



# Stock Market Data Analysis and Stock Price Prediction

A Comparative Analysis using Neural Networks

**Submitted to:**

Dr. Somesh K. Mathur  
Professor, IIT Kanpur  
[skmathur@iitk.ac.in](mailto:skmathur@iitk.ac.in)

**Submitted by:**

Abhishek Yadav	(190043)	(J3)
Deepalok Kaushik	(190261)	(J3)
Neeraj Pratap Singh	(190542)	(J2)
Nitin Patil	(190572)	(J3)
<b><u>Group(19) Leader - Prajwal Arya</u></b>	<b>(190613)</b>	<b>(J4)</b>
Rishabh Arora	(190698)	(J1)
Rishabh Chaudhary	(190700)	(J2)
Rohan Kumar	(190715)	(J2)
Rohit Sharma	(190719)	(J1)
Sandipan Mitra	(190755)	(J4)

# Abstract

- This paper will present a model based on **Artificial Neural Networks** to successfully predict the stock prices and compare it with the actual prices of a particular firm.
- We are using **multi-dimensional data** of five companies listed on the **National Stock Exchange (NSE) of India** from various sectors for the last six months.
- To train the network faster, we reduce the multi-dimensionality of the data by using **Principal Component Analysis (PCA)**.

# Introduction

- 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.
- ML models can be used to make predictions. We can also use such models to make stock market predictions.
- Traditionally, there are two approaches to predict the prices of stocks: 1)Technical analysis 2)Qualitative analysis
- In our Paper, We are using Technical Analysis, as it is based upon the use of statistics like data sets of different companies etc.

# Objective

- The basic of stock market prediction is to predict the future stock using the market statistics of the past few years.
- In our research work, we will use the Artificial Neural Network (ANN), as it is the one of the most powerful tool to predict and analyze data.
- An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons
- It is perfect for finding out the unknown relation among different data variables and recognizing complex patterns.

# Literature Review

- We use the National Stock Exchange, India, as our data source and select five companies from different categories and collect their last six months' data.
- Principal Component Analysis (PCA) is used to reduce the data dimension because a large dataset needs more time to train in a neural network. Then we implement ANN to prepare the data set.
- After specifying these experimental parameters, we implement the Backpropagation.
- We can test our network using existing data and how well it can predict using different plotting and using various diagrams.
- We calculate the error rate and how much data can be predicted in proximity to the original data.

# Hypothesis

**Random Walk Hypothesis** says that in financial economics:

- We cannot predict stock market prices since it evolves according to a random walk.
- The past movement or trend of a stock price or market cannot be used to predict its future movement.

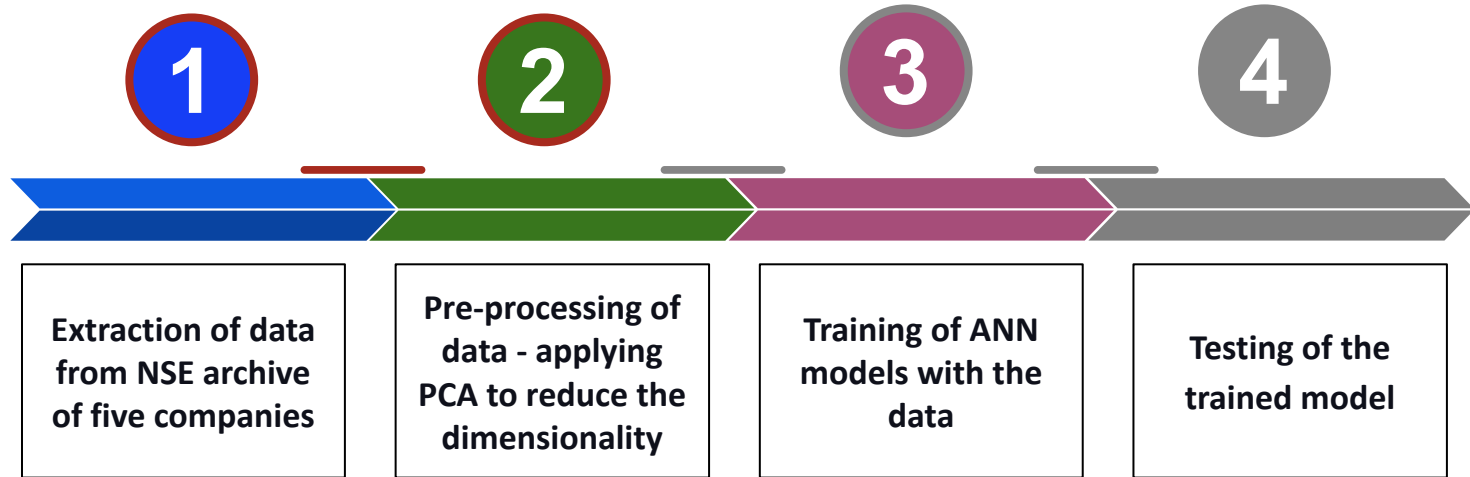
On the other hand, the **Efficient-Market Hypothesis (EMH)** says that:

