**Mini Project**

“PREDICTION OF STOCKS”

**Submitted in Partial Fulfillment of the Academic**

**Requirement for the Award of Degree of**

BACHELOR OF TECHNOLOGY

IN

**Computer Science & Engineering**

Submitted

**By**

**D. SHIVANI (17R01A0575)**

**K. SAI ANUHYA (17R01A0592)**

**K. SAI DEEKSHITH (17R01A0593)**

**M. SRAVANI (17R01A05A2)**

**Under the esteemed guidance of**

Mrs. B.VASAVI

(Assistant Professor)



CMR INSTITUTE OF TECHNOLOGY

**(Approved by AICTE, Affiliated to JNTU, Hyderabad)**

**Kandlakoya, Medchal Road, R.R. Dist., Hyderabad.**

2020-21

**CMR institute of technology**

**(UGC AUTONOMOUS)**

**(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad) Kandlakoya, Medchal Road, Hyderabad.**



# **CERTIFICATE**

This is to certify that an Industry oriented Mini Project entitled with: “PREDICTION OF STOCKS” is being

Submitted By

D. Shivani (17R01A0575)

K. Sai Anuhya (17R01A0592)

K. Sai Deekshith (17R01A0593)

M. Sravani (17R01A05A2)

In partial fulfillment of the requirement for award of the degree of B. Tech in CSE to the JNTUH, Hyderabad is a record of a bonafide work carried out under our guidance and supervision.

The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University for award of any other degree or diploma.

**Signature of Guide Signature of Coordinator Signature of HOD**

(Mrs. B. Vasavi) (Mr. Alagumuthu Krishnan) (Mr. A.Prakash)

**ACKNOWLEDGEMENT**

We are extremely grateful to **Dr. M. Janga Reddy**, **Director**, **Dr. B. Satyanarayana**, **Principal** and **A .Prakash**, **Head of Department**, Dept of Computer Science and Engineering, CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to **Mrs. B. Vasavi**, **Mini Project Guide** and coordinator **Mr. Alagumuthu Krishnan**, Dept of Computer Science and Engineering, CMR Institute of Technology for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

D. Shivani (17R01A0575)

K. Sai Anuhya (17R01A0592)

K. Sai Deekshith (17R01A0593)

M. Sravani (17R01A05A2)

**I**

**ABSTRACT**

The aim of the mini project is to analyze the different machine learning forecast techniques to profit the site invest in the future for better returns based on past returns and various other metrics of the company. We are going to produce a portfolio of multiple stocks in order to minimize the risk factor. We do this by the application of supervised learning algorithm and classification methods of Machine Learning .The motivation of this mini project is to provide an investor an insight on which stocks to invest in based on its previous returns and the company's previous price movement, etc.

The solubility and violent nature of the stocks makes it difficult to predict when to invest and as the data keeps on increasing it makes it nearly impossible for a person to come on a decision without any help. So, we provide a machine learning aided approach to evaluate few different tracks using diverse attributes present in the data and to provide an analysis on which ones are better to invest in.

**TABLE OF CONTENTS**

**ACKNOWLEDGEMENT**

**ABSTRACT**

**LIST OF FIGURES**

**1. INTRODUCTION**

1.1 MOTIVATION

1.2 DOMAIN BACKGROUND

1.3 PROBLEM DEFINITION

1.4 PROBLEM OBJECTIVE

1.5 LIMITATIONS

**2. PROJECT DESCRIPTION**

2.1 MACHINE LEARNING

2.1.1 LINEAR REGRESSION 2.1.2 NEURAL NETWORKS

2.2 DEEP LEARNING

2.2.1 Long-Short Term Memory (LSTM) network

2.3 EXISTING SYSTEMS

2.4 PROPOSED SYSTEMS

**3. PROJECT ANALYSIS**

3.1 DATA EXPLORATION

3.2 EXPLORATORY VISUALIZATION

3.3 PROJECT REQUIREMENTS

3.3.1 PROGRAMMING LANGUAGE

3.3.1.1 PYTHON

3.3.2 LIBRARIES NEEDED

3.3.3 SOFTWARE NEEDED

3.3.3.1 ANACONDA DISTRIBUTION

3.3.3.2 ANACONDA NAVIGATOR

3.3.3.3 CONDA

3.4 ALGORITHMS AND TECHNIQUES

**4. PROJECT DESIGN**

4.1 PROJECT STEPS

4.2 ARCHITECTURE DIAGRAM

4.3 SEQUENCE FLOW DIAGRAM

4.4 ACTIVITY DIAGRAM

4.5 DATA FLOW DIAGRAM

4.6 USE CASE DIAGRAM

4.7 RECENT ADVANCES IN STOCK MARKET PREDICTION

**5. PROJECT METHODOLOGY**

5.1 PREPROCESS OF DATASET

5.2 LINEAR REGRESSION MODEL

5.3 LONG-SHORT TERM MEMORY NETWORK

**6. OUTPUT**

**7. ANALSIS OF THE RESULT**

**8. CONCLUSION**

**9. FUTURE SCOPE**

**10. REFERENCES**

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **FIG.NO** | **NAME OF THE FIGURE** |
| 2.1.1 | Linear Regression model plot |
| 4.1.1 | A basic neural network model |
| 4.2 | LSTM network |
| 4.3 | Google stock plot |
| 4.4 | Anaconda Navigator |
| 4.5 | Conda prompt |
| 4.6 | Preprocessing code |
| 4.7 | Linear Regression Code(1) |
| 4.8 | Linear Regression Code(2) |
| 5.1.1 | LSTM Model Code(1) |
| 5.2.1 | LSTM Model Code(2) |
| 5.3.1 | LSTM Model Code(3) |
| 5.4.1 | Predicted Values and Accuracy |
| 5.4.2 | Linear Regression Predicted Values |
| 6.1.1 | Forecast values of model |
| 6.2.1 | Plot of initial dataset |
|  | Graph of LSTM predictions(1) |
|  | Error and predictions of LSTM(1) |
|  | Graph of LSTM predictions(1) |
|  |  |

1. **INTRODUCTION:**

**1.1 Motivation:**

The motivation of this project is to learn and implement the algorithms of a basic machine learning and a Deep Learning concept in order to show case the difference and the development in the Artificial Intelligence field. The sector that we chose to implement these algorithms in stock market. The idea is to implement a Long Short Term Memory (LSTM) network to calculate the actual output and the hypothesis (output) predicted by the model and to provide information on stock prices to investors and traders.

**1.2 Domain Background:**

Investment firms, hedge funds and even individuals are using financial models to better understand and interpret market patterns and make investments and trade for better profit. Abundance of information, which is available in the form of previous stock prices, company data like (Revenue, balance sheet, etc) and sometimes external factors like trends, inflation, liquidity, etc and there are various machine learning concepts and algorithms to do so.

Can we actually predict the stock prices of shares with machine learning? Most of the investors make educated guesses by analyzing data. They will read news, study the company history, industry trends and other tors are taken into consideration as well. The general perception is that stock prices are totally random and predictable but in the last decade many investment firms like AlphaSense, Quandl, Morgan Stanley, etc have led expert data scientists to build predictive models for them. This has led to many other firms to adapt to this new way of analyzing stocks rather than having experts collect data and call their clients to provide data. Most the orders in Wall Street are now placed by software.

In this project we are going utilize the concept of Neural Networks in Machine Learning to build an algorithm to predict stock prices.

**1.3 Problem Definition:**

The challenge of this project is to accurately predict the future closing value of a given stock across a period of time in the future using predictive models. In the past few years there have been lots of academic papers published using neural nets to predict stock prices with varying degrees of success but until recently to build these models has been restricted to academics. Now with libraries like tensor flow anyone can build powerful predictive models trained on massive of datasets.

**1.4 Project Objective:**

For this project we will be using a Long Short Term Memory networks- usually just called "ISTM", which is an improved model of Recurrent Neural Network, and Linear Regression model to predict the closing price of the google and other stock prices using a dataset of past prices. In the end, we are going to compare the results of both models to see the improvement and the difference between the two.

**Goal of the project:**

1. Explore stock prices (dataset)
2. Clean the dataset and preprocess it for the models
3. Implement basic model of the algorithm using Linear Regression
4. Implement a LSTM Neural Network
5. Finally compare the output with the dataset.

**1.5 Limitations:**

The biggest factor that effects the hypothesis projected by our models is the dataset. If the dataset is too small, then we will end up with a poor predictive model. We are also taking into effect the stock prices of the past and not considering other factors that might affect the outcome of our hypothesis. Our dataset should also should be in a certain format as we are taking only specified attributes for building our predictive models.

