A PROJECT REPORT ON- MARKET SEGMENTATION

"Bitcoin Price Prediction"

Submitted in partial fulfillment for the award of the Internship

by

ASHUTOSH MISHRA

FEYNN LABS

ABSTRACT

After the boom and bust of cryptocurrencies' prices in recent years, Bitcoin has been increasingly regarded as an investment asset. Because of its highly volatile nature, there is a need for good predictions on which to base investment decisions. Although existing studies have leveraged machine learning for more accurate Bitcoin price prediction, few have focused on the feasibility of applying different modeling techniques to samples with different data structures and dimensional features. To predict Bitcoin price at different frequencies using machine learning techniques, we first classify Bitcoin price by daily price and high-frequency price. A set of high-dimension features including property and network, trading and market, attention and gold spot price are used 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 with high-dimensional features achieve an accuracy of 66%, outperforming more complicated machine learning algorithms. Compared with benchmark results for daily price prediction, we achieve a better performance, with the highest accuracies of the statistical methods and machine learning algorithms of 66% and 65.3%, respectively. 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%. Our investigation of Bitcoin price prediction can be considered a pilot study of the importance of the sample dimension in machine learning techniques.

Chapter 1

PREAMBLE

1.1 Introduction

Bitcoin is a crypto currency used worldwide for digital payment or simply for investment purposes. Bitcoin is decentralized i.e. it is not owned by anyone. Transactions made by bitcoins are easy as they are not tied to any country. Investment can be done through various marketplaces known as "bitcoin exchanges." These allow people to sell/buy bitcoins using different currencies. The largest bitcoin exchange is Mt Gox. Bitcoins are stored in a digital wallet which is basically like a virtual bank account. The record of all the transactions, the timestamp data is stored in a place called Blockchain. Each block contains a pointer to a previous block of data. The data on blockchain is encrypted. During transactions the users name is not revealed, but only their wallet ID is made public.

Predicting the future is no easy task. Many have tried and many have failed. But many of us would want to know what will happen next and would go to great lengths to figure that out. Imagine the possibilities of knowing what will happen in the future! Imagine what you would have done back in 2012 when Bitcoin was less than $15 knowing that it would surpass
$18,000! Many people may regret not buying Bitcoin back then but how were they supposed to know in the first place? This is the dilemma we now face in regards to Cryptocurrency. We do not want to miss out on the next jump in price but we do not know when that will or will not happen. So how can we potentially solve this dilemma? Maybe machine learning can tell us the answer.

Machine learning models can likely give us the insight we need to learn about the future of Cryptocurrency. It will not tell us the future but it might tell us the general trend and direction to expect the prices to move. Let's try and use these machine learning models to our advantage and predict the future of Bitcoin by coding them out in Python!

1.2 Existing System

After the boom and bust of crypto currencies' prices in recent years, Bitcoin has been increasingly regarded as an investment asset. Because of its highly volatile nature, there is a need for good predictions on which to base investment decisions. Although existing studies have leveraged machine learning for more accurate Bitcoin price prediction, few have focused on the feasibility of applying different modeling techniques to samples with different data structures and dimensional features. To predict Bitcoin price at different frequencies using machine learning techniques, we first classify Bitcoin price by daily price and high- frequency price. A set of high-dimension features including property and network, trading and market, attention and gold spot price are used for Bitcoin daily price prediction, while the basic trading features acquired from a cryptocurrencyexchange are used for 5-minute interval price prediction.

1.3 Proposed System

Statistical methods including Logistic Regression and Linear Discriminant Analysis for Bitcoin daily price prediction with high-dimensional features achieve an accuracy of 66%, outperforming more complicated machine learning algorithms. Compared with benchmark results for daily price prediction, we achieve a better performance, with the highest accuracies of the statistical methods and machine learning algorithms of 66% and 65.3%, respectively. Machine learning models such as Long Short-term Memory for Bitcoin 5- minute interval price prediction are superior to statistical methods, with accuracy reaching 67.2%. Our investigation of Bitcoin price prediction can beconsidered a pilot study of the importance of the sample dimension in machine learning techniques.

1.4 Problem Statement

The objective of this proposed system is to develop an application which will predict the bitcoin prices in future with decent accuracy. This allows the investors to invest wisely in bitcoin trading as the prices of bitcoin have gone up to an exaggerating amount in the last ten years.

1.5 Objective of the Project

The objectives of the “Bitcoin Price Prediction” can be stated as follows:

1. Develop an application which will predict the bitcoin prices in future with decent accuracy.
2. Allow the investors to invest wisely in bitcoin trading as the prices of bitcoin have gone up to an exaggerating amount in the last ten years.
3. Make use of machine learning algorithms to increase the accuracy of Bitcoin price prediction

Chapter 3

THEORITICAL BACKGROUND

Theoretical background highlighting some topics related to the project work is given below. The description contains several topics which are worth to discuss and also highlight some of their limitations that encourage going on finding solutions as well as highlights some of their advantages for which reason these topics and their features are used in this project.

3.1 Prediction Models

3.1.1 Deep Neural Networks

The first prediction model was based on a deep neural network (DNN). A DNN usually consists of an input layer, several hidden layers, and an output layer as shown in Figure 1.