- Asset prices reflect all available information about the stock
- Stock prices fully reflect all available information and expectations, so current prices are the best approximation of a company's intrinsic value.

In our paper, we formally took in consideration both the above mentioned hypothesis as:

- **Null hypothesis( $H_0$ )** which says, “the past asset price of the stock does not affect the present asset price.”
- **Alternate hypothesis( $H_1$ )** which says “the past asset price of the stock affects the present asset price.”

# Method And Methodology



# Data extraction from NSE

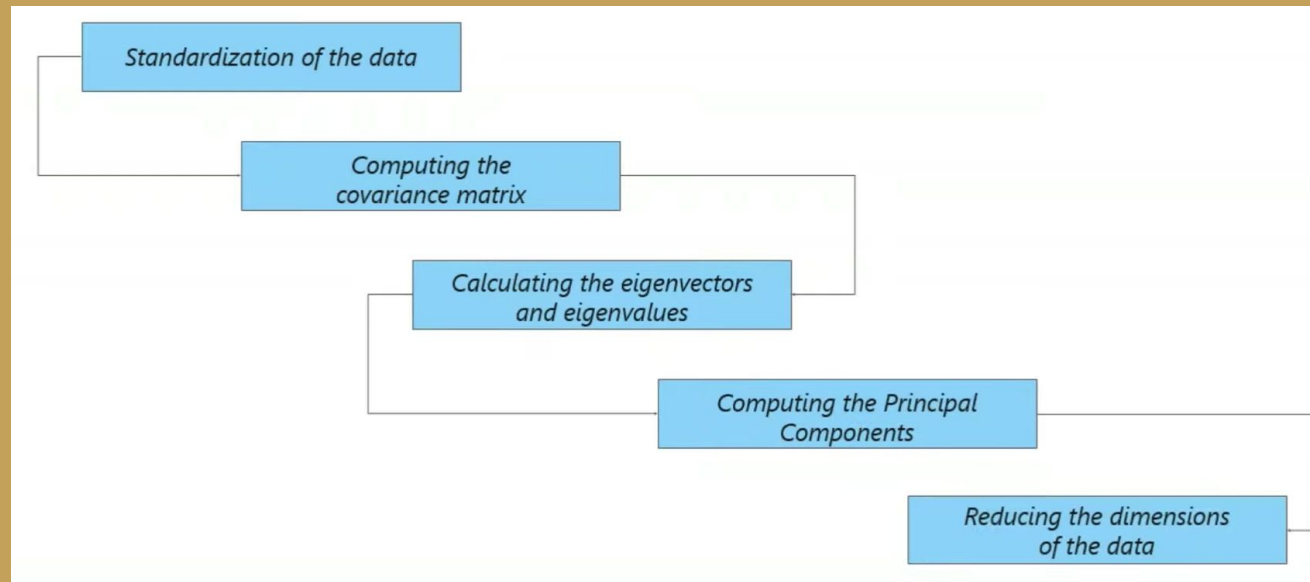
- Data is extracted from Prowess dx of Centre for Monitoring Indian economies (CMIE).
- We include the Stocks Trading data of NSE of current Sensex companies
- We have included the daily stocks trading for past 6 months
- Data set includes several variables like *opening price, high price, trade quantity etc.*
- Data Sheet:

[https://docs.google.com/spreadsheets/d/1M5\\_W7K4tut\\_v0w2Zzu3guWj1iDeocXMJ1bwCs340kNM/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1M5_W7K4tut_v0w2Zzu3guWj1iDeocXMJ1bwCs340kNM/edit?usp=sharing)



# Principal Component Analysis

- Principal Component Analysis (PCA) is a dimensionality reduction method that is used to minimize the dimension of large datasets.
- This is done by transforming several variables into a smaller one which yet contains a lot of facts in the large set.



