MACHINE LEARNING

(INDIAN FOREIGN EXCHANGE RATES PREDICTION)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of **Bachelor of Technology**

In

Computer Science Engineering

By

PRANAV SUNDARESAN BABU

221710309048

Under the Guidance of

GITAM

(DEEMED TO BE UNIVERSITY)

(Estd. u/s 3 of the UGC Act, 1956)

VISAKHAPATNAM • HYDERABAD • BENGALURU

Accredited by NAAC with A+ Grade

Assistant Professor

Department Of Computer Science Engineering GITAM School of Technology

GITAM (Deemed to be University) Hyderabad-502329

June 2020

DECLARATION

I submit this industrial training work entitled "INDIAN FOREIGN EXCHANGE RATES PREDICTION" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "Bachelor of Technology" in "Computer Science Engineering". I declare that it was carried out independently by me under the guidance of , GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD **Name: PRANAV SUNDARESAN BABU**

Date: JUNE 2020

221710309048

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India Dated:

CERTIFICATE

This is to certify that the Industrial Training Report entitled“INDIAN FOREIGN EXCHANGE RATES” is being submitted by PRANAV SUNDARESAN BABU (221710309048) in partial fulfillment of the requirement for the award ofBachelor of Technology in Computer Science Engineering at GITAM (Deemed To Be University), Hyderabad during the academic year 2020-21

It is faithful record work carried out by her at the Computer Science Engineering Department, GITAM University Hyderabad Campus under my guidance and supervision.

Assistant Professor

Department of CSE

Professor and HOD

Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected Dr. N. Siva Prasad, Pro Vice Chancellor, GITAM Hyderabad and Dr. CH. Sanjay, Principal, GITAM Hyderabad

I would like to thank respected Prof.S Phani Kumar, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties Mr. who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

PRANAV SUNDARESAN BABU

221710309048

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms .The primary task is to build models of higher accuracy .My perception of understanding the given data set has been in the view of undertaking the reviews of various viewers who watch a movie and give their reviews. Based on sentiment analysis under Machine learning, we try to focus our task of sentiment analysis on IMDB movie review databases. We examine the sentiment expression to classify the polarity of the movie review as either positive or negative.A comparative study on different

classification algorithms has been performed to determine the most suitable classifier to suit our problem.. We conclude that our proposed approach to sentiment classification adds a new feature to the existing features and brings in a progress to future research in this domain. Our approach using classification techniques has the best accuracy of 100% with the logistic regression algorithm".

TABLE OF CONTENTS:

CHAPTER 1:MACHINE LEARNING.....13

1.1INTRODUCTION.....13

1.2IMPORTANCE OF MACHINE LEARNING……13

1.3 USES OF MACHINE LEARNING.....14

1.4 TYPES OF LEARNING ALGORITHMS.....15

1.4.1 Supervised Learning.....15

1.4.2 Unsupervised
Learning.....16

1.4.3 Semi Supervised
Learning.....16

CHAPTER 2:DEEP LEARNING
.....17

2.1
INTRODUCTION…….....17

2.1.1 Importance Of Deep
Learning...…….....18

2.1.2 Uses Of Deep Learning
…....….....19

2.1.3 Relationship Between DataMining,Machine Learning,Deep
Learning…….....1
9

CHAPTER 3: PYTHON
…….20

3.1 INTRODUCTION TO
PYTHON……...……..20

3.2 HISTORY OF PYTHON……..
21

3.3 FEATURES OF
PYTHON……...…….21

3.4 INSTALLATION OF
PYTHON……...…….21

3.4.1 Installation(using python IDLE).....22

3.4.2 Installation(using Anaconda).....22

3.5 PYTHON VARIABLE
TYPES.....23

3.5.1 Python Numbers.....24

3.5.2 Python
Strings.....24

3.5.3 Python
Lists.....25

3.5.4 Python
Tuples.....25

3.5.5 Python
Dictionary.....26

3.6 PYTHON
FUNCTION.....26

3.6.1 Defining a
Function.....26

3.6.2 Calling a
Function.....27

3.7 PYTHON USING OOPs
CONCEPTS.....27

3.7.1Class.....27

3.7.2 __init__method in
class.....28

CHAPTER 4:INDIAN FOREIGN EXCHANGE
RATES.....28

4.1 PROJECT
REQUIREMENTS……..……..28

4.1.2 Versions Of
Packages……..29

4.1.3 Algorithms Used
…….30

4.2 PROBLEM
STATEMENT…….....31

4.3 DATA SET
DESCRIPTION.....31

4.3.1 Reading The Data
……...31

4.3.2 Data Set
Shape…….31

4.3.3 Data Set
Head……..32

4.4 OBJECTIVE OF THE CASE
STUDY.....32

CHAPTER 5: DATA PROCESSING/FEATURE ENGINEERING AND EDA……..33

5.1 STATISTICAL ANALYSIS…….....33

5.2 GENERATING PLOTS……..34

5.3 DATA TYPE CONVERSIONS……34

5.4 DETECTION OF OUTLIERS……...35

5.5 HANDLING MISSING VALUES……..35

5.6 ENCODING CATEGORICAL DATA……...36

CHAPTER 6: FEATURE SELECTIONS……36

6.1 SELECT RELEVANT FEATURES OF ANALYSIS…….37

6.2 DROP IRRELEVANT FEATURES……...37

6.3 TRAIN-TEST-SPLIT……..38

CHAPTER 7: MODEL BUILDING AND EVALUATION…….40

7.1 BRIEF OF THE ALGORITHMS USED……...40

7.2 TRAIN THE MODELS……..41

7.3 VALIDATING THE DATA……....42

7.4 MAKE PREDICTIONS……..43

7.5 PREDICTIONS FROM RAW DATA…….44

CONCLUSION.....46

LIST OF FIGURES:

Fig 1 : The Process Flow.....14

Fig 2 : Unsupervised Learning.....16

Fig 3 : Semi Supervised Learning.....17

Fig 4: Anaconda download.....23

Fig 5 : Jupyter notebook.....23

Fig 6 : Defining a Class.....28

Fig 4.1.1 : Importing Libraries.....29

Fig 4.1.2(1): Panda
Version……..29

Fig 4.1.2(2): Numpy
Version……....30

Fig 4.1.2(3): Tensorflow
Version……..30

Fig 4.1.2(4) Keras
Version……....30

Fig4.3.1: Reading the
Dataset.....31

Fig4.3.2 Dataset
shape.....31

Fig 4.3.3 Dataset
head……..32

Fig 5.1 Preprocessing the
data……..33

Fig 5.2
Plotting…….33

Fig 5.5 Handling missing datas……....34

Fig 6.1: DataFrame……...36

Fig 6.3(a): Training model ……...37

Fig 6.3(b): DataFrame……..38

Fig 6.3(c): Testing model …….39

Fig 7.1: LSTM model ……..39

Fig 7.2(a):Processing train LSTM model ……....41

Fig 7.2(b): Training ……..41

Fig 7.3: Compiling……42

Fig 7.4(a):
Prediction……42

Fig 7.4(b): Processing test
shape……..43

Fig 7.5(a): Plotting for actual and predicted
rates……....44

Fig 7.5(b): Mean squared
error……45

Fig 7.5(c): Predicted and actual rates
……..45

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are

all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

Fig 1 : The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a

known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

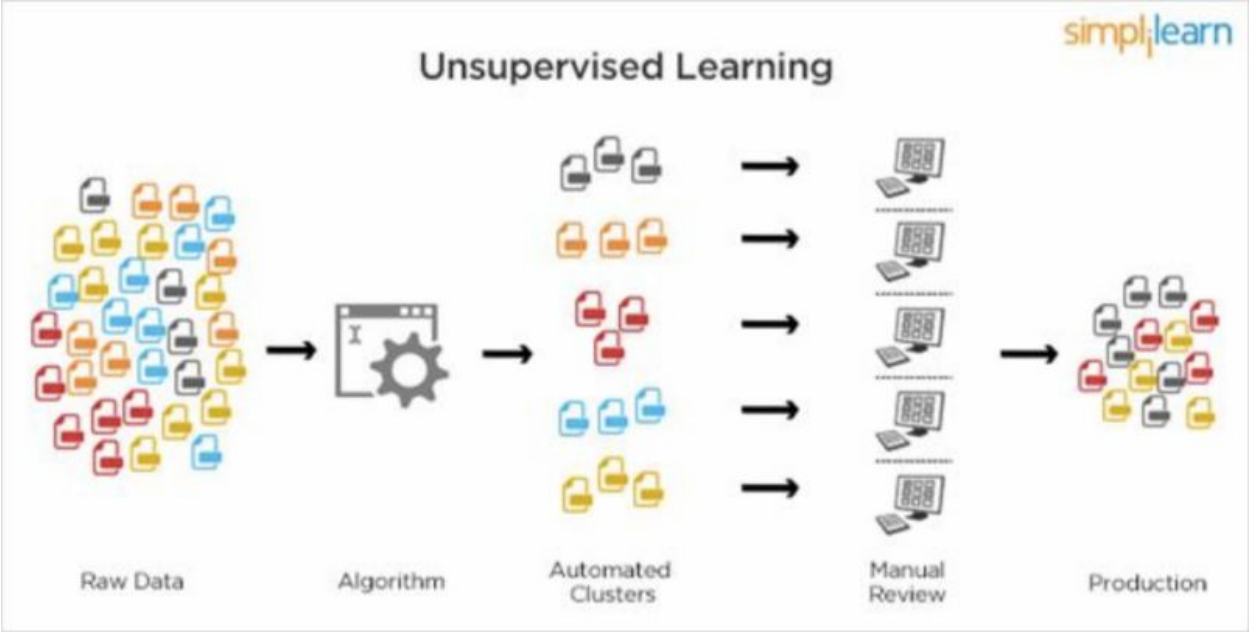

Fig 2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

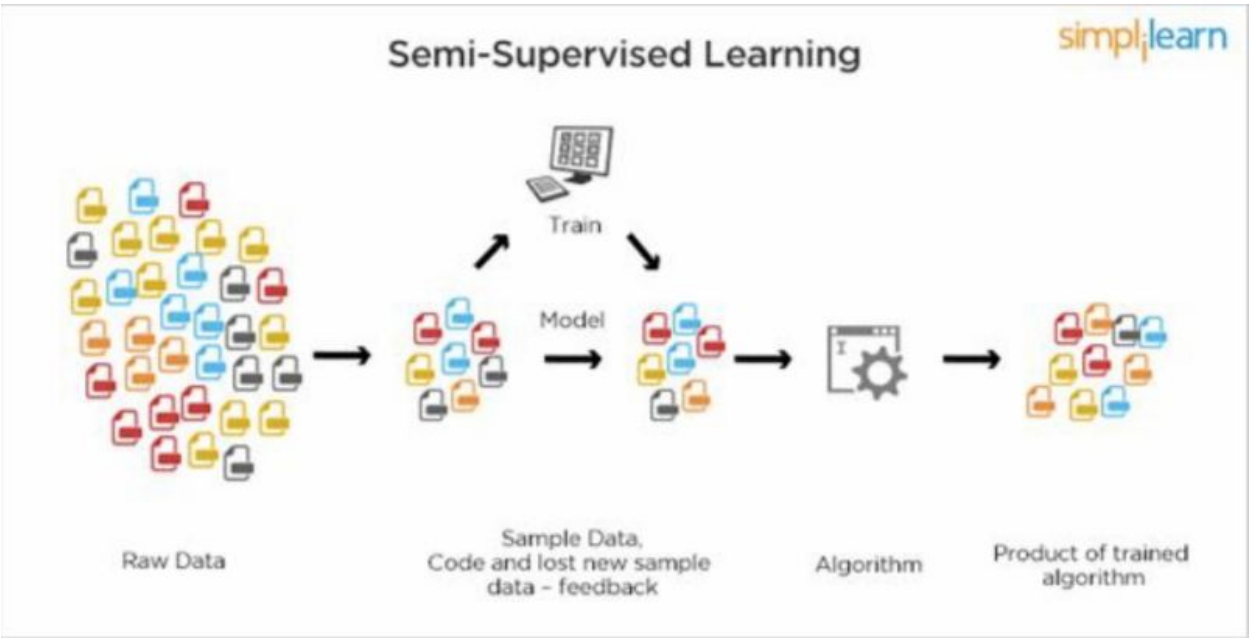

Fig 3: Semi Supervised learning

CHAPTER 2

DEEP LEARNING

2. INTRODUCTION:

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as the neural network is going to mimic the human brain so deep learning is also a kind of mimic of the human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came into the picture.

2.1.1 IMPORTANCE OF DEEP LEARNING:

Artificial Intelligence as the name suggests is to make a machine artificially intelligent i.e making the machine think or act like humans. It's not that the concept of AI is totally new instead it has been explored many decades ago but in recent years it has gained popularity and has led to various miraculous discoveries because of 2 major factors. The 2 factors which have made all the world invest in this field are increase in computational speed and the amount of useful data available. In fact the 90% of the total data available as of now has been gathered in the previous 3 to 4 years itself .If a robot is hard coded i.e. all the logic has been coded manually into the system then it's not AI so it can't be said that simply robots means AI.

To understand AI more clearly let's dive into its most popular and useful subset, Machine learning. Machine learning simply means making a machine learn from its experience and improving its performance with time just like the case of a human baby. Concept of machine learning became feasible only when sufficient amounts of data were made available for training machines. Machine learning helps in dealing with complex and robust systems. But most of the miraculous discoveries which we have come across in recent years have been made possible out of Deep Learning.

Now the question arises what is it in deep learning which has brought such a revolution in our lives . Basically deep learning is itself a subset of machine learning but in this case the machine learns in a way in which humans are supposed to learn. The structure of deep learning models is highly similar to a human brain with a large number of neurons and nodes like neurons in the human brain thus resulting in artificial neural networks. In applying traditional machine learning algorithms we have to manually select input features from complex data set and then train them which becomes a very tedious job for ML scientist but in neural networks we don't have to manually select useful input features, there are various layers of neural networks for handling complexity of the data set and algorithm as well. In my recent project on human activity recognition , when we applied traditional machine learning algorithm like K-NN then we have to separately detect human and its activity also had to select impactful input parameters manually which became a very tedious task as data set was way too complex but the complexity

dramatically reduced on applying artificial neural network, such is the power of deep learning. Yes it's correct that deep learning algorithms take lots of time for training sometimes even weeks as well but its execution on new data is so fast that it's not even comparable with traditional ML algorithms. Deep learning has enabled Industrial Experts to overcome challenges which were impossible, a decade ago like Speech and Image recognition and Natural Language Processing. Majority of the Industries are currently depending on it , be it Journalism, Entertainment, Online Retail Store, Automobile, Banking and Finance, Healthcare, Manufacturing or even Digital Sector. Video recommendations, Mail Services, Self Driving cars, Intelligent Chat bots, Voice Assistants are just trending achievements of Deep Learning.

