

A MAJOR PROJECT REPORT ON

PREDICTING BITCOIN PRICES USING

DEEP LEARNING

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF
DEGREE OF

**BACHELOR OF TECHNOLOGY IN ELECTRONICS AND
COMMUNICATION ENGINEERING**



Submitted By:

ESHA MAHENDRA (9916102024)

HARSHITA MADAN (9916102069)

SHIVANGI GUPTA (9916102126)

Under the Guidance Of

Dr. Sajaivir Singh

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING, JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY,
NOIDA (U.P.) May, 2020**

CERTIFICATE

This is to certify that the major project report entitled, “PREDICTING BITCOIN PRICES USING DEEP LEARNING” submitted by Esha Mahendra, Shivangi Gupta, Harshita Madan in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in **Electronics and Communication Engineering** of the Jaypee Institute of Information Technology, Noida is an authentic work carried out by them under my supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Signature of Supervisor:

Name of the Supervisor: Dr. Sajaivir Singh

ECE Department, JIIT, Sec-128, Noida-201304

Dated:

DECLARATION

We hereby declare that this written submission represents our own ideas in our own words and where others' ideas or words have been included, have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

Place: Noida

Date:

Name: Esha Mahendra

Enrollment: 9916102024

Name: Harshita Madan

Enrollment: 9916102069

Name: Shivangi Gupta

Enrollment: 9916102126

ABSTRACT

The quick improvement of computerized monetary standards during the most recent decade is one of the most disputable and questionable developments in the cutting edge worldwide economy. Huge variances in the conversion scale of digital currencies and their high unpredictability, just as the absence of lawful guideline of their exchanges in many nations brought about huge dangers related with venture into crypto resources. This has prompted warmed conversations about their place and job in the advanced economy. Along these lines, the issue of creating fitting techniques and models at foreseeing costs for bitcoin is applicable both for established researchers and for money related examiners, financial specialists and brokers. Methodological ways to deal with determining costs for monetary resources rely upon an expert's comprehension of the causal connections in the estimating procedure.

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project coordinator, Dr.Sajaivir Singh for his valuable inputs, guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department for encouraging and allowing us to present the project on the topic “PREDICTING BITCOIN PRICES USING DEEP LEARNING” at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree. We take this opportunity to thank all our lecturers who have directly or indirectly helped our project.

We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career. Last but not the least we express our thanks to our friends for their cooperation and support.

Signature:

Names:

Esha Mahendra (9916102024)

Harshita Madan (9916102069)

Shivangi Gupta (9916102126)

Table of Contents

Contents	Page No.
<i>Abstract</i>	<i>i</i>
<i>Acknowledgement</i>	<i>ii</i>
<i>Table of Contents</i>	<i>iii</i>
<i>List of Figures</i>	<i>iv</i>
CHAPTER 1: INTRODUCTION	1
1.1 Introduction to Deep learning	1
1.2 Introduction to Cryptocurrency and Bitcoin	2
1.3 Project Goal	3
CHAPTER 2: LITERATURE SURVEY	4
2.1 Forecasting cryptocurrency prices time series using machine learning approach	4
2.2 Predictive Analysis of Cryptocurrency Price Using Deep Learning	4
2.3 Predicting Bitcoin Prices using Deep Learning	4
CHAPTER 3: ALGORITHMS USED	6
3.1 Recurrent Neural Networks	6
3.1.1 Long Short Term Memory (LSTM)	6
3.1.2 Gated Recurrent unit (GRU)	8
3.2 Convolution Neural Network	10
3.3 Support Vector Machine	11
CHAPTER 4: DETAILED EXPLANATION	13
4.1 Softwares and libraries used	13
4.2 Application Programming Interface (API)	14
4.3 Work Flow	16
4.3.1 Building Deep Learning Models for Prediction	16
4.3.2 Deployment of Model using API	17
CHAPTER 5: RESULTS AND DISCUSSION	20
CHAPTER 6: CONCLUSION AND FUTURE SCOPE	27
References	
Appendices	

LIST OF FIGURES

S. No	Title	Page No.
1.1	Difference Between Machine Learning and Deep Learning	1
3.1	RNN Algorithm	6
3.2	LSTM Gates	7
3.3	GRU Gates	9
3.4	SVM Algorithm	12
4.1	Working of API	15
4.2	Workflow of the Proposed Approach	17
4.3	Flowchart Of the Model Deployment	18
4.4	Dataset of Bitcoin Price	18
4.5	API Design	19
5.1	LSTM Model	20
5.2	GRU Model	20
5.3	LSTM Prediction Matrix	21

5.4	GRU Prediction Matrix	21
5.5	Plot of LSTM Model	22
5.6	Plot of GRU Model	22
5.7	Mean Square Error	23
5.8	CNN Matrix	23
5.9	Plot between actual and predicted prices	24
5.10	Jupyter code line showing SVM Accuracy	24
5.11	SVM Predicted Matrix	24
5.12	Terminal Execution	25
5.13	API while giving input	25
5.14	API while giving output	26

CHAPTER 1: INTRODUCTION

1.1 Introduction to Deep Learning

Deep learning is a piece of a wide group of AI techniques which depend on fake neural systems. Deep learning techniques are supervised, semi- supervised or unsupervised. [1]

Deep learning systems, for example, deep neural systems, profound conviction systems, intermittent neural systems (RNN) and convolutional neural networks (CNN) have been applied to different fields including acknowledgment of discourse and sound, characteristic language handling, informal community clearing, machine version, clinical picture examination and material assessment where they have given outcomes like and in different cases higher than human specialists. [Fig.1.1]

Deep learning is a field of AI calculations that utilizes various layers to continuously remove more significant level highlights from the crude information. For instance, in picture handling, base layers may perceive edges, while top layers perceive the ideas appropriate to a human, for example, digits, letters or faces. [2]

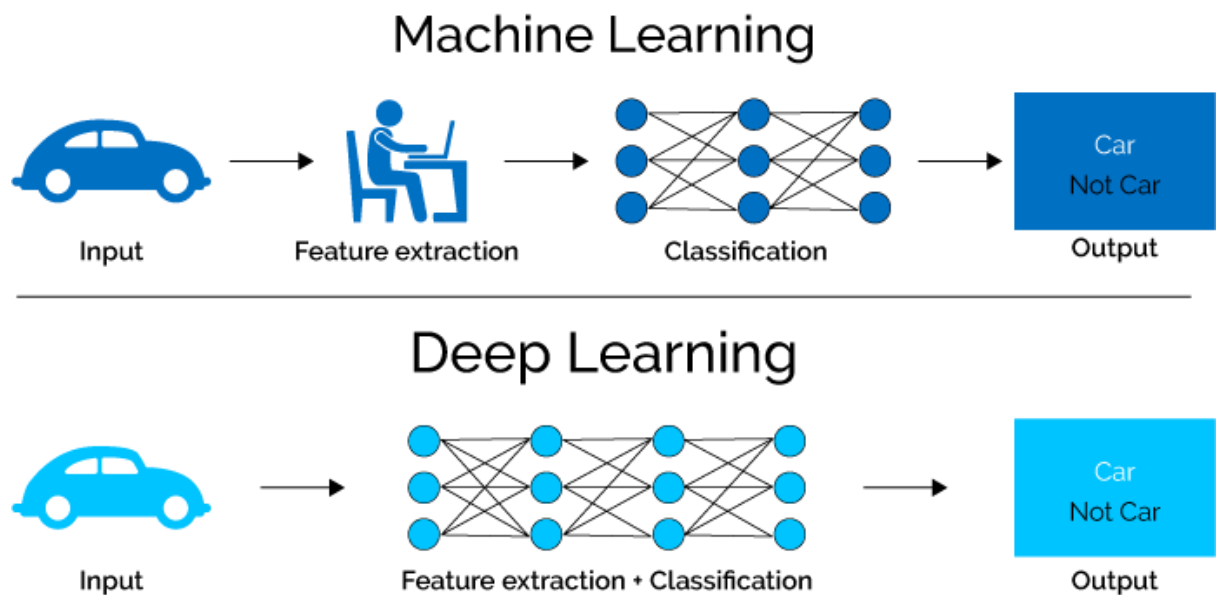


Fig. 1.1 Difference between Deep learning and Machine learning [2]

1.2 Introduction to Cryptocurrency and Bitcoin

Digital currency is a method for trade which is web based and utilizes cryptographic capacities to perform money related exchanges. The principle highlight of a cryptographic money is that it isn't constrained by any fundamental power: the circulated pith of the blockchain makes digital forms of money hypothetically invulnerable to the old methods of government control and obstruction. Digital currencies support blockchain innovation to pick up decentralization, straightforwardness, and permanence. Cryptographic forms of money can be posted straightforwardly between two gatherings by means of the utilization of private and open keys. These exchanges can be done with least handling expenses, allowing clients to keep away from the precarious expenses charged by conventional money related foundations. Bitcoin is an advanced installment that uses cryptographic money and distributed (P2P) mechanization to produce and oversee financial exchanges instead of a focal power. The open source Bitcoin P2P organize produces the bitcoins and controls all the exchanges of bitcoins. It is alluded as money for the Web, Bitcoin is one of the most well-known advanced installment monetary forms alongside Litecoin, Peercoin and Namecoin. Bitcoin is viewed as the greatest digital money. It was first presented in 2009 and is the most generally exchanged digital money. Bitcoin as an execution of the cryptographic money idea was developed by Wei Dai in 1998 in the cypherpunks mailing list. Dai proposed another type of cash that utilizes cryptography to control its creation and exchanges, as opposed to a principle focal power. In 2009, the Bitcoin detail and confirmation of idea was given in a cryptography mailing list by Satoshi Nakamoto. Installments are made through a Bitcoin wallet application that lives on a client's PC or cell phone, and an individual just needs to enter the beneficiary's Bitcoin address data and installment sum before sending to finish installment. [3]

New bitcoins are made by a serious and decentralized procedure called mining. Bitcoin diggers are preparing exchanges and making sure about the system utilizing specific equipment and are gathering new bitcoins in return. The Bitcoin convention guarantees new bitcoins are made at a fixed rate, making the procedure of bitcoin mining an extremely serious business. While assailants are following Bitcoin-related destinations, there is a significant qualification between the security of the Bitcoin organize and the Bitcoin trades.

1.3 Project Goal

Time arrangement estimating or expectation is a notable issue. Much research has been accomplished for anticipating markets, for example, the stock market. Digital forms of money can be viewed as a type of virtual cash proposed to fill in as a mode of trade and presents a fascinating point since it tends to be treated as a period arrangement expectation issue. This difficult despite everything stays in incipient stages. Thusly, there is high unpredictability in the market, and this offers open doors for additional exploration on the forecast of digital currency cost. In addition, digital currencies, for example, the Bitcoin are progressively received over the world. Due to the open idea of the digital money, it works on a decentralized, distributed, and trustless framework in which all exchanges are passed to an open record known as the blockchain. Such straightforwardness is obscure in the realm of old style money related markets.

The spirit reason for the venture to give a joke thought of how profound learning can be utilized to identify, foresee bitcoin costs at a lot quicker rate and consequently intricacy. The present work is centered primarily in two ways: Foreseeing the costs of bitcoin and understanding the example of digital money.

In this task, we will utilize the strategy of profound learning at forecast of bitcoin costs of the state-wise wrongdoing dataset. The bitcoin information is separated from the official site Kaggle.com. It comprises of the data like timestamp, open value, close value, significant expense, low cost and volume. Before preparing of the model information preprocessing will be finished after this component determination and scaling will be done so precision get will be high. The Long momentary memory (LSTM) and Gated Repetitive Unit (GRU) calculations will be tried at forecast of bitcoin costs and one with better exactness will be utilized for preparing.

When the calculations were handled, exactness of various calculations were estimated and the calculation with the most precision is utilized for the forecast. Perception of dataset is done as far as graphical portrayal is watched.

