A MINI PROJECT REPORT

On

STOCK PREDICTION

Submitted By:

Shivam Upadhyay

(171500324)

Submitted To:

Mr. Subhash Chand Agrawal

Assistant Professor, Dept. of CEA

Department of Computer Engineering & Applications

GLA University, Mathura

2019

Contents

Student Information:

Name :Shivam Upadhyay	University Roll. No.171500324
Mobile : 7078384588	Email: shivam.upadhyay_cs17@gla.ac.in

Project Information:

Title of Project/Training/Task	**STOCK PRICE PREDICTION**
Role & Responsibility	Group leader and gui developer
Technical Details	**Hardware Requirements:** I3 and above Internet connectivity 8 GB RAM 50 GB hard disk space **Software Requirements:** ANACONDA Python Jupyter notebook Github
Implementation Details	Fully Implemented

Department of Computer Engineering and Applications

GLA University, Mathura

17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,

Mathura – 281406

Declaration

I hereby declare that the work which is being presented in the Mini Project "**Stock Prediction",** in partial fulfilment of the requirements for Mini Project, is an authentic record of my own work carried under the supervision of **Mr. Subhash Chand Agrawal,** Assistant Professor, Dept. of CEA, GLA University, Mathura.

Signature of Candidate:

Shivam Upadhyay

Univ. Roll. No. : 171500324

B.Tech

Year: 3rd

Semester: 5th

Department of Computer Engineering and Applications

GLA University, Mathura

17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,

Mathura – 281406

ABSTRACT

In this report we analyse existing and new methods of stock market prediction. We take three different approaches at the problem: Fundamental analysis, Technical Analysis, and the application of Machine Learning. We find evidence in support of the weak form of the Efficient Market Hypothesis, that the historic price does not contain useful information but out of sample data may be predictive. We show that Fundamental Analysis and Machine Learning could be used to guide an investor's decisions. We demonstrate a common flaw in Technical Analysis methodology and show that it produces limited useful information. Based on our findings, algorithmic trading programs are developed and simulated using Quantopian. In this project, linear regression is used for doing the stock prediction.

Department of Computer Engineering and Applications

GLA University, Mathura

17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,

Mathura – 281406

CERTIFICATE

This is to certify that the project entitled **"Stock prediction"** carried out in Mini Project – I Lab is a bonafide work done by ***Shivam Upadhyay (171500324)*** and is submitted in partial fulfilment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Name of Supervisor: Mr. Subhash Agrawal

Name : Shivam Upadhyay

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that I have received.

My heartiest thanks to **Prof. (Dr.) Anand Singh Jalal,** Head of Dept., Department of CEA for providing me with an encouraging platform to develop this project, which thus helped me in shaping my abilities towards a constructive goal.

I owe special debt of gratitude to **Mr. Subhash agrawal,** Assistant Professor, Department of CEA, for his constant support and guidance throughout the course of my work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.

I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of my project. Last but not the least, I acknowledge my friends for their contribution in the completion of the project.

Shivam Upadhyay

1. INTRODUCTION

Predicting the Stock Market has been the bane and goal of investors since its existence. Everyday billions of dollars are traded on the exchange, and behind each dollar is an investor hoping to profit in one way or another. Entire companies rise and fall daily based on the behaviour of the market. Should an investor be able to accurately predict market movements, it offers a tantalizing promises of wealth and influence. It is no wonder then that the Stock Market and its associated challenges find their way into the public imagination every time it misbehaves. The 2008 financial crisis was no different, as evidenced by the flood of films and documentaries based on the crash. If there was a common theme among those productions, it was that few people knew how the market worked or reacted. Perhaps a better understanding of stock market prediction might help in the case of similar events in the future.

1.1 Project Goals and Scope

Despite its prevalence, Stock Market prediction remains a secretive and empirical art. Few people, if any, are willing to share what successful strategies they have. A chief goal of this project is to add to the academic understanding of stock market prediction. The hope is that with a greater understanding of how the market moves, investors will be better equipped to prevent another financial crisis. The project will evaluate some existing strategies from a rigorous scientific perspective and provide a quantitative evaluation of new strategies.

