow.keras import callbacks, Sequential
import tensorflow as tf
from datetime import datetime
import pytz
import warnings
warnings.filterwarnings("ignore")
from tensorflow.keras import backend as K
import shutil, os
import pickle
```

Figure 7.10

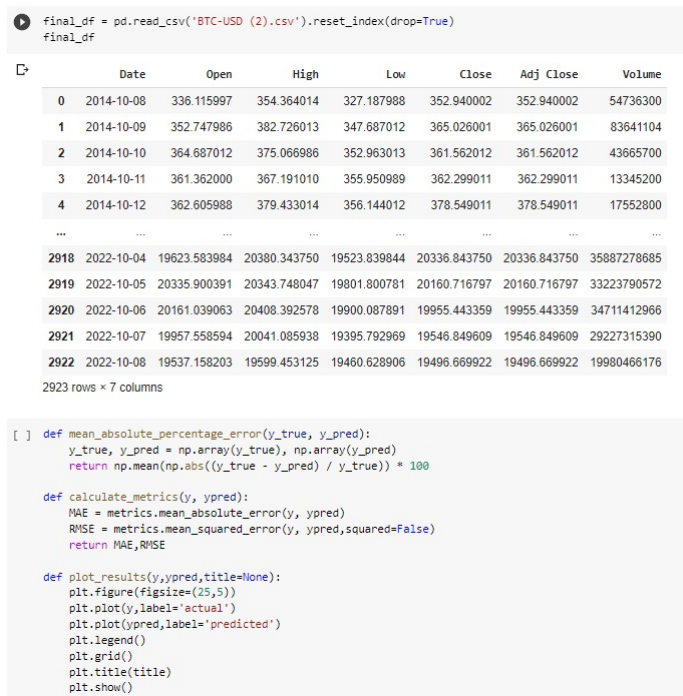


Figure 7.11

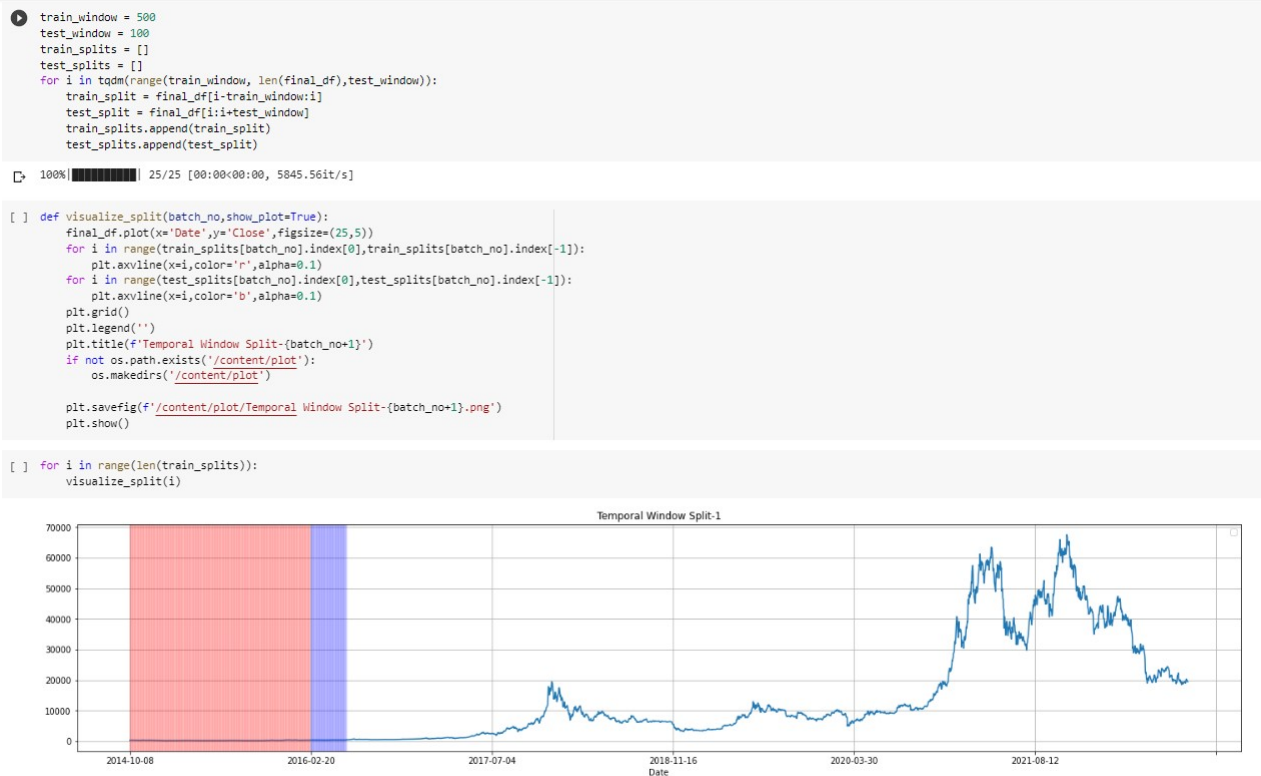
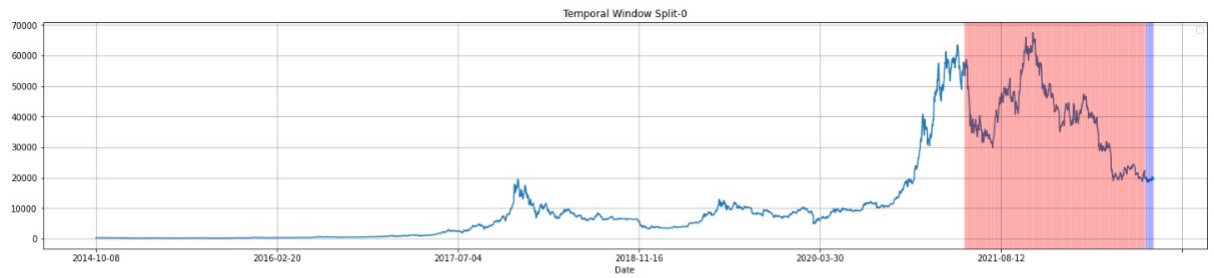


Figure 7.12