CHAPTER 2: LITERATURE SURVEY

2.1 Forecasting cryptocurrency prices time series using machine learning approach

In this paper, it depicts the development of the transient estimating model of cryptographic forms of money costs utilizing AI approach. The altered model of Paired Auto Backward Tree (BART) is adjusted from the standard models of relapse trees and the information of the time arrangement. BART joins the exemplary calculation order and relapse trees (C&RT) and autoregressive models ARIMA. Utilizing the BART model, they have made a transient estimate (from 5 to 30 days) for the 3 most promoted cryptographic forms of money: Bitcoin, Ethereum and Wave. We found that the proposed approach was more precise than the ARIMA-ARFIMA models in anticipating digital forms of money time arrangement both in the times of moderate rising (falling) and in the times of progress elements (change of pattern). [3]

2.2 Predictive Analysis of Cryptocurrency Price Using Deep Learning

This paper proposes a novel strategy to foresee cryptographic money cost by considering different factors, for example, advertise top, volume, coursing gracefully, and greatest flexibly dependent on profound learning methods, for example, the repetitive neural system (RNN) and the long momentary memory (LSTM), which are successful learning models for preparing information, with the LSTM being better at perceiving longer-term affiliations. The proposed approach is actualized in Python and approved for benchmark datasets. The outcomes check the appropriateness of the proposed approach for the precise expectation of digital currency cost. [4]

2.3 Predicting Bitcoin Prices using Deep Learning

The objective for this research is to show how a prepared machine model can foresee the cost of a digital money on the off chance that we give the perfect measure of information and computational influence. It shows a diagram with the anticipated qualities. The most mainstream innovation is the sort of mechanical arrangement that could assist humankind with foreseeing future occasions. With tremendous measure of information being produced and recorded consistently, we have at long last approached a time where forecasts can be exact and be created dependent on concrete truthful information. Moreover, with the ascent of the crypto computerized time more heads have turned

towards the advanced market for ventures. This offers us the chance to make a model equipped for foreseeing cryptographic forms of money basically Bitcoin. This can be cultivated by utilizing a progression of AI strategies and procedures. Building calculations and models to anticipate costs and future occasions has been given noteworthy measure of consideration in the previous decade. With client information being gathered through different types of ways, there has never been a plenitude in crude information like there is currently. Any model equipped for foreseeing a future occasion whether it be to discover what the following huge pattern is or to anticipate the following conduct of an anything have an incredible potential in this day and age. [5]

CHAPTER 3: ALGORITHMS USED

3.1 Recurrent Neural Networks

The Time-Series data is the sequential data where the occurrence of the events is defined by time. If some reasonable output is to be achieved by using the data, a network is required which has access to prior knowledge to completely understand the data. This is achieved through Recurrent Neural Network. It uses its internal state(memory) for processing the sequential data. There are mainly 3 layers: The Input layer which receives the input, hidden layer which is characterized by its own weights and biases and activation function is applied here and the last is the output layer where the output is received. Fig [3.1]

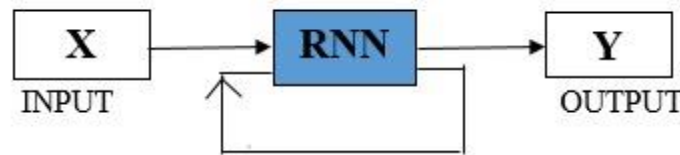


Fig. 3.1 RNN Algorithm

Using Forward propagation, the input neuron transforms the input to the hidden state using weights while the weights are updated using back propagation. The result of the information is dependent on the previous state. This type of neural network suffers the problem of vanishing gradient when the long term dependencies are present. The training of data also becomes difficult as the parameters are extremely large. To solve this problem, many variations have been developed. The LSTM and GRU have been developed to solve the long term dependencies.

3.1.1 Long Short Term Memory (LSTM)

The Long Short Term Memory is the advancement of recurrent neural network architecture used in building deep learning models which deal with the problem of vanishing gradient of RNN. LSTM differs from RNN in module architecture. RNN has single neural network layer while LSTM has four layers interacting to produce output. The Cell state is the key of the LSTM architecture. The structures responsible for the flow of information to cell state are called gates. The three gates control and protect the cell state by letting only the useful information to pass through.

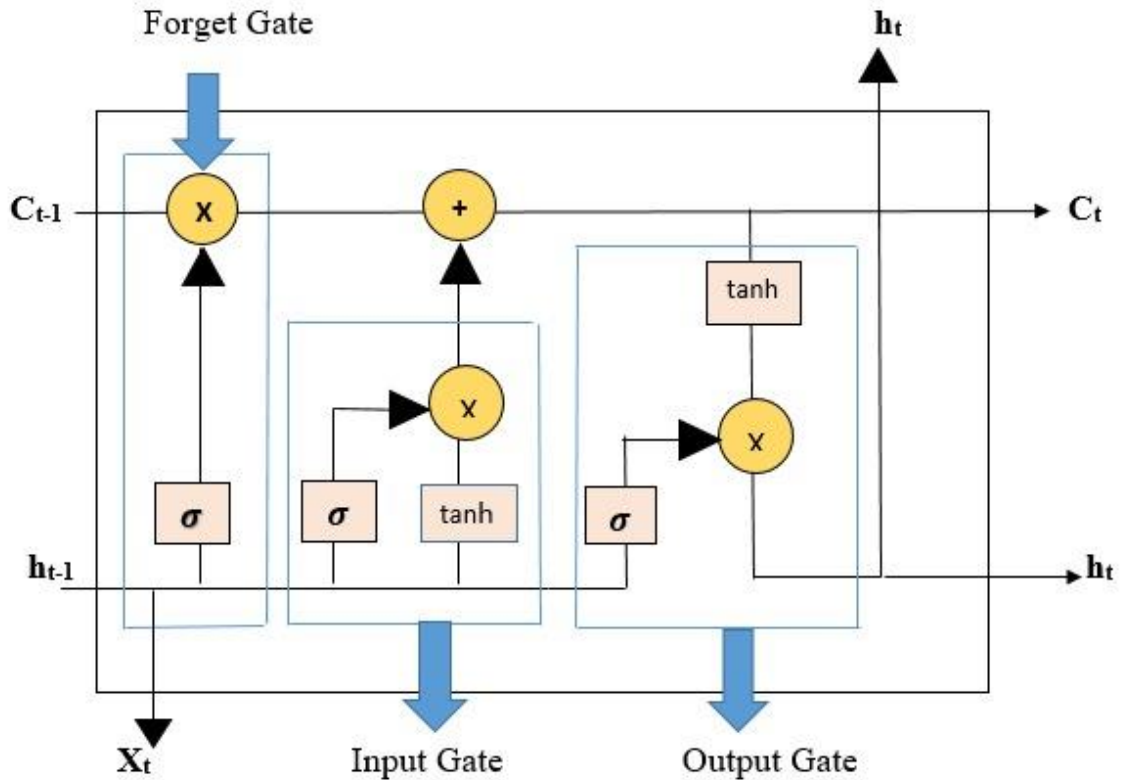


Fig 3.2 LSTM Gates

The above figure shows the LSTM unit consisting of the gates which regulate the information before producing the output. X_t represents the current input while h_t represents the current hidden state calculated. h_{t-1} represents the previous hidden state which is most important in deciding which values are to be updated in current state and C_{t-1} denotes the cell state of previous time step. [Fig 3.2] The function of each gate is explained below:

1. **Forget Gate:** As the name suggests, the forget gate keeps the important information to pass to next gate while the irrelevant information is thrown away from block. The hidden state of previous time step and the current input state are passed to sigmoid function and the resulting value is obtained as 0 or 1. The value close to 0 represents the information that is to be forgotten while the value closer to 1 represents the value to keep.
2. **Input Gate:** The gate next to forget gate is the input gate. It decides the value of input to keep or to throw away to update the cell state. Firstly, the current input and the previous hidden state is passed through sigmoid function which transforms the value between 0 and 1. This is helpful in deciding which values are to be updated. To regulate the network, the above two values are passed through the tanh function which make the values between -1 to 1. The sigmoid output and the tanh output are finally multiplied to produce final output suggesting the values to be update.

3. Output Gate: The output gate decides the next hidden state. This hidden state is the most important in any neural network architecture as it is used for predictions. The reason behind is that the hidden state contains the information on previous inputs. First, the hidden state of previous time step and the current input is pass through the sigmoid function. Then the newly modified cell state is passed through the tanh function so that the value becomes between -1 to 1. The hidden state is calculated and the important information to keep and which to throw away is decided when sigmoid output is multiplied with the tanh output. The modified cell state and the current hidden state are then carried to the next time step. [6]

The cell state is the key to LSTM architecture. It runs along the whole chain and without any minor interactions it is possible for the information to just flow through it without being unchanged. The regulation of the information into it is carried out by gates. The new cell state in each time step is obtained by multiplying the cell state of previous time step with the forget vector. In this stage, if the value come to approximately zero then the cell state is dropped. To get the values which cell state find relevant, the pointwise addition is carried out with the output of the input gate. The same process of all the three gates are carries out for the given time steps. To reduce the number of gates in LSTM architecture and thereby reducing the complexity a lot of research is being carried out.

3.1.2 Gated Recurrent Unit (GRU)

Gated Recurrent Unit is the advance version of LSTM neural architecture that also solves long term dependencies problem. The LSTM and GRU neural networks produce same results at times but differ in their architecture. Both possess the ability to remember the previous state rather than replacing the entire activation as that of RNN architecture which make them capable to predict data having long term dependencies. The gated architecture in GRU network helps to transfer the important information while neglecting the irrelevant information. The cell state in GRU network is absent and the hidden state is used to transfer the information between the different time steps. This network is capable to handle both the long-term and short-term dependencies at the same time. The three gates present the LSTM network is replaced by two gates in GRU thus reducing its complexity. The GRU cell has only 2 gates: Reset Gate and Update Gate. These two gates are basically vectors that are trained to throw out any irrelevant information and the important information to kept for the next time step. The combined operation of input and forget gate in LSTM is done with the reset gate in GRU network. [7]

The details of the gates are explained below:

1. **Reset Gate:** This gate is useful in deciding which past information to forget. This gate is calculated by the hidden state from the hidden state of previous time step and the input data of the current time step. Mathematically, this happens when the hidden state from previous time step and the input from the current time step is multiplied by their respective weights and then summing them together before passing them through the sigmoid function. The entire network is trained through back propagation and the weights are updated. The hidden state of previous time step is multiplied with the current time step weight and then the Hadamard product is taken with the reset vector. The current input is multiplied by the trained weight and then summed to the previous product. The output is then pass through the tanh function as it squishes the value between -1 to 1 for the regulation of the network.

