Stock Market Prediction using Hybrid Machine Learning System

A PROJECT REPORT

Submitted by

SHRUSHTI GOVARDHAN [Reg No: RA1611008010099]

SHREYAS KULKARNI [Reg No: RA1611008010186]

PRATIK DHARAMDASANI [Reg No: RA1611008010298]
ASTHA MAHESH KUMAR [Reg No: RA1611008010752]

Under the Guidance of

Mr. S Deepan
(Assistant Professor, Department of Information Technology)

In partial fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603203

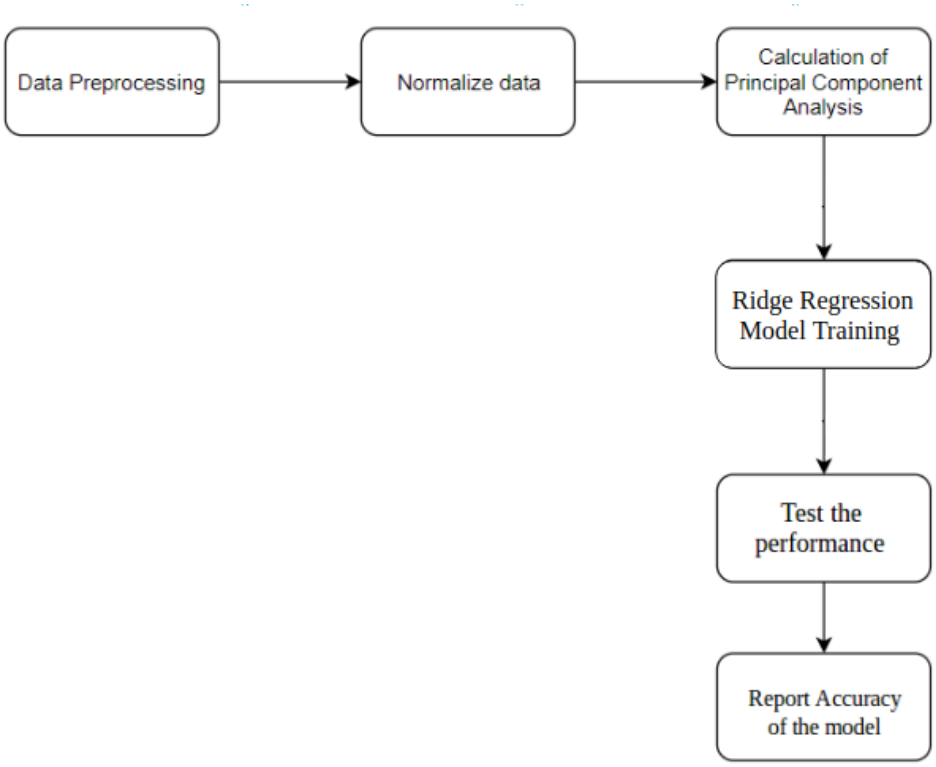
MAY 2020

CHAPTER 3

PROPOSED WORK

3.1 Proposed System Architecture

The proposed system makes use of principal component analysis (PCA) for obtaining the most relevant features and regression technique for predicting the value of NIFTY of National Stock Exchange (NSE) market of India. The system consists of the following modules:



3.2 System Modules

The system has been divided into modules in order to make the implementation easy and trackable. The modularization of the project has also made it easier for task division and possible for future enhancements. The proposed system consists the following modules:

1. Data Preprocessing
2. Normalization of data
3. PCA calculation
4. Training the regression model
5. Testing model performance

3.2.1 Data Preprocessing

This module takes the raw data from the dataset as its input and gives clean, complete and well formatted data as its output. This modules is responsible for performing the following major functions:

- ➔ Removal or replacement of missing values: All the rows in the dataset are checked for missing values. Rows containing single missing data are filled with the mean value of the row. Rows containing multiple missing values are removed from the dataset.
- ➔ Conversion of Timestamp into individual columns(day, month, year) as timestamp object cannot be processed by the algorithm.
- ➔ The data is first split into X (features for input) and Y (target for output) sets.
- ➔ The X and Y sets are then split into training and test set. Since a high amount of data is available, the split is done in the recommended ratio of '70-30' using train_test_split module.