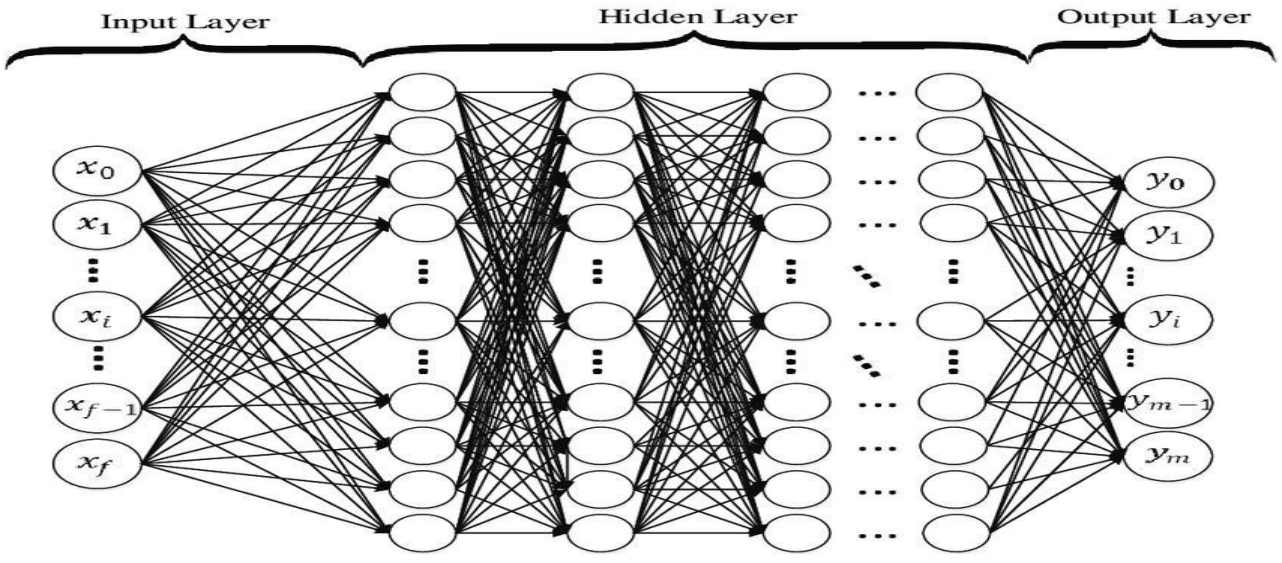


Figure 1 : An example of a fully-connected deep neural network (DNN) model.

In DNN prediction model, every node in one layer is connected to every node in the next layer, i.e., fully-connected. That is,each node in the hidden layer takes input vectors x1, . . ., xn from the previous layer, where n is the number of nodes in the previous layer and the size of each Vector is m.Its output h is then defined as

$$h=\sigma h(w1 \square x1+\cdots+wn \square xn+b)$$

where the weight vectors w1, . . ., wn and the bias b are parameters to be learned using training data. In addition, is an element-wise multiplication (Hadamard product) and σh is a nonlinear function, such as the logistic sigmoid, defined by the activation function

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

and the rectified linear unit (ReLU) , defined as g(x) = max(0,x). σh is applied element-wise, and thus the size of his also m. In the output layer, the set of input vectors are flattened into a single vector and then translated into a single value by using a dot product between the flattened vector and a weight vector. For a regression problem, the final value y is used as the predicted Bitcoin price. For a classification problem y is further translated into a value between 0 and 1 using a sigmoid function and then rounded off. In particular, 1 means the rise of the price and 0 means the fall or no change.
If we represent the whole DNN model as a function f , the parameters of the model are optimized using the back-propagation algorithm and training data so that the following mean squared error can be minimized:

Where f(Xi) is a predicted value based on an input sequence Xi and Yitrue is the true Bitcoin price at the day after the day corresponding to Xi[m] for regression. For classification, Yitrueis 1 if the true price is higher than the price at Xi[m]and otherwise 0.

3.1.2 Recurrent Neural Networks and Long Short – Term Memory

For time series data, it is well known that a recurrent neural network (RNN) model and a long short-term memory (LSTM) model are effective. Although a DNN model can be used to predict time series data, it does not exploit the temporal relationship between the data points in the input sequence. To exploit the temporal relationship, both RNN and LSTM models use an internal state (memory). Figure 2 shows example RNN and LSTM models for the Bitcoin price prediction.

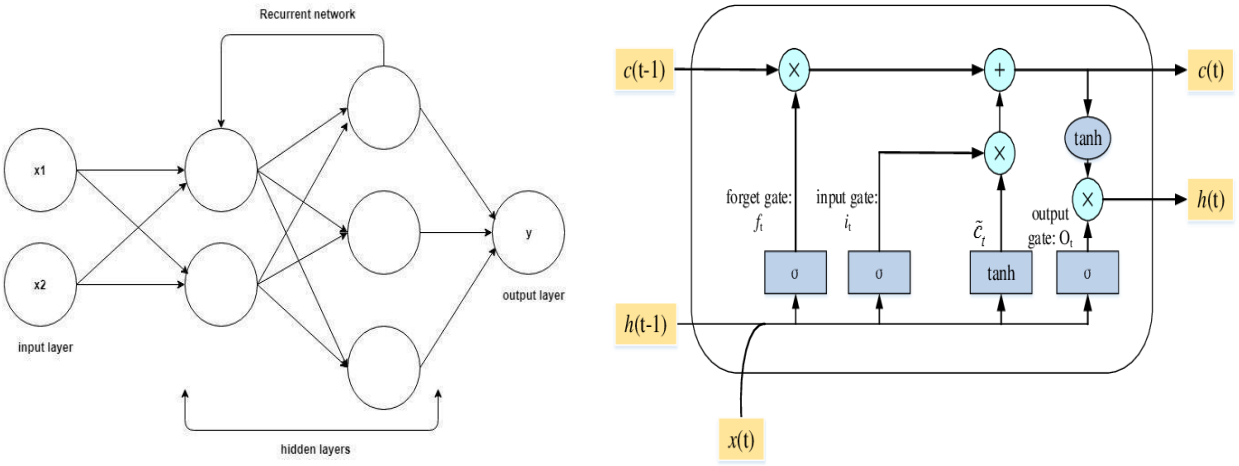


Figure 2. Recurrent neural network (RNN)model (left) and long short-term memory (LSTM) model (right).

To illustrate, suppose that we use three days data to predict the price. Then, the input layer of an RNN model consists of three nodes each of which takes the whole data of a single day, i.e., a vector of 18 features. Given xtas an input, each hidden state htis then computed as follows:

$$ht=tanh(Whxt+Uhht−1+bh)$$

where Wh , Uh, and bh are parameters to be learned and tanh is a hyperbolic tangent function. The main difference from DNNs is that RNNs use not only the input xt, but also the previous hidden state ht−1, to compute the current hidden state ht, thus exploiting the temporal dependency between the input sequence data. Assuming that we use three days sequence data, the final output y for regression is computed as follows,

$$y=wyh3+by$$

where wy and by are also parameters to be learned. One drawback of RNNs is that when the length of the input sequence is large, the long-term information is rarely considered due to the so-called vanishing gradient problem. LSTM avoids the vanishing gradient problem mainly by using several gate structures, namely input, output, and forget gates, together with a cell unit (the memory part of an LSTM unit), and additive connections between cell states. For an LSTM unit at time t, its hidden state ht, i.e., the output vector of the LSTM unit, is computed as follows,

$$ft=σ(Wfxt+Ufht−1+bf)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$g_t = \tanh(W_g x_t + U_g h_{t-1} + b_g)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

where W, U, and b are parameters to be learned and is a sigmoid function. ft , it , and ot, respectively, correspond to the forget, input, and output gates, and ctis the cell state vector.

Thanks to the cell state, the long-term dependencies between the data points in the input sequence is maintained well and LSTMs can be applied to long sequence data. In the experiment, only LSTM-based prediction models were considered.