1. **PROJECT DESCRIPTION:**

**2.1 Machine Learning:**

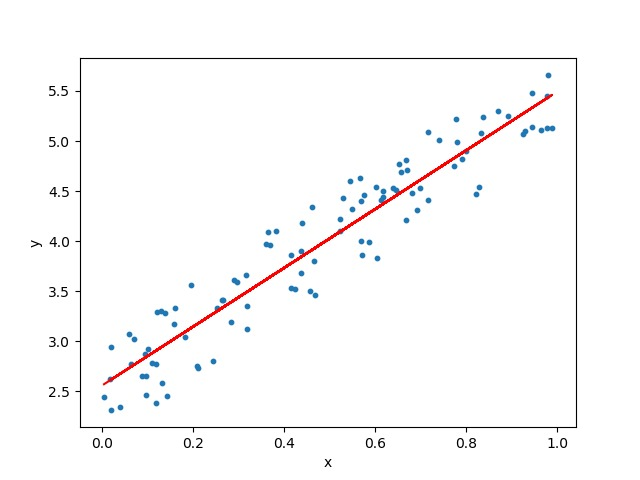
Machine learning is an application of artificial intelligence (Al) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or reconstruction, in order to look for patters in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or distance and adjust actions accordingly.

The project can be classified as a Supervised Machine Learning problem, as we are going to predict the output values of known training dataset, using a hypothesis that is going to be tested using a test dataset and used for future data.

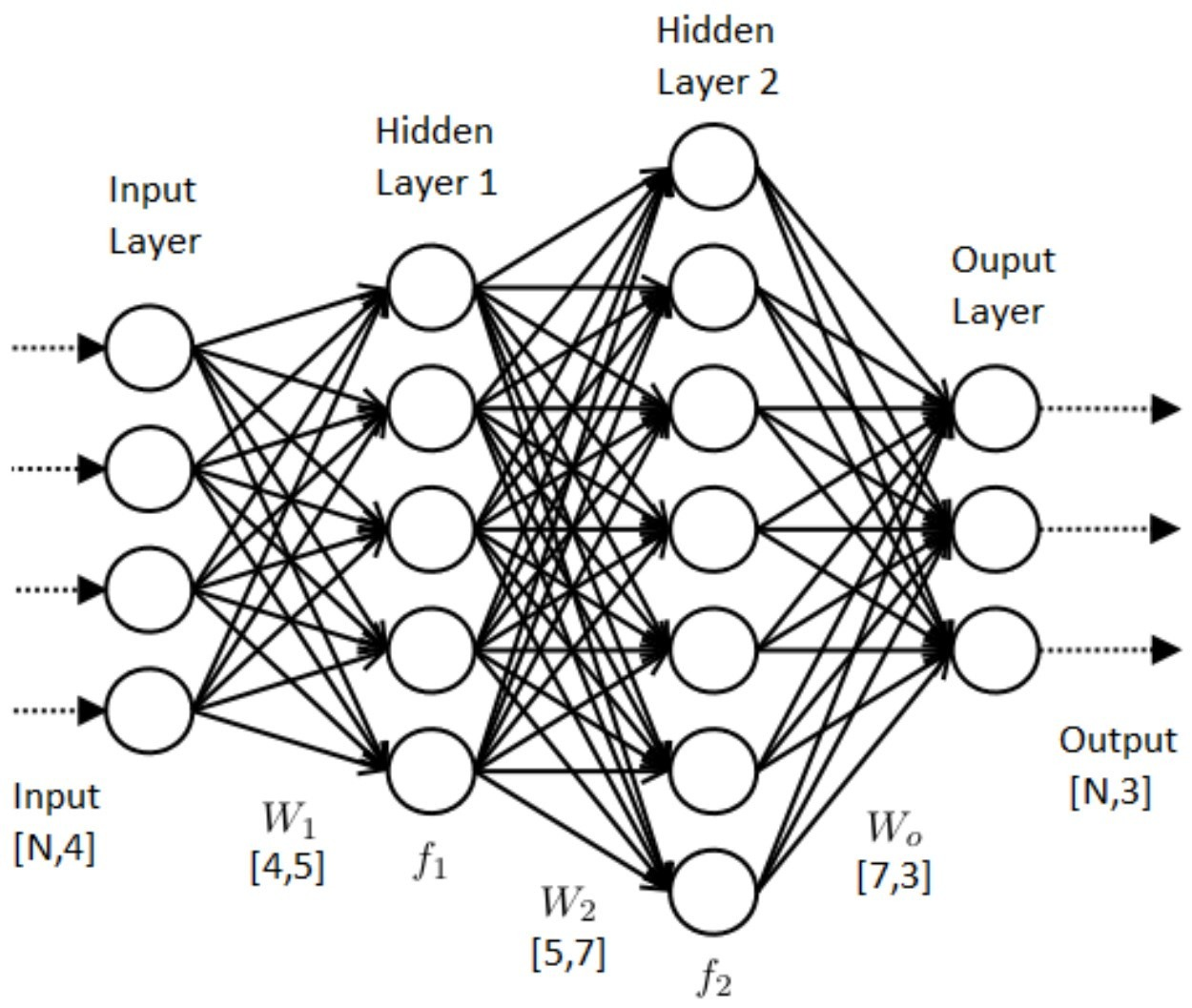
**2.1.1 Linear Regression:**

Linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm. Linear Regression predicts a real-valued output based on some input value of its features. This concept is mostly used for a basic prediction model**.**



**2.1.2 Neural Network:**

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in union to solve specific problems. ANNS, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.



**2.2 Deep Learning:**

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.

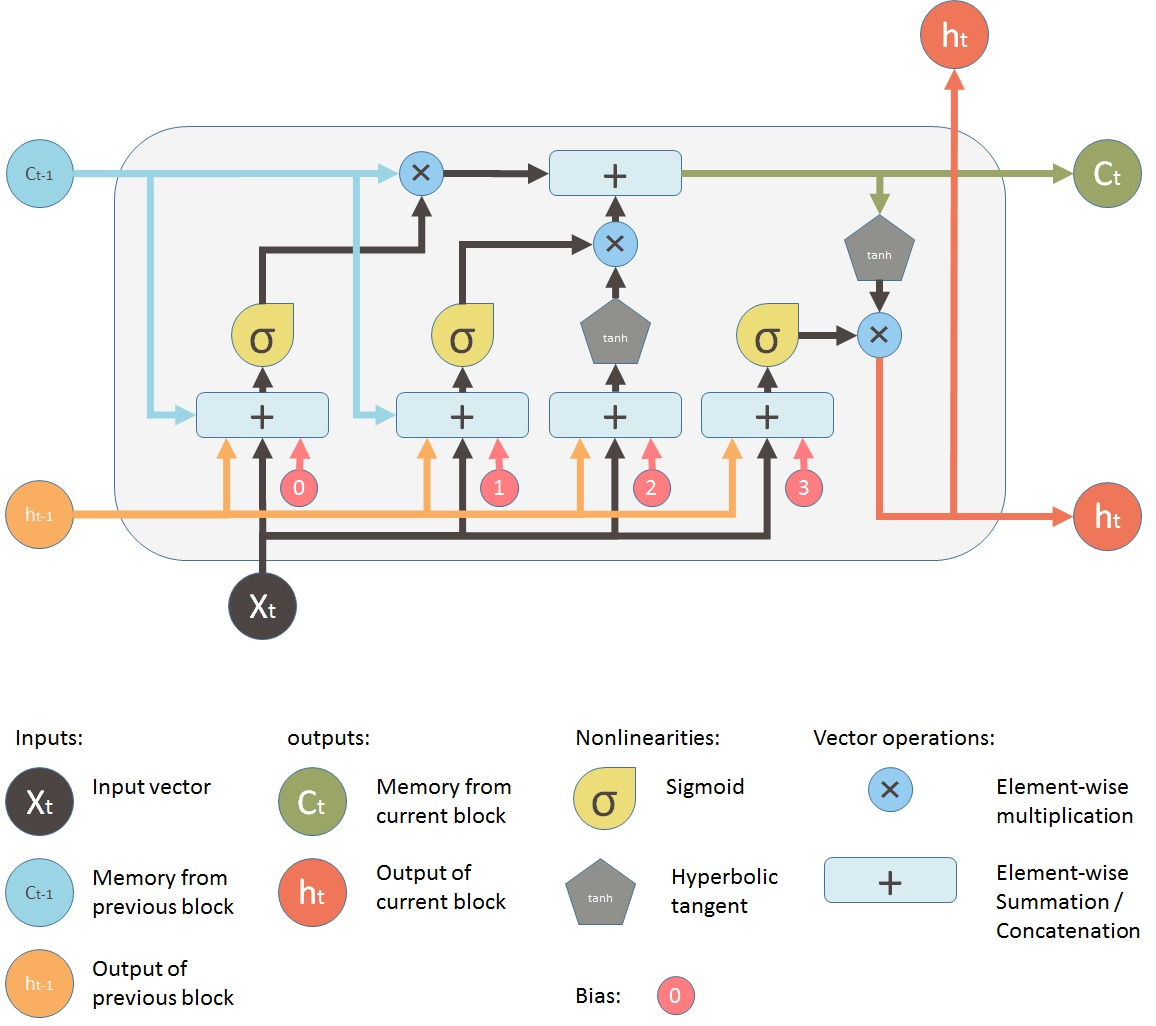
Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human

The concept of Deep learning that we are going to use in this project is Long Short Term Memory network improved version of Recurrent Neural Networks.