```
[ ] visualize_split(-1)
```



```
[ ] #dummy regressor
dmy_date_array = []
dmy_y_test_array = []
dmy_y_test_pred_array = []
dmy_batch_id_array = []
dmy_batch_id_array_result = []
dmy_batch_mae_train_array = []
dmy_batch_rmse_train_array = []
dmy_batch_mae_test_array = []
dmy_batch_rmse_test_array = []

for i in tqdm(range(len(train_splits))):
    Xtrain_split = train_splits[i].drop(['Close', 'Date'], axis=1).values
    Xtest_split = test_splits[i].drop(['Close', 'Date'], axis=1).values

    ytrain_split = train_splits[i]['Close'].reset_index(drop=True).values
    ytest_split = test_splits[i]['Close'].reset_index(drop=True).values

    dmy = DummyRegressor(strategy="quantile", quantile=0.9)
    dmy.fit(Xtrain_split, ytrain_split)

    ytrain_pred = dmy.predict(Xtrain_split)
    ytest_pred = dmy.predict(Xtest_split)

    MAE_train, RMSE_train = calculate_metrics(ytrain_split, ytrain_pred)
    MAE_test, RMSE_test = calculate_metrics(ytest_split, ytest_pred)
```

Figure 7.13

```

dmy_date_array.extend(test_splits[i]['Date'])
dmy_y_test_array.extend(test_splits[i]['Close'])
dmy_y_test_pred_array.extend((ytest_pred.flatten()))
dmy_batch_id_array.extend([i]*len(test_splits[i]))

dmy_batch_id_array_result.append(i)
dmy_batch_mae_train_array.append(MAE_train)
dmy_batch_rmse_train_array.append(RMSE_train)

dmy_batch_mae_test_array.append(MAE_test)
dmy_batch_rmse_test_array.append(RMSE_test)

dmy_result_test_df = pd.DataFrame()
dmy_result_test_df['batch_id'] = dmy_batch_id_array
dmy_result_test_df['Date'] = dmy_date_array
dmy_result_test_df['y_test'] = dmy_y_test_array
dmy_result_test_df['y_test_pred'] = dmy_y_test_pred_array
dmy_y_test_array = dmy_result_test_df['y_test']
dmy_y_test_pred_array = dmy_result_test_df['y_test_pred']
dmy_result_metrics_df = pd.DataFrame()
dmy_result_metrics_df['batch_id'] = dmy_batch_id_array_result
dmy_result_metrics_df['mae_train'] = dmy_batch_mae_train_array
dmy_result_metrics_df['rmse_train'] = dmy_batch_rmse_train_array
dmy_result_metrics_df['mae_test'] = dmy_batch_mae_test_array
dmy_result_metrics_df['rmse_test'] = dmy_batch_rmse_test_array
```

100% |██████████| 25/25 [00:00:00:00, 328.33it/s]