3.1.3 Convolutional Neural Networks

A convolutional neural network (CNN), has many applications in image analysis and classification problems. Recently, it is shown to be also effective for sequence data analysis, and thus we also developed a CNN-based prediction model. Normally, a CNN consists of a series of convolution layers, ReLU layers, pooling layers, and fully-connected layers, where a convolution layer convolves the input with a dot product. In this work, we developed a simple CNN model consisting of a single 2D convolution layer (Conv2D) as shown in Figure 7. (A more sophisticated CNN modelis discussed in the next section.) The input is an $m \times 18$ matrix, where m is the number of days to be consulted for the prediction and 18 is the number of the features. A total of 36 2D convolution filtersof size 3×18 are used for convolution, where single real values are extracted from consecutive three days data through each filter. That is, 3×18 feature values are translated into a single value and thus each filter produces a real-valued vector of size $m-2$, which is then applied to an element-wise ReLU activation function. Then, the 36 output vectors produced by the convolution filters are flattened into a single vector of size $(m-2) \times 36$, which is then translated into a single prediction value through a fully connected layer. The number of filters was determined experimentally. Moreover, unlike other image analysis applications, simply adding more convolution layers togetherwith pooling layers did not improve the performance of CNN models for Bitcoin price prediction, and therefore we present only a simple CNN model.

3.1.4 Deep Residual Networks

Although stacking many layers allows a CNN model to encode a very complex function from the input to the output, it is difficult to train such a model due to the vanishing gradient problem as in RNNs. A deep residual network (ResNet) resolves this problem by introducing a so-called "identity shortcut connection" that skips one or more layers. Figure 3 shows stacked nonlinear layers without/with a shortcut connection. Assuming that H(x) is the desired underlying mapping, residual learning lets the stacked nonlinear layers fit a residual mapping of H(x)−x. The original mapping is then recast to F(x) + x. In the work by the authors of, it is shown that the residual mapping is easier to optimize than the original. By simply adding identity shortcut connections, deep convolutional networks can easily be optimized and produce a good result. Figure 3 shows our ResNet model used in the experiment. It consists of an input layer, one 2D convolution layer, four residual blocks, one average pooling layer, and an output layer. A residual block implements a shortcut connection between two convolution layers. For ResNet, we use convolution layers different from the one used for CNN. More precisely, the first Conv2D for input data uses 36 convolution filters of size 4 × 4. Then, each Conv2D in the four residual blocks uses filters of size 2× 2. The number of filters used in each Conv2D of the first, second, third, and last residual blocks is 36, 72, 144, and 288, respectively. Although it is not shown in the figure, we apply batch normalization and the ReLU activation function after each Conv2D in each residual block. After applying 2D average pooling of size 3 × 3, we flatten the resultant 288 vectors of size 1 × 1 into a one-dimensional vector. It is translated into single prediction value through a fully connected layer as in the CNN model.

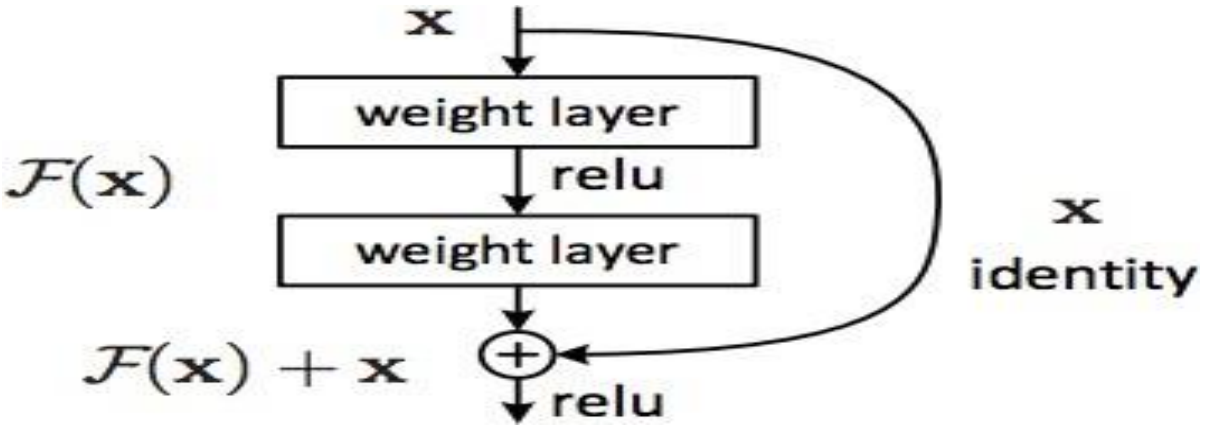


Figure 3. Residual Learning

3.1.5 Combinations of CNN and RNNs

As CNNs are effective for structural analysis, whereas RNNs are effective for temporal analysis, developed a prediction model called CRNN by combining CNN and LSTM models so as tointegrate their strengths . Figure 4 shows the structure of our CRNN model. The same input data are fed into both the CNN and LSTM models discussed in the previous sections. In particular, for the CNN part, after applying 2D convolution, we also apply batch normalization and flattening. As for the LSTM part, we additionally apply dropout . The results of CNN and LSTM are then combined into a single vector, which is then translated into a single prediction value through a fully connected layer. As a combining operator, we tested element-wise addition, subtraction, multiplication, average, maximum and minimum operators, and a concatenation operator. In the experiment, the subtraction operator was used for regression problems while the concatenation operator was used for classification problems because they showed better performance.

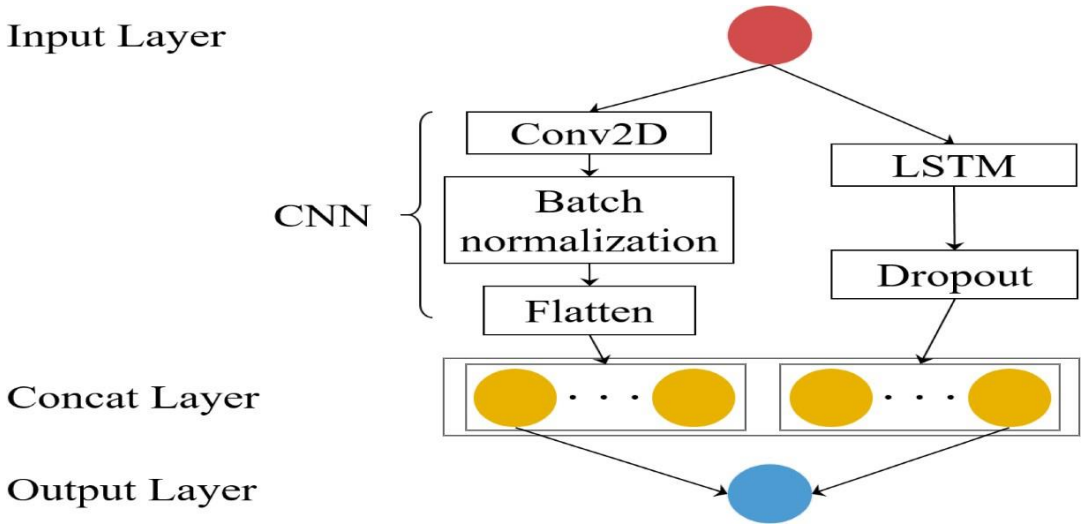


Figure 4. CRNN model (combination of CNN and RNN)