**2.2.1 Long-Short Term Memory (LSTM) network:**

Long Short Term Memory networks - usually just called "LSTMs"-are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were redefined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour, not something they struggle to learn!



**2.3 Existing Systems:**

* Predicting how the stock market will perform is one of the most difficult things to do.
* There are so many factors involved in the prediction – physical factors vs. psychological, rational and irrational behavior, etc.
* All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy.

**2.4 Proposed Systems:**

* It allows the investors or online traders to peek into the reasons behind a certain market trend, pricing and understand price behavior, which was otherwise impossible to know just a few years back.
* Access to Machine Learning helps mitigate possible risks in online trading and enables the investor to make precise decisions.

1. **PROJECT ANALYSIS**

**3.1 Data Exploration:**

The data used in this project is of the Alphabet Inc 3 from January 1, 2005 to June 20, 2017, this is a series data points indexed in time order or a time series. Our goal is to predict the closing price for any date after training. For case of reproducibility and reusability, all data was pulled from the Google

The prediction has to be made for Closing (Adjusted closing) price of the data. Since Google Finance y adjusts the closing prices for us, we just need to make prediction for ""CLOSE"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DATE | OPEN | HIGH | LOW | CLOSE | VOLUME |
| 30-Jun-17 | 943.99 | 945.00 | 929.61 | 929.68 | 2287662 |
| 29-Jun-17 | 951.35 | 951.66 | 929.60 | 937.82 | 3206674 |
| 28-Jun-17 | 950.66 | 963.24 | 936.16 | 961.01 | 2745568 |

The dataset is of following form:

Table: The Whole data can be found out in ‘Google.csv’

**3.2 Exploratory Visualization**:

Visualize the data, I have used matplotlib library, I have plotted closing stock price of the data with number of items (number of days) available.



Google stock plot

X-axis: represents trading days

Y-axis: represents closing price in USD

**3.3 Project Requirements:**

**3.3.1 Programming Languages:**

**3.3.1.1 Python:**

The core part of the project is going to be done in Python, which is a general purpose, dynamic, high level and interpreted programming language. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library. It is simple and easy to learn and provides lots of high-level data structures. Python's syntax and dynamic typing with its interpreted nature. makes it an ideal language. The primary factor for choosing this language is due to its that support the implementation of the neural network concept, plot graphs, etc.

**3.3.2 Libraries Needed:**

1. **NumPy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

2. **SciPy:** It is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering

**3. Matplotlib:** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a product "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged SciPy makes use of matplotlib..

**4. Tensor flow:** TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

**5. Keras:** It is an open source neural network library written in Python. It is capable of running on top of Tensor flow, Microsoft Cognitive Toolkit or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

**6. Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

**7. Sklearn:** It has features various classification, regression and clustering algorithms including support sector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**3.3.3 Software Needed:**

**3.3.3.1Anaconda Distribution:**

Anaconda Enterprise is an Al/ML enablement platform that empowers organizations to develop, govern and automate Al/ML and data science from laptop through training to production. It lets organizations scale individual data scientists to collaborative teams of thousands, and to go from a single server to thousands nodes for model training and deployment.

Anaconda distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, called Anaconda Navigator, so it eliminates the need to learn to install each library independently.

The open source data packages can be individually installed from the Anaconda repository with the Conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of Conda packages and in most cases they can work together. You can also make your own custom packages using the Conda build command, and you can share them with others by uploading them to Anaconda Cloud, PyPy or other repositories.

The default installation of Anaconda-2 includes Python 2.7 and Anaconda-3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with Conda.

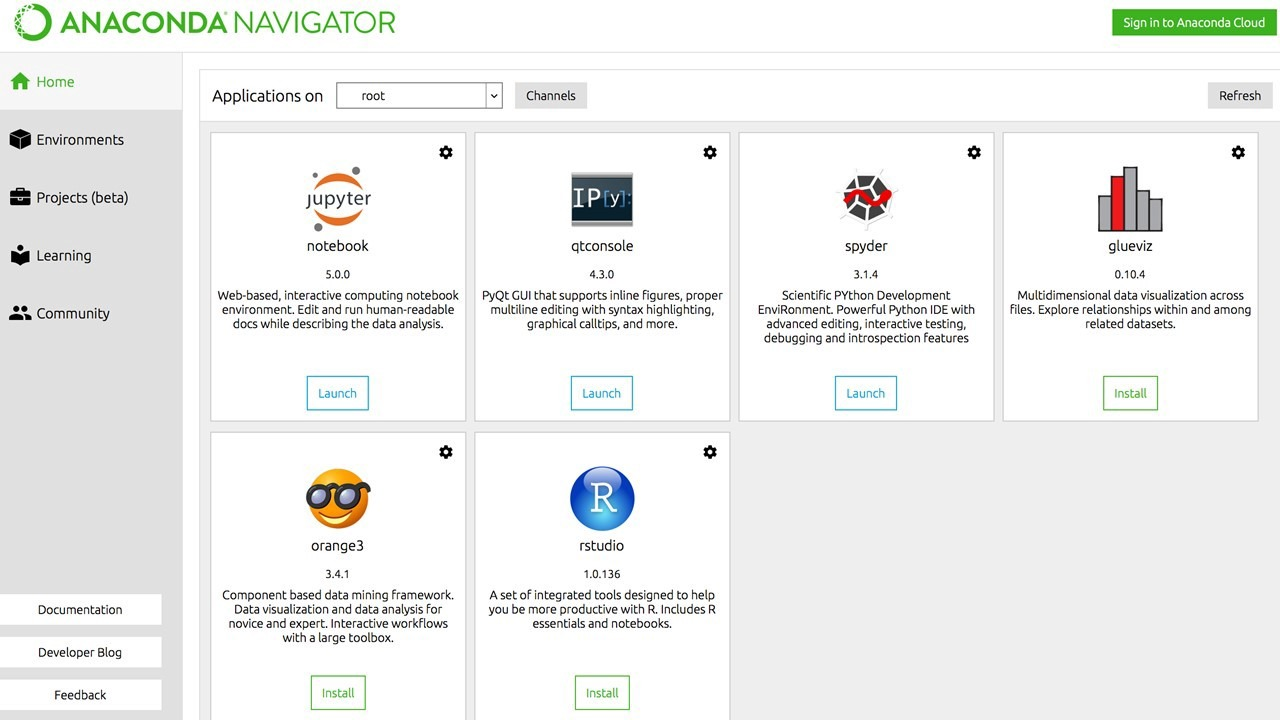
**3.3.3.2 Anaconda Navigator:**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that s users to launch application and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, OS and Linux.

Navigator is automatically included with Anaconda version 4.0.0 or higher.

The following applications are available by default in Navigator:

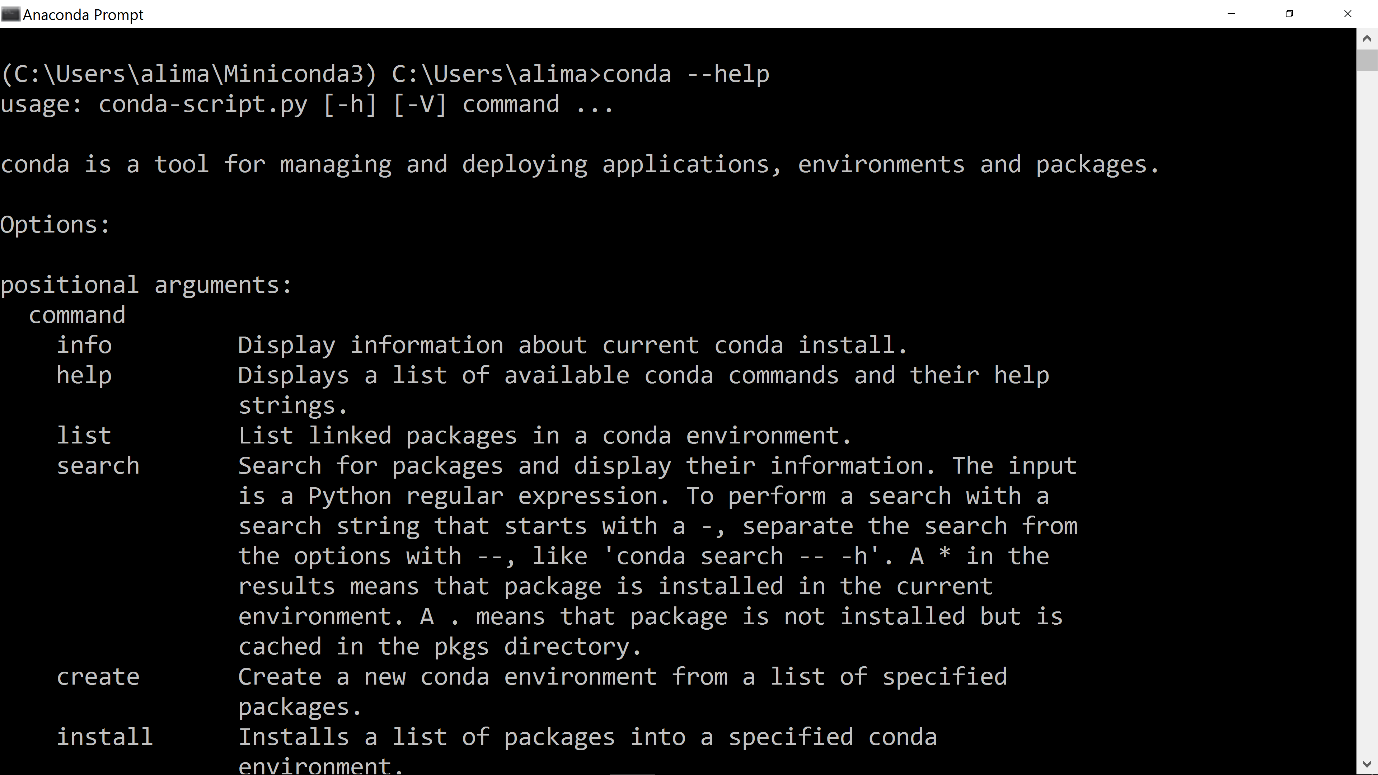
* JupyterLab
* Jupyter Notebook
* QtConsole
* Spyder
* Glueviz
* Orange
* Rstudio
* Visual Studio Code



**ANACONDA NAVIGATOR**

**3.3.3.3 Conda:**

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs but it can package and distribute software for any language (e.g. R), including multi-language projects. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.



Conda Prompt

**3.4 Algorithms and Techniques:**

The goal of this project was to study time-series data and explore as many options as possible to accurately predict the Stock Price. Through our research we came to know about Recurrent Neural Nets (RNN) which is used specifically for sequence and pattern learning. As they are networks with loops in them, this allows information to persist and this ability to memorize the data accurately. But Recurrent Neural Nets have vanishing descent problem which does not allow it to learn from past data as was expected. The remedy of this problem was solved in Long Short Term Memory Networks, usually referred as LSTMs There are a special kind of RNN, capable of learning long-term dependencies.

In addition to adjusting the architecture of the Neural Network, the following full set of parameters can he tuned the prediction model:

**Input Parameters:**

* Preprocessing and Normalization

**Neural Network Architecture:**

* Number of Layers
* Number of Nodes

**Training Parameters:**

* Training /Test Split (In this project we divided dataset into 75/25 for Linear Regression Model and 80/20 for LSTM)
* Batch Size
* Optimizer Function
* Epochs (How many times to in through training process)

1. **PROJECT DESIGN:**

**4.1 Project Steps:**

The project will be implemented through the Keras/Tensor Flow using LSTM Neural Networks. Development work flow will follow the below sequence.

1. **Set up Infrastructure:**

* Anaconda
* Incorporate required Libraries (Keras, Tensor flow. Pandas, Matplotlib, Sklearn, Numpy)
* Download cloudera Manager and Hadoop

1. **Prepare Dataset:**

* Incorporate data of S&P 500 companies
* Import the data into the Hadoop's HDFS environment
* Perform clean-up operations on dataset using a Map Reduce program
* Extract the cleaned dataset
* Process the requested data into Pandas Data frame
* Develop function for normalizing data
* Dataset will be used with a 80/20 split on training and test data across all models

1. **Develop Benchmark Model:**

* Set up basic Linear Regression model with Scikit-Learn
* Calibrate parameters

1. **Develop Basic LSTM Model:**

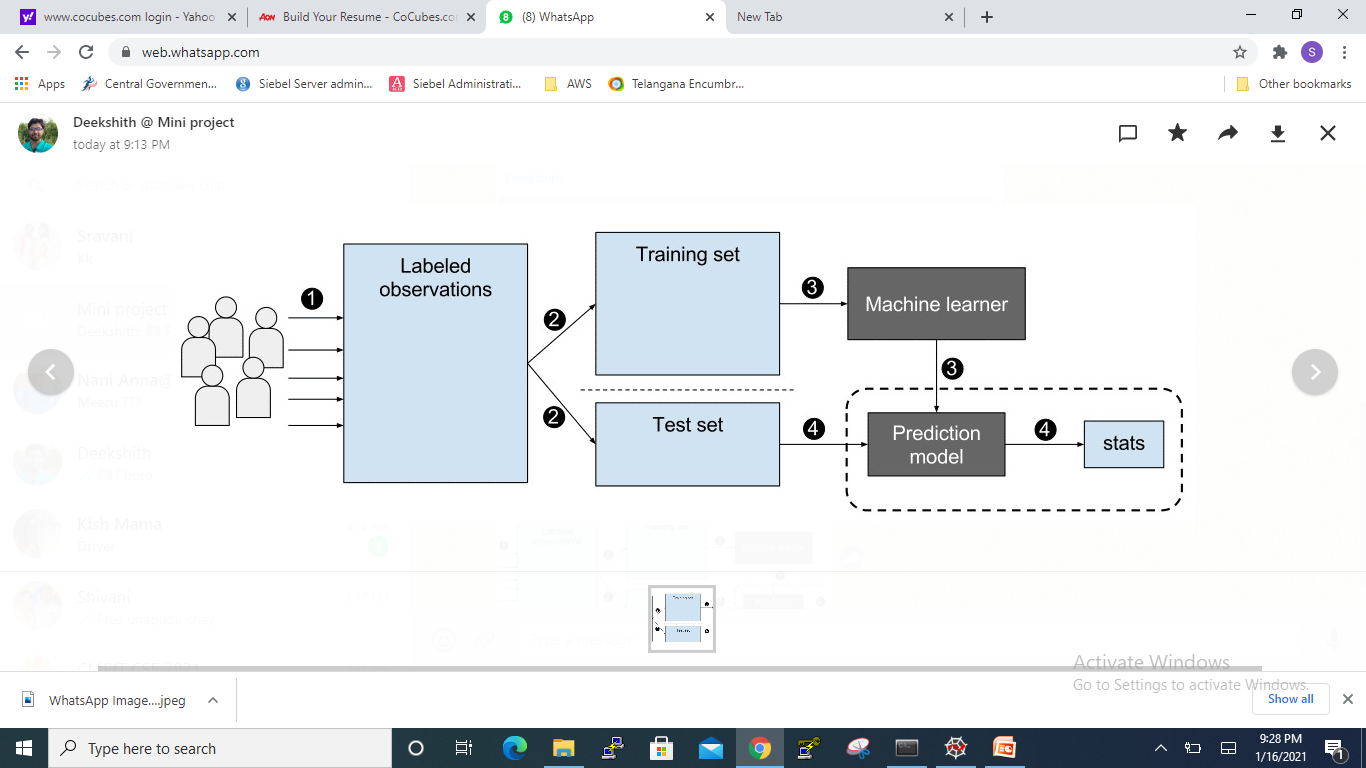
* Set up basic LSTM model with Keras utilizing parameters from Benchmark Model

1. **Improve LSTM Model:**

* Develop, document, and compare results using additional labels for the LSTM model

1. **Document and Visualize Results:**

* Plot Act, Benchmark Predicted Values, and LSTM Predicted Values per time series.
* Analyze and describe results
  1. **Architecture Diagram:**