It is important here to define the scope of the project. Although vital to any investor operating in the real world, no attempt is made in this project at portfolio management. Portfolio management is largely an extra step done after an investor has made a prediction on which direction any particular stock will move. The investor may choose to allocate funds across a range of stocks in such a way to minimize his or her risk. For instance, the investor may choose not to invest all of their funds into a single company lest that company takes unexpected turn. A more common approach would be for an investor to invest across a broad range of stocks based on some criteria he has decided on before. This

project will focus exclusively on predicting the daily trend (price movement) of individual stocks. The project will make no attempt to deciding how much money to allocate to each prediction. More so, the project will analyse the accuracies of these predictions.

Additionally, a distinction must be made between the trading algorithms studied in this project and high frequency trading (HFT) algorithms. HFT algorithms make little use of intelligent prediction and instead rely on being the fastest algorithm in the market. These algorithms operate on the order of fractions of a second. The algorithms presented in this report will operate on the order of days and will attempt to be truly predictive of the market.

Analysis of stocks using data mining will be useful for new investors to invest in stock market based on the various factors considered by the software.Stock market includes daily activities like sensex calculation, exchange of shares. The exchange provides an efficient and transparent market for trading in equity, debt instruments and derivatives.
Our software will be analysing sensex based on company's stock value. The stock values of company depend on many factors, some of them are:

1) **Demand and Supply:** Demand and Supply of shares of a company is a major reason price change in stocks. When Demand Increase and Supply is less, price rises and vice versa.
2) **Corporate results:** This will be regarding to the profits or progress of the company over a span of time say 3 months.
3) **Popularity:** Main Strength in hands of share buyer. Popularity of a company can effect on buyers. Like if any good news of a company, may result in rise of stock price. And a bad news may break dreams.

The stock value depends on other factors as well, but we are taking into consideration only these main factors.

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests

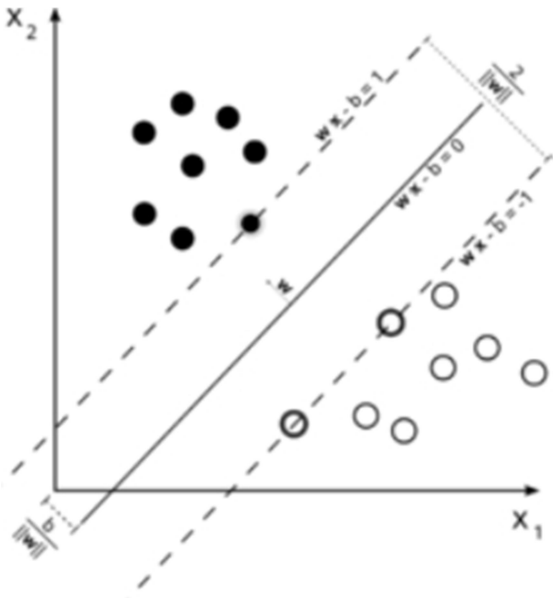
that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information.

2. Basic information

2.1 Why Machine Learning?

Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*." This definition of the tasks in which machine learning is concerned offers a fundamentally operational The name *machine learning* was coined in 1959 by Arthur Samuel definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?" In Turing's proposal the various characteristics that could be possessed by a *thinking machine* and the various implications in constructing one are exposed.

Machine learning tasks



A support vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white.

Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model from a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback. Semi-supervised learning algorithms develop mathematical models from incomplete training data, where a portion of the sample input doesn't have labels.

Classification algorithms and regression algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "spam" or "not spam", represented by the Boolean values true and false. Regression algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object.

In unsupervised learning, the algorithm builds a mathematical model from a set of data which contains only inputs and no desired output labels. Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as in feature learning. Dimensionality reduction is the process of reducing the number of "features", or inputs, in a set of data.

2.2 Knowing Your Task

Your Data Quite possibly the most important part in the machine learning process is understanding the data you are working with and how it relates to the task you want to solve. It will not be effective to randomly choose an algorithm and throw your data at it. It is necessary to understand what is going on in your dataset before you begin building a model. Each algorithm is different in terms of what kind of data and what problem setting it works best for. While you are building a machine learning solution, you should answer, or at least keep in mind, the following questions:

2.3 Why Python?

Python has become the lingua franca for many data science applications. It combines the power of general-purpose programming languages with the ease of use of domain-specific scripting languages like MATLAB or R. Python has libraries for data loading, visualization, statistics, natural language processing, image processing, and more. This vast toolbox provides data scientists with a large array of general- and special-purpose functionality. One of the main advantages of using Python is the ability to interact directly with the code, using a terminal or other tools like the Jupyter Notebook, which we'll look at shortly. Machine learning and data analysis are fundamentally iterative processes, in which the data drives the analysis. It is essential for these processes to have tools that allow quick iteration and easy interaction. As a general-purpose programming language, Python also allows for the creation of complex graphical user interfaces (GUIs) and web services, and for integration into existing systems.