Chapter 4

SYSTEM REQUIREMENT SPECIFICATION

A System Requirement Specification (SRS) is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point prior to anyactual design or development work. The information gathered during the analysis is translated into a document that defines a set of requirements. It gives a brief description of the services that the system should provide and also the constraints under which the system should operate. Generally, SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It's a two-way insurance policy that assures that both theclient and the organization understand the other's requirements from that perspective at a given pointin time.
SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an ecommerce website and so on) must provide, as wellas states any required constraints by which the system must abide. SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting therequirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

4.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulationand processing and other specific functionality. In this system following are the functional requirements: -

Following are the functional requirements on the system:

1. The entire control model set must be translated to C output Code.
2. Inputs must be models designed using CLAW design components along with standard design components,
3. Multiple design models must be processed and the result must be combined to obtain a single output file.

4.2 Non-Functional Requirement

Nonfunctional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Nonfunctional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as: -

- Product Requirements
- Organizational Requirements
- User Requirements
- Basic Operational Requirements

4.2.1 Product Requirements

Platform Independency: Standalone executables for embedded systems can be created so the algorithm developed using available products could be downloaded on the actual hardware and executed without any dependency to the development and modeling platform.

Correctness: It followed a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.

Ease of Use: Model Coder provides an interface which allows the user to interact in an easy manner.

Modularity: The complete product is broken up into many modules and well- defined interfaces are developed to explore the benefit of flexibility of the product.

Robustness: This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevance and correctness
Nonfunctional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security & usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

4.2.2 Organizational Requirements

Process Standards: The standards defined by DRDO are used to develop the application which is the standard used by the developers inside the defense organization.

Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

4.2.3 User Requirements

- The coder must request the name of the model file to be processed
- In case of multiple files, the coder must ask the names of the files sequentially.
- The output file must be a C code translated from the model.
- Only a single output file must be created even if multiple input files are provided.

4.2.4 Basic Operational Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points: -

Mission profile or scenario: It describes the procedures used to accomplish mission objectives. It also finds out the effectiveness or efficiency of the system.

Performance and related parameters: It points out the critical system parameters to accomplish the mission

Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.

Operational life cycle: It defines the system lifetime.

4.2.5 System Configuration

H/W System Configuration:

Processor	-	Pentium –IV
Speed	-	1.1 Ghz
RAM	-	4GB RAM
Hard Disk	-	20 GB
Key Board	-	Standard Windows Keyboard
Mouse	-	Two or Three Button Mouse
Monitor	-	SVGA

S/W System Configuration:

Operating System	: XP/7/8/8.1/10
Coding Language	: Python – 3.7.0

Chapter 5

SYSTEM ANALYSIS

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluate the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

5.1 Feasibility Study

All systems are feasible when provided with unlimited resources and infinite time. But unfortunately, this condition does not prevail in the practical world. So it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill- conceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

5.1.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.2 Analysis

5.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of a healthy networking environment. Performance analysis is done to find out whether the proposed system. It is essential that the process of performance analysis and definition must be conducted in parallel.

5.2.2 Technical Analysis

System is only beneficial only if it can be turned into information systems that will meet the organization's technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed & installed, whether there are any major barriers to implementation. Regarding all these issues in technical analysis there are several points to focus on: -

Changes to bring in the system: All changes should be in a positive direction, there will be increased level of efficiency and better customer service.

Required skills: Platforms & tools used in this project are widely used. So the skilled manpower is readily available in the industry.

Acceptability: The structure of the system is kept feasible enough so that there should not be any problem from the user's point of view.

5.2.3 Economical Analysis

Economic analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we need not haveany routers which are highly economical. So, the system is economically feasible enough.