**4.3 Sequence Flow Diagram:**

**Defining Training**

**Parameters**

Step-1

**Define Model Architecture**

**Linear Regression (Basic) &**

**LSTM (Optimized**)

Step-2

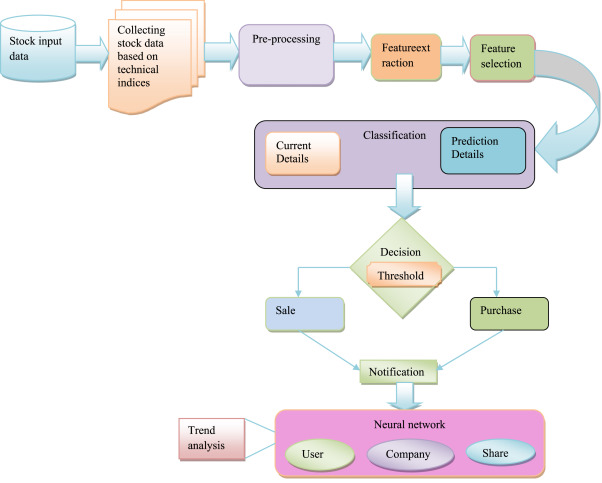
**Train the model and generate prediction output**

Step-3

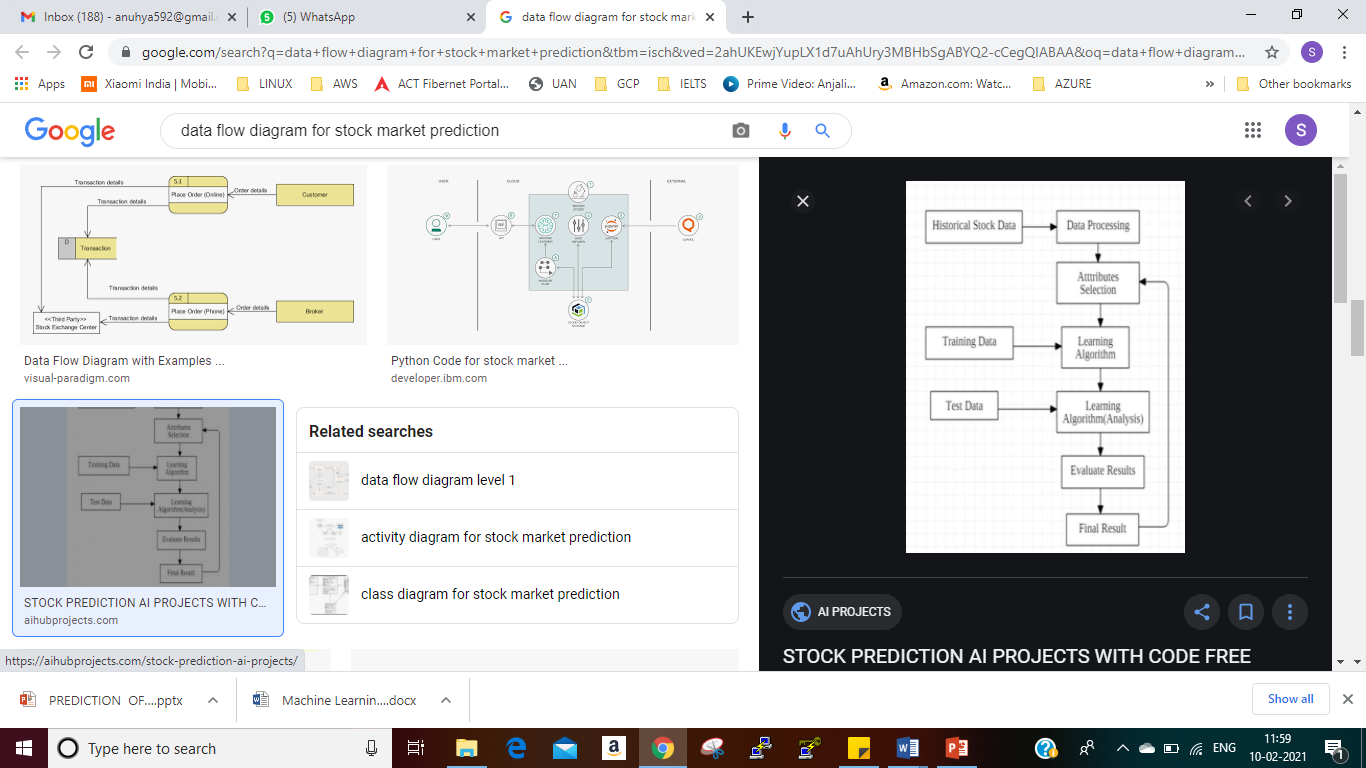
**Plot prediction, analyze and compare results**

Step-4

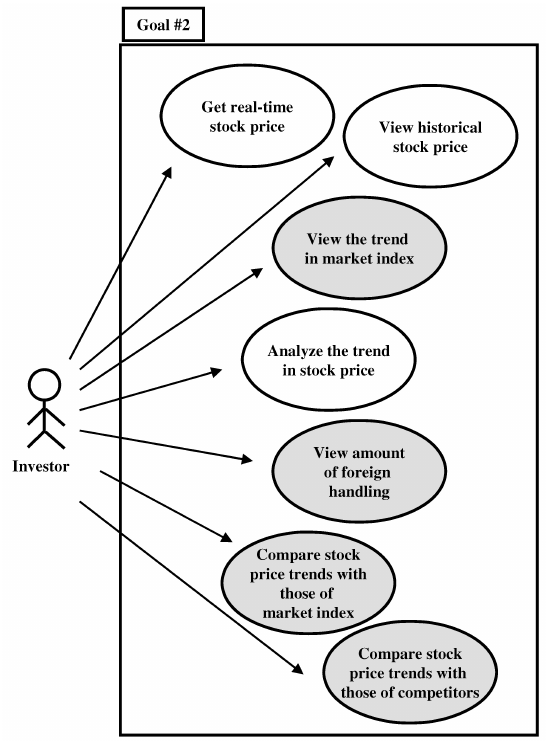
**4.4 Activity Diagram:**

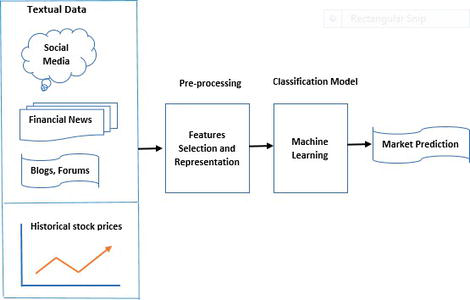


**4.5 Data Flow Diagram:**

****

**4.6 Use Case Diagram:**





**4.7 Recent Advances in Stock Market Prediction:**

1. **PROCESS METHODOLOGY:**
   1. **Preprocess of Dataset:**

The code for the preprocessing process of the dataset in saved and done using the execution of the code preprocessing.py file. The function of this file is to create a time series dataset from ID data if the necessity exists. This is particularly useful when we want to plot the data.

**Preprocessing.py**

import numpy as np

# FUNCTION TO CREATE 1D DATA INTO TIME SERIES DATASET

def new\_dataset(dataset, step\_size):

data\_X, data\_Y = [], []

for i in range(len(dataset)-step\_size-1):

a = dataset[i:(i+step\_size), 0]

data\_X.append(a)

data\_Y.append(dataset[i + step\_size, 0])

return np.array(data\_X), np.array(data\_Y)

# THIS FUNCTION CAN BE USED TO CREATE A TIME SERIES DATASET FROM ANY 1D ARRAY

* 1. **Linear Regression Model:**

In this code we will initially get the dataset from quandl, which contains financial accessible via an API, we will build a linear regression model to plot and predict the stock prices for the next month.

In this code:

* Getting dataset from quandl: The required dataset is obtained from quandl using API.
* We will get the attributes and store them in an array form.
* Split the dataset into training and test set using model selection.
* Build the Linear Regression Model and giving the parameters to it
* Training the model
* Finally plotting the graph

**LinearRegressionModel.py**

import pandas as pd

import numpy as np

import quandl, math, time, datetime

import matplotlib.pyplot as plt

import pickle

from matplotlib import style

from sklearn import preprocessing, model\_selection, svm

from sklearn.linear\_model import LinearRegression

quandl.ApiConfig.api\_key = 'nwE\_WyQ\_7VLxsXDpgU7H'

df = quandl.get('WIKI/GOOGL')

df=df[['Adj. Close']]

df.fillna(-99999, inplace=True)

forecast = int(math.ceil(0.01\*len(df)))

df['Forecast'] = df['Adj. Close'].shift(-forecast)

'''

if this shift is used,

[1% of stocks lifetime] (e.g. 33) days ago is the forecasted price for that day.

this shift is built to learn from the stocks trend.