```
[ ] plot_results(dmy_y_test_array, dmy_y_test_pred_array, 'results-test')
```

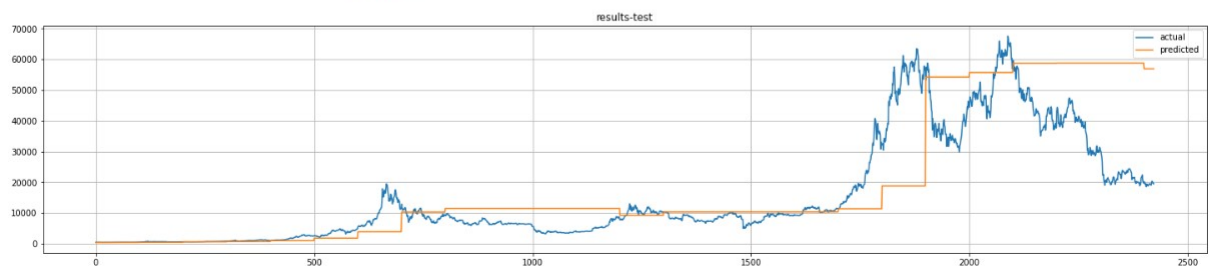


Figure 7.14

dmy_result_metrics_df					
	batch_id	mae_train	rmse_train	mae_test	rmse_test
0	0	107.457681	123.053146	34.106453	40.144413
1	1	127.334693	152.016829	181.186844	188.862211
2	2	227.158595	257.158550	71.707930	94.321301
3	3	231.699700	274.862597	307.025655	334.554098
4	4	393.392617	435.310267	879.674751	1080.550096
5	5	1008.846357	1058.131756	1817.986463	1994.627022
6	6	2418.418368	2629.463655	7065.428198	8292.618290
7	7	7595.720798	7940.655350	1607.842330	1887.930443
8	8	6866.128738	7643.043082	4019.817583	4139.402746
9	9	5574.169627	6314.669265	4976.563672	4983.389815
10	10	4647.016656	5126.747091	7668.714712	7680.469691
11	11	4607.462863	5056.723558	5989.959902	6201.127565
12	12	2960.802891	3435.647911	1386.011874	1601.942399
13	13	3798.204059	4347.207809	1976.809611	2195.934370
14	14	3629.344733	4256.077616	2172.135191	2580.528393
15	15	3292.180638	4056.557658	1714.066685	2033.771521
16	16	2324.972634	2943.813839	828.693797	955.728667
17	17	2194.289071	2615.817796	10609.784172	13528.253918
18	18	9136.142951	9733.143334	32719.078125	33762.266340
19	19	34743.794774	38147.943240	15882.593563	16918.823943
20	20	29890.332802	34263.790623	7388.118516	8097.804205
21	21	24137.510399	29608.865044	13519.988813	14870.762934
22	22	17284.847766	21496.009454	20987.167715	21761.304414
23	23	14118.353598	16478.946316	36960.775879	37026.920606
24	24	18389.717172	21268.481209	37513.971315	37516.496532

Figure 7.15

```
[ ] pd.DataFrame(dmy_result_metrics_df.mean()).drop(['batch_id'],axis=0)

0
mae_train    7988.212007
rmse_train    9186.565480
mae_test      8731.168390
rmse_test     9190.741437

[ ] dmy_result_test_df.to_csv('BTC-USD (2).csv')
dmy_result_metrics_df.to_csv('BTC-USD (2).csv')
```

```
#svr
svr_date_array = []
svr_y_test_array = []
svr_y_test_pred_array = []
svr_batch_id_array = []
svr_batch_id_array_result = []
svr_batch_mae_train_array = []
svr_batch_rmse_train_array = []
svr_batch_mae_test_array = []
svr_batch_rmse_test_array = []

for i in tqdm(range(len(train_splits))):
    Xtrain_split = train_splits[i].drop(['Close','Date'],axis=1).values
    Xtest_split = test_splits[i].drop(['Close','Date'],axis=1).values

    ytrain_split = train_splits[i]['Close'].reset_index(drop=True).values
    ytest_split = test_splits[i]['Close'].reset_index(drop=True).values

    svr = SVR(C=10000,gamma='auto',kernel='rbf')
    svr.fit(Xtrain_split, ytrain_split)

    ytrain_pred = svr.predict(Xtrain_split)
    ytest_pred = svr.predict(Xtest_split)