Chapter 6

SYSTEM DESIGN

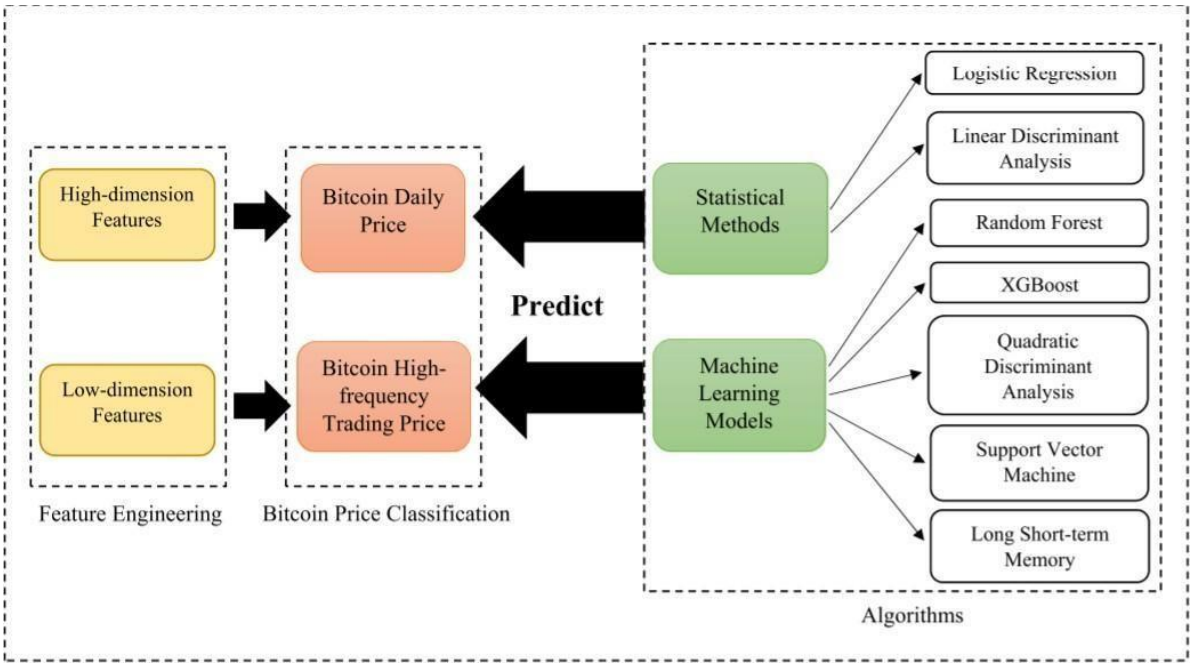


Fig 5

Our project addresses leveraging appropriate machine learning techniques to engineer sample dimensions for Bitcoin price prediction. Inspired by the principle of Occam's razor and the characteristics of our datasets, we tackle the problem as follows. First, the prediction sample is divided into daily intervals with small sample size and 5-minute intervals with a big sample size. Second, we conduct the features engineering: select high-dimension features for daily price and few features for 5-minute interval trading data respectively. Third, we conduct simple statistical models including Logistic Regression and Linear Discriminant Analysis and the more complicated machine learning models including Random Forest, XGBoost, Quadratic Discriminant Analysis, SupportVector Machine and Long Short-term Memory. Fourth, we adopt the simple statistical methods to predicting Bitcoin daily price with high- dimensional features to avoid overfitting. Meanwhile, the machine learning models are leveraged in high-frequency price few features. Fig. 5 shows the overview of our research framework.

Our project makes observations in two ways. One is to extend the feature dimensions, and the other is to evaluate different machine learning techniques for solving problems of multiple frequency

Bitcoin prices. The study makes the following contributions. (1) To the best of our knowledge, we are at the forefront of establishing higher dimensional features for problems of Bitcoin daily price prediction by integrating investor attention, media hype and XAU gold spot features with common and traditional features such as network and market. (2) We address the importance of the sample dimension by classifying Bitcoin price data by interval. The real-time 5-minute interval trading data acquired from the top cryptocurrency exchange is high-frequency and large scale, and the aggregated Bitcoin daily price obtained from CoinMarketCap is low-frequency and small scale. Hence, the problem of Bitcoin price prediction is addressed from a broad perspective. (3) To find appropriately complex models and meet the requirement of accuracy, we evaluate different machine learning techniques using problems of multiple frequency Bitcoin price. Specifically, we lower the complexity of algorithms for low-frequency daily price prediction with higher-dimension features and apply more complicated models for high-frequency price prediction with a few features. The results show that simple statistical methods outperform machine learning models for daily Bitcoin price prediction while more complicated models should be adopted for high frequency Bitcoin price prediction. We envision this study as a pilot for dealing with datasets with different scales and intervals, which can shed light on other industrial prediction problems in the context of machine learning.

6.1 Experimental Design

Two datasets are employed. The first includes the aggregated Bitcoin daily price, with a big interval and small scale, from CoinMarketCap.com. It also includes property and network data, trading and market data, media and investor attention and gold spot price, for the period from February 2, 2017, to February 1, 2019. Fig. 3 plots the distribution of the Bitcoin daily price. A complete cycle for Bitcoin price rise and fall is considered. The price continued to rise from February 2017 and crashed from January 2018 to February 2019.

Summary statistics of features used for bitcoin daily data prediction.

Feature	Count	Mean	SD	Min	Max
Block size	740	819710.5	157884.5	361626.6	998175.2
Hash rate	740	23412238	17352387	2917084	61866256
Mining difficulty	740	3.18E+12	2.41E+12	4.22E+11	7.45E+12
Number of transactions	740	255215.8	57038.29	131875	490644
Confirmed transactions per Day	740	255694.5	57012.15	131875	490644
Mempool transaction Count	740	255215.8	57038.29	131875	490644
Mempool size	740	26489513	35357624	35369.5	1.37E+08
Market capitalization	740	9.87E+10	6.1E+10	1.53E+10	3.23E+11
Estimated transaction value	740	193095	96494.26	37558.21	629491.3
Total transaction fees	740	165.9894	192.5094	11.2287	1495.946
Google trend search volume index	740	8.452703	10.53606	2	100
Gold spot price	740	1268.682	43.20863	1174.2	1357.91

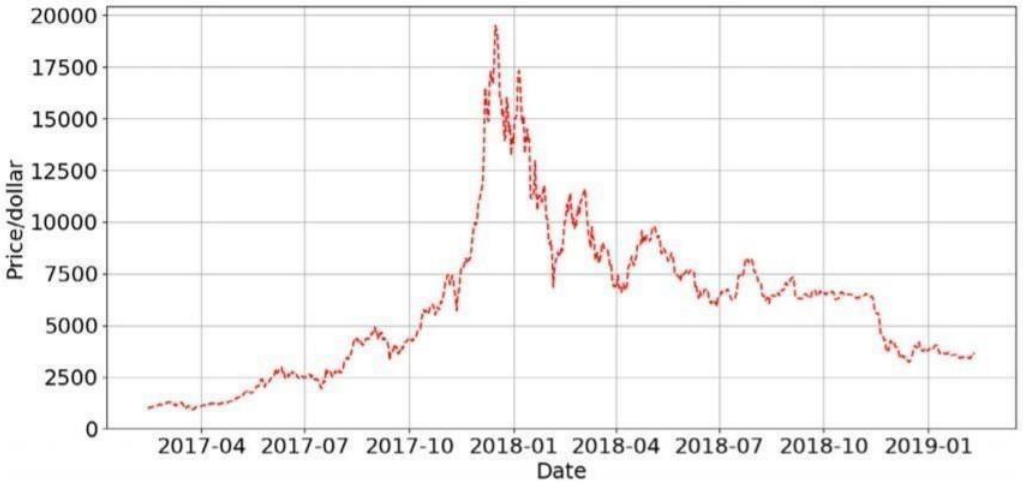


Figure 6: Bitcoin daily price distribution

The second dataset consists of 5-minute interval Bitcoin real-time trading price data at high- frequency and large scale pulled from Binance, the top cryptocurrency exchange in the world. We collected tick data by building an automated real-time Web scraper that pulled data from the APIs of the Binance cryptocurrency exchange from July 17, 2017 to January 17, 2018, obtaining roughly 50,000 unique trading records including Price, Trading Volume, Open, Close, High, and Low points for use in our modeling. Fig. 4 illustrates the distribution of the Bitcoin 5-minute interval price. We can observe moderate growth during the period of January to May 2017 and a rapid rise to a peak at the beginning of 2018, which is the price turning point. A laptop is configured to process the data for our experiments, with four cores of 3.60 GHz CPU and a total memory of 500 GB. We ordered multiple frequency Bitcoin price datasets and used the first 75% for training and the remaining 25% for testing.

