PUNE INSTITUTE OF COMPUTER TECHNOLOGY,
DHANKAWADI PUNE-43.

A Seminar Report

On

Stock Market Prediction

SUBMITTED BY

NAME: Anurag Gujarathi
ROLL NO: 3134
CLASS: TE-1

GUIDED BY

PROF. A. G. Phakatkar

COMPUTER ENGINEERING DEPARTMENT

Academic Year: 2018-19

PUNE INSTITUTE OF COMPUTER TECHNOLOGY,
DHANKAWADI PUNE-43.

CERTIFICATE

This is to certify that Mr. ***Anurag Gujarathi***, *R*oll No**. 3134** a student of T.E. (Computer Engineering Department) Batch 2016-2019, has satisfactorily completed a seminar report on "**Stock Market Prediction**" under the guidance of Prof. A. G. Phakatkar towards the partial fulfillment of the third year Computer Engineering Semester II of Pune University.

Prof. A. G. Phakatkar
Internal Guide

Dr. R.B.Ingle
Head of Department,
Computer Engineering

Date:
Place:

Abstract:

A stock market is the aggregation of buyers and sellers which represent ownership claims on businesses. Intelligent investments in the stock market can potentially earn high returns to investors. However, due to the non-linear nature of stock market fluctuations, it is difficult to make an intelligent decision. This leads to the need of a model, which can predict the stock market on the basis of historical data. Concepts like Regression, Artificial Neural Networks and Support Vector Machines make it possible to predict values on the basis of historical data. A model for estimating the daily opening and closing prices of the stock market has been proposed.

Keywords*: Stock market prediction, Regression, Neural Networks, Support Vector Machine*

Prediction of Stock Market using historical data

Contents

List of Figures

List of Tables

1 Introduction

The stock market is an aggregation of buyers and sellers of stocks, which represent ownership claims on businesses. Shares of stocks, equities and other securities of publicly listed companies can be traded on the Stock Exchange. An individual investor can participate in the trade of stocks on a Stock Exchange. The primary motive of an investor trading in the stock exchange is profitability.

The stock market is volatile in nature. The price of a stock of a particular company may undergo multiple fluctuations within a day. An investor in the hope of making profit from the investment always looks towards an intelligent and well informed decision. It is difficult to gauge projected prices of a stock in future. Machine learning techniques are being utilized to predict the future prices of stocks.

A predictive model which uses polynomial regression and support vector regression has been proposed to forecast prices of stocks on the basis of historical data. The patterns of fluctuations in historical data are recognized and future predictions are projected in accordance with these patterns. The stock prices data of technology companies like Google and Microsoft has been used in this prediction model.

1.1 Motivation

The stock market is known for providing a high return on investments. Intelligently designed and well executed decisions have the potential to earn exceptional profit. Many successful investors have become billionaires with the stock market being their sole source of income. For example, Mr Warren Buffett. Conversely, poorly thought investment decisions can also lead to financial losses. Many unsuccessful investors have faced bankruptcy owing to the financial losses sustained while trading in the stock market.

The stock market undergoes fluctuations routinely over the course of a day. The non-linear nature of these fluctuations makes it difficult to forecast the future prices of stocks at the time of investments. The quantification of an intelligent decision can be made after the return on investments is earned. Thus, a prediction model, which predicts future prices of a stock can aid in making an informed investment decision. Considering the prediction made by the model, an investor can judge the probability of profit and invest accordingly. Machine learning techniques like polynomial regression and support vector regression have been used.

1.2 Literature Survey:

1.2.1 Regression

[1] paper discusses the usage of regression algorithms in prediction of stock market. Support vector machines have a number of applications in prediction as mentioned in the paper. It also gives an overview of the shortcomings of support vector regression and the subsequent need to use particle swarm optimization. It was published in IEEE transactions on neural networks and learning systems.

1.2.2 Particle Swarm Optimization

[1] applies particle swarm optimization to a support vector regression model to increase the accuracy of prediction. Particle swarm optimization is a technique for iteratively selecting the best cost solution.

1.2.3 Deep Learning, Artificial Neural Networks

[2] was published in IEEE Access. It compares different techniques used to predict stock market prices. Deep learning, back propagation neural networks are some techniques that are discussed.

1.3 Challenges

Being a non-linear model, finding a best fit curve exposes the algorithm to the problem of overfitting and underfitting.

1.3.1 Underfitting

Underfitting occurs when the model is unable to adequately capture the underlying structure of the data. It leads to a highly biased representation. Underfitting is typically observed when fitting a linear model to a non-linear data distribution. It leads to poor performance during prediction.