2. **Update Gate:** This gate in GRU network decides which information to keep and which to through away in a time step. It carries out the same mechanism as the forget and input gate does in the LSTM architecture. The output value of this gate is calculated using hidden state of previous time step and the current input. The update gate vector is calculated using the same formula for the reset gate but the weights multiplied with the hidden state and the current input are different so that the two gates serve their own purpose. The main purpose is to determine the relevant information that must be kept for the future time steps and which can be helpful in maintaining predictions. Fig [3.3]

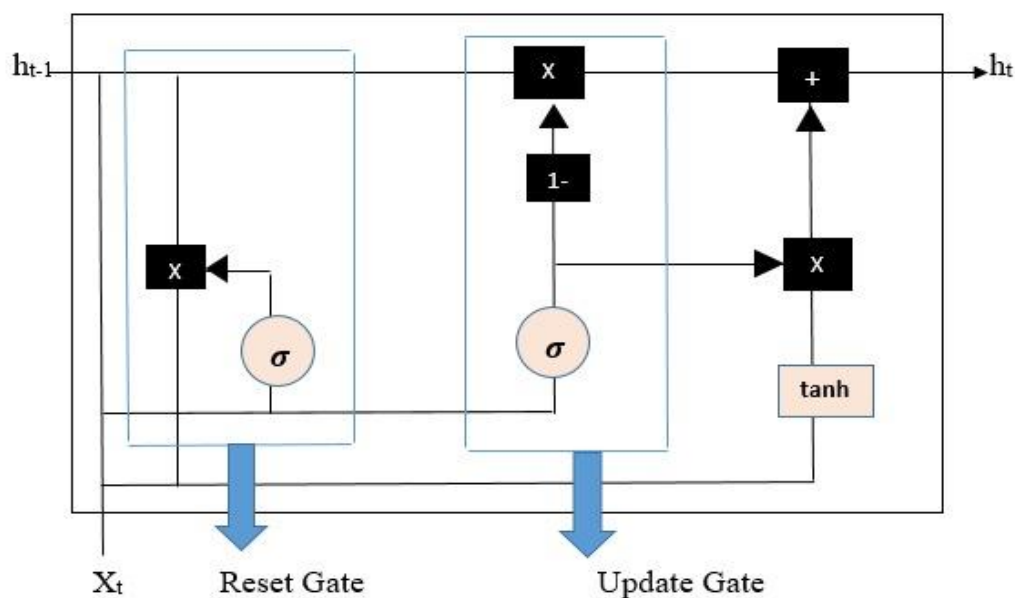


Fig 3.3 GRU Gates

In the above GRU unit cell, h_{t-1} represents the hidden state of previous time step, X_t represents the input value of data in current time step and h_t represents the hidden state of current time step which would be passed to next time step. The reset gate performs the operation which the input and the forget gate does in LSTM thereby reducing the number of operations and simplifying the architecture to reduce complexity. The short term and the long term dependencies are handled at the same time in GRU network. This makes the model to train sequential data at the faster rate. The gated mechanism in this neural network eliminate the problem of vanishing gradient problem. [8]

3.2 Convolution Neural Network

A convolution neural network (CNN) is another most important algorithm in deep learning which has also done many advancements in the field of computer vision. As the name suggest, this neural network is somewhat related to mathematical term convolution. The preprocessing of data is much lower as compared to other algorithms in deep learning and therefore mainly used in image analysis. The architecture of CNN is similar to fully connected networks as that of neurons inside the brain. Single neuron of a layer is connected to the all neurons in the nearby layer. This fully connected networks cause the overfitting of the data. This network lacks the capability to remember thereby consider only the input and not the hidden state as that of in RNN. The architecture of CNN contains the four layers which are as follows:

1. **Convolution Layer:** This layer is considered first in building the CNN model as it is the key of this neural network. Through the convolution filtering, this layer detects the set of features in an image received as input. The filters are the building block of CNN architecture. With the help of filters, the convolution layer calculates the convolution of several images which are received as input and then form the feature map. The features are learnt during the training phase. The convolution layer weights are referred to as filters which are updated during back propagation using various gradient.
2. **Pooling Layer:** This layer reduces the computation power required to process the data by reducing the dimensions of the image through which effective learning can be achieved. It is stacked between the two convolution layers and then applying the pooling operations to several feature maps which it receives preserving the dominant features of the image. The number of feature map are same as applies in the input but are relatively small. This is helpful in reducing the number of parameters to be trained and thereby increasing the efficiency of the model.

3. **ReLU Correction Layer:** ReLU stands for rectified linear units. This layer acts as an activation layer by changing all the values from negative to zero.

4. **Fully connected Layer:** To learn the non-linear features as represented by the output of the convolution layer, it is added to the network. This type of layer receives the input vector and the output vector is produced by applying the linear combinations and the activation functions over the input. [9]

These four layers are parameterized differently and the hyper parameters used by the convolution and pooling layers are defined by the user first before building the model. These convolution layers are stacked differently making it different from other neural networks used in deep learning. This algorithm is used in our project to show that CNN doesn't produce good predictions while dealing with the sequential data i.e. Time-series data but best suited for image analysis.

Differences Between CNN and RNN Algorithm are:

1. The CNN algorithm is used for discrete data such as images while the RNN algorithm is used for sequential data i.e. Time-series data.
2. CNN is considered to have more features compatibility as compared to RNN.
3. CNN comes under the category of feed-forward neural network takes fixed sized input and cannot handle random inputs and output produce is also of the same size while RNN can handle random input/output length.
4. RNN uses internal memory to process sequential data whereas CNN produces output with minimal preprocessing of data.
5. RNN has architecture similar to neural networks while the CNN differs as its layers are stacked differently.
6. RNN is good at natural language processing while CNN is the most important algorithm used in computer vision along with deep learning.

3.3 Support Vector Machine

Support vector machine has gained a lot of importance in the field of machine learning as it has proved to be a good prediction algorithm to predict the stock prices as compared to other back propagation algorithm. This algorithm can both be used in classification as well as regression problem but is mostly used in the classification problem. In SVM, each data value is plotted in n-dimensional space with each feature represented as the coordinate in the space. The two classes are differentiated by classification which forms the hyper-plane for differentiates the classes very well. The hyper plane is identified by thumb rule which states to choose the hyper plane which segregates the two classes in a better way. It can also be used in non-separable problems where it converts the problem to separable problem using the kernel function. This is known as kernel trick.

The major advantage of SVM is the global optimum because the other artificial neural networks are local optimum. The second most important thing is that it works well for the separable data and where the numbers of dimensions are greater than the samples but becomes unfit when large data is to be trained. This algorithm also has more classification errors as compared to other neural network and doesn't perform well with the continuous sequential data. This algorithm is used in the project to showcase that for the time series data which is sequential as well as continuous the machine learning algorithms like SVM don't work here. The algorithm took more time to train the data as well as prediction is of low accuracy. Fig [3.4] For the above reasons, it is clear that it is better to use deep learning algorithms when dealing with the sequential data which is time dependent as in the case of Time-series data. In the below figure, A, B, C are the hyperplanes the space where B is well suited hyperplane since it segregated the two classes better. [10]

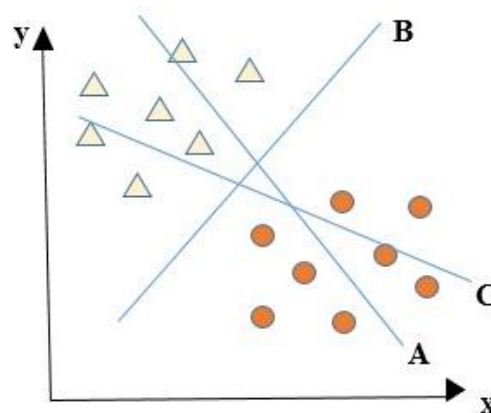


Fig 3.4 SVM Algorithm

CHAPTER 4: DETAILED EXPLANATION

4.1 Software and Libraries Used

- **ANNACONDA:** The most popular python distribution platform for data science and machine learning along with dealing the large data and its predictions is one and only Anaconda. It provides various environments such as Spyder, Jupyter notebook, Jupyter lab, Pycharm and these can be run and maintained separately without any interference. There are around 7500 packages available in its cloud repository. Instead of using the command-line commands one can use the Anaconda Navigator which is Desktop GUI automatically installed with the Anaconda individual edition for the launching of applications and managing the environment. Being an open-source community it provides tools where data can be fetched easily from any available databases and this data can be used to build data science project that can be further deployed for its production. [11]
- **SPYDER:** Spyder is a cross-platform Integrated Development Environment(IDE) developed in 2009 for scientific programming in Python. This open source IDE is available through Anaconda. It uses Qt for GUI. The core building blocks of this IDE are: Editor, IPython Console, Variable Explorer, Profile, Debugger and help. Numpy, Scipy, Pandas, IPython, QtConsole, Matplotlib are among the most popular scientific packages offered by this IDE to build Machine learning models. The deep learning models in the project are built using this software and predictions of prices has been done using various libraries which are supported by this IDE.
- **FLASK:** Flask is a web framework written in Python which allow user to use its tools and libraries to build we applications such as web-pages, blog or a wiki. It is the Unicode-based development server and debugger that has Integrated support for unit testing and RESTful request dispatching. It uses Jinja templating and also as Google App Engine compatibility. It provides the template engine that save a lot time in building web applications. This framework is used in the project for the deployment of the built model.
- **KERAS:** Keras is the open-source Python library which is used to build deep learning models and do training by the data provided. It is the Tensor flow's high level API that uses Tensor flow or Thenao as its backend engine. It contains tools such as layers, activation function, optimizers and objective to build the neural network blocks.

Keras library supports recurrent and convolution neural network along with the standard ones. There are two types of model in Keras data structure: Sequential Model Class and the Model class used with the function API. Keras is being used widely because of its ease of the model building, supported by various backend engines and most importantly support multiple GPUs.

- **MATPLOTLIB:** Matplotlib is the visualization library for the Python Programming language for 2D plots of the array. It is used as an alternative of MATLAB when used along with NumPy. With GUI toolkit, it can also be used for embedding plots into the applications. It has various plots such as line, histogram, scatter plot for visualizing data.
- The **Pandas** and **Numpy library** in data science are used for the data cleaning, its manipulation and analysis. Pandas has 2D table object called data frame to store data while Numpy provide objects for multidimensional arrays.

4.2 Application Programming Interface

Application programming interface (API) is a kind of software which is used in building software applications. It is the set of protocols, routines and tools that helps the two software's to interact between themselves without knowing their individual implementation. [12] API's are also used to program the Graphical User Interface (GUI) components. It basically provides all the building blocks for a program thus making it easier for a programmer to design new tools and the products. It provides a way to simplify design, increasing flexibility and usage of the web app. API's play a major role in collaboration of business and IT teams as they provide an easier way to connect the application to the cloud-native app development. [Fig. 4.1] There are mainly 4 different types of API as follows:

1. **OPEN/PUBLIC API:** They are available to users without any restrictions to enable the access to data and services. They can be accessed via any activation key or the user registration or can be open completely.
2. **PARTNER API:** To access the partner API, one has to follow the specific validation workflow as the on-boarding process to use it in a self-service mode. They are the most visible API's on the developer portal which is used beyond company boundaries.
3. **INTERNAL/PRIVATE API:** This API is used within the internal departments of the company for re usage of services and productivity and is not available for the outside users.
4. **COMPOSITE API:** This API is used to access several end points by developers using

single call. This task is achieved by combining multiple data using the API creation tool. This is useful to perform single task where the information required to do is taken from various other services.

To call an API there are certain set of rules. These set of rules that specify commands and accepted data types to call an API are called as protocols. [13] The major types of protocol used are listed below:

1. **REST:** REST API is the short for Representation State Transfer. They are useful in building web applications. Some of the well-known web app which have used this API are Amazon, Netflix, Uber. Because of its simplicity, portability and performance this API stands out from the other API's. This API is used where there is a communication using the HTTP protocol between the server and JSON conventions are used in designing. As this API is stateless in nature it is supported by a layered architecture. Using this API, the user experiences faster services on a condition that the responses are being cached.
2. **SOAP:** Simple object access protocol (SOAP) is similar to REST API and a type of WEB API. It is used in managing services through network connections by providing the way one should use them through applications. It has gained a lot popularity in late 1990's before development of REST API. This API has rigid standards and becomes heavy when used in web applications.
3. **RPC:** Remote procedural call protocol (RPC) is the simplest protocol. The most important task is to execute code on server. E.g.: For JSON, JSON-RPC is used for encoding. This API is similar to REST API except of the fact that it is difficult to update or to maintain it. To do any change one has to go through the RPC's documentation to observe the effect of changes of one other. [14]

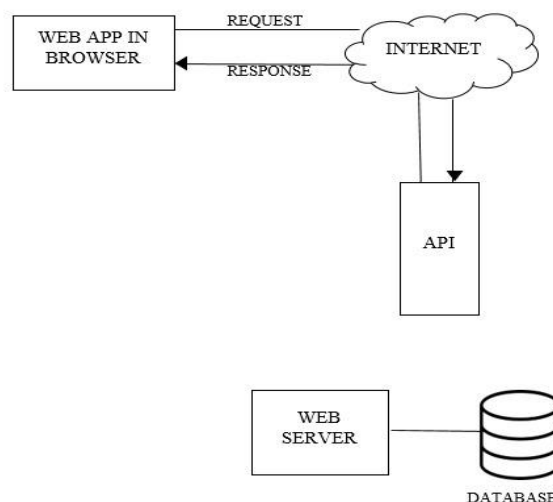


Fig. 4.1 Working of API [14]