Z. Chen, C. Li and W. Sun / Journal of Computational and Applied Mathematics 365 (2020) 112395

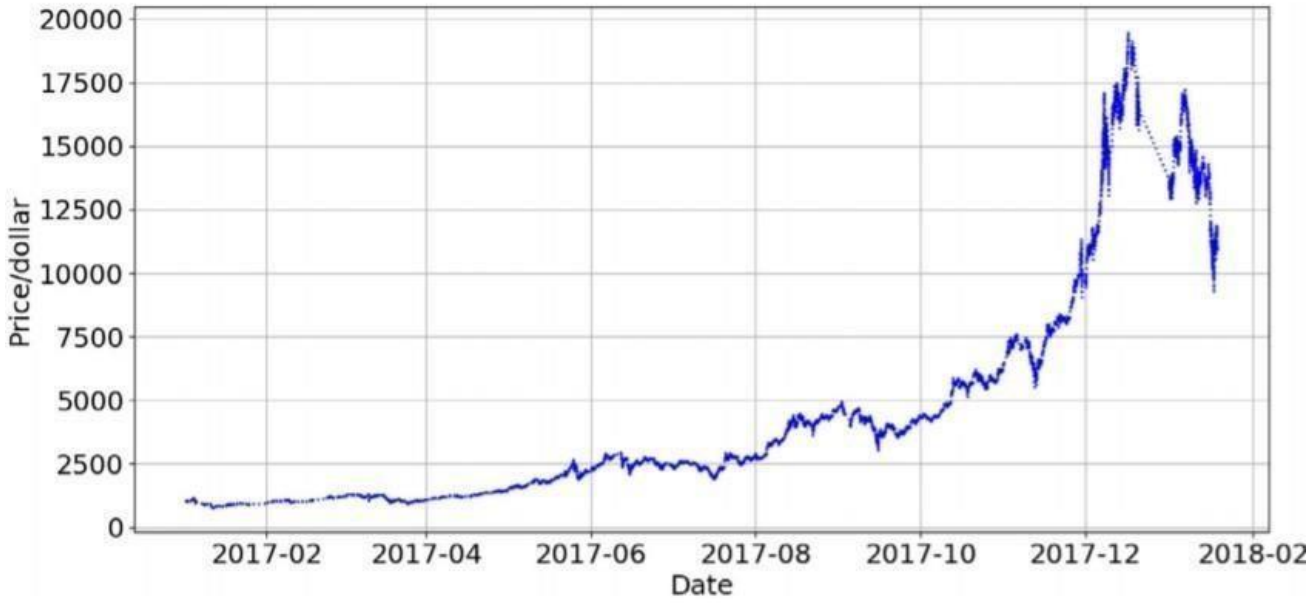


Figure 7: Bitcoin 5 - minute interval price distribution

6.2 Machine Learning Algorithms

This section presents the implementation of different machine learning techniques. For Bitcoin daily price with higher dimensional features, we implement two statistical methods: logistic regression (LR) and linear discriminant analysis (LDA), and the 5-minute interval price with a few features is predicted using machine learning models including random forest (RF), XGBoost (XGB), quadratic discriminant analysis (QDA), support vector machine (SVM), and long short-term memory (LSTM).

Z. Chen, C. Li and W. Sun / Journal of Computational and Applied Mathematics 365 (2020) 112395

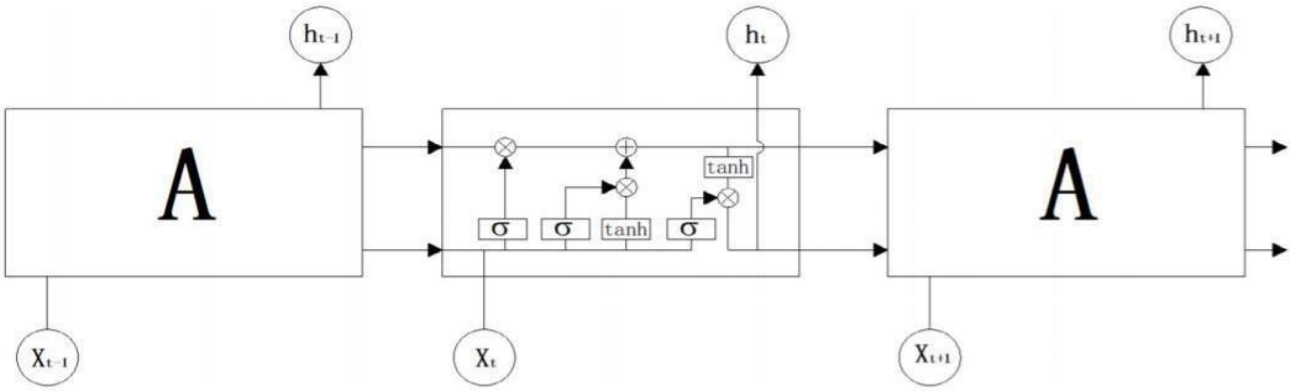


Figure 8: Structure of Long short – term memory model

6.3 Parameter Settings and Configuration

We apply the default parameters in Python as our optimal values. The Keras package is applied for the LSTM algorithm and Sklearn is used for other algorithms. Tables 4–10 show the parameters and configuration that we used.

Table 4
Parameters and configuration for logistic regression.

Penalty	Tol	C	Fit-intercept	Intercept scaling	Class weight	Random state	Max iter	Verbose	N jobs
L2	1E−4	1.0	True	1	None	None	100	0	None

Table 5
Parameters and configuration for linear discriminant analysis.

Solver	Tol	Shrinkage	Priors	N components	Store covariance
Svd	1E−4	None	None	None	False

Table 6
Parameters and configuration for random forest.

Crite-rion	Max depth	Min samples split	Min samples leaf	Min weight fraction leaf	Max features	Max leaf nodes	Mini impurity decrease	Mini impurity split	Boot-strap	Oob score	N jobs	Random state	Wer-bose	Warm start	Class weight
Gini	None	2	1	0	Auto	None	0	None	True	False	None	None	0	False	None

Table 7
Parameters and configuration for XGBoost.

Max depth	Learning rate	N estimators	Silent bool	Objective	Booster	N jobs	Nthread	Gamma	Min child weight
Int = 3	Float = 0.1	Int = 100	True	Str = binary: logistic	Str = gbtree	Int = 1	Int = none	Float = 0	Int = 1

Max delta step	Subsample	Colsample bytree	Closample bylevel	Reg alpha	Reg lambda	Scale pos weight	Base score	Random state	Seed
Int = 0	Float = 1	Float = 1	Float = 1	Int = 0	Int = 1	Float = 1	Float = 0.5	Int = 0	Int = 0

Table 8
Parameters and configuration for quadratic discriminant analysis.