3.2.2 Normalization of data

Normalizing is a technique used to bring the numeric values of the entire dataset to a common scale without changing the distribution curve of the data. If different features have different ranges, gradients may oscillate back and forth and take a long time before they finally find their way to the point where loss is minimum. When the data is normalized, the gradients converge more quickly hence decreasing the time for training. Another benefit is that if the columns which have the data in high ranges (for eg, 10,000 to 100,000) are converted to a lower range (for eg, 1 to 10) the matrix operations would become faster and hence take less computation time.

In our work, we have normalized all the values of the dataset in the range of 0 to 1. Additionally, normalization is necessary before calculation of Principal Component Analysis, which is done in the next step.

3.2.3 PCA Calculation

Principal Component Analysis is a method used for transforming the data in such a way that resulting variables become linearly independent of each other. It helps in identifying the independent vectors can describe the pattern of the data, hence reducing the dimensionality of training set for achieving good results.

Eigenvectors represent direction of data for each column, and eigenvalues represent the magnitude or importance of the eigenvectors. PCA takes into consideration the directions (eigenvectors) and their magnitude (eigenvalues) in describing the pattern of the data, which are calculated using the covariance matrix.

PCA calculation helps in analyzing the importance of each eigenvector and enables us to select only the most important features in order to obtain highly accurate results. The vectors (or components) obtained from PCA calculation are fed to regression as input.

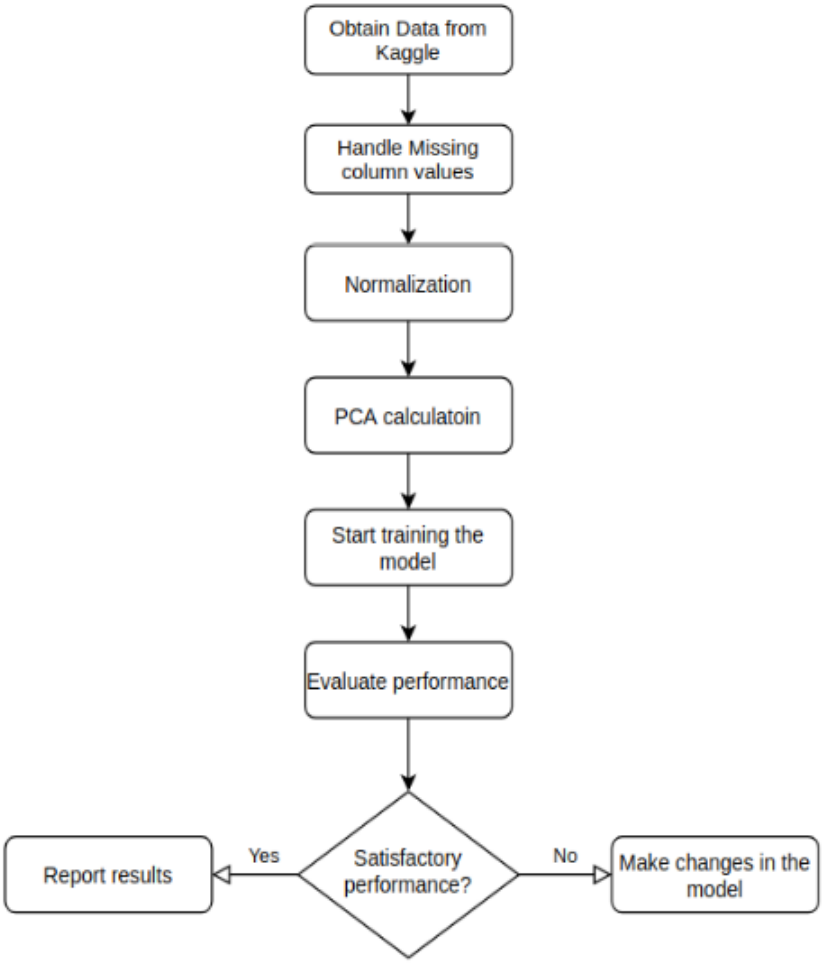
3.2.4 Model Training

Stock market data is highly complex in nature and hence we would have to select a model which can be trained to learn complex features and not overfit on the training set. Therefore, the Ridge Regression algorithm was chosen this project. In this regression model, a small amount of bias is added which acts as a regularizer in the model. This regularization prevents the model from overfitting and helps it achieve better results on the test data.