    MAE_train,RMSE_train = calculate_metrics(ytrain_split,ytrain_pred)
    MAE_test,RMSE_test = calculate_metrics(ytest_split,ytest_pred)

    svr_date_array.extend(test_splits[i]['Date'])
    svr_y_test_array.extend(test_splits[i]['Close'])
    svr_y_test_pred_array.extend(ytest_pred.flatten())
    svr_batch_id_array.extend([i]*len(test_splits[i]))
```

Figure 7.16

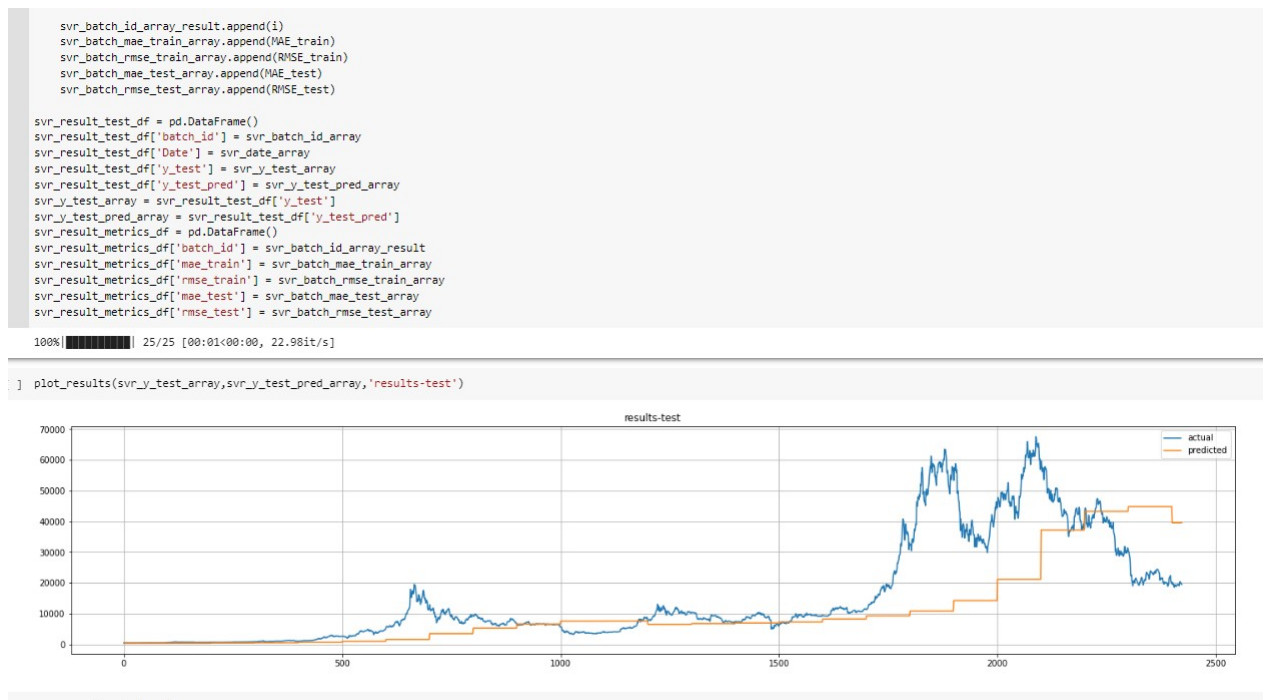


Figure 7.17

svr_result_metrics_df

	batch_id	mae_train	rmse_train	mae_test	rmse_test
0	0	0.099981	0.099982	135.144032	136.795208
1	1	0.100010	0.100010	305.742649	310.352751
2	2	0.099987	0.099987	285.308488	292.127770
3	3	0.099996	0.099997	530.651963	547.040398
4	4	0.099980	0.099980	1248.299814	1397.543484
5	5	0.099982	0.099983	2698.072067	2820.116143
6	6	0.099998	0.099998	9396.287822	10350.723033
7	7	182.742463	794.353350	5674.557108	5861.595560
8	8	74.087805	436.299323	2225.558219	2434.919111
9	9	29.454373	238.124862	197.225328	269.405312
10	10	12.066450	136.066974	3704.474782	3728.748149
11	11	11.583260	132.572045	2337.705790	2615.668205
12	12	0.100002	0.100002	3829.734426	4001.005330
13	13	0.100003	0.100003	1657.371387	1913.305074
14	14	0.099999	0.100000	1655.507513	1900.958640
15	15	0.100008	0.100008	1568.503624	1788.382237
16	16	0.099981	0.099981	2580.874767	2719.949698
17	17	0.099988	0.099988	12722.555074	15242.891722
18	18	802.862198	3308.709817	40721.375077	41564.202666
19	19	6017.630402	13013.696097	24564.413912	25507.403241
20	20	6413.631125	10993.264284	32012.406624	32920.065588
21	21	7460.926775	10135.362552	8153.983510	10173.835473
22	22	3744.125076	6975.998686	6175.059702	7908.906493
23	23	1408.583254	2836.279299	22938.210823	23044.639544
24	24	2930.668614	5294.418299	20115.742873	20120.451773

[] pd.DataFrame(svr_result_metrics_df.mean()).drop(['batch_id'])

Figure 7.18