4.3 Workflow

4.3.1 Building Deep Learning Models for Prediction

The proposed approach for building the deep learning model to predict bitcoin coin prices involved various steps as discussed below:

1. **GATHERING DATA:** The dataset requirement entirely depends on the project requirement. The dataset can be collected from various sources such as file, database or even a sensor but the data cannot be used directly for building the model. Data preprocessing is therefore considered as the most important step. [15]
2. **DATA PREPROCESSING:** This is the most important step in deep learning which helps in building the model more accurately to perform the analysis. In this step, data collected from the site is converted to the clean data set. The missing values in the dataset can either be removed or can be filled manually. The ordinal or the categorical data can be converted to the numerical data whereas the null or the irrelevant values is removed. Then the data is split into testing set and training set. For e.g.: data can be divided into 80% of training data while 20% testing data.
3. **DATA SCALING PHASE:** In this step the data is scaled according to model requirements. It reshapes data to make the model more suitable.
4. **MODEL-BUILDING PHASE:** The preprocessed data is used to build the best performing model. Using Tensor flow as the backend engine, Keras is used to increase the accuracy of the prediction model. The Keras sequential model contains layers which process the data and analyze the different patterns in dataset to make the model more precise. [16]
5. **MODEL LEARNING PHASE:** After the training data is defined it is configured with the defined model to start the learning phase. This is achieved using the **compile() method** of the Keras sequential model. After the fully configured neural network is defined the data is passed to the model for training. This is achieved by calling **fit() method** which takes the input and target tensors as the arguments. In the project, data is trained using various LSTM and GRU units. [17]
6. **EVALUATION:** This is the integral part as it helps in finding the best model that represents data and how well the prediction is achieved. Input values are passed to the model and the output are the predicted values. The output is compared with the testing data to calculate accuracy and the RMSE values. The accuracy can be improved by tuning the hyper parameters of the model. [Fig.4.2]

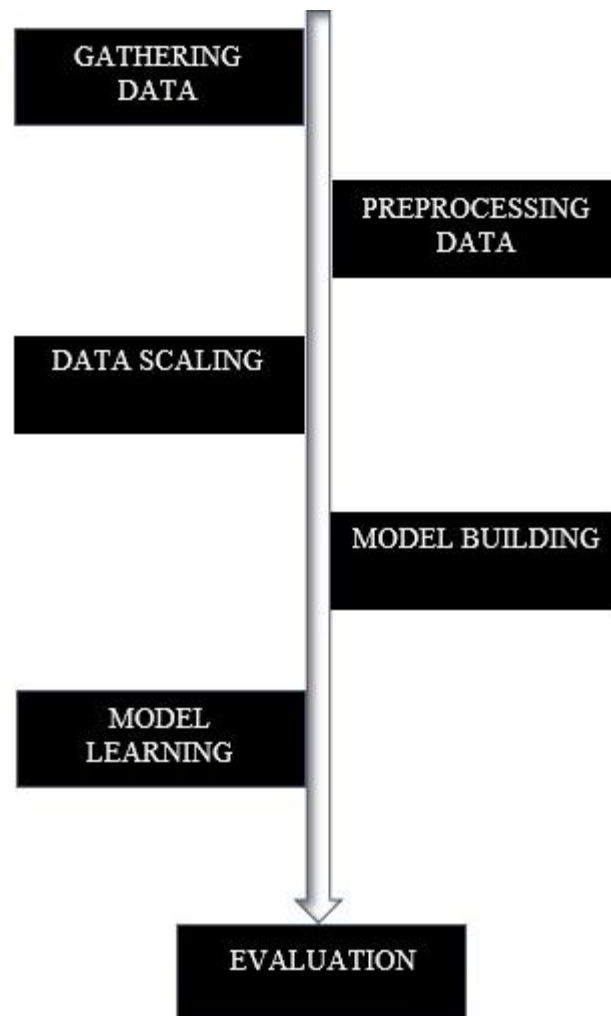


Fig. 4.2 Workflow of the Proposed Approach [17]

4.3.2 Deployment of Model using API

The LSTM and the GRU model is built in Spyder using various libraries and can be used by the user who created it but by its deployment the model can be used by other users. This model is deployed to the website using Python and Flask. [Fig. 4.3] Following steps are followed for the deployment. [18]

1. The file that contains code of LSTM model for the bitcoin price prediction is saved in the same location as the below files are created.
2. The other file is created which contains Flask APIs that receive data when the user input through GUI or API calls.
3. The request module is to be created inside the file which is used to call API from the file to display the returned value.
4. Lastly the HTML templates and CSS styling is done to develop the GUI of Flask and stored in the file of the same directory as that of the above files. [19]

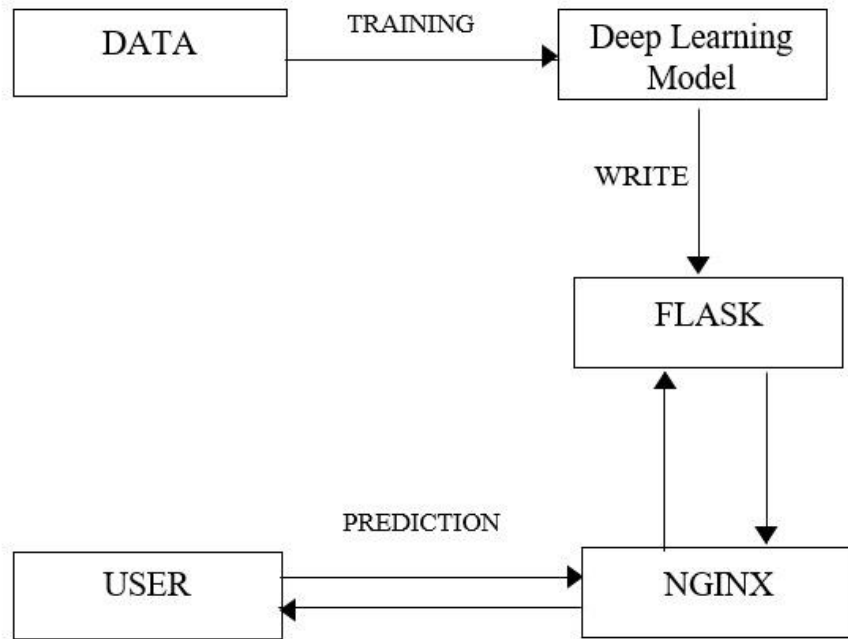
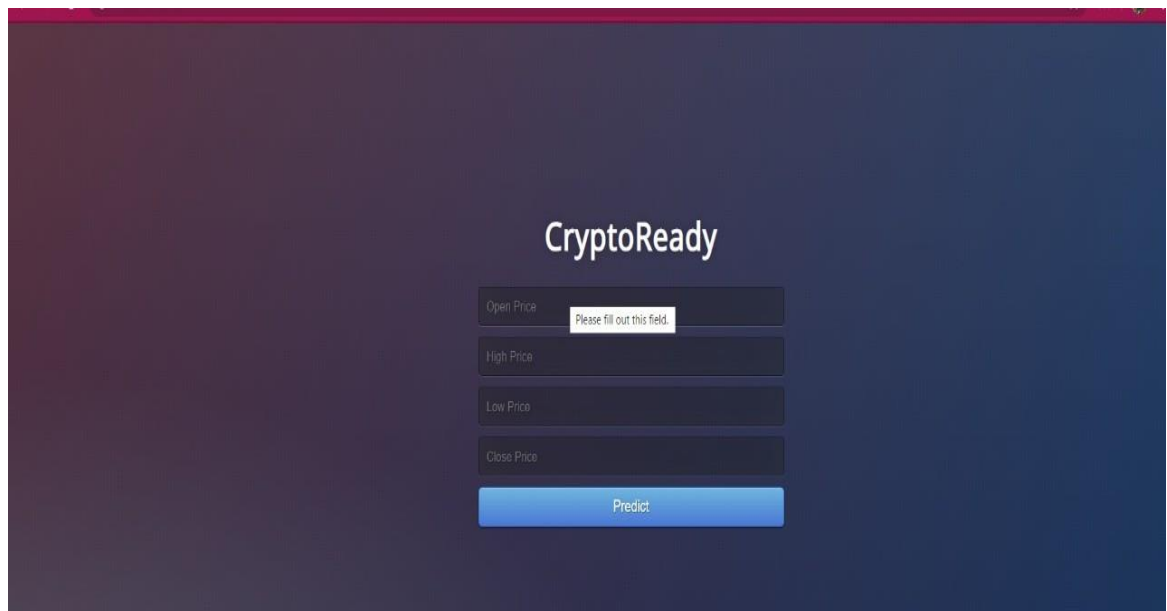


Fig. 4.3 Flowchart of Model Deployment [19]

The dataset of the bitcoin prices used in the project to build deep learning model is downloaded from kaggle.com. [Fig.4.4] The CSV file has prices based upon various factors such as open, high price, closing market price, market volume and the weighted price. [20] The snapshot of the same is as below:

A8		1/7/2017 12:00:00 AM								
	A	B	C	D	E	F	G	H	I	J
1	Timestamp	Open	High	Low	Close	Volume (B	Volume (C	Weighted Price		
2	#####	966.34	1005	960.53	997.75	6850.59	6764742	987.47		
3	#####	997.75	1032	990.01	1012.54	8167.38	8273577	1013		
4	#####	1011.44	1039	999.99	1035.24	9089.66	9276500	1020.56		
5	#####	1035.51	1139.89	1028.56	1114.92	21562.46	23469645	1088.45		
6	#####	1114.38	1136.72	885.41	1004.74	36018.86	36211400	1005.35		
7	#####	1004.73	1026.99	871	893.89	27916.7	25523261	914.26		
8	#####	894.02	907.05	812.28	906.2	20401.11	17624310	863.89		
9	#####	906.2	941.81	881.3	909.75	8937.49	8168170	913.92		
10	#####	909.8	912.87	875	896.23	8716.18	7780059	892.6		
11	#####	896.09	912.47	889.41	905.05	8535.52	7704271	902.61		
12	#####	905.24	918.4	755	778.62	35893.77	29459969	820.75		
13	#####	778.7	832.99	751.34	807.47	17400.14	13632251	783.46		
14	#####	807.51	831.4	775	825.86	11409.52	9224730	808.51		
15	#####	825.98	837.76	810	818.27	6614.72	5470215	826.98		
16	#####	819.52	823.45	808	821.86	4231.46	3455366	816.59		
17	#####	821.86	835	818.09	831.81	6166.04	5107031	828.25		
18	#####	831.76	908.5	827	905.99	12264.17	10775498	878.62		
19	#####	905.95	915.99	851.74	887.7	11181.9	9830118	879.11		
20	#####	887.69	910	878.89	901.01	11094.6	9929647	895		
21	#####	902.23	902.43	880	895.8	6618.63	5915865	893.82		
22	#####	895.81	928	895	921.98	5865.63	5373391	916.08		
23	#####	921.98	937.74	886.76	923.76	7166.67	6569177	916.63		
24	#####	923.75	927.57	913.21	913.52	3514.74	3234454	920.25		

Fig 4.4 Dataset of Bitcoin Price [20]



The image shows a web form titled "CryptoReady" on a dark blue gradient background. The form is centered and consists of four input fields stacked vertically, each with a label to its left: "Open Price", "High Price", "Low Price", and "Close Price". The "Open Price" field is currently empty and has a small white error message "Please fill out this field." displayed next to it. Below these four fields is a blue button with the text "Predict" in white. The entire form is enclosed in a dark blue border.

Fig. 4.5 API Design

CHAPTER 5: RESULTS AND DISCUSSION

The first algorithm implemented was LSTM. This was the primary algorithm useful in deep learning for predicting Bitcoin Prices. It had of 3 LSTM layers and 1 Dense layer. It consisted of 75 neurons in 1st LSTM layer and 30 each neuron in subsequent 2nd and 3rd layer. The final and output layer had 1 neuron. The model was of 50 training epochs and a batch size of 8. Its RMSE value came to be 0.022.