- Python is an interpreted high-level programming language. It has advantages of both scripting and programming languages.
- It is very useful for rapid application development
- It is easy to learn when comparing with other programming languages and has a design philosophy that emphasizes code readability, mainly using indentation.
- The language was originally created by Guido van Rossum and first released on 1991.
- Python programming language has a wide range of applications from Web Development, scientific and mathematical computing.
- Nowadays, Python is gaining more attention as it's great for data analysis, artificial intelligence and scientific computing
- It does not need a compiler to run the application. It's basically an interpreter language.

Few IDE'S for Python Development

- IDLE (Integrated Development Environment for Python)

This is the default IDE available with Python installation. In this tutorial, I am running my example programs only on IDLE. There are other IDEs as well which have lots of features. I will give details about those IDEs in upcoming parts.

- Visual Studio
- PyCharm
- Anaconda Navigator
- Tkinter

2.4 NumPy

NumPy is one of the fundamental packages for scientific computing in Python. It contains functionality for multidimensional arrays, high-level mathematical functions such as linear algebra operations and the Fourier transform, and pseudorandom number generators. In scikit-learn, the NumPy array is the fundamental data structure. scikit-learn takes in data in the form of NumPy arrays. Any data you're using will have to be converted to a NumPy array. The core functionality of NumPy is the ndarray class, a multidimensional (n-dimensional) array. All elements of the array must be of the same type. A NumPy array looks like this:

In:-

```
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]]) print("x:\n{}".format(x))
```

Out:-

```
x:

[[1 2 3]

 [4 5 6]]
```

We will be using NumPy a lot in this book, and we will refer to objects of the NumPy ndarray class as "NumPy arrays" or just "arrays."

2.5 matplotlib

matplotlib is the primary scientific plotting library in Python. It provides functions for making publication-quality visualizations such as line charts, histograms, scatter plots, and so

on. Visualizing your data and different aspects of your analysis can give you important insights, and we will be using matplotlib for all our visualizations. When working inside the Jupyter Notebook, you can show figures directly in the browser by using the %matplotlib notebook and %matplotlib inline commands. We recommend using %matplotlib notebook, which provides an interactive environment (though we are using %matplotlib inline to produce this book). For example, this code produces the plot in Figure 1-1:

In:-

```
%matplotlib inline

import matplotlib.pyplot as plt

# Generate a sequence of numbers from -10 to 10 with 100 steps in between

x = np.linspace(-10, 10, 100)

# Create a second array using sine

y = np.sin(x)

# The plot function makes a line chart of one array against another

plt.plot(x, y, marker="x")
```

2.6 pandas

pandas is a Python library for data wrangling and analysis. It is built around a data structure called the DataFrame that is modeled after the R DataFrame. Simply put, a pandas DataFrame is a table, similar to an Excel spreadsheet. pandas provides a great range of methods to modify and operate on this table; in particular, it allows SQL-like queries and joins of tables. In contrast to NumPy, which requires that all entries in an array be of the same type, pandas allows each column to have a separate type (for example, integers, dates, floating-point numbers, and strings). Another valuable tool provided by pandas is its ability to ingest from a great variety of file formats and databases, like SQL, Excel files, and comma-separated values (CSV) files. Going into detail about the functionality of pandas is out of the scope of this book. However, Python for Data Analysis by Wes McKinney

(O'Reilly, 2012) provides a great guide. Here is a small example of creating a DataFrame using a dictionary:

In:-

```
import pandas as pd

# create a simple dataset of people

 data = {'Name': ["John", "Anna", "Peter", "Linda"],        'Location' : ["New York", "Paris", "Berlin", "London"],        'Age' : [24, 13, 53, 33]       }

data_pandas = pd.DataFrame(data)

# IPython.display allows "pretty printing" of dataframes

# in the Jupyter notebook display(data_pandas)
```