# Code of PCA

```
#IMPORTING LIBRARIES
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
#LOADING DATA
```

```
data = pd.read_csv('Axis Bank.csv')
df = pd.DataFrame(data)
df
```

```
#STANDARDIZING AND FITTING DATA
```

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
scalar.fit(data)
scaled_data = scalar.transform(data)
```

```
#APPLYING PRINCIPAL COMPONENT ANALYSIS
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(scaled_data)
data_pca = pca.transform(scaled_data)
print(data_pca)
data_pca.shape
```

```
#PLOTING THE GRAPH OF PRINCIPAL COMPONENTS 1 AND 2
```

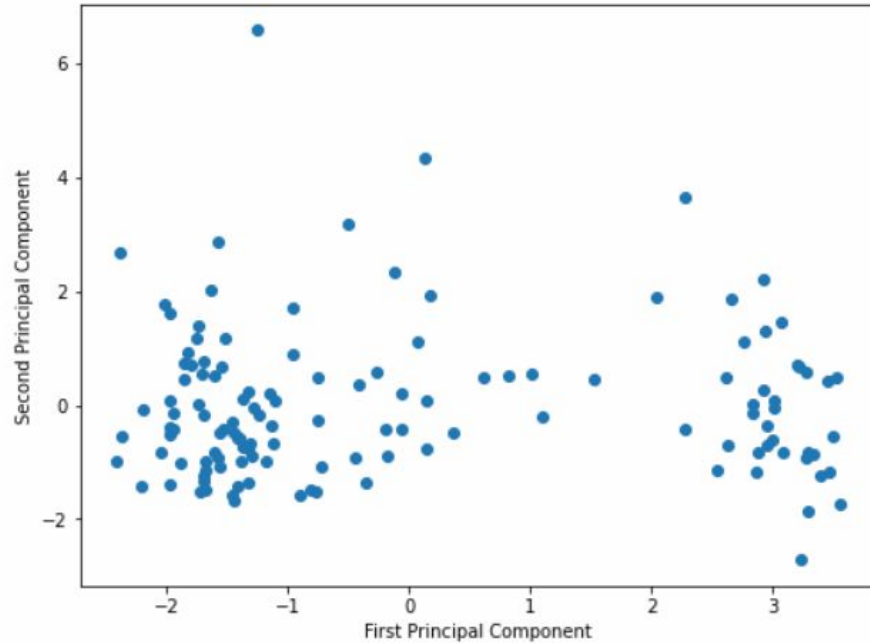
```
plt.figure(figsize =(8, 6))
plt.scatter(data_pca[:, 0], data_pca[:, 1],cmap ='plasma')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
```

```
#PRINCIPAL COMPONENTS 1 AND 2
```

```
pca.components_.T
```

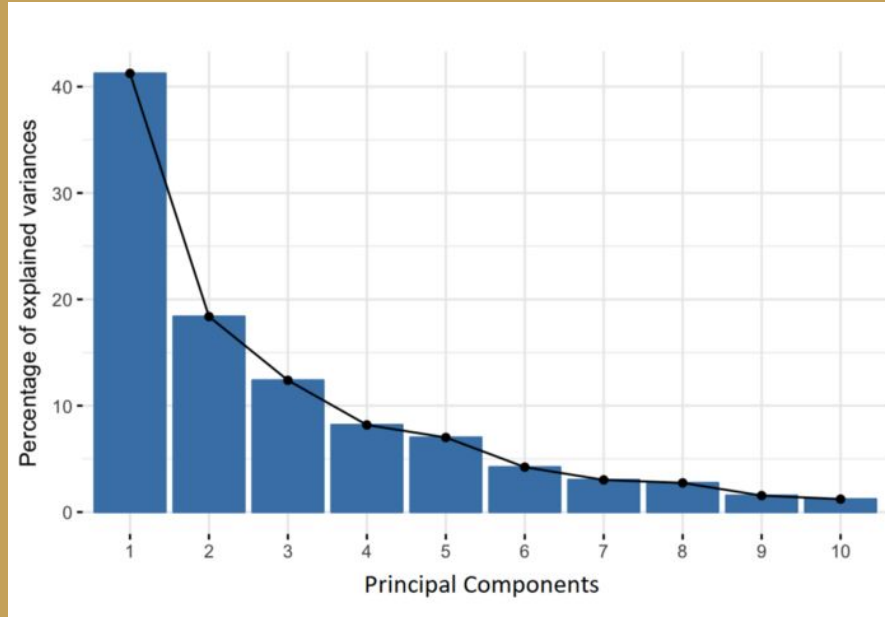
# Graph Of PCA