The next algorithm implemented was GRU. As GRU is the advanced version of LSTM so it had same structure as LSTM but different output. LSTM had more number of parameters while GRU had less, so this makes GRU less complicated and more efficient. Its RMSE value came to be 0.014.

```
In [78]: runfile('C:/Users/user/Desktop/tryinglstm.py', wdir='C:/Users/user/I
```

Layer (type)	Output Shape	Param #
lstm_48 (LSTM)	(None, None, 75)	23100
lstm_49 (LSTM)	(None, None, 30)	12720
lstm_50 (LSTM)	(None, 30)	7320
dense_72 (Dense)	(None, 1)	31

```
Total params: 43,171  
Trainable params: 43,171  
Non-trainable params: 0
```

Fig. 5.1 LSTM Model

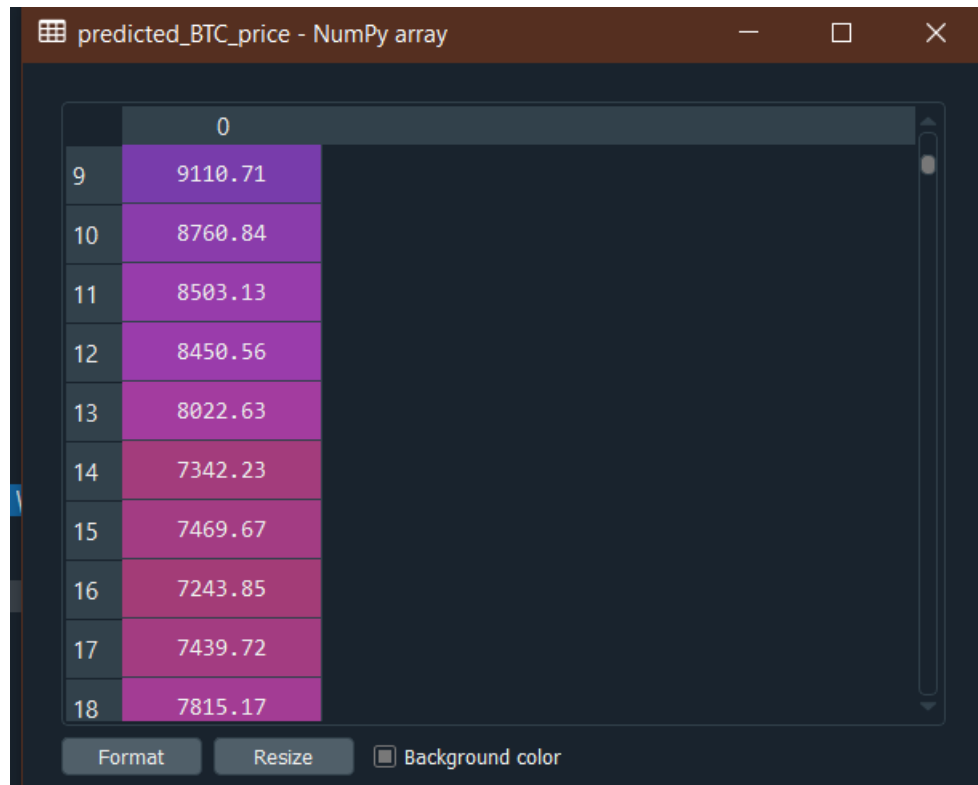
Layer (type)	Output Shape	Param #
gru_42 (GRU)	(None, None, 75)	17325
gru_43 (GRU)	(None, None, 30)	9540
gru_44 (GRU)	(None, 30)	5490
dense_74 (Dense)	(None, 1)	31

```
Total params: 32,386  
Trainable params: 32,386  
Non-trainable params: 0
```

```
Test RMSE:0.014
```

Fig. 5.2 GRU Model

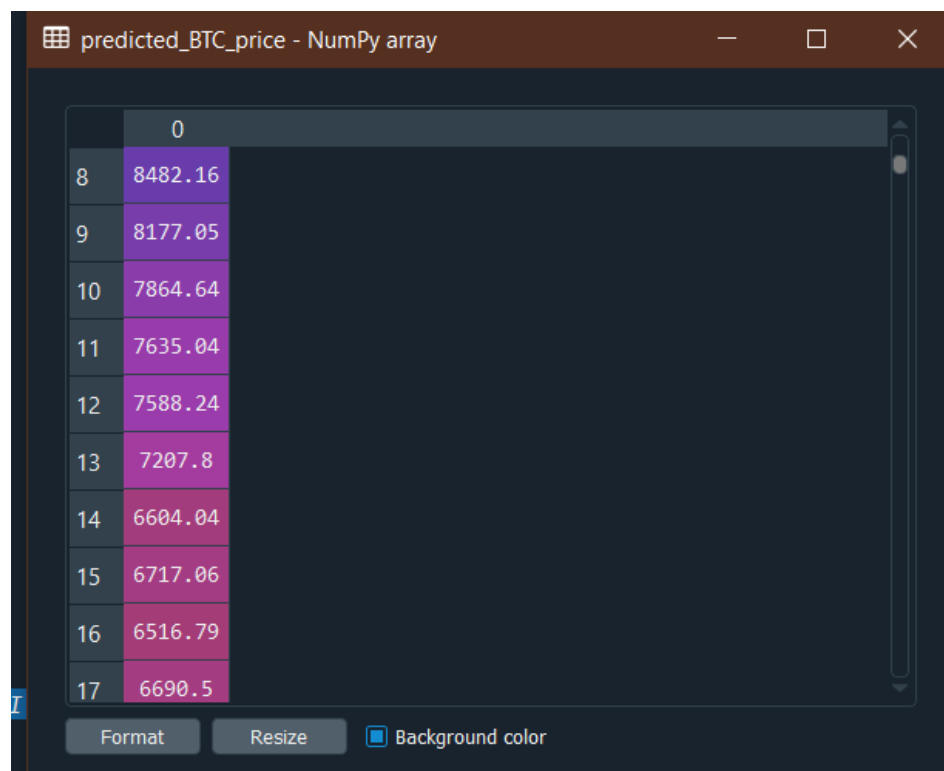
Following are the few cells of predicted matrix of LSTM Model.



	0
9	9110.71
10	8760.84
11	8503.13
12	8450.56
13	8022.63
14	7342.23
15	7469.67
16	7243.85
17	7439.72
18	7815.17

Fig 5.3 LSTM Prediction matrix

Following is the Predicted matrix of GRU Model showing few cells out of all.



	0
8	8482.16
9	8177.05
10	7864.64
11	7635.04
12	7588.24
13	7207.8
14	6604.04
15	6717.06
16	6516.79
17	6690.5

Fig. 5.4 GRU Prediction matrix

The prediction is done for 240 days starting from 12-04-2017 and the BTC price versus Time for the actual and the predicted data is plotted using matplotlib library for the two models as shown below:

BTC PRICE PREDICTION

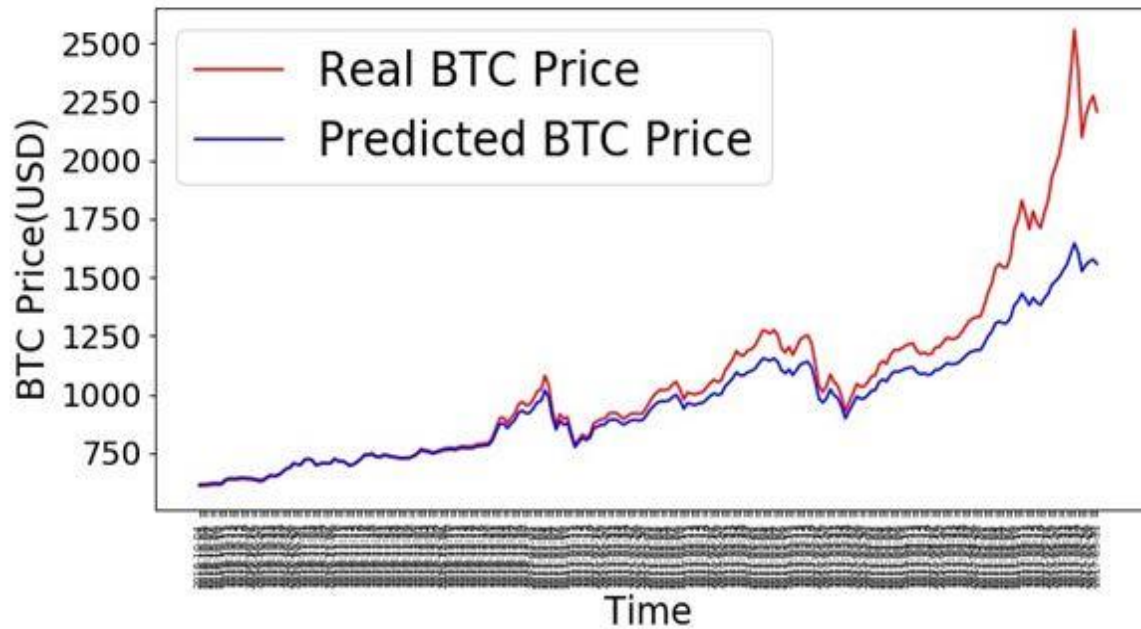


Fig. 5.5 Plot for LSTM model

BTC PRICE PREDICTION

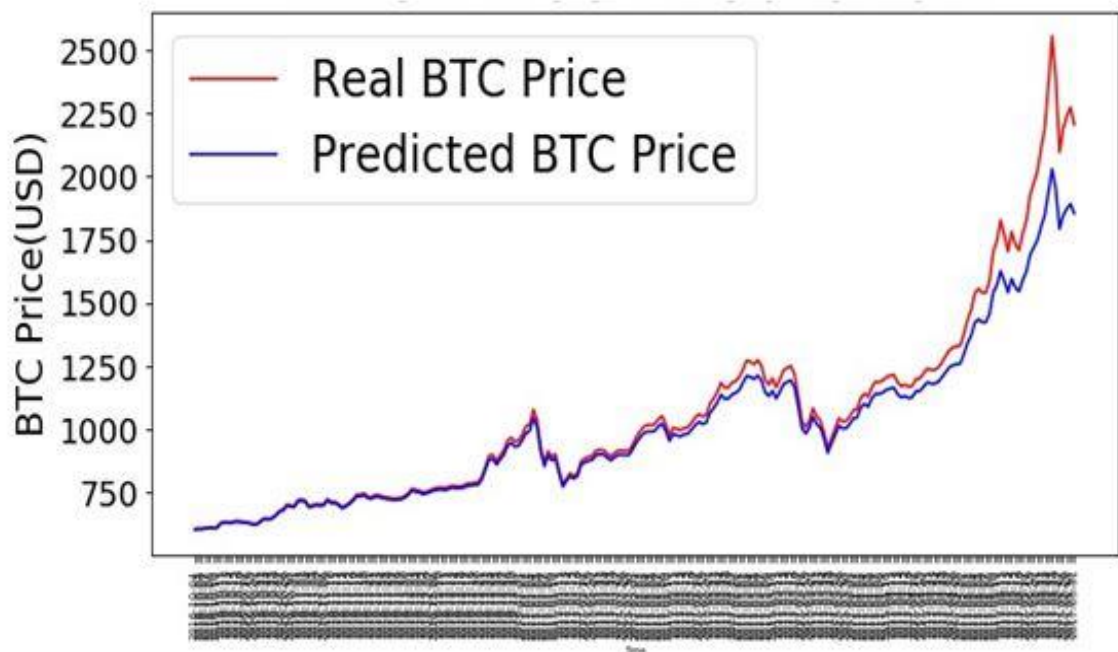


Fig. 5.6 Plot for GRU model

It can be clearly seen that in both the cases, algorithms could not perform well for prediction for the higher BTC price, though GRU performed better than the LSTM model. Because of time series data and its non-linear nature prediction is difficult. GRU prediction is able to pick up the shape of observed data, but not quite there yet. Next deep learning algorithm used was CNN. It had only 1 convolutional layer whereas many hidden layers.