Priors	Reg param	Store covariance
None	0	Bool = False Tol = 1.0E−4

a

Table 9
Parameters and configuration for support vector machine.

C	Kernel	Degree	Gamma	Coef	Shrinking	Probability
1.0	Rbf	3	Auto deprecated	0.0	True	False

Tol	Cache size	Class weight	Verbose	Max iter	Decision function shape	Random state
1E−3	200	None	False	−1	ovr	None

Table 10
Parameters and configuration for long short-term memory (LSTM).

Return sequences	Input shape	Units	Activation	Loss	Optimizer	Metrics	Win
True	1,11	1	Tanh	Mse	Nadam	Accuracy	7

Chapter 7

IMPLEMENTAION

```
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

In [2]:
```
#Get the stock quote
df = pd.read_csv("bitcoin16.csv")
#Show teh data
```

In [3]:
```
#Get the number of rows and columns in the data set
df.shape
```

Out[3]: (43575, 7)

In [4]:
```
#Visualize the closing price history
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

In [5]:
```
#Create a new dataframe with only the 'Close column
data = df.filter(['Close'])
#Convert the dataframe to a numpy array
dataset = data.values
#Get the number of rows to train the model on
training_data_len = math.ceil( len(dataset) * .8 )

training_data_len
```

Out[5]: 34860

In [6]:
```
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

Out[6]:
```
array([[0.01445348],
       [0.01445348],
       [0.01445348],
       ...,
       [0.33851114],
       [0.33851114],
       [0.33851114]])
```

In [7]:
```
#Create the scaled training data set
train_data = scaled_data[0:training_data_len , :]
#Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60, len(train_data)):
  x_train.append(train_data[i-60:i, 0])
  y_train.append(train_data[i, 0])
  if i<= 61:
    print(x_train)
    print(y_train)
    print()
```

In [8]:
```
x_train, y_train = np.array(x_train), np.array(y_train)
```

In [9]:
```
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
x_train.shape
```

Out[9]:
```
(34800, 60, 1)
```

In [10]:
```
#Build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences= False))
model.add(Dense(25))
model.add(Dense(1))
```

In [11]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
```

In [*]:
```
#Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

```
WARNING:tensorflow:From C:\Users\kashi\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_
variables is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/1
21323/34800 [=================>............] - ETA: 8:27 - loss: 2.3740e-04
```

In [25]:
```
#Create the testing data set
test_data = scaled_data[training_data_len - 60: , :]
#Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
  x_test.append(test_data[i-60:i, 0])
```

Chapter 8

TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. In the testing stage following goals are tried to achieve: -

- To affirm the quality of the project.
- To find and eliminate and residual errors from previous stages.
- To validate the software as solution to the original problem.
- To provide operational reliability of the system.

8.1 Testing Methodologies

There are different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

8.1.1 White Box Testing

White box testing (clear box testing, glass box testing, and transparent box testing or structural testing) uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs. While white box testing is applicable at the unit, integration and system levels of the software testing process, it is typically applied to the unit. While it normally tests paths within a unit, it can also test paths betweenunits during integration, and between subsystems during a system level test.

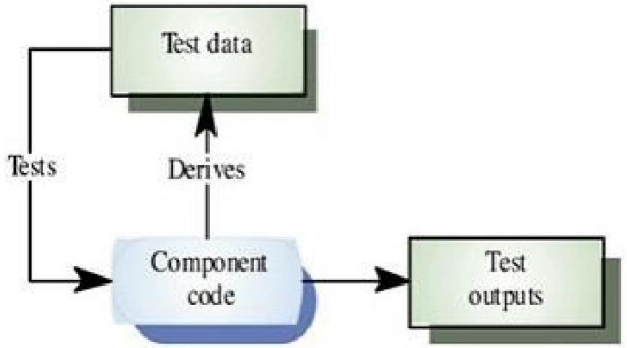


Figure 9 White box Testing

Though this method of test design can uncover an overwhelming number of test cases, it might not detect unimplemented parts of the specification or missing requirements, but one can be sure that all paths through the test object are executed. Using white box testing we can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structure to assure their validity

8.1.1.1 Advantages of White Box Testing

- To start the white box testing of the desired application there is no need to wait for user face (UI) to be completed. It covers all possible paths of code which will ensure a thorough testing.
- It helps in checking coding standards.
- Tester can ask about implementation of each section, so it might be possible to remove unused/deadlines of codes helps in reducing the number of test cases to be executed during the black box testing.
- As the tester is aware of internal coding structure, then it is helpful to derive which type of input data is needed to test the software application effectively.
- White box testing allows you to help in code optimization.

8.1.1.2 Disadvantages of White Boxing Testing

- To test the software application a highly skilled resource is required to carry out testing who has good knowledge of internal structure of the code which will increase the cost.
- Updating the test script is required if there is change in requirement too frequently.
- If the application to be tested is large in size, then exhaustive testing is impossible.
- It is not possible for testing each and every path/condition of software program, which might miss the defects in code.
- White box testing is a very expensive type of testing.
- To test each paths or conditions may require different input conditions, so in order to test full application, the tester need to create range of inputs which may be a time consuming.

8.1.2 Black Box Testing

Black box testing focuses on the functional requirements of the software. It is also known as functional testing. It is a software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs.

The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications. It enables us to derive sets of inputs that will fully exercise all functional requirements for a program.

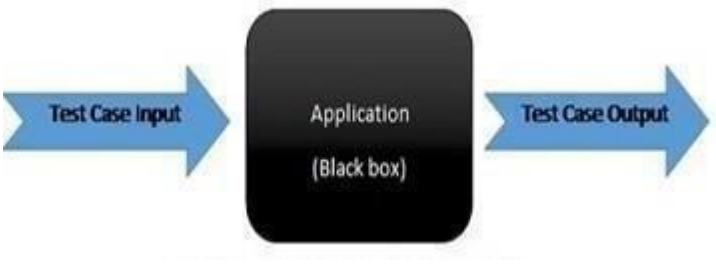


Figure 10: Black Box Testing

Black box testing is an alternative to white box technique. Rather it is a complementary approach that is likely to uncover a different class of errors in the following categories: -

- Incorrect or missing function.
- Interface errors.
- Performance errors.
- Initialization and termination errors.
- Errors in objects.

8.1.2.1 Advantages of Black Box Testing

- The test is unbiased as the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifications are complete.

8.1.2.2 Disadvantages of Black Box Testing

- The test inputs need to be from large sample space. That is, from a huge set of data this will take time.
- Also, it is difficult to identify all possible inputs in limited testing time. So, writing test cases is slow and difficult.
- Chances are more that there will be unidentified paths during this testing.

8.2 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and

internal code flow should be validated. It is the testing of individual software units of the application. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.3 System Testing

This information contributes towards reducing the ambiguity about the system. For example, when deciding whether to release a product, the decision makers would need to know the state of the product including aspects such as the conformance of the product to requirements, the usability ofthe product, any known risks, the product's compliance to any applicable regulations,

Software testing enables making objective assessments regarding the degree of conformance of the system to stated requirements and specifications.

System testing checks complete end-end scenarios, as a user would exercise the system. The system has to be tested for correctness of the functionality by setting it up in a controlled environment. System testing includes testing of functional and nonfunctional requirements. It helps to verify and validate the system. All components

system should have been successfully unit tested and then checked for any errors after integration.

8.4 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confident that the product quality is meeting its goals. This is an "umbrella activity" that is applied throughout the engineering process. Software quality assurance encompasses: -

- Analysis, design, coding and testing methods and tools

- Formal technical reviews that are applied during each software engineering
- Multi-tiered testing strategy
- Control of software documentation and the change made to it.
- A procedure to ensure compliance with software development standards.
- Measurement and reporting mechanisms.

8.4.1 Quality Factors

An important objective of quality assurance is to track the software quality and assess the impact of methodological and procedural changes on improved software quality. The factors that affect the quality can be categorized into two broad groups:

- Factors that can be directly measured.
- Factors that can be indirectly measured

These factors focus on three important aspects of a software product

- Its operational characteristics
- Its ability to undergo changes
- Its adaptability to a new environment.
- Effectiveness or efficiency in performing its mission
- Duration of its use by its customer.

8.5 Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : Interfacing systems or procedures must be invoked.

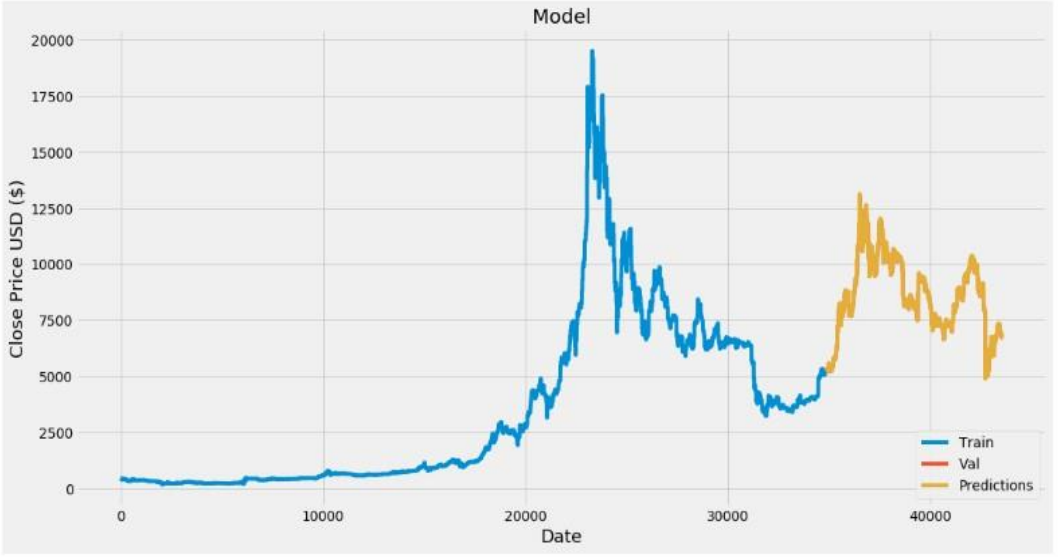
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

Chapter 9

RESULT AND PERFORMANCE ANALYSIS

9.1 Snapshots