1.3.2 Overfitting

Overfitting is observed when the predictive model is designed to fit the dataset too perfectly. In overfitting, some boundary line cases are also correctly classified or predicted. However, the downfall of this phenomenon is the increase in the degree of the polynomial representing the curve.

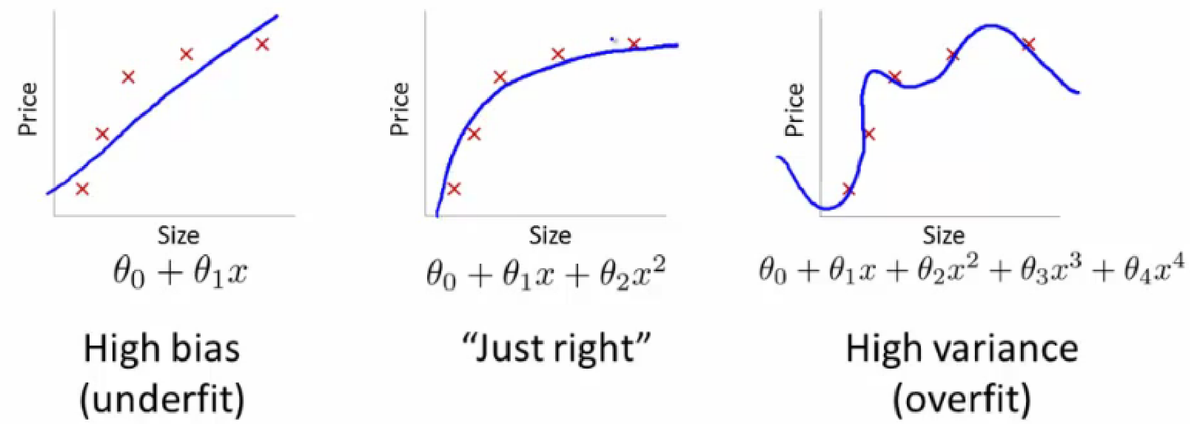

Figure 1. Underfitting and Overfitting.

2 Polynomial Regression

2.1 Dataset

The dataset is obtained from Quandl API and accessed using Python's Numpy array. The parameters of the dataset are Open, High, Low, Close, Volume, Adj. Open, Adj. High, Adj. Close, Adj. Low, Adj. Volume.

In order to obtain the features of the dataset, following operations are performed.

```
df = df[['Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close', 'Adj. Volume']]
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Low']) / df['Adj. Low'] * 100.0
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj. Open'] * 100.

df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]

forecast_col = 'Adj. Close'
df.fillna(-99999, inplace=True)

forecast_out = int(math.ceil(0.1*len(df)))

df['label'] = df[forecast_col].shift(-forecast_out)

X = np.array(df.drop(['label'],1))
X = preprocessing.scale(X)
X_lately = X[-forecast_out:]
X = X[:-forecast_out]

df.dropna(inplace=True)
y = np.array(df['label'])
```

Figure 2. Obtaining features for regression.

Following image represents the data obtained by the Quandl API.

```
  Open     High      Low     Close       Volume  Ex-Dividend  \

100.01  104.06    95.96  100.335  44659000.0          0.0
101.01  109.08   100.50  108.310  22834300.0          0.0
110.76  113.48   109.05  109.400  18256100.0          0.0
111.24  111.60   103.57  104.870  15247300.0          0.0
104.76  108.00   103.88  106.000   9188600.0          0.0

Split Ratio  Adj. Open  Adj. High   Adj. Low  Adj. Close  \

        1.0  50.159839  52.191109  48.128568   50.322842
        1.0  50.661387  54.708881  50.405597   54.322689
        1.0  55.551482  56.915693  54.693835   54.869377
        1.0  55.792225  55.972783  51.945350   52.597363
        1.0  52.542193  54.167209  52.100830   53.164113

Adj. Volume

 44659000.0
 22834300.0
 18256100.0
 15247300.0
  9188600.0
```

Figure 3. Parameters in the dataset.

2.2 Features and Label

Features obtained after manipulating raw data are Adj. Close, HL_PCT, PCT_Change, Adj. Volume. The objective of the regression model is to predict the closing stock prices. The forecast column of closing stock prices is shifted up by 10 percent of the size of the dataset and serves as the label for the regression model.