Its mean square error value came out to be:

```
In [102]: from sklearn.metrics import mean_squared_error
          mean_squared_error(validation_original_outputs[:, :, 0].reshape(-1), predicted_inverted)

Out[102]: 66989.53361326191
```

Fig.5.7 Mean square error

Below shown is the CNN layer matrix values of epoch and loss with few cells.

```
1 epoch, loss, val_loss
2 0, 0.3149545303097478, 0.3596475507531847
3 1, 0.19547608605137579, 0.1518878319433757
4 2, 0.10558914144833882, 0.08568656125238963
5 3, 0.0720601506807186, 0.05330683557050569
6 4, 0.06250524355305566, 0.03439063525625637
7 5, 0.06036739520452641, 0.02471945248544216
8 6, 0.05474598319442184, 0.020889981516769955
9 7, 0.04944624724211516, 0.020312636026314328
10 8, 0.04696778690925351, 0.020759809762239456
11 9, 0.04357719600752548, 0.02191086140062128
12 10, 0.04759169480314961, 0.025078512728214264
13 11, 0.04005510815315776, 0.030859517199652537
14 12, 0.042680877847252066, 0.03383993171155453
15 13, 0.037286861489216484, 0.03420305039201464
16 14, 0.035893064406183034, 0.030580527282186916
17 15, 0.0362026643835836, 0.03027414876435484
18 16, 0.03780028065321622, 0.025199461994426593
19 17, 0.034006778357757464, 0.019487386835472926
20 18, 0.03146845467940525, 0.016305522061884403
21 19, 0.028999546887697996, 0.017194349451788833
22 20, 0.034550449638455, 0.019302504137158394
23 21, 0.03326969786926552, 0.019992828901324953
24 22, 0.033752102266859124, 0.017448322847485542
25 23, 0.031562769302615416, 0.01694029356752123
26 24, 0.028680027979943488, 0.018771831478391374
27 25, 0.027746249266244746, 0.019177813082933426
28 26, 0.028208468609523994, 0.01974669231900147
29 27, 0.028035704149968095, 0.01994865414287363
30 28, 0.026427986169302906, 0.017338651631559645
31 29, 0.027943178873371194, 0.015902199250246798
```

Fig. 5.8 CNN Matrix

And the subsequent plot comparing actual and predicted values is as follows:

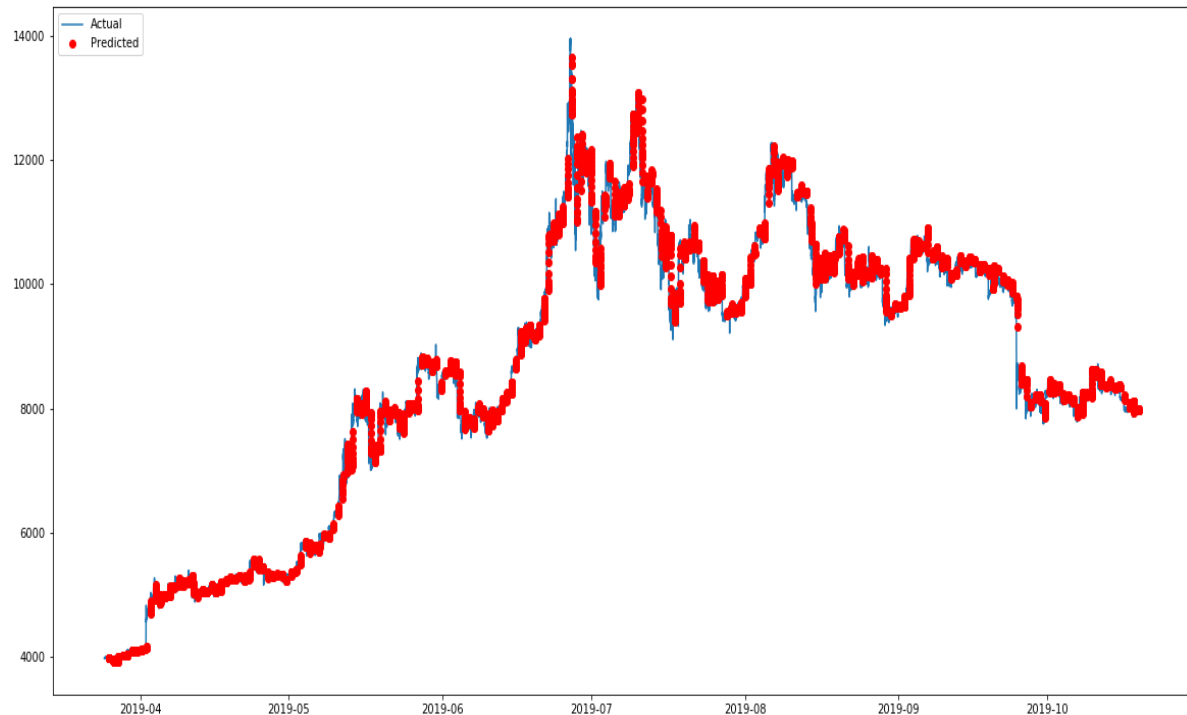


Fig. 5.9 Plot comparing actual and predicted prices

We also deployed SVM, the machine learning algorithm to see how ML looks into the deep learning data set. To prove that deep learning algorithms are always better as compared to machine learning models, accuracy of SVM came out to be negative. This shows that Deep learning models are best in case of time series data.

```
In [11]: svr_rbf_confidence = svr_rbf.score(x_test, y_test)
          print("svr_rbf accuracy: ", svr_rbf_confidence)

svr_rbf accuracy:  -0.0028574795295994715
```

Fig. 5.10 Jupyter code line showing Accuracy of SVM Algorithm

Below is the predicted value matrix using SVM model:

```
[11101.    3781.13  9833.7   8693.98  6967.64  4329.91  8199.83  1333.93
 4202.34  6588.18  9971.    1033.72 16650.01  4053.87  4822.01  1917.72
17700.    6010.85  2516.98 11400.98  8504.57  8258.54  2538.41 11685.58
1269.98  7675.    4322.44  7737.26  8259.42  5741.35  1171.    1024.
1009.15  1503.25 10433.68 11417.39  3662.95  2504.    11393.97 16927.99
13394.2   8892.18  5700.    14107.87  1056.4   17477.98  5653.58  1774.52
7373.3   5967.38  8904.02  2793.34 13829.28  6745.04  4170.6   15143.67
8095.19 11479.68  1036.23  6120.    2851.01  7150.61  1515.39  8560.
9225.31  1203.98  3864.    1114.39  1051.8   7791.69  1055.75  7860.83
1287.38  2502.26 16466.98  1030.84  1146.42  1205.2   2424.96  2383.97
8195.99  5773.03  7988.96  8541.96  2899.01]
```

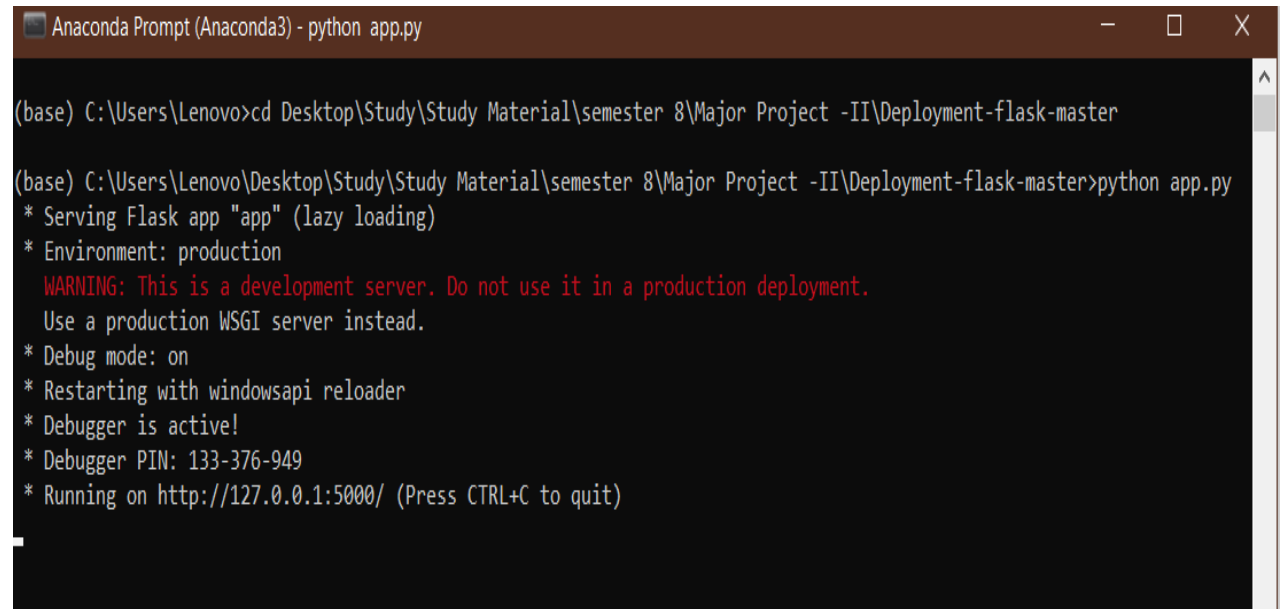
Fig 5.11 SVM Predicted matrix

The SVM model cannot be plotted because it is a continuous data.

Further the applied algorithms were deployed on Flask framework to get the real time values on web API.

Following is the output of Price Prediction on API:

Below shown is the anaconda terminal while executing flask API and displaying local URL to be used.