'''

# X - Features (Adj. Close) | y - Labels (Forecast)

X = np.array(df.drop(['Forecast'], 1))

X = preprocessing.scale(X)

X\_lately = X[-forecast:]

X = X[:-forecast]

df.dropna(inplace=True)

y = np.array(df['Forecast'])

# Training - test size?

X\_train, X\_test, y\_train, y\_test = model\_selection.train\_test\_split(X, y, test\_size=0.2)

# Classifier

clf = LinearRegression(n\_jobs=-1)

clf.fit(X\_train, y\_train)

# Storing + Accessing Training Data

with open('prediction.pickle','wb') as f:

pickle.dump(clf, f)

pickle\_in = open('prediction.pickle', 'rb')

clf = pickle.load(pickle\_in)

accuracy = clf.score(X\_test, y\_test)

forecast\_set = clf.predict(X\_lately)

print(forecast\_set, accuracy, forecast)

df['Forecast'] = np.nan

last\_date = df.iloc[-1].name

last\_unix = time.mktime(last\_date.timetuple())

one\_day = 86400

next\_unix = last\_unix + one\_day

for i in forecast\_set:

next\_date = datetime.datetime.fromtimestamp(next\_unix)

next\_unix += one\_day

df.loc[next\_date] = [np.nan for \_ in range(len(df.columns)-1)] + [i]

# Plotting on Graph

style.use('ggplot')

df['Adj. Close'].plot()

df['Forecast'].plot()

plt.legend(loc=4)

plt.xlabel("Date")

plt.ylabel("Price")

plt.show()

print(df.tail(35))

* 1. **Long-Short Term Memory Network:**

In this model we are trying to implement an LSTM network model to predict the stock prices of the past that is given as an input in the code itself. For building an LSTM Model, we are going to use the Keras library and as output give the error of the predicted values of both training and test set, as well the predicted value of next day.

In this code:

* Importing Dataset: This step includes reading the required attributes of the dataset stored in an csv file and reshaping the indexes
* Taking different indicators from attributes for prediction: Here, we are creating indicators like OHLP(mean of attributes Open, High, Low, Close), HLC(mean of attributes High, Low, Close)
* Plotting the indicators: We are going to plot the above indicators in a graph using matplotlib library.
* Preparing time series dataset
* Splitting dataset into training and test sets: We are splitting the dataset into training set (75%) and test set(25%).
* Reshaping the datasets
* Building the LSTM Model: This involves using the Keras library to build a LSTM network by specifying the density of layers and number of layers
* Model training: This step is to compile the model and specifying the epochs, batch size, etc
* Prediction of training and test sets
* De-normalizing dataset for plotting
* Calculating Root mean square error for training and test sets
* Creation of plots: To showcase the difference between the actual and predicted values in a graph
* Plot the training and test predictions
* Predicting future values: We are going to print the predicted values for the next day

**StockPricePrediction.py**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import math

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

from keras.models import Sequential

from keras.layers import Dense, Activation

from keras.layers import LSTM

import preprocessing

np.random.seed(7)

# IMPORTING DATASET

dataset = pd.read\_csv('google.csv', usecols=[1,2,3,4])

dataset = dataset.reindex(index = dataset.index[::-1])

# CREATING OWN INDEX FOR FLEXIBILITY

obs = np.arange(1, len(dataset) + 1, 1)

# TAKING DIFFERENT INDICATORS FOR PREDICTION

OHLC\_avg = dataset.mean(axis = 1)

HLC\_avg = dataset[['High', 'Low', 'Close']].mean(axis = 1)

close\_val = dataset[['Close']]

# PLOTTING ALL INDICATORS IN ONE PLOT

plt.plot(obs, OHLC\_avg, 'r', label = 'OHLC avg')

plt.plot(obs, HLC\_avg, 'b', label = 'HLC avg')

plt.plot(obs, close\_val, 'g', label = 'Closing price')

plt.legend(loc = 'upper right')

plt.show()

# PREPARATION OF TIME SERIES DATASE

OHLC\_avg = np.reshape(OHLC\_avg.values, (len(OHLC\_avg),1))

scaler = MinMaxScaler(feature\_range=(0, 1))

OHLC\_avg = scaler.fit\_transform(OHLC\_avg)

# TRAIN-TEST SPLIT

train\_OHLC = int(len(OHLC\_avg) \* 0.75)

test\_OHLC = len(OHLC\_avg) - train\_OHLC

train\_OHLC, test\_OHLC = OHLC\_avg[0:train\_OHLC,:], OHLC\_avg[train\_OHLC:len(OHLC\_avg),:]

# TIME-SERIES DATASET (FOR TIME T, VALUES FOR TIME T+1)

trainX, trainY = preprocessing.new\_dataset(train\_OHLC, 1)

testX, testY = preprocessing.new\_dataset(test\_OHLC, 1)

# RESHAPING TRAIN AND TEST DATA

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

step\_size = 1

# LSTM MODEL

model = Sequential()

model.add(LSTM(32, input\_shape=(1, step\_size), return\_sequences = True))

model.add(LSTM(16))

model.add(Dense(1))

model.add(Activation('linear'))

# MODEL COMPILING AND TRAINING

model.compile(loss='mean\_squared\_error', optimizer='adagrad')

model.fit(trainX, trainY, epochs=10, batch\_size=1, verbose=2)

# PREDICTION

trainPredict = model.predict(trainX)

testPredict = model.predict(testX)

# DE-NORMALIZING FOR PLOTTING

trainPredict = scaler.inverse\_transform(trainPredict)

trainY = scaler.inverse\_transform([trainY])

testPredict = scaler.inverse\_transform(testPredict)

testY = scaler.inverse\_transform([testY])

# TRAINING RMSE

trainScore = math.sqrt(mean\_squared\_error(trainY[0], trainPredict[:,0]))

print('Train RMSE: %.2f' % (trainScore))

# TEST RMSE

testScore = math.sqrt(mean\_squared\_error(testY[0], testPredict[:,0]))

print('Test RMSE: %.2f' % (testScore))

# CREATING SIMILAR DATASET TO PLOT TRAINING PREDICTIONS

trainPredictPlot = np.empty\_like(OHLC\_avg)

trainPredictPlot[:, :] = np.nan

trainPredictPlot[step\_size:len(trainPredict)+step\_size, :] = trainPredict

# CREATING SIMILAR DATASSET TO PLOT TEST PREDICTIONS

testPredictPlot = np.empty\_like(OHLC\_avg)

testPredictPlot[:, :] = np.nan

testPredictPlot[len(trainPredict)+(step\_size\*2)+1:len(OHLC\_avg)-1, :] = testPredict

# DE-NORMALIZING MAIN DATASET

OHLC\_avg = scaler.inverse\_transform(OHLC\_avg)

# PLOT OF MAIN OHLC VALUES, TRAIN PREDICTIONS AND TEST PREDICTIONS

plt.plot(OHLC\_avg, color='blue', label = 'original dataset')

plt.plot(trainPredictPlot, color='red', label = 'training set')

plt.plot(testPredictPlot, color='green', label = 'predicted stock price/test set')

plt.legend(loc = 'upper right')

plt.xlabel('Time in Days')

plt.ylabel('OHLC Value of Google Stocks')

plt.show()

# PREDICT FUTURE VALUES

last\_val = testPredict[-1]

last\_val\_scaled = last\_val/last\_val

next\_val = model.predict(np.reshape(last\_val\_scaled, (1,1,1)))

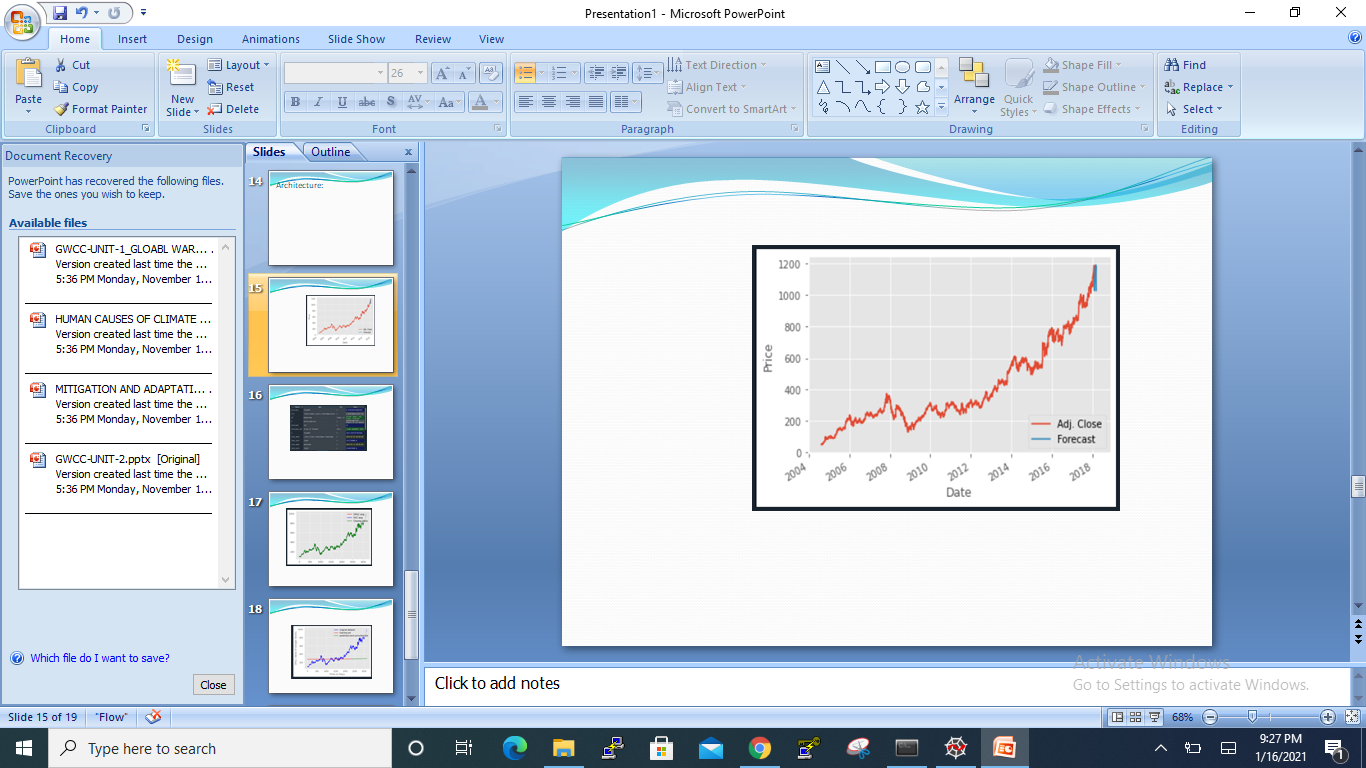
print ("Last Day's Value:", np.asscalar(last\_val))