2.1.2 Uses of Deep Learning:

1. Self Driving Cars
2. News Aggregation and Fraud News Detection
3. Natural Language Processing
4. Virtual Assistants
5. Entertainment
6. Visual Recognition
7. Fraud Detection
8. Healthcare
9. Personalisation,etc.

2.1.3 Relation between Data Mining, Machine Learning and Deep Learning:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovered previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 3

PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 INTRODUCTION TO PYTHON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

3.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990’s
- Its latest version is 3.7 , it is generally called as python3

3.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax,This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

3.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

3.4.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

Figure 4 : Python download

3.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager that quickly installs and manages packages.

- In WINDOWS:

- In windows
    - Step 1: Open Anaconda.com/downloads in a web browser.
    - Step 2: Download python 3.4 version for (32-bits graphic installer/64 -bit graphic installer)
    - Step 3: select installation type(all users)
    - Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
    - Step 5: Open jupyter notebook (it opens in default browser)

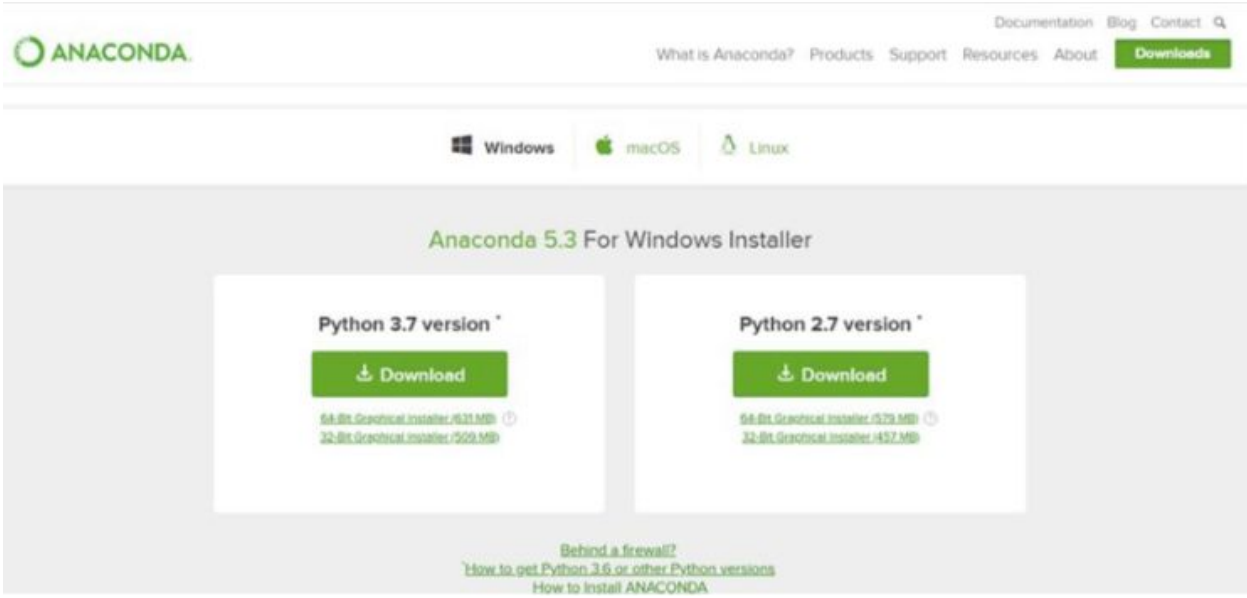

Fig 4 : Anaconda download

Fig 5: Jupyter notebook

3.5 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what

can be stored in the reserved memory.

- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –

o Numbers

o Strings

o Lists

o Tuples

o Dictionary

3.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

3.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

3.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets11([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

3.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple.

Tuples have no remove or pop method.