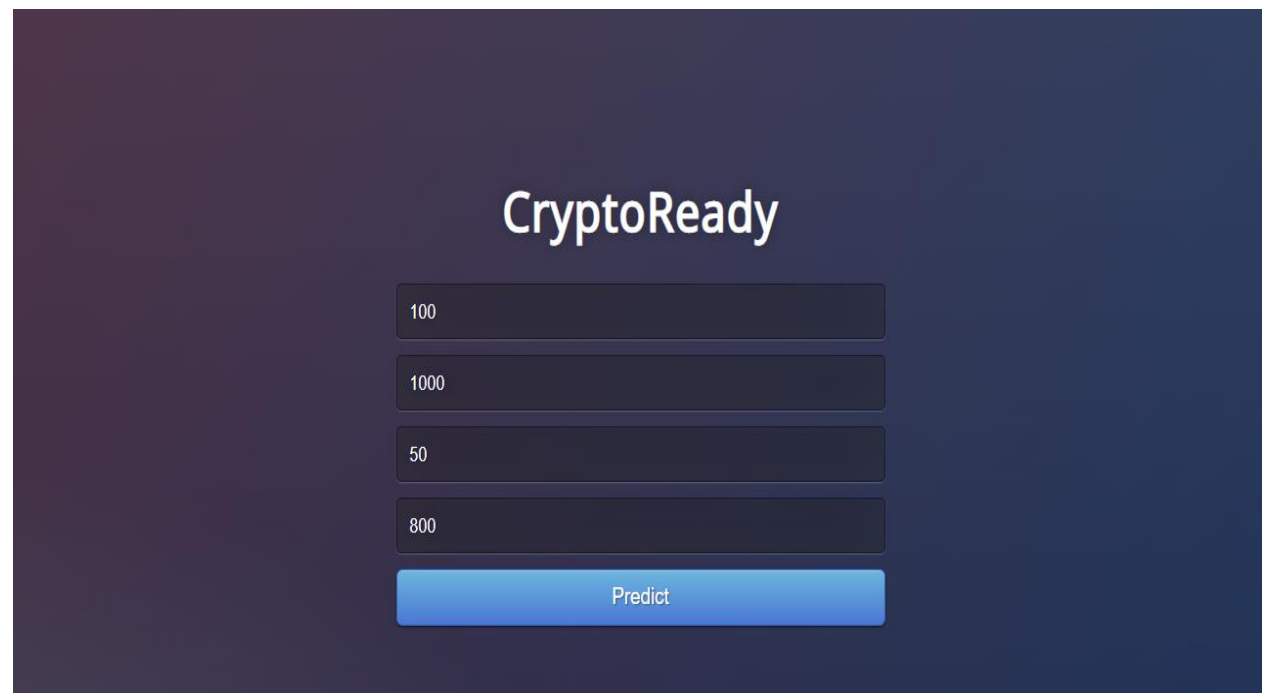
```
Anaconda Prompt (Anaconda3) - python app.py

(base) C:\Users\Lenovo>cd Desktop\Study\Study Material\semester 8\Major Project -II\Deployment-flask-master

(base) C:\Users\Lenovo\Desktop\Study\Study Material\semester 8\Major Project -II\Deployment-flask-master>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 133-376-949
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig. 5.12 Terminal execution

Giving open price, high price, low price and close price by our side as an input.



The screenshot shows a web application titled "CryptoReady" with a dark blue background. It features four input fields for numerical data, each with a light blue border and a dark blue background. The values entered in the fields are 100, 1000, 50, and 800. Below these fields is a prominent blue button with the text "Predict" in white.

Fig. 5.13 API while giving input

Bitcoin price comes as an output.

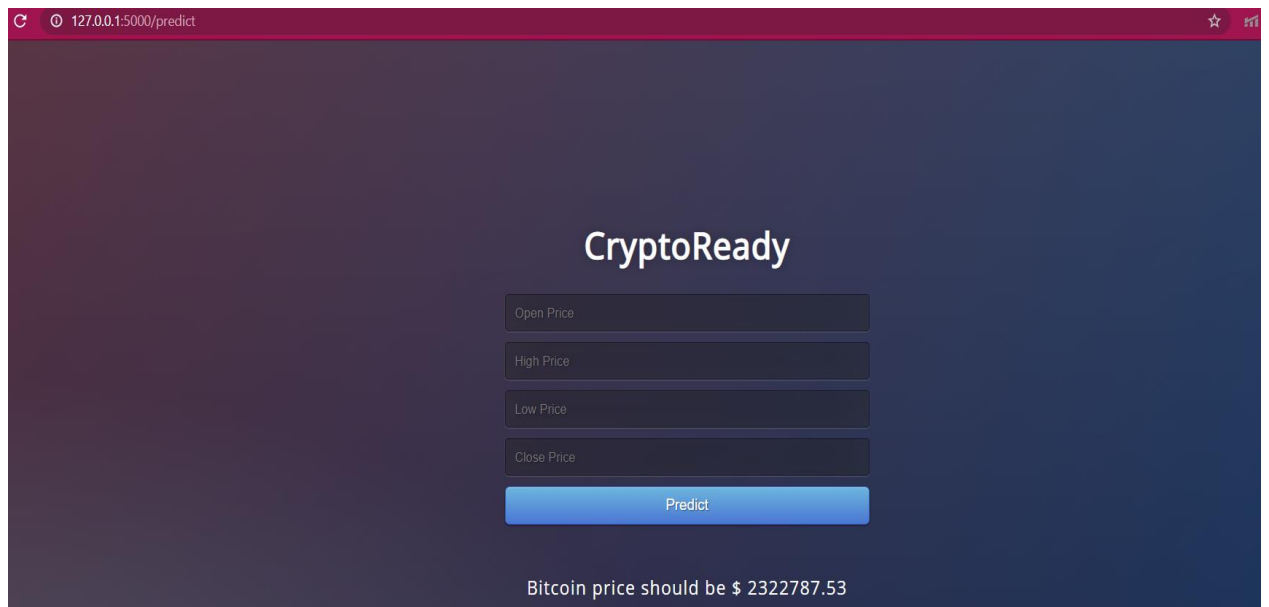


Fig. 5.14 API while giving output

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

Prediction has always been an exciting and interesting topic for people around the globe. As the technology has grown so far, still predicting things before hand is in demand.

Bitcoin has been of great importance in past few years. From blockchain to machine learning, it has occupied all markets. Predicting bitcoin prices was of great enthusiasm and knowledge.

It was found that various deep learning algorithms were successful in prediction ranging from GRU, LSTM and CNN then comparing them with machine learning algorithm SVM.

It can be inferred that deep learning models take long time to train. Prediction of bitcoin price was a complicated task as it is based on large dataset.

Machine learning model or deep learning model without real time deployment are of no use. So API was designed for this purpose. This can be seen as our work in production.

As we work, there is always a room for improvement. No. of neurons can be increased. Dataset of large columns will be of better results. Including more features or parameters is always for betterment.

In coming future as per our needs, we can also develop an android app or website for same.

REFERENCES

- [1] Y. Yao, J. Yi, S. Zhai, Y. Lin, T. K. G. Zhang, L. Yoonjae Lee-“ *Predictive Analysis of Cryptocurrency Price Using Deep Learning*” International Journal of Engineering & Technology, pp: 258-264, 2018
- [2] S. Yogeshwaran, M. J. Kaur, P. Maheshwari- “*Predicting Bitcoin Prices using Deep Learning*”, IEEE Global Engineering Education Conference (EDUCON), pp: 1449-1454, April 2019
- [3] S. Velankar, S. Valecha, S. Maji, “*Bitcoin Price Prediction using Machine Learning*”, International Conference on Advanced Communications Technology, pp:144-147, Feb 2018
- [4] V. Derbentsev, N. Datsenko, O. Stepanenko, and V. Bezkorovainyi- “*Forecasting cryptocurrency prices time series using machine learning approach*” SHS Web of Conferences, pp:1-7, 2019
- [5] T. Phaladisailoed, “*Machine Learning Models Comparison for Bitcoin Price Prediction*”, Thanisa Numnonda Faculty of Information Technology, pp:506-511, January 2018
- [6] McNally, S. Roche, J. Caton, “*Predicting the Price of Bitcoin Using Machine Learning*” 26th Euro micro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp:339–343, March 2018
- [7] F. Tschorsch, B. Scheuermann, “*Bitcoin and beyond: A technical survey on decentralized digital currencies*”, IEEE Communications Surveys & Tutorials Vol.18 No.3, pp:2084-2123, 2016.
- [8] P. Meesad, R.I Rasel, “*An efficient modelling approach for forecasting financial time series data using support vector regression*”, International Journal of Computational Intelligence Studies Vol.4 No.2, pp:134-150, January 2015.
- [9] RC Phillips, D. Gorse, “*Predicting cryptocurrency price bubbles using social media data and epidemic modeling*”, IEEE Symposium Series on Computational Intelligence (SSCI), pp: 1-7, 2017.
- [10] S. Mohapatra, N. Ahmed, P. Alencer, “*Krypto Oracle: A Real-Time Cryptocurrency Price Prediction Platform Using Twitter Sentiments*”, IEEE International Conference on Big Data, pp: 5544-5551, 2019.
- [11] F. A. Aslam, Mohammed, “*Efficient Way of Web Development Using Python and Flask*”, International Journal of Advanced Computer Research Vol.6, pp:54-57, January 2015.
- [12] <https://hackernoon.com/machine-learning-w22g322x>

- [13]<https://medium.com/swlh/how-i-used-ml-to-predict-bitcoin-prices-82af7c655092>
- [14]https://blog.quantinsti.com/rnn-lstm-gru-trading/?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com
- [15] <https://blog.floydhub.com/gru-with-pytorch/>
- [16] https://www.google.com/deeplearning_and_deeplearning/
- [17] <https://stackabuse.com/time-series-analysis-with-lstm-using-pythons-keras-library/>
- [18] <https://blog.paperspace.com/deploying-deep-learning-models-flask-web-python/>
- [19]<https://www.analyticsvidhya.com/blog/2017/09/machine-learning-models-as-apis-using-flask/>
- [20] <https://www.kaggle.com/atuljhasg/bitcoin-price-prediction-using-rnn>

APPENDICES

LSTM Code:

```
import numpy as np
import pandas as pd
import math
from math import sqrt
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
import keras
from sklearn.preprocessing import MinMaxScaler
df = pd.read_csv(r'C:\Users\user\Desktop\bitstampUSD_1-mindata_.csv')
df['date'] = pd.to_datetime(df['Timestamp'], unit='s').dt.date
group = df.groupby('date')
Real_Price = group ['Weighted Price']. mean ()
df.head()
prediction_days=240
df_train= Real Price [: len(Real_Price)-prediction_days]
df_test= Real_Price[len(Real_Price)-prediction_days:]
training_set = df_train.values
training_set= training_set.astype('float32')
training_set = np.reshape(training_set, (len(training_set), 1))
sc = MinMaxScaler(feature_range= (0,1))
training_set = sc.fit_transform(training_set)
X_train = training_set[0: len(training_set)-1]
y_train = training_set[1: len(training_set)]
X_train = np.reshape(X_train, (len(X_train), 1, 1))
regressor = keras.Sequential()
```

```

regressor.add(keras.layers.LSTM(units=75,activation='sigmoid',return_sequences=True,
input_shape = (None, 1)))
regressor.add(keras.layers.LSTM(units=30,return_sequences=True))
regressor.add(keras.layers.LSTM(units=30))
regressor.add(Dense(units = 1))
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
regressor.summary()
regressor.fit(X_train, y_train, epochs = 50, batch_size=8, verbose=2)
testpredict= regressor.predict(X_train,batch_size=5)
score= regressor.evaluate(X_train,y_train,batch_size=5,verbose=2)
MSE=score
RMSE=math.sqrt(score)
print ('Test RMSE: %.3f%RMSE)
test_set = df_test.values
inputs = np.reshape(test_set, (len(test_set), 1))
inputs = sc.transform(inputs)
inputs = np.reshape(inputs, (len(inputs), 1, 1))
predicted_BTC_price = regressor.predict(inputs)
predicted_BTC_price = sc.inverse_transform(predicted_BTC_price)
plt.figure(figsize=(10,5), dpi=80, facecolor='w', edgecolor='k')
ax = plt.gca()
plt.plot(test_set, color = 'red', label = 'Real BTC Price')
plt.plot(predicted_BTC_price, color = 'blue', label = 'Predicted BTC Price')
plt.title('BTC Price Prediction', fontsize=40)
df_test = df_test.reset_index()
x=df_test.index
labels = df_test['date']
plt.xticks(x, labels, rotation = 'vertical')
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(6)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(18)
plt.xlabel('Time', fontsize=20)
plt.ylabel('BTC Price(USD)', fontsize=20)
plt.legend(loc=2, prop={'size': 25})

```

```
plt.show()
```

GRU Code:

```
import numpy as np
import math
import pandas as pd
from math import sqrt
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import GRU
import keras
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv(r'C:\Users\user\Desktop\bitstampUSD_1-mindata_.csv')
df['date'] = pd.to_datetime(df['Timestamp'],unit='s').dt.date
group = df.groupby('date')
Real_Price = group['Weighted Price'].mean()
df.head()

prediction_days=240
df_train= Real_Price[:len(Real_Price)-prediction_days]
df_test= Real_Price[len(Real_Price)-prediction_days:]
training_set = df_train.values
training_set = np.reshape(training_set, (len(training_set), 1))
sc = MinMaxScaler(feature_range=(0,1))
training_set = sc.fit_transform(training_set)
X_train = training_set[0:len(training_set)-1]
y_train = training_set[1:len(training_set)]
X_train = np.reshape(X_train, (len(X_train), 1, 1))
regressor = Sequential()
regressor.add(keras.layers.GRU(units=75,
activation='relu',return_sequences=True,input_shape = (None, 1)))
regressor.add(keras.layers.GRU(units=30,return_sequences=True))
regressor.add(keras.layers.GRU(units=30))
regressor.add(Dense(units = 1))
```