3.2.5 Model Performance Testing

The performance of the model is tested on the test set which contains data that the model has never seen before in order to get accurate results. Accuracy is selected as the performance metric and it is reported as the model is trained on different numbers of eigenvectors.

3.3 Proposed Data Flow Diagram



The dataset of National Stock Exchange of India is acquired from kaggle after which the data is preprocessed by handling the missing values and converting the whole data to numerical format. The cleaned and processed dataset is then normalized in the range of [0-1] and is fed to the PCA module where eigenvectors and their corresponding eigenvalues are calculated. These eigenvectors of training set are used for training and [“CLOSE”] column is predicted. The performance of the model is evaluated on the test set. Number of eigenvectors chosen for training are varied and performance in terms of accuracy is recorded.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Implementation Procedure

The aim is to predict the "Close" value for each day of the stock market dataset using all other features for the same day. Following procedure is used to achieve the desired result:

- ➔ The dataset is preprocessed and normalized for making it computationally efficient.
- ➔ Since the dataset is large, a feature elimination algorithm is employed for considering the most independent features from the dataset dataset for the prediction of target column. This reduces the amount of data to be processed hence increases the speed of algorithm.
- ➔ These extracted features are fed to our selected algorithm which makes the prediction.
- ➔ The dataset is split into training and test set. The algorithm is trained on the training set and its performance is evaluated on the test set.
- ➔ The results are reported with accuracy and other relevant performance metrics.

Following tools were used for implementation:

- ★ Python: It was chosen for implementation as a lot of machine learning and large data handling libraries are available.
- ★ Numpy: A mathematical library of python which efficiently handles large matrices and provides built in tools for mathematical operations.
- ★ Pandas: Data manipulation library which converts the data in "Data Frame" format that supports very fast data handling operations. This library was used to read data from the csv dataset obtained from kaggle.
- ★ Matplotlib: A graphical library which supports 2D graphs and charts. This was used for visualizing the data for understanding it.
- ★ Scikit-learn: A machine learning library which contains machine learning algorithms. This was used for normalizing the data, splitting the dataset into training and test, calculating PCA and training our selected algorithm.
- ★ Google Colab: A jupyter notebook interface provided by Google for performing high computation machine learning operations in python language. It was chosen for the project because it comes pre-loaded with all the dependencies installed and can be easily shared among the team to be used for collaboration. It provided us with high hardware resources which enabled us to complete our project on time.

★ Kaggle: A platform which hosts competitions and contains a lot of structured datasets which can be used for training machine learning algorithms. We made use of this tool for obtaining a National Stock Market dataset of India.

4.2 Implementation Code

```
mean_vec = np.mean(X_std, axis=0)
cov_mat = (X_std - mean_vec).T.dot((X_std - mean_vec)) / (X_std.shape[0]-1)
print('Covariance matrix \n%s' %cov_mat)
```

```
cov_mat = np.cov(X_std.T)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
principalComponents = pca.fit_transform(X_std)
pdf1 = pd.DataFrame(data = principalComponents, columns = ['pca1'])
pdf1.head()
```

```
from sklearn import linear_model
from math import sqrt

X_train,X_test,y_train,y_test=get_data(pdf1)

rr = linear_model.Ridge()
rr.fit(X_train, y_train)
pr=rr.predict(X_test)
acc=0
for i in range(len(pr)):
  if(sqrt((y_test[i]-pr[i])**2)<15):
    acc+=1
print('Accuracy = ',(acc/(len(y_test)))*100)
accuracy.append(acc/(len(y_test)))
```

```
from sklearn.model_selection import train_test_split

def get_data(pdf_no):
  X_train, X_test, y_train, y_test = train_test_split(pdf_no, y, test_size=0.3, random_state=0)
  y_train = np.array(y_train)

  y_train = np.array(y_train)
  y_test = np.array(y_test)
  X_train = np.array(X_train)
  X_test = np.array(X_test)
  X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1]))
  X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1]))
  return X_train, X_test, y_train, y_test
```