```
[ ] pd.DataFrame(svr_result_metrics_df.mean()).drop(['batch_id'])
```

	0
mae_train	1163.586468
rmse_train	2171.857820
mae_test	8297.390695
rmse_test	8782.841304

```
[ ] svr_result_test_df.to_csv('BTC-USD (2).csv')
svr_result_metrics_df.to_csv('BTC-USD (2).csv')
```

Figure 7.19

CHAPTER 8

SOCIAL IMPACT ON SOCIETY

The rise of bitcoin has encouraged people to accept and value the role that technology plays in facilitating money transfers. In the digital era we live in, Bitcoin opens up new possibilities for us to perceive and take advantage of. You are not required to invest in that was like stocks and bonds, for instance. You could buy Bitcoin.

8.0.1 Sparking Innovation

A fantastic innovation is bitcoin. Consider how thousands of people currently use it if you have any doubts. More significantly, Bitcoin is supported by blockchain technology and currently has more over 109 million monthly active users at this time. And this is a ground-breaking technology that is having an effect on people, businesses, and entire sectors. Blockchain is already having an influence on the real estate industry by improving contract management. Smart contracts in particular are now possible because to blockchain technology.

8.0.2 Building Trust

Society has lost faith in governments and big corporations because of their control and interference in business and life. One of the main reasons Satoshi Nakamoto created Bitcoin was to provide an alternative. Bitcoin is decentralized, so governments and big corporations have no control or influence over it. Efforts to block Bitcoin began with these big companies, and some governments even banned Bitcoin.

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

The Bitcoin Price Prediction is a centralized application that allows you to track all of the money that are currently functioning in the online transactions in national as well as international. The following are some of the software's primary features:

1. Workflow - Tracks the transfer of funds from the federal government to state governments. It also documents the plans and accomplishments related to the projects that have been released.

2. Reports - Various reports are provided to assist even the most inexperienced user. For the better convenience of the users of the application, a large number of reports is available. There are a lot of detailed reports included.

3. Availability — The programmer is web based software and anytime accessible. It just requires proper authentication.

4. Security - Encrypted passwords are employed, as well as a password expiration feature to prevent passwords from being entered incorrectly again.

Bitcoin Price Prediction can be used for increments in economy to enhance the development world wide and web services available with this application can also be used for general purpose.

Future enhancements are: we can attempt to predict 7th & 15th & 30th day price by using similar approach. Similar to original reference, we can attempt to predict increase or decrease price.

CHAPTER 10

REFERENCE

- 1- <https://www.slideshare.net/AnishSojan/bitcoin-close-price-prediction-report>
- 2- <https://www.slideshare.net/KadambiniIndurkar/bitcoin-price-prediction>
- 3- <https://taxguru.in/finance/bitcoin-impact-society.html>
- 4- <http://rafalab.dfci.harvard.edu/dsbook/smoothing.html>
- 5- <https://medium.com/@rohansawant7978/forecasting-of-bitcoin-price-using-machine-learning-deep-learning-techniques-93bf662f46ab>