```

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
regressor.fit(X_train, y_train, epochs =50, batch_size=8, verbose=2)
regressor.summary()
testpredict= regressor.predict(X_train,batch_size=8)
score= regressor.evaluate(X_train,y_train,batch_size=8,verbose=2)
MSE=score
RMSE=math.sqrt(score)
print("Test RMSE:%.3f%RMSE)
test_set = df_test.values
inputs = np.reshape(test_set, (len(test_set), 1))
inputs = sc.transform(inputs)
inputs = np.reshape(inputs, (len(inputs), 1, 1))
predicted_BTC_price = regressor.predict(inputs)
predicted_BTC_price = sc.inverse_transform(predicted_BTC_price)
plt.figure(figsize=(10,5), dpi=80, facecolor='w', edgecolor='k')
ax = plt.gca()
plt.plot(test_set, color = 'red', label = 'Real BTC Price')
plt.plot(predicted_BTC_price, color = 'blue', label = 'Predicted BTC Price')
plt.title('BTC Price Prediction', fontsize=40)
df_test = df_test.reset_index()
x=df_test.index
labels = df_test['date']
plt.xticks(x, labels, rotation = 'vertical')
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(6)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(18)
plt.xlabel('Time', fontsize=5)
plt.ylabel('BTC Price(USD)', fontsize=20)
plt.legend(loc=2, prop={'size': 25})
plt.show()

```

CNN Code:

```

from keras import applications
from keras.models import Sequential

```

```
from keras.models import Model
from keras.layers import Dropout, Flatten, Dense, Activation
from keras.callbacks import CSVLogger
import tensorflow as tf
#from scipy.ndimage import imread
import numpy as np
import random
from keras.layers import LSTM
from keras.layers import Conv1D, MaxPooling1D, LeakyReLU
from keras import backend as K
import keras
from keras.callbacks import CSVLogger, ModelCheckpoint
from keras.backend.tensorflow_backend import set_session
from keras import optimizers
import h5py
from sklearn.preprocessing import MinMaxScaler
import os
import pandas as pd
import matplotlib.pyplot as plt
os.environ['CUDA_DEVICE_ORDER'] = 'PCI_BUS_ID'
os.environ['CUDA_VISIBLE_DEVICES'] = '0'
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True
#set_session(tf.Session(config=config))
with h5py.File(".join(['bitcoin2016-12to2019-10_close.h5']), 'r') as hf:
    datas = hf['inputs'].value
    labels = hf['outputs'].value
    input_times = hf['input_times'].value
    output_times = hf['output_times'].value
    original_inputs = hf['original_inputs'].value
    original_outputs = hf['original_outputs'].value
    original_datas = hf['original_datas'].value
    scaler=MinMaxScaler()
    training_size = int(0.8* datas.shape[0])
```

```

training_datas = datas[:training_size,:,:]
training_labels = labels[:training_size,:,:]
validation_datas = datas[training_size:,:,:]
validation_labels = labels[training_size:,:,:]
validation_original_outputs = original_outputs[training_size:,:,:]
validation_original_inputs = original_inputs[training_size:,:,:]
validation_input_times = input_times[training_size:,:,:]
validation_output_times = output_times[training_size:,:,:]
ground_true = np.append(validation_original_inputs,validation_original_outputs, axis=1)
ground_true.shape
ground_true_times = np.append(validation_input_times,validation_output_times, axis=1)
ground_true_times.shape
step_size = datas.shape[1]
batch_size= 8
nb_features = datas.shape[2]
epochs = 1
model = Sequential()
model.add(Conv1D(activation='relu',
input_shape=(step_size, nb_features), strides=3,
filters=8, kernel_size=8))
model.add(LeakyReLU())
model.add(Dropout(0.5))
model.add(Conv1D(activation='relu', strides=2,
filters=8, kernel_size=8))
model.add(LeakyReLU())
model.add(Dropout(0.5))
model.add(Conv1D(strides=2, filters=nb_features,
kernel_size=8))
model.load_weights('NewWeights/bitcoin2016-10to2019-10_close_CNN_2_relu-96-
0.00018.hdf5')
model.compile(loss='mse', optimizer='adam')
predicted = model.predict(validation_datas)
predicted_inverted = []
for i in range(original_datas.shape[1]):
scaler.fit(original_datas[:,i].reshape(-1,1))

```

```

predicted_inverted.append(scaler.inverse_transform(predicted[:,i]))
print(np.array(predicted_inverted).shape)
ground_true = ground_true[:,0].reshape(-1)
ground_true_times = ground_true_times.reshape(-1)
ground_true_times = pd.to_datetime(ground_true_times, unit='s')
# since we are appending in the first dimension
predicted_inverted = np.array(predicted_inverted)[0,:].reshape(-1)
print(np.array(predicted_inverted).shape)
validation_output_times = pd.to_datetime(validation_output_times.reshape(-1), unit='s')
ground_true_df = pd.DataFrame()
ground_true_df['times'] = ground_true_times
ground_true_df['value'] = ground_true
ground_true_df.head()
ground_true_df = pd.DataFrame()
ground_true_df['times'] = ground_true_times
ground_true_df['value'] = ground_true
ground_true_df.set_index('times').reset_index()
ground_true_df.shape
prediction_df = pd.DataFrame()
prediction_df['times'] = validation_output_times
prediction_df['value'] = predicted_inverted
prediction_df.shape
prediction_df.head()
prediction_df=prediction_df.loc[(prediction_df["times"].dt.year==2019
)&(prediction_df["times"].dt.month > 1 ),:]
ground_true_df=ground_true_df.loc[(ground_true_df["times"].dt.year==2019
)&(ground_true_df["times"].dt.month > 1 ),:]
plt.figure(figsize=(20,10))
plt.plot(ground_true_df.times,ground_true_df.value, label = 'Actual')
plt.plot(prediction_df.times,prediction_df.value,'ro', label='Predicted')
plt.legend(loc='upper left')
plt.savefig('result_paper/bitcoin2015to2019_close_CNN_2_relu_result.png')
plt.show()
from sklearn.metrics import mean_squared_error
mean_squared_error(validation_original_outputs[:,0].reshape(-1),predicted_inverted)

```

```
correct_prediction=tf.equal(tf.argmax(pred,1),tf.argmax(y,1))
acc=tf.reduce_mean(tf.cast(correct_prediction,tf.float32))
```

SVM Code:

```
import numpy as np
import pandas as pd
df = pd.read_csv(r'C:\Users\Lenovo\Desktop\Study\Study Material\semester 8\Major
Project -II\bitcoin_data.csv')
df.head(7)
df.drop(['Timestamp'],1, inplace=True)
prediction_days = 30
df['Prediction'] = df[['Open']].shift(-prediction_days)
X = np.array(df.drop(['Prediction'],1))
X= X[:len(df)-prediction_days]
print(X)
y = np.array(df['Prediction'])
y = y[:-prediction_days]
print(y)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
prediction_days_array = np.array(df.drop(['Prediction'],1))[:-prediction_days:]
from sklearn.svm import SVR
svr_rbf = SVR(kernel='rbf', C=1e5, gamma=0.00005)
#Create the model
svr_rbf.fit(x_train, y_train)
svr_rbf_confidence = svr_rbf.score(x_test, y_test)
print("svr_rbf accuracy: ", svr_rbf_confidence)
svm_prediction = svr_rbf.predict(x_test)
print(svm_prediction)
print()
print(y_test)
svm_prediction = svr_rbf.predict(prediction_days_array)
print(svm_prediction)
```

API Code:

Model .py

```
import math
import pandas as pd
import numpy as np
from math import sqrt
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import GRU
import keras
import pickle
from sklearn.preprocessing import MinMaxScaler

df = pd.read_csv(r'C:\Users\Lenovo\Desktop\Study\Study Material\semester 8\Major
Project -II\CryptoReady\bitstampUSD_1-min_data_2012-01-01_to_2018-11-11.csv')
df['date'] = pd.to_datetime(df['Timestamp'],unit='s').dt.date
group = df.groupby('date')
Real_Price = group['Weighted_Price'].mean()
df.head()
prediction_days=240
df_train= Real_Price[:len(Real_Price)-prediction_days]
df_test= Real_Price[len(Real_Price)-prediction_days:]
training_set = df_train.values
training_set = np.reshape(training_set, (len(training_set), 1))
sc = MinMaxScaler(feature_range=(0,1))
training_set = sc.fit_transform(training_set)
X_train = training_set[0:len(training_set)-1]
y_train = training_set[1:len(training_set)]
X_train = np.reshape(X_train, (len(X_train), 1, 1))
regressor = Sequential()
regressor.add(keras.layers.GRU(units=75,activation='relu',return_sequences=True,input_
shape = (None, 1)))
regressor.add(keras.layers.GRU(units=30,return_sequences=True))
regressor.add(keras.layers.GRU(units=30))
```

```

regressor.add(Dense(units = 1))
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
regressor.fit(X_train, y_train, epochs =50, batch_size=8, verbose=2)
regressor.summary()
testpredict= regressor.predict(X_train,batch_size=8)
score= regressor.evaluate(X_train,y_train,batch_size=8,verbose=2)
MSE=score
RMSE=math.sqrt(score)
print("Test RMSE:%.3f%RMSE)
test_set = df_test.values
inputs = np.reshape(test_set, (len(test_set), 1))
inputs = sc.transform(inputs)
inputs = np.reshape(inputs, (len(inputs), 1, 1))
predicted_BTC_price = regressor.predict(inputs)
predicted_BTC_price = sc.inverse_transform(predicted_BTC_price)

# Saving model to disk
pickle.dump(predicted_BTC_price, open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))
print(model.predict([[2, 9, 6, 8]]))

```

app.py

```

import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict',methods=['POST'])
def predict():
    int_features = [int(x) for x in request.form.values()]

```

```

    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)
    output = round(prediction[0], 2)
return render_template('index.html', prediction_text='Bitcoin price should be $
{}'.format(output))
@app.route('/predict_api',methods=['POST'])
def predict_api():
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])
    output = prediction[0]
    return jsonify(output)
if __name__ == "__main__":
    app.run(debug=True)

```

Request.py

```

import requests
url = 'http://localhost:5000/predict_api'
r = requests.post(url)
print(r.json())

```

Index.html

```

<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
    <meta charset="UTF-8">
    <title>Bitcoin Price Prediction API</title>
    <linkhref='https://fonts.googleapis.com/css?family=Pacifico'rel='stylesheet'
type='text/css'>
    <linkhref='https://fonts.googleapis.com/css?family=Arimo'rel='stylesheet' type='text/css'>
    <linkhref='https://fonts.googleapis.com/css?family=Hind:300'rel='stylesheet'
type='text/css'>
    <linkhref='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" href="{ { url_for('static', filename='css/style.css') } }">

```



```
</head>
<body>
<div class="login">
    <h1>CryptoReady</h1>
    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}" method="post">
        <input type="text" name="Open Price" placeholder="Open Price"
required="required" />
        <input type="text" name="High Price" placeholder="High Price"
required="required" />
        <input type="text" name="Low Price" placeholder="Low Price" required="required" />
        <input type="text" name="Close Price" placeholder="Close Price"
required="required" />
        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>
    <br>
    <br>
    {{ prediction_text }}
</div>
</body>
</html>
```

9916102024_PREDICTING BITCOIN PRICES USING DEEP LEARNING

ORIGINALITY REPORT

17%

SIMILARITY INDEX

6%

INTERNET SOURCES

7%

PUBLICATIONS

13%

STUDENT PAPERS

PRIMARY SOURCES

- | | | |
|---|--|----|
| 1 | www.coursehero.com
Internet Source | 3% |
| 2 | Submitted to Victorian Institute of Technology
Student Paper | 2% |
| 3 | Submitted to Engineers Australia
Student Paper | 2% |
| 4 | Yecheng Yao, Jungho Yi, Shengjun Zhai, Yuwen Lin, Taekseung Kim, Guihongxuan Zhang, Leonard Yoonjae Lee. "Predictive Analysis of Cryptocurrency Price Using Deep Learning", International Journal of Engineering & Technology, 2018
Publication | 2% |
| 5 | www.shs-conferences.org
Internet Source | 1% |
| 6 | S. Yogeshwaran, Maninder Jeet Kaur, Piyush Maheshwari. "Project Based Learning: Predicting Bitcoin Prices using Deep Learning", | 1% |