4.3 Outputs and Results

The proposed work made use of a number of modules. This section contains information about inputs and results from some of those modules.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 846404 entries, 0 to 846403
Data columns (total 13 columns):
SYMBOL          846404 non-null object
SERIES          843947 non-null object
OPEN            846404 non-null float64
HIGH            846404 non-null float64
LOW             846404 non-null float64
CLOSE           846404 non-null float64
LAST            846404 non-null float64
PREVCLOSE       846404 non-null float64
TOTTRDQTY       846404 non-null int64
TOTTRDVAL       846404 non-null float64
TIMESTAMP       846404 non-null object
TOTALTRADES     846404 non-null int64
ISIN            846404 non-null object
dtypes: float64(7), int64(2), object(4)
memory usage: 83.9+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 846404 entries, 0 to 846403
Data columns (total 12 columns):
OPEN            846404 non-null float64
HIGH            846404 non-null float64
LOW             846404 non-null float64
CLOSE           846404 non-null float64
LAST            846404 non-null float64
PREVCLOSE       846404 non-null float64
TOTTRDQTY       846404 non-null int64
TOTTRDVAL       846404 non-null float64
TOTALTRADES     846404 non-null int64
YEAR            846404 non-null int64
MONTH           846404 non-null int64
DATE            846404 non-null int64
dtypes: float64(7), int64(5)
memory usage: 77.5 MB
```

Figure: Input to the preprocessing modules (left) and output from the module (right)

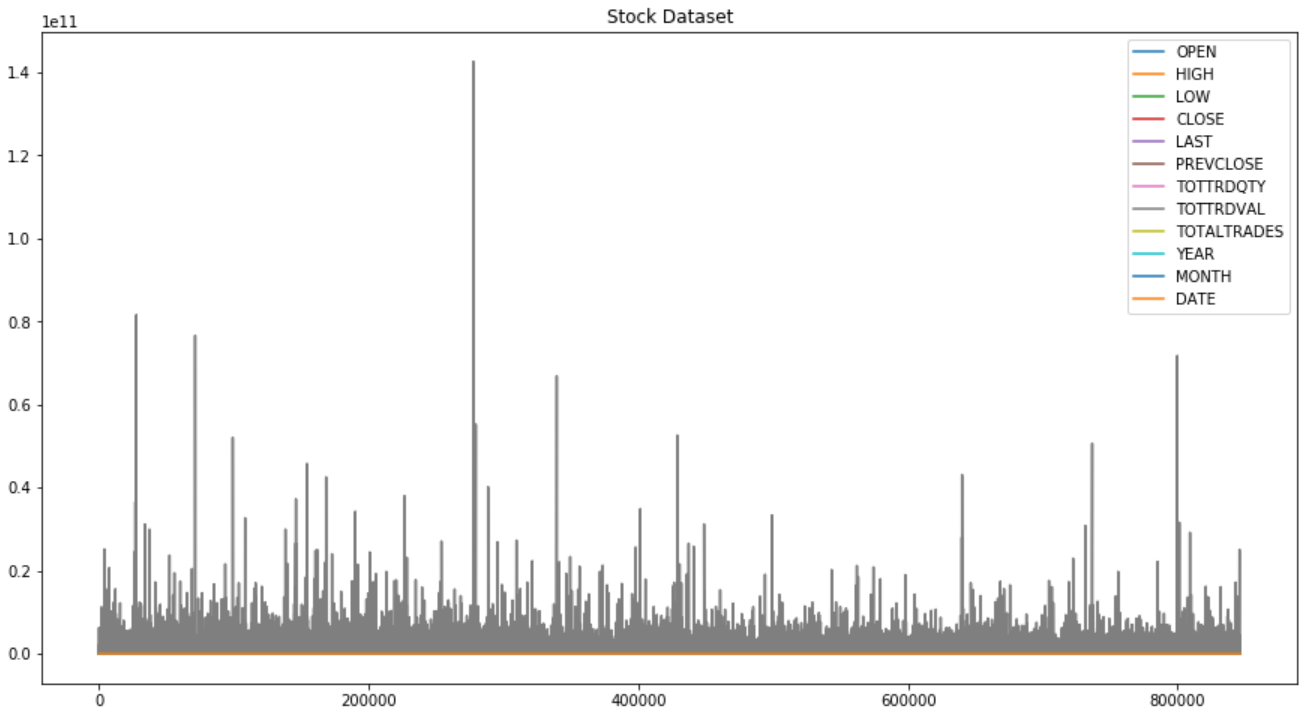


Figure: Visualization of dataset