Linear regression

Linear regression, or ordinary least squares (OLS), is the simplest and most classic linear method for regression. Linear regression finds the parameters w and b that minimize the mean squared error between predictions and the true regression targets, y, on the training set. The mean squared error is the sum of the squared differences

between the predictions and the true values. Linear regression has no parameters,

which is a benefit, but it also has no way to control model complexity.

```
from sklearn.linear_model import LinearRegression

X, y = mglearn.datasets.make_wave(n_samples=60)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

lr = LinearRegression().fit(X_train, y_train)
```

```
print("lr.coef_: {}".format(lr.coef_))

print("lr.intercept_: {}".format(lr.intercept_))
```

Out:-

```
lr.coef_: [ 0.394]

lr.intercept_: -0.031804343026759746
```

Logistic Regression

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

x = np.arange(-10, 10, 0.01)

sig = 1/(1 + np.power(np.e,-x))

sig_1 = np.power(np.e,-x)/(1 + np.power(np.e,-x))

line = 3 * x + 7

plt.plot(x,sig)

plt.show()

plt.plot(x, sig_1)

plt.show()

plt.plot(x, line)

plt.show()

sig_line = 1/( 1 + np.power(np.e, line) )

plt.plot(x, sig_line)

plt.show()
```

3. Introduction to project

3.1 Statement of the problem

Financial analysts investing in stock market usually are not aware of the stock market behaviour. They are facing the problem of trading as they do not properly understand which stocks to buy or which stocks to sell in order to more profits. In today's world, all the information pertaining to stock market is available. Analysing all this information individually or manually is tremendously difficult. As such, automation of the process is required. This is here data mining techniques help.

3.2 Advantage

The research helps a lot of new investors in deciding when to buy or sell a particular stock. It also helps in understanding the sentiments of experienced financial analysts and financial news data more quickly than doing the same manually.

3.3 Snapshots

- We removed the unnecessary columns and will take only the required columns in fig(1).

```
In [6]: 1 import matplotlib.pyplot as plt

In [7]: 1 f.columns

Out[7]: Index(['adjClose', 'adjHigh', 'adjLow', 'adjOpen', 'adjVolume', 'close',
               'divCash', 'high', 'low', 'open', 'splitFactor', 'volume'],
              dtype='object')

In [8]: 1 f.reset_index(inplace=True)
        2 f.set_index('date',inplace=True)
        3 f=f[['adjClose', 'adjHigh', 'adjLow', 'adjOpen', 'adjVolume',]]
```

Fig(1)

- We than made two different data sets x and y having different respective values as a fig(2).

```
In [15]: 1 x=f.drop(['adjClose','newclose'],axis=1)
         2 y=f['newclose'].dropna()
```

```
In [16]: 1 f.head()
```

Out[16]:

date	adjClose	adjHigh	adjLow	adjOpen	adjVolume	newclose
2010-01-04	24.529440	24.648322	24.244122	24.267898	38409100	22.555989
2010-01-05	24.537365	24.648322	24.283749	24.450185	49749600	22.690723
2010-01-06	24.386781	24.632471	24.188643	24.473961	58182400	22.064608
2010-01-07	24.134750	24.331302	23.927101	24.275824	50559700	22.207267
2010-01-08	24.299600	24.473961	23.966729	23.998431	51197400	21.969502

Fig(2)

- We than made two new data set x1 which has data in which last no_days =20 is as NAN as we will predict the stock price of next 20 days in fig(3).
- We will split this data set into 80% train,and 20% test. The model will be trained using the train set in fig(3).

```
In [17]: 1 x1=x[:-no_days]
         2 #doing negative shifting.....removing below 20 data and putting NaN
         3 x_pr=x[-no_days:]
         4 #above line is for predicting
```

```
In [22]:
1 from sklearn.model_selection import train_test_split
2 x_tr,x_ts,y_tr,y_ts=train_test_split(x1,y,test_size=0.2)
3 #prefer to give test_size as in above line
4 #0.2 is 20%
```

fig(3)

As we get the data so now applying Linear regression into the made data set in fig(4)

```
In [25]:
1 alg=LinearRegression()
2 alg.fit(x_tr,y_tr)
3
4 alg.score(x_ts,y_ts)
5 #score is predicting that the random values are correct or not