3.5.5 PythonDictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays12 or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

● Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.6 PYTHON FUNCTION:

3.6.1 DEFINING A FUNCTION:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.7 PYTHON USING OOPs CONCEPTS:

3.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.
- Defining a Class:

o We define a class in a very similar way how we define a function.

o Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is

indented like a functions body is.

```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Fig 6 : Defining a Class

3.7.2 __init__ method in Class:

● The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

● The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 4

INDIAN FOREIGN EXCHANGE RATES(INFORMATION ABOUT THE PROJECT):

4.1 PROJECT REQUIREMENTS:

4.1.1 PACKAGES USED:

1. pandas
2. numpy
3. matplotlib.pyplot
4. tensorflow
5. keras
6. from keras.models import Sequential
7. from keras.layers import Dense,LSTM

IMPORTING LIBRARIES

```
[1] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline
    import tensorflow as tf
    import keras
    from keras.models import Sequential
    from keras.layers import Dense,LSTM
```

```
Using TensorFlow backend.
```

Fig:4.1.1:Packages Used

4.1.2 VERSIONS OF THE PACKAGES:

1. Version of pandas package

```
[2] print(pd.__version__)
```

```
1.0.5
```

Fig:4.1.2(1):Panda version

2. **Version of numpy package**

```
[3] print(np.__version__)
    1.18.5
```

Fig:4.1.2(2):Numpy Version

3. **Version of tensorflow package**

```
[4] print(tf.__version__)
    2.2.0
```

Fig:4.1.2(3):Tensorflow Version

4. **Version of keras package**

```
[5] print(keras.__version__)
    2.3.1
```

Fig:4.1.2(4):Keras Version

4.1.3: Algorithms Used

The algorithm which is used in this project is **LSTM (Long Short-Term Memory).** Long short-term memory is an artificial recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points, but also entire sequences of data.

4.2: PROBLEM STATEMENT:

The main aim is to compare the Indian-Indian Rupee with USA -US$. We have to predict the Indian Rupee rates over USA $ rates and see if the model predicts the rates as close as the original rates.

4.3 DATASET DESCRIPTION:

4.3.1: READING THE DATA

pd_read.csv(): This is the command which lets the user read the data or the DataFrame from the csv file.

```
READING THE DATASET

[6] data=pd.read_csv('Foreign_Exchange_Rates.csv',na_values='ND')
```

Fig:4.3.1:Reading the data

4.3.2: DATASET SHAPE

data.shape: This command is used to display the number of rows and columns in the DataFrame.

```
DATASET SHAPE

[7] data.shape

    (5217, 24)
```

Fig:4.3.2:Dataset shape

4.3.3: DATASET HEAD

data.head(): This command is used to display the top 10 rows of the entire DataFrame.

DATASET HEAD

```
[8] data.head()
```

	Unnamed: 0	Time Serie	AUSTRALIA - AUSTRALIAN DOLLAR/US$	EURO AREA - EURO/US$	NEW ZEALAND - NEW ZELAND DOLLAR/US$	UNITED KINGDOM - UNITED KINGDOM POUND/US$	BRAZIL - REAL/US$	CANADA - CANADIAN DOLLAR/US$	CHINA - YUAN/US$	HONG KONG - HONG KONG DOLLAR/US$	INDIA - INDIAN RUPEE/US$	KORE WON/
0	0	2000-01-03	1.5172	0.9847	1.9033	0.6146	1.8050	1.4465	8.2798	7.7765	43.55	112
1	1	2000-01-04	1.5239	0.9700	1.9238	0.6109	1.8405	1.4518	8.2799	7.7775	43.55	112
2	2	2000-01-05	1.5267	0.9676	1.9339	0.6092	1.8560	1.4518	8.2798	7.7780	43.55	113
3	3	2000-01-06	1.5291	0.9686	1.9436	0.6070	1.8400	1.4571	8.2797	7.7785	43.55	114
4	4	2000-01-07	1.5272	0.9714	1.9380	0.6104	1.8310	1.4505	8.2794	7.7783	43.55	113

Fig:4.3.3:Dataset head

4.4: OBJECTIVE OF THE CASE STUDY:

The objective of the case study is to see whether the LSTM (Long Short-Term Memory) model can predict the Indian Rupee rates/US $ rates as close to the original rates.

The original rates:

DATA FRAME

```
[10] df=data['INDIA - INDIAN RUPEE/US$']
     df
```

```
0       43.55
1       43.55
2       43.55
3       43.55
4       43.55
        ...
5212      NaN
5213    71.28
5214    71.45
5215    71.30
5216    71.36
Name: INDIA - INDIAN RUPEE/US$, Length: 5217, dtype: float64
```

Fig:4.4:DataFrame

CHAPTER 5

DATA PROCESSING/FEATURE ENGINEERING AND EDA

5.1: STATISTICAL ANALYSIS:

Here we preprocess the data by forming arrays

PREPROCESSING DATASET

```
[13] df=np.array(df).reshape(-1,1)
     from sklearn.preprocessing import MinMaxScaler
     scaler=MinMaxScaler()
     df=scaler.fit_transform(df)
     print (df)
```