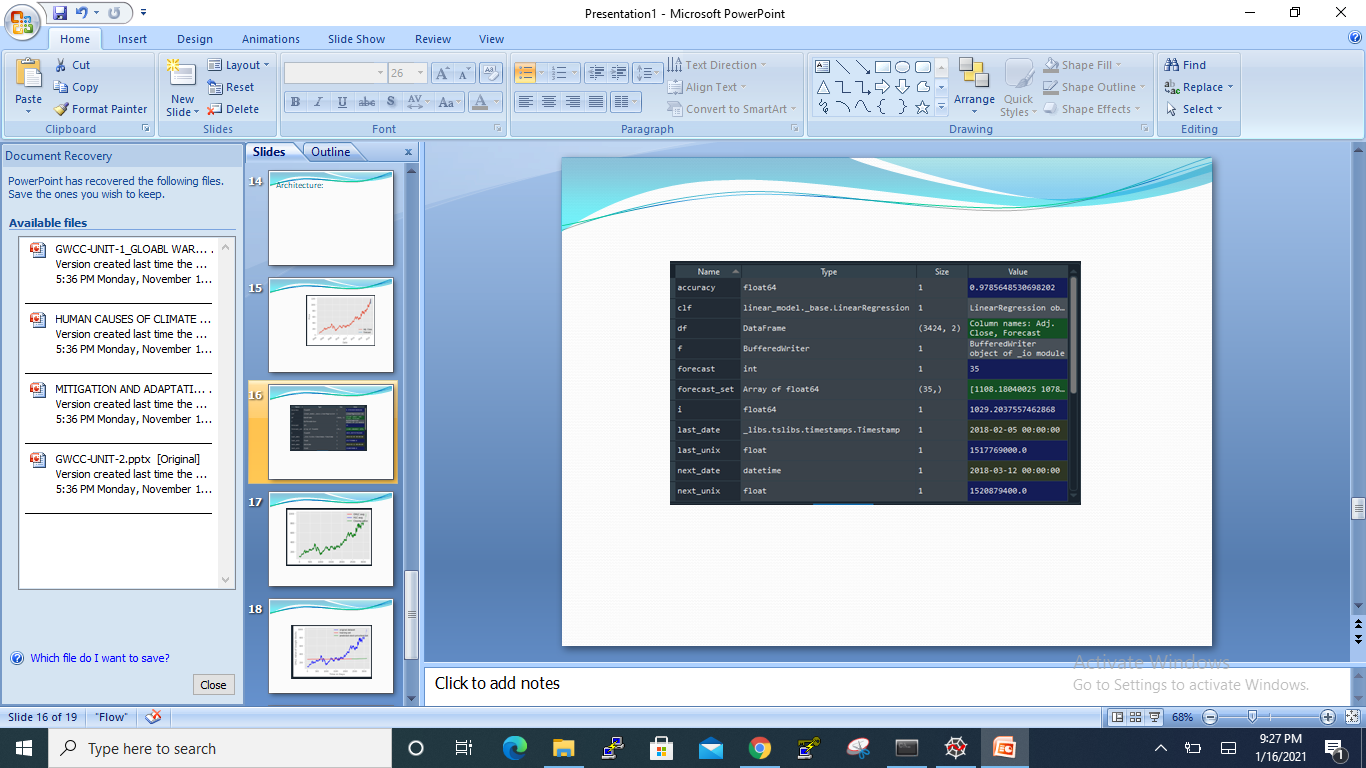
print ("Next Day's Value:", np.asscalar(last\_val\*next\_val))

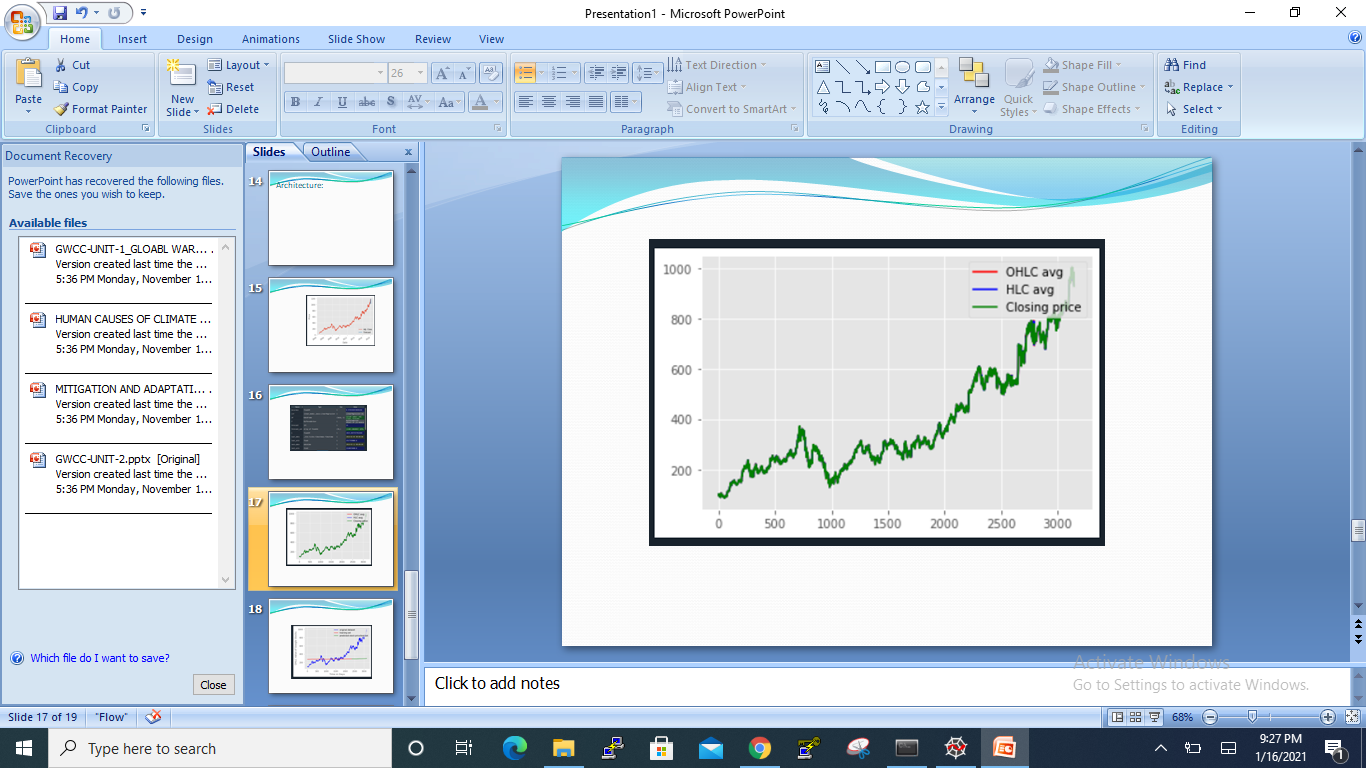
# print np.append(last\_val, next\_val)