Out[25]: 0.9917925583321744
```

Fig(4)

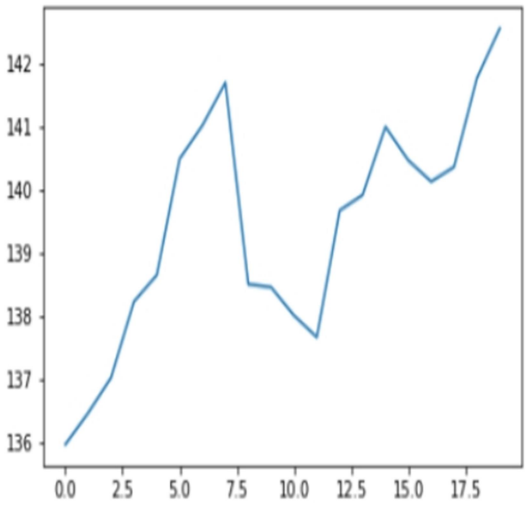
- This data is 99% accurate, it's means that it will be predicting 99% right predictions. Now for calculating the score of the respective no days =20 in fig(5).

```
In [26]: 1 from sklearn.metrics import r2_score

In [27]: 1 r2_score(y_ts,alg.predict(x_ts))

Out[27]: 0.9917925583321744

In [28]: 1 alg.predict(x_pr)

Out[28]: array([135.95434172, 136.45554152, 137.01546129, 138.2212379 ,
                138.64239702, 140.4870041 , 141.0195361 , 141.69695504,
                138.49810758, 138.44974767, 138.00130175, 137.65607628,
                139.66783466, 139.91253829, 140.99565175, 140.46236508,
                140.12631208, 140.35770204, 141.76585181, 142.5571461 ])
```

fig(5)

- The graph of the predicted score of the 20 days seem as a fig(6)

```
In [29]: 1 plt.plot(alg.predict(x_pr))
         2 plt.show()
```



fig(6)

- For making the graph of the new values that's predicted values and old values that's past values in fig(7).

In [35]:
```
import numpy as np
#making new column
f['forecast']=np.nan
```

In [36]:
```
prd=alg.predict(x_pr)
lastday=f.iloc[-1].name
for i in prd:
    lastday+=datetime.timedelta(1)
    f.loc[lastday]=[np.nan for _ in range(6)]+[i]
    #in loop we r using 6 to give NaN value in 6 columns and to put i value in column forecast
```

In [37]:
```
f['adjClose'].plot()
f['forecast'].plot()
plt.show()
```

fig(7)

- The end point in the graph tells the prediction of the next 20 days as a fig(8).

```
In [38]:  1 f.tail()
```

Out[38]:

date	adjClose	adjHigh	adjLow	adjOpen	adjVolume	newclose	forecast
2019-07-27	NaN	NaN	NaN	NaN	NaN	NaN	140.462365
2019-07-28	NaN	NaN	NaN	NaN	NaN	NaN	140.126312
2019-07-29	NaN	NaN	NaN	NaN	NaN	NaN	140.357702
2019-07-30	NaN	NaN	NaN	NaN	NaN	NaN	141.765852
2019-07-31	NaN	NaN	NaN	NaN	NaN	NaN	142.557146

fig(8)

CODE IS BELOW:-

```
import numpy as np

from sklearn.linear_model import LinearRegression

from pandas_datareader import data as dt

import datetime

import pandas as pd

from sklearn.model_selection import train_test_split

import datetime

def fun(stock,test,days,lab,message):

    global f

    start = datetime.datetime(2010, 1, 1)
```

```
    end = datetime.datetime.now()

    f = dt.DataReader(stock, 'tiingo', start,
end,access_key='911ee28d70118f9cea5a84d2b8f1436fa32d3116')

    f.reset_index(inplace=True)

    f.set_index('date',inplace=True)

    f=f[['adjClose', 'adjHigh', 'adjLow', 'adjOpen', 'adjVolume',]]

    no_days=int(days)

    f['newclose']=f['adjClose'].shift(-no_days)

    x=f.drop(['adjClose','newclose'],axis=1)

    y=f['newclose'].dropna()

    x1=x[:-no_days]

    x_pr=x[-no_days:]

    x_tr,x_ts,y_tr,y_ts=train_test_split(x1,y,test_size=float(test))

    alg=LinearRegression()

    alg.fit(x_tr,y_tr)

    lab.config(text=str(alg.score(x_ts,y_ts)))

    prd=alg.predict(x_pr)

    message.config(text=str(prd))