```
[[0.14142259]
 [0.14142259]
 [0.14142259]
 ...
 [0.91966527]
 [0.91548117]
 [0.91715481]]
```

Fig:5.1:PreProcessing the data

5.2:GENERATING PLOTS:

Here we plot the graph of the required data for the model. In this case it is INDIAN-INDIAN RUPEE/USA $.

plt.plot(): This command is used to plot the graph for the required data.

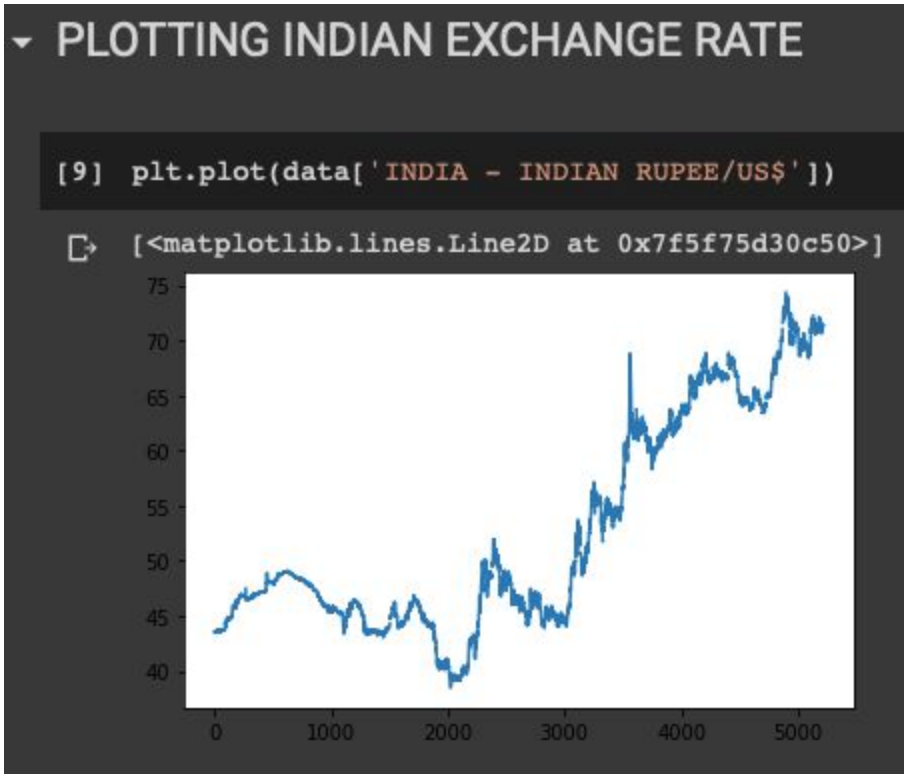

Fig:5.2:Plotting

5.3 DATA TYPE CONVERSIONS:

Type Conversion is the conversion of an object from one data type to another data type. Implicit Type Conversion is automatically performed by the Python interpreter. Python avoids the loss of data in Implicit Type Conversion.

Here, in this LSTM model we do not use any Data Type Conversions as the data we require is numerical data, and the data is already in numerical format.

5.4 DETECTION OF OUTLIERS:

In most of the cases a threshold of 3 or -3 is used i.e if the Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers. We will use the Z-score function defined in scipy library to detect the outliers.

Here, in this LSTM model there are no outliers because all the values present in the data set are close to each other and this information is required in predicting the rates.

5.5: HANDLING MISSING VALUES:

Remove Rows With Missing Values. The simplest strategy for handling missing data is to remove records that contain a missing value. The simplest approach for dealing with missing values is to remove the entire predictors and/or samples that contain missing values.

In the DataFrame as you can see from the figure above(Fig:4.4) there are few missing values in the DataFrame, and we can drop/remove the rows containing NaN values.

df.dropna(): This command is used to drop all the rows in the data frame which contains Nan values in the data frame.

```
[11] df.dropna(inplace=True)

[12] df

0        43.55
1        43.55
2        43.55
3        43.55
4        43.55
         ...
5211     71.23
5213     71.28
5214     71.45
5215     71.30
5216     71.36
Name: INDIA - INDIAN RUPEE/US$, Length: 5018, dtype: float64
```

Fig:5.5:Handling missing values

5.6: ENCODING CATEGORICAL DATA:

This means that categorical data must be encoded to numbers before we can use it to fit and evaluate a model. There are many ways to encode categorical variables for modeling, although the three most common are as follows: Integer Encoding: Where each unique label is mapped to an integer.

As per the definition the data must be converted to numerical format. But the above data is already in the numerical data format. (Fig:5.5)

CHAPTER 6

FEATURE SELECTIONS

6.1: SELECT RELEVANT FEATURES FOR ANALYSIS:

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested.

Here, we are training the model to predict the Indian Rupee / USA $ so therefore we will need the Indian Rupee/USA $ rates.

DATA FRAME