```
In [15]: x_test = np.array(x_test)

In [16]: x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

In [17]: #Get the models predicted price values
         predictions = model.predict(x_test)
         predictions = scaler.inverse_transform(predictions)

In [18]: #Get the root mean squared error (RMSE)
         rmse=np.sqrt(np.mean(((predictions- y_test)**2)))
         rmse

Out[18]: 81.32658959270121

In [19]: train = data[:training_data_len]
         valid = data[training_data_len:]
         valid['Predictions'] = predictions
         #Visualize the data
         plt.figure(figsize=(16,8))
         plt.title('Model')
         plt.xlabel('Date', fontsize=18)
         plt.ylabel('Close Price USD ($)', fontsize=18)
         plt.plot(train['Close'])
         plt.plot(valid[['Close', 'Predictions']])
         plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
         plt.show()

C:\Users\kashi\anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```



In [20]:
```
#Show the valid and predicted prices
valid
```

Out[20]:

	Close	Predictions
34860	5251.937988	5244.632812
34861	5251.937988	5244.634277
34862	5298.385742	5244.635254
34863	5298.385742	5284.603516
34864	5298.385742	5292.470703
...	...	...
43570	6717.900391	6723.784180
43571	6717.900391	6723.757812
43572	6717.900391	6723.737305
43573	6717.900391	6723.723145
43574	6717.900391	6723.711914

8715 rows × 2 columns

In [ ]:
```
#Show predicted price of next day
```

In [22]:
```
apple_quote = pd.read_csv("bitcoin20.csv")
#Create a new dataframe
new_df = apple_quote.filter(['Close'])
#Get teh last 60 day closing price values and convert the dataframe to an array
last_60_days = new_df[-60:].values
#Scale the data to be values between 0 and 1
last_60_days_scaled = scaler.transform(last_60_days)
#Create an empty list
X_test = []
#Append teh past 60 days
X_test.append(last_60_days_scaled)
#Convert the X_test data set to a numpy array
X_test = np.array(X_test)
#Reshape the data
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
#Get the predicted scaled price
pred_price = model.predict(X_test)
#undo the scaling
pred_price = scaler.inverse_transform(pred_price)
print(pred_price)
```

```
[[6723.7036]]
```

Chapter 10

CONCLUSION AND FUTURE SCOPE

Deep learning models such as the RNN and LSTM are evidently effective for Bitcoin prediction with the LSTM more capable for recognizing longer-term dependencies. However, a high variance task of this nature makes it difficult to transpire this into impressive validation results. As a result, it remains a difficult task. There is a fine line between overfitting a model and preventing it from learning sufficiently. Dropout is a valuable feature to assist in improving this. However, despite using Bayesian optimization to optimize the selection of dropout it still couldn’t guarantee good validation results. Despite the metrics of sensitivity, specificity and precision indicating good performance, the actual performance of the ARIMA forecast based on error was significantly worse than the neural network models.