```
Covariance matrix
[[ 1.00000118e+00  9.99888164e-01  9.99906703e-01  9.99743955e-01
   9.99165298e-01 -3.18362066e-02  6.02346392e-02  2.21014042e-02
   2.73070489e-02  1.60158408e-02  6.63192113e-04]
 [ 9.99888164e-01  1.00000118e+00  9.99819335e-01  9.99830881e-01
   9.99112191e-01 -3.17537313e-02  6.09119284e-02  2.27654361e-02
   2.71809532e-02  1.60769189e-02  6.71109101e-04]
 [ 9.99906703e-01  9.99819335e-01  1.00000118e+00  9.99809380e-01
   9.99082112e-01 -3.18612717e-02  5.98635678e-02  2.16776007e-02
   2.75607764e-02  1.60098894e-02  6.89207290e-04]
 [ 9.99743955e-01  9.99830881e-01  9.99809380e-01  1.00000118e+00
   9.98969750e-01 -3.17435395e-02  6.04512204e-02  2.24285659e-02
   2.73468870e-02  1.60586260e-02  7.12022075e-04]
 [ 9.99165298e-01  9.99112191e-01  9.99082112e-01  9.98969750e-01
   1.00000118e+00 -3.18736960e-02  6.00442895e-02  2.20856880e-02
   2.72731293e-02  1.60707525e-02  5.86383386e-04]
 [-3.18362066e-02 -3.17537313e-02 -3.18612717e-02 -3.17435395e-02
  -3.18736960e-02  1.00000118e+00  4.47042017e-01  4.75124179e-01
   2.22980458e-02  8.54989398e-03  7.29005137e-03]
 [ 6.02346392e-02  6.09119284e-02  5.98635678e-02  6.04512204e-02
   6.00442895e-02  4.47042017e-01  1.00000118e+00  8.56671867e-01
   2.95757562e-02  1.50255155e-02  1.08804008e-02]
 [ 2.21014042e-02  2.27654361e-02  2.16776007e-02  2.24285659e-02
   2.20856880e-02  4.75124179e-01  8.56671867e-01  1.00000118e+00
   1.96001295e-02  1.35533958e-02  5.97686496e-03]
 [ 2.73070489e-02  2.71809532e-02  2.75607764e-02  2.73468870e-02
   2.72731293e-02  2.22980458e-02  2.95757562e-02  1.96001295e-02
   1.00000118e+00 -3.01871264e-03  5.96937945e-05]
 [ 1.60158408e-02  1.60769189e-02  1.60098894e-02  1.60586260e-02
   1.60707525e-02  8.54989398e-03  1.50255155e-02  1.35533958e-02
  -3.01871264e-03  1.00000118e+00  2.63859017e-02]
 [ 6.63192113e-04  6.71109101e-04  6.89207290e-04  7.12022075e-04
   5.86383386e-04  7.29005137e-03  1.08804008e-02  5.97686496e-03
   5.96937945e-05  2.63859017e-02  1.00000118e+00]]
```

Figure: Input to PCA Module