```
Text(0, 0.5, 'Second Principal Component')
```



```
array([[ 0.         ,  0.         ],  
       [ 0.49622191,  0.02631746],  
       [ 0.49586778,  0.05184345],  
       [ 0.497116   ,  0.01237118],  
       [ 0.49682003,  0.030788   ],  
       [-0.11480085,  0.69478453],  
       [ 0.0272995  ,  0.71609577],  
       [ 0.         ,  0.         ]])
```

# Dimension reduction using PCA

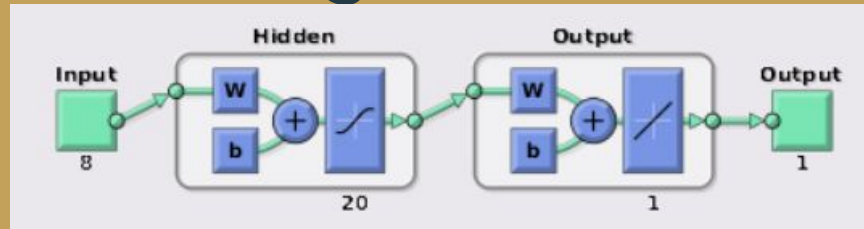


$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

# Training of the Model

- After PCA, we get reduced data which is used as the input of ANN.
- The available data is divided into two sets: training set, and testing set.
- Then we specify the parameters of our Neural Network.
- After specification of the parameters, we move to implementation of our Neural Network.
- Initially, we assign random values to the **hyperparameters**(like weights and the biases).
- Then, in the **feed forward step**, input patterns are propagated through the network one by one, and actual outputs are calculated.
- Then, comparing actual and target outputs, gives us the error.