```
[10] df=data['INDIA - INDIAN RUPEE/US$']
     df
```

```
0        43.55
1        43.55
2        43.55
3        43.55
4        43.55
         ...
5212       NaN
5213     71.28
5214     71.45
5215     71.30
5216     71.36
Name: INDIA - INDIAN RUPEE/US$, Length: 5217, dtype: float64
```

Fig:6.1: Data Frame

6.2: DROP IRRELEVANT FEATURES:

Having irrelevant features in your data can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression.

Three benefits of performing feature selection before modeling your data are:

- Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy: Less misleading data means modeling accuracy improves.
- Reduces Training Time: Less data means that algorithms train faster.

Here, we do not require to drop any irrelevant features as there is no irrelevant features in the data set.

6.3: TRAIN-TEST-SPLIT:

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on.

Training model:

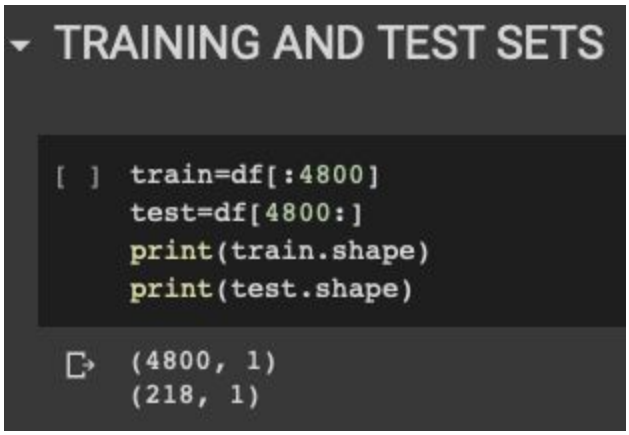

Fig:6.3(a):Training model

```
[ ] def get_data(data,look_back):
      data_x,data_y=[],[]
      for i in range(len(data)-look_back-1):
        data_x.append(data[i:(i+look_back),0])
        data_y.append(data[i+look_back,0])
      return np.array(data_x),np.array(data_y)
    look_back=1
    x_train,y_train=get_data(train,look_back)

    print(x_train.shape)
    print(y_train.shape)

    (4798, 1)
    (4798,)
```

Fig:6.3(b):Training model

Testing Model:

```
[ ] x_test,y_test=get_data(test,look_back)
    print(x_test.shape)
    print(y_test.shape)

    (216, 1)
    (216,)
```

Fig:6.3(c):Testing model

CHAPTER 7

MODEL BUILDING AND EVALUATION

7.1: BRIEF OF THE ALGORITHMS USED:

The algorithm which is used to predict the rates of INDIAN-RUPEE/USA$ is LSTM(Long Short-Term Memory).

LSTM(Long Short-Term Memory):

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

IDEA: In theory, classic (or "vanilla") RNNs can keep track of arbitrary long-term dependencies in the input sequences. The problem of vanilla RNNs is computational (or practical) in nature: when training a vanilla RNN using back-propagation, the gradients which are back-propagated can "vanish" (that is, they can tend to zero) or "explode" (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers. RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged. However, LSTM networks can still suffer from the exploding gradient problem.

DEFINING THE LSTM MODEL

```
[ ] n_features=x_train.shape[1]
    model=Sequential()
    model.add(LSTM(100,activation='relu',input_shape=(1,1)))
    model.add(Dense(n_features))

    ## Model Summary
    model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 100)               40800
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 40,901
Trainable params: 40,901
Non-trainable params: 0
_________________________________________________________________
```

Fig:7.1:LSTM model

7.2: TRAIN THE MODELS:

Training the models:

PROCESSING TRAIN AND TEST SETS FOR LSTM MODELS

```
[ ] x_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1)
    x_test=x_test.reshape(x_test.shape[0],x_test.shape[1],1)
    print(x_train.shape)
    print(x_test.shape)

    (4798, 1, 1)
    (216, 1, 1)
```

Fig:7.2(a):Processing Train LSTM model

Training

```
[ ] model.fit(x_train,y_train, epochs = 5, batch_size=1)

    Epoch 1/5
    4798/4798 [==============================] - 10s 2ms/step - loss: 0.0044
    Epoch 2/5
    4798/4798 [==============================] - 9s 2ms/step - loss: 9.4519e-05
    Epoch 3/5
    4798/4798 [==============================] - 9s 2ms/step - loss: 7.3005e-05
    Epoch 4/5
    4798/4798 [==============================] - 9s 2ms/step - loss: 7.3690e-05
    Epoch 5/5
    4798/4798 [==============================] - 9s 2ms/step - loss: 7.3439e-05
    <keras.callbacks.callbacks.History at 0x7f5f6e3e6208>
```

Fig:7.2(b):Training

7.3: VALIDATING THE DATA:

As we know without compiling the data no model works. So in this step we compile the model with default optimizer and loss.

COMPILING