    lastday=f.iloc[-1].name

    f['forecast']=np.nan
```

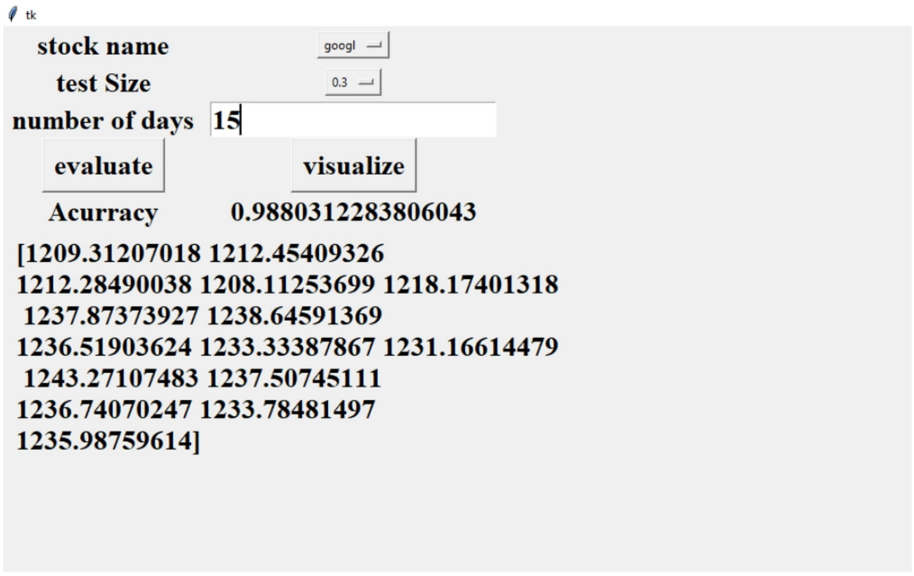
```
    for i in prd:
        lastday+=datetime.timedelta(1)
        f.loc[lastday]=[np.nan for _ in range(6)]+[i]
def vis():
    %matplotlib tk
    f['adjClose'].plot()
    f['forecast'].plot()
    plt.show()
from tkinter import *
def gui():
    scr=Tk()
    label=Label(scr,font=('times',20,'bold'),text='stock name')
    label.grid(row=0,column=0)
    stock=StringVar()
    op=OptionMenu(scr,stock,'tcs','aple','googl','msft')
    op.grid(row=0,column=1)
    label1=Label(scr,font=('times',20,'bold'),text='test Size')
    label1.grid(row=1,column=0)
    test=DoubleVar()
    op1=OptionMenu(scr,test,0.1,0.2,0.3)
```

```
    op1.grid(row=1,column=1)
    label2=Label(scr,font=('times',20,'bold'),text='number of days')
    label2.grid(row=2,column=0)
    e=Entry(scr,font=('times',20,'bold'))
    e.grid(row=2,column=1)
    b=Button(scr,text='evaluate',font=('times',20,'bold'),command=lambda
:fun(stock.get(),test.get(),e.get(),label4,m))
    b.grid(row=3,column=0)
    b1=Button(scr,text='visualize',font=('times',20,'bold'),command=vis)
    b1.grid(row=3,column=1)
    label3=Label(scr,font=('times',20,'bold'),text='Acurracy')
    label3.grid(row=4,column=0)
    label4=Label(scr,font=('times',20,'bold'))
    label4.grid(row=4,column=1)
    m=Message(scr,font=('times',20,'bold'))
    m.grid(row=5,columnspan=7)
    scr.mainloop()
gui()
```

- After it we had made GUI through which it becomes very easy to calculate the value s of next desired days and getting the graph also in fig(9).

fig(9)

Fig 10 shows Stock prediction GUI for stock prediction of Google company for next 15 days



fig(10)

5. Future Enhancement and Limitation

• The limitation of the proposed system is its computational speed, especially with respect to sliding-window validation as the computational cost increases with the number of forward day predictions.

• The proposed model does not predict well for sudden changes in the trend of stock data.

• This occurs due to external factors and real-world changes affecting the stock market.

• We can overcome this by implementing Sentiment Analysis and Neural Networks to enhance the proposed model.

• We can modify the same system to an online-learning system that adapts in real-time. Stock Market Prediction

6. Conclusion

Thus, as we can see above in our proposed method, we train the data using existing stock dataset that is available. We use this data to predict and forecast the stock price of n-days into the future. The average performance of the model decreases with increase in number of days, due to unpredictable changes in trend. The current system can update its training set as each day passes so as to detect newer trends and behave like an online-learning system that predicts stock in real-time. Stock Market Prediction