- Weight updates take place through the **back propagation** to minimize the error in the cost function. Thus,  
**$$\text{Updated Weight} = \text{weight}(\text{old}) + [(\text{learning rate}) * (\text{output error}) * \text{output}(\text{neurons } i) * \text{output}(\text{neurons } i+1) * (1 - \text{Output}(\text{neurons } i+1))]$$**
- When all the data is passed through the model and hyper parameters are set accordingly, this is called an **epoch**. Multiple epochs are performed to maximize the efficiency of our model.

# Training of the Model



## Algorithms

Data Division: Random (dividerand)  
Training: Bayesian Regularization (trainbr)  
Performance: Mean Squared Error (mse)  
Calculations: MEX

## Progress

Epoch:	0	232 iterations	1000
Time:		0:00:02	
Performance:	2.82e+04	12.3	0.00
Gradient:	1.13e+05	1.98	1.00e-07
Mu:	0.00500	5.00e+10	1.00e+10
Effective # Param:	161	13.7	0.00
Sum Squared Param:	153	3.08	0.00

## Plots

Performance	(plotperform)
Training State	(plottrainstate)
Error Histogram	(ploterrhist)
Regression	(plotregression)
Fit	(plotfit)

Plot Interval: 50 epochs

# Code Of ANN

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created 17-Apr-2021 08:07:08
%
% This script assumes these variables are defined:
%
%   data - input data.
%   data_1 - target data.

x = data';
t = data_1';

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainbr'; % Bayesian Regularization backpropagation.

% Create a Fitting Network
hiddenLayerSize = 20;
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
```

```
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y);

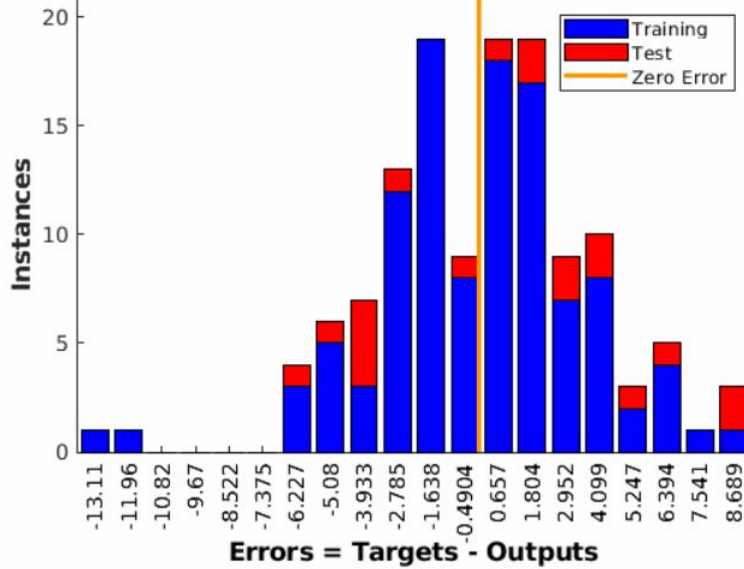
% View the Network
view(net);

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

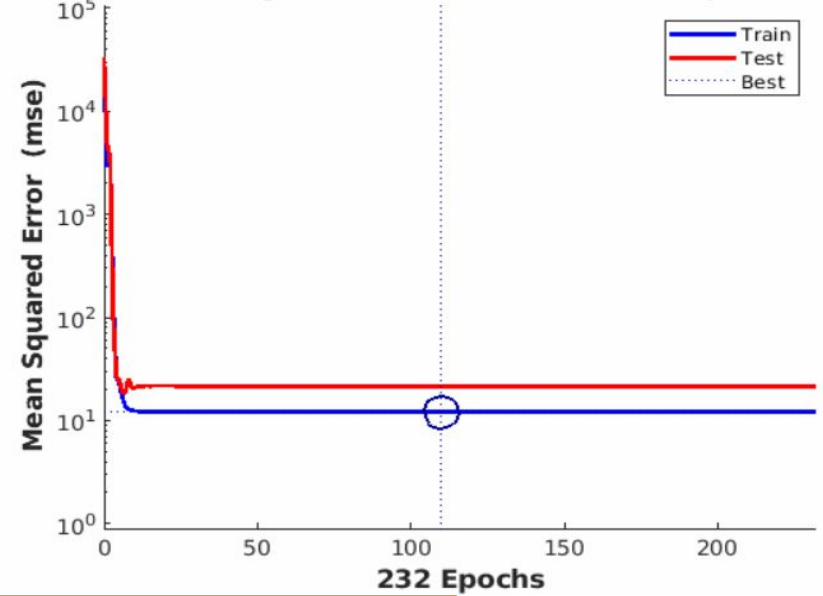


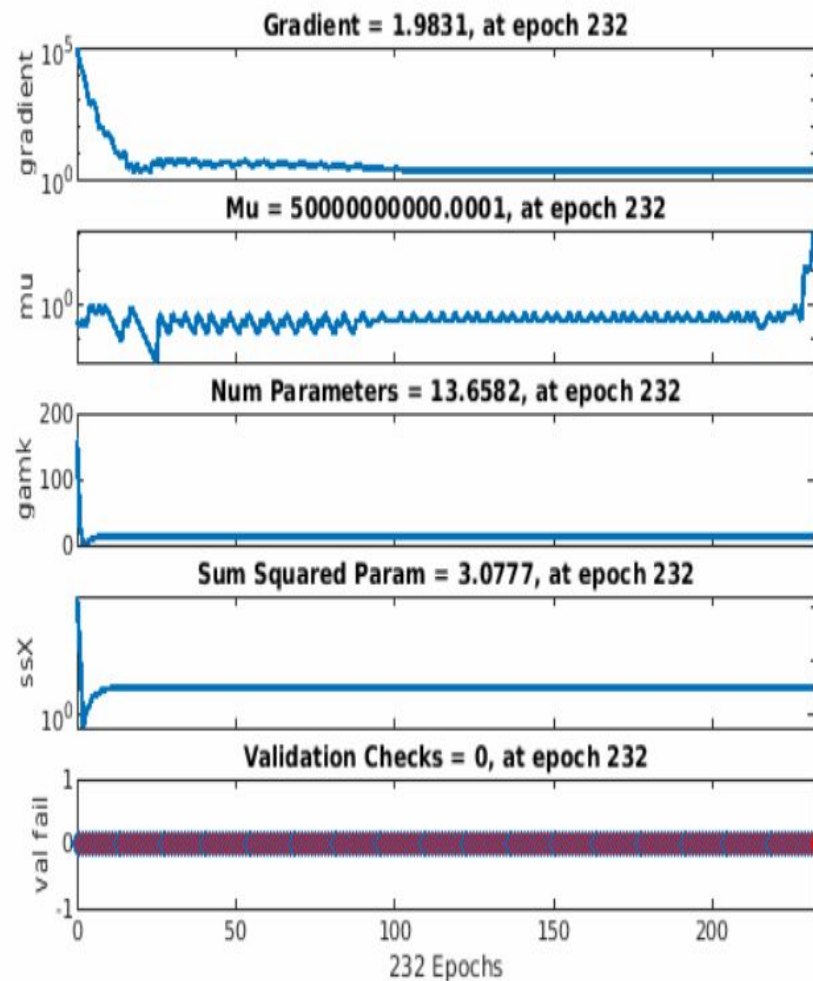
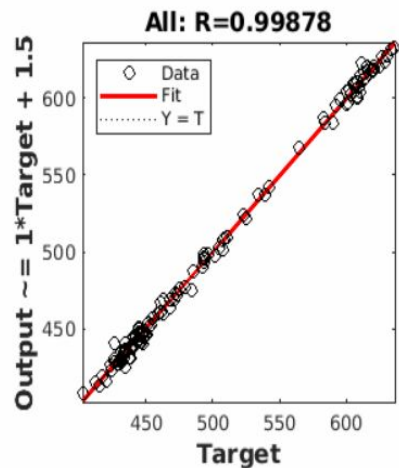
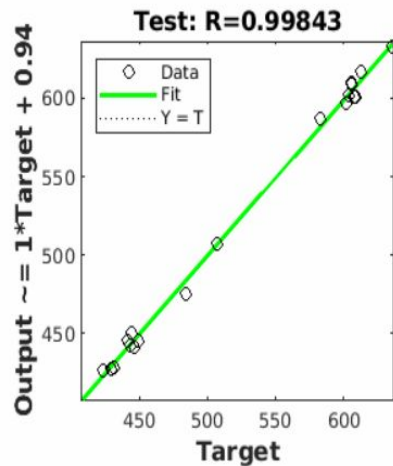
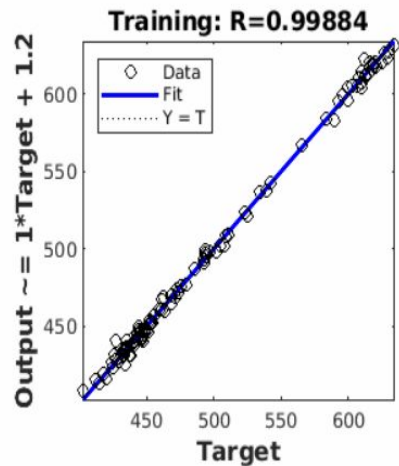
# Graph of Training

Error Histogram with 20 Bins




Best Training Performance is 12.2701 at epoch 110



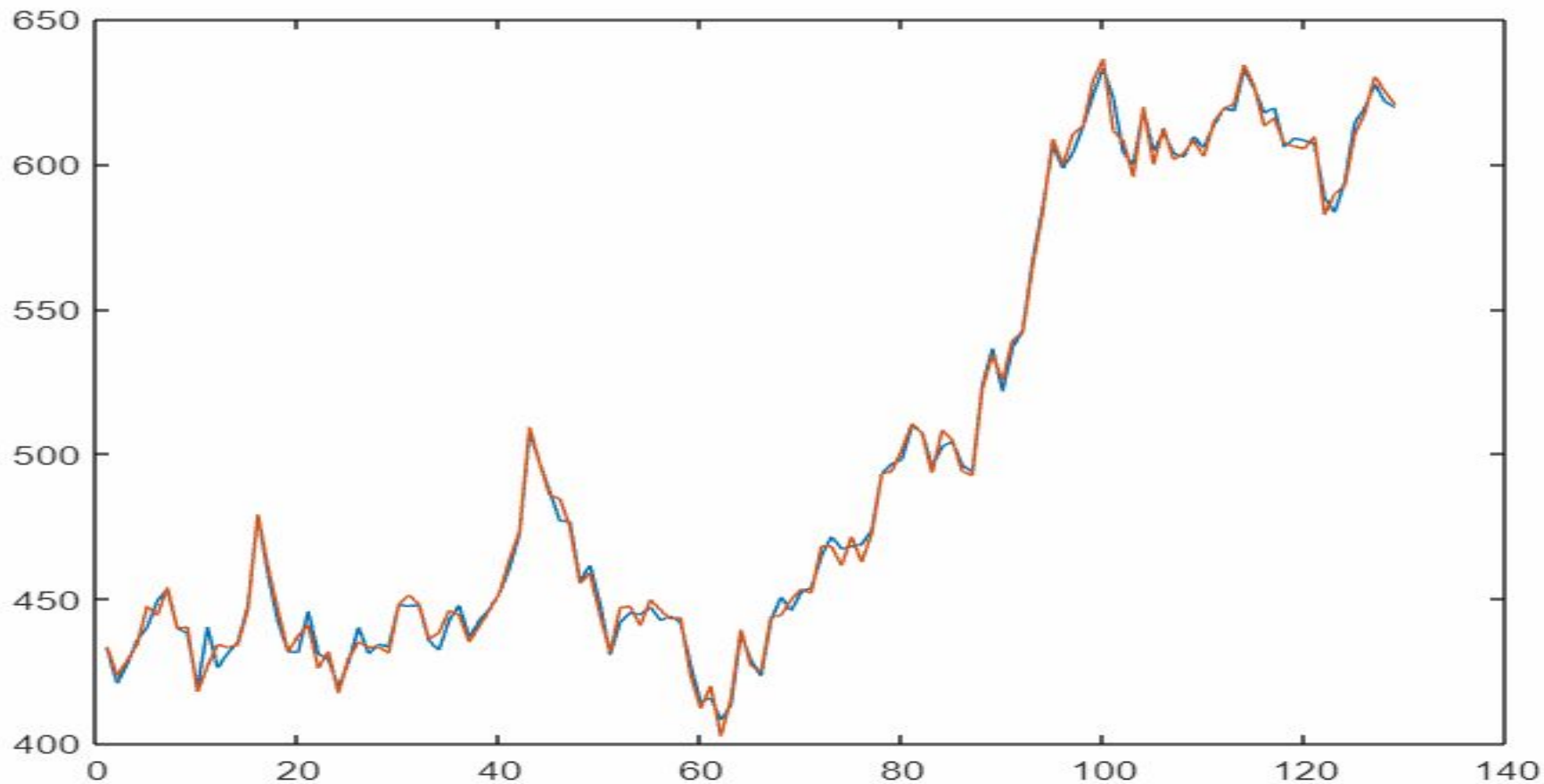




# Testing

- Earlier, we have divided the data into two parts, the second part would now be used for testing of our model.
  - The resulting data is then used for plotting the graph against the actual data.
  - Then **error** is calculated.
  - A number of epochs are performed to reduce this error.
  - Also, while choosing the epoch we should try to avoid **overfitting**.
- 

# Output - Target Plot



# Tentative conclusion

- In our suggested system, we will develop a model to anticipate a particular firm's stock rate by training their preceding data in a neural network.
- First, we use Principal Component Analysis (PCA) to reduce the data dimension, which helps train our system faster. We can get the most influential features of data after the reduction of the data dimension.
- Artificial Neural Network, which is one of the finest neural network methods, will be used, which can reduce the error between the actual output and the desired output by using gradient descent.
- Since it will be quite strenuous to predict with 100% accuracy, the performance is not expected to be satisfactory every time.