```
Eigenvectors
[[-4.46782830e-01  1.33522099e-02  1.31318508e-02  2.06386312e-03
   3.45185520e-03  6.34702421e-03  4.96650099e-03  1.65586477e-01
   6.71774493e-01 -5.55489125e-01  1.12628729e-01]
 [-4.46785052e-01  1.29583390e-02  1.25459446e-02  2.01576337e-03
   3.43353835e-03  6.50523681e-03  4.94482121e-03  2.07156856e-01
  -4.54510375e-01  2.25657343e-02  7.41601187e-01]
 [-4.46771624e-01  1.35768262e-02  1.35034318e-02  2.07087649e-03
   3.44174798e-03  6.07033682e-03  4.67906216e-03  2.27060611e-01
  -5.03254638e-01 -3.08974561e-01 -6.32191850e-01]
 [-4.46754406e-01  1.31758433e-02  1.29704295e-02  2.01126553e-03
   3.45242245e-03  6.31267032e-03  5.51718584e-03  2.90677778e-01
   2.97879880e-01  7.68114284e-01 -1.91720797e-01]
 [-4.46569479e-01  1.34041872e-02  1.31475434e-02  2.06818611e-03
   3.35104940e-03  6.37333833e-03  5.74958760e-03 -8.90851644e-01
  -1.18994089e-02  7.38738752e-02 -3.03049338e-02]
 [ 1.13006382e-02 -4.76462747e-01  8.78514451e-01  9.78718881e-03
   5.59796176e-03  4.24379300e-03 -3.03745620e-02  8.61521180e-06
  -1.21204424e-05  1.37440094e-04  3.70244587e-04]
 [-3.65681366e-02 -6.16960479e-01 -3.58536955e-01  8.00657215e-03
   5.31481341e-03  2.04596786e-02 -6.99258355e-01 -5.44300087e-04
   2.34055174e-04  1.78822491e-04 -4.58202773e-04]
 [-1.89717182e-02 -6.24469569e-01 -3.14050826e-01  1.16276287e-02
   3.22154370e-03  3.16306081e-02  7.14074976e-01  3.21450333e-04
  -2.04910536e-04 -6.17156000e-04 -4.87004345e-04]
 [-1.55334187e-02 -3.24845859e-02 -1.29735342e-02  1.16607502e-01
  -1.11758227e-01 -9.86097183e-01  7.75903788e-03 -5.35370678e-05
   5.57858219e-05  5.43321075e-05  2.74991030e-04]
 [-9.11599399e-03 -1.73472588e-02  3.10252020e-03 -7.02817979e-01
  -7.11090274e-01 -1.83904986e-03  6.85652151e-04  3.70396324e-05
   1.23767342e-05 -3.63358954e-05 -2.88651776e-05]
 [-5.37772862e-04 -1.18656730e-02 -2.11513660e-03 -7.01522308e-01
   6.94069203e-01 -1.61158119e-01  4.10938220e-03 -8.70766183e-05
  -3.25616646e-07 -2.23426785e-05  2.35476097e-05]]