```
[ ] model.compile(optimizer='adam',loss='mse')
```

Fig:7.3:Compiling

7.4: MAKE PREDICTIONS:

PREDICTING USING THE TRAINED MODEL

```
[ ] scaler.scale_
    y_pred=model.predict(x_test)
    y_pred=scaler.inverse_transform(y_pred)
    print(y_pred[:15])
```

```
[[71.19835 ]
 [71.03855 ]
 [70.958664]
 [70.709076]
 [70.908745]
 [71.03855 ]
 [70.71906 ]
 [70.78894 ]
 [70.79892 ]
 [70.43962 ]
 [69.91095 ]
 [69.98075 ]
 [69.89101 ]
 [69.71155 ]
 [69.5122  ]]
```

Fig:7.4(a):Prediction

PROCESSING TEST SHAPE

```
[ ] y_test = np.array(y_test).reshape(-1,1)
    y_test = scaler.inverse_transform(y_test)
    print(y_test[:10])
```

```
[[71.15]
 [71.07]
 [70.82]
 [71.02]
 [71.15]
 [70.83]
 [70.9 ]
 [70.91]
 [70.55]
 [70.02]]
```

Fig:7.4(b):Processing test shape

7.5: PREDICTIONS FROM RAW DATA:

VISUALIZING THE RESULTS

```
[ ] plt.figure(figsize=(10,5))
    plt.title('Foreign Exchange Rate of India')
    plt.plot(y_test , label = 'Actual', color = 'b')
    plt.plot(y_pred , label = 'Predicted', color = 'r')
    plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f5f651215f8>
```

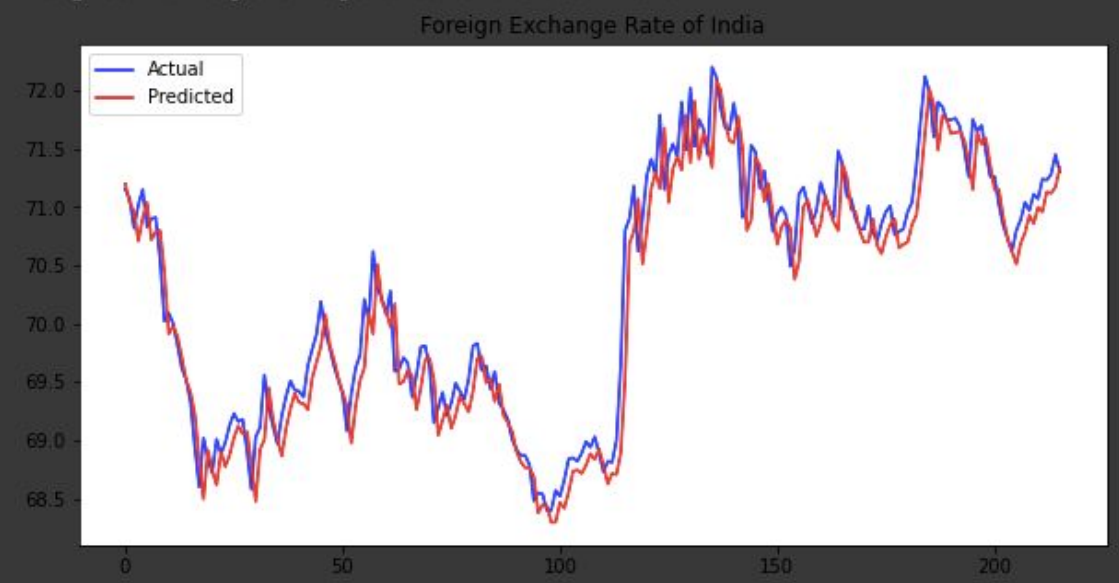

Fig:7.5(a):Plotting for actual and predicted rates

MEAN SQUARED ERROR:

MEAN SQUARED ERROR

```
[ ] from sklearn.metrics import mean_squared_error
    mean_squared_error(y_test, y_pred)
```

```
0.07726862930499022
```

Fig:7.5(b):Mean squared error:

PREDICTED AND ORIGINAL RATES:

10 original and predicted rates together

```
[ ] display(y_test[:10])
    display(y_pred[:10])

array([[71.15],
       [71.07],
       [70.82],
       [71.02],
       [71.15],
       [70.83],
       [70.9 ],
       [70.91],
       [70.55],
       [70.02]])
array([[71.19835 ],
       [71.03855 ],
       [70.958664],
       [70.709076],
       [70.908745],
       [71.03855 ],
       [70.71906 ],
       [70.78894 ],
       [70.79892 ],
       [70.43962 ]], dtype=float32)
```

Fig:7.5(c):predicted and actual rates

CONCLUSION

The LSTM (Long Short-Term Memory) model has predicted almost the perfect values as the original rates. The LSTM model had a mean square error of 0.077 which is a really low error value and which is required for the perfect prediction of the Indian Foreign exchange rates.