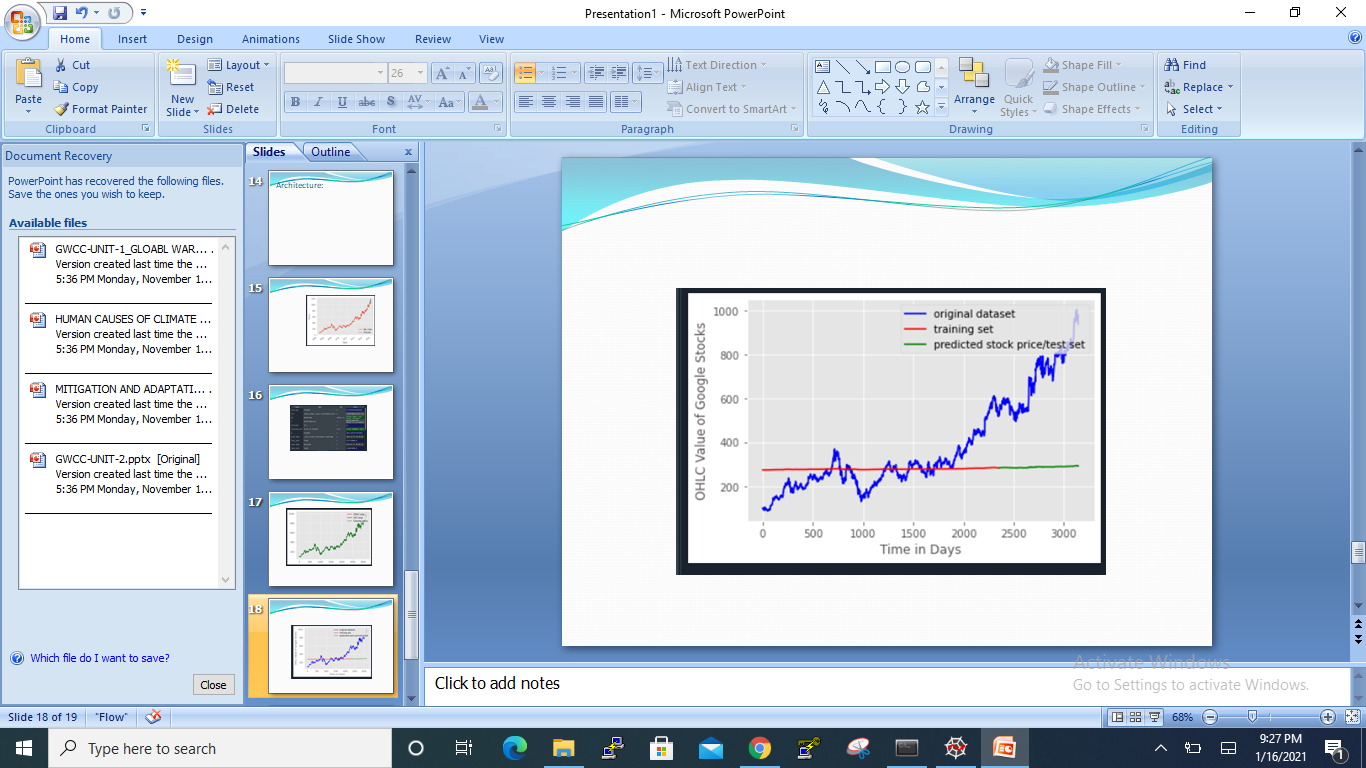
1. **Output:**

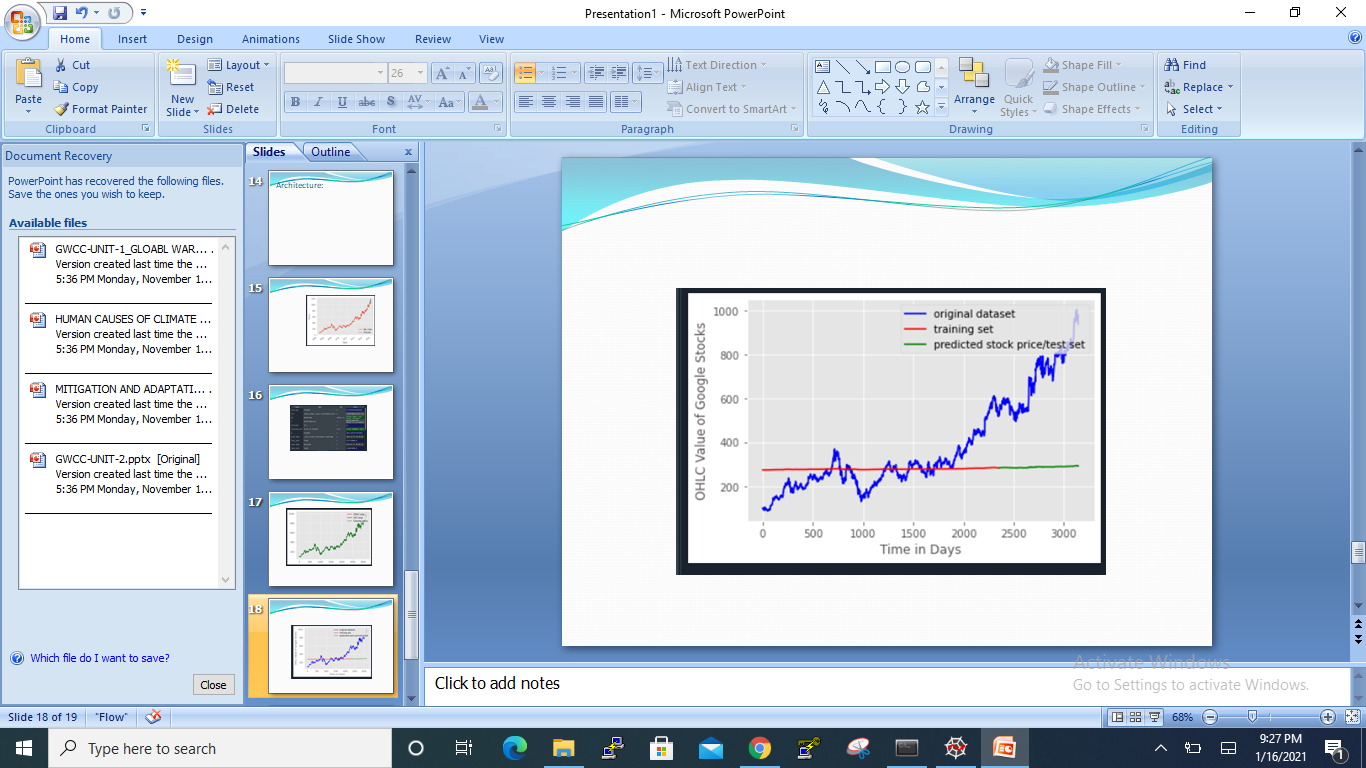
**LinearRegressionModel.py**





StockPricePrediction.py





1. **Analysis of the results:**

The Linear Regression model is giving us good results as it accuracy is around 97% but coming to the LSTM network, we are getting a very low accuracy when our training parameters for the LSTM model are set to a high value. The RMSE error value and the predictions are getting better when the training parameters are modified as seen above when epoch is modified to 10 from 5.

Both the models are giving us excellent results for such databases and attributes. But, when it comes to choosing one of the models from Linear Regression or LSTM, the LSTM seems to be the better choice as it improves on every epoch.

1. **Conclusion:**

* In conclusion the advancements in the concepts of machine learning and deep learning given us predictive models for prediction of stock prices and in other fields.
* As, we have seen in this project that LSTM networks give us better prediction values than the linear regression model, by tuning its parameters to a right degree.
* Further advancements in deep learning can give an even better models and algorithms and as a we can provide more information to traders in stock market.

1. **Future Scope:**

* The Linear Regression model is giving us good results as it accuracy is around 97% but coming to the LSTM network, we are getting a very low accuracy when our training parameters for the LSTM model are set to a high value. The RMSE error value and the predictions are getting better when the training parameters are modified as seen above when epoch is modified to 10 from 5.
* Both the models are giving us excellent results for such databases and attributes. But, when it comes to choosing one of the models from Linear Regression or LSTM, the LSTM seems to be the better choice as it improves on every epoch.

1. **REFERENCES:**

* <https://en.wikipedia.org/wiki/Stock_market_prediction>
* <https://keras.io/models/sequential/>
* <https://www.tensorflow.org/api_docs/>
* <https://scikit-learn.org/stable/>
* <https://github.com/Rajat-dhyani/Stock-price-predictor>
* [httpsandas.pydata.org/pandas-docs/stable/](https://pandas.pydata.org/pandas-docs/stable/)
* <https://colab.github.io/posts/2015-08-Understanding-LSTMs/>
* <https://en.wikipedia.org/wiki/Deep_learning>