Eigenvalues
[5.00609523e+00 2.20803190e+00 6.45027083e-01 1.02609126e+00
 9.73144855e-01 9.97790925e-01 1.41959320e-01 1.37819219e-03
 5.04141131e-05 2.67940296e-04 1.75878650e-04]
```

Figure: Output from PCA Module

The following plot is obtained between the variance of features and number of considered eigenvectors. The plot shows that the amount of variance is increasing steadily as we are increasing the number of eigenvectors, however, the amount of variance becomes constant after the consideration of the 6th eigenvector. This change signifies that the variance achieves a standstill and becomes max after 6th eigenvector.

This graph is used in choosing the minimum number of eigenvectors that should be selected in order to achieve good results. Considering all the eigenvectors would give better results but would increase the cost of computation.

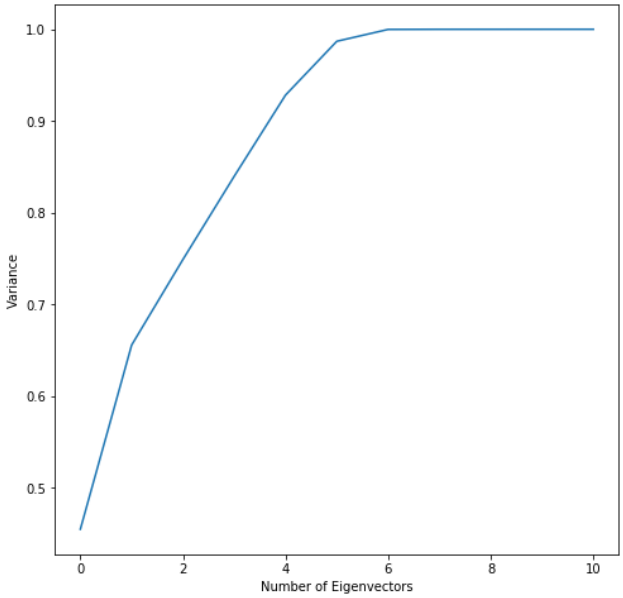


Figure: Number of eigenvectors and variance between them

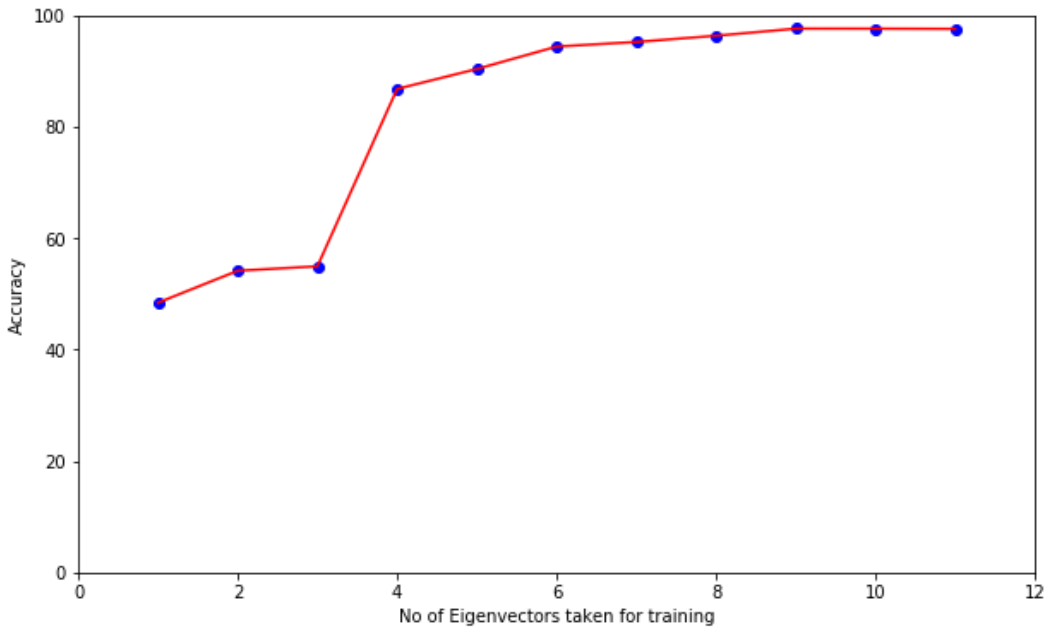


Figure: Effect of number of eigenvectors used for training versus accuracy achieved

4.4 Conclusion

In this work, an attempt is made at predicting the closing value of National Stock Exchange of India using other values like Open, High, Low, etc. A methodology proposed for this task, which involves calculating the Principal Components from the input features and then giving these components as an input to Ridge Regression algorithm. The proposed methodology achieves an accuracy of 97.59% using 10 components from the PCA. A steady increase is seen in the prediction accuracy with increase of number of components considered for training. Although the stock market is complex in nature and depends on a lot of factors, the results obtained in this study confirm that highly accurate prediction of stock market value is possible using machine learning techniques.