2.3 Algorithm

Following is the regression algorithm for prediction of closing stock prices of Google's data. The library used for performing the prediction is scikit-learn. Pickle is the standard functionality provided by scikit-learn to load data into the model.

```
pickle_in = open('stock_classifier.pickle', 'rb')
clf = pickle.load(pickle_in)
accuracy = clf.score(X_test, y_test)
forecast_set = clf.predict(X_lately)
print('Accuracy is ',accuracy)

df['Forecast'] = np.nan
last_date = df.iloc[-1].name
last_unix = last_date.timestamp()
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]

df['Adj. Close'].plot()
df['Forecast'].plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

Figure 4. Code snippet of algorithm.

2.4 Result

The last 10 percent of the dataset is forecasted by using the prediction model and compared with the values of the data. An accuracy of 70% is observed. Following graph depicts the result.

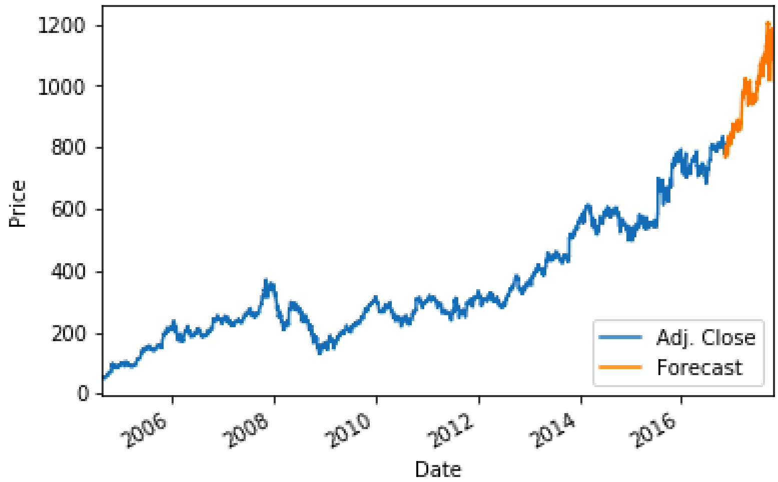

Figure 5. Result of prediction.

3 Support Vector Regression.

3.1 Dataset

Microsoft's stock prices were used as the training data. This data was obtained from Kaggle. Following image represents the data.

	A	B	C	D	E	F	G
1		date	open	low	high	close	volume
2	544	2010-01-04	30.620001	30.59	31.1	30.950001	38409100
3	1012	2010-01-05	30.85	30.639999	31.1	30.959999	49749600
4	1480	2010-01-06	30.879999	30.52	31.08	30.77	58182400
5	1948	2010-01-07	30.629999	30.190001	30.700001	30.450001	50559700
6	2416	2010-01-08	30.280001	30.24	30.879999	30.66	51197400
7	2884	2010-01-11	30.709999	30.120001	30.76	30.27	68754700
8	3352	2010-01-12	30.15	29.91	30.4	30.07	65912100
9	3820	2010-01-13	30.26	30.01	30.52	30.35	51863500
10	4288	2010-01-14	30.309999	30.26	31.1	30.959999	63228100
11	4756	2010-01-15	31.08	30.709999	31.24	30.860001	79913200
12	5224	2010-01-19	30.75	30.68	31.24	31.1	46575700
13	5692	2010-01-20	30.809999	30.309999	30.940001	30.59	54849500
14	6160	2010-01-21	30.610001	30	30.719999	30.01	73086700
15	6628	2010-01-22	30	28.84	30.200001	28.959999	102004600
16	7096	2010-01-25	29.24	29.1	29.66	29.32	63373000
17	7564	2010-01-26	29.200001	29.09	29.85	29.5	66639900
18	8032	2010-01-27	29.35	29.02	29.82	29.67	63949500
19	8500	2010-01-28	29.84	28.889999	29.870001	29.16	117513700
20	8968	2010-01-29	29.9	27.66	29.92	28.18	193888500
21	9436	2010-02-01	28.389999	27.92	28.48	28.41	85931100
22	9904	2010-02-02	28.370001	28.139999	28.5	28.459999	54413700
23	10372	2010-02-03	28.26	28.120001	28.790001	28.629999	61397900
24	10840	2010-02-04	28.379999	27.809999	28.5	27.84	77850000
25	11308	2010-02-05	28	27.57	28.280001	28.02	80960100
26	11776	2010-02-08	28.01	27.57	28.08	27.719999	52820600
27	12244	2010-02-09	27.969999	27.75	28.34	28.01	59195800
28	12712	2010-02-10	28.030001	27.84	28.24	27.99	48591300
29	13180	2010-02-11	27.93	27.700001	28.4	28.120001	65993700
30	13648	2010-02-12	27.809999	27.58	28.059999	27.93	81117200
31	14116	2010-02-16	28.129999	28.02	28.370001	28.35	51935600
32	14584	2010-02-17	28.530001	28.360001	28.65	28.59	45882900
33	15052	2010-02-18	28.59	28.51	29.030001	28.969999	42856500

msft_prices

Find | Find All | Formatted Displa

Sheet 1 of 1 | Default

Figure 6. Microsoft's dataset

3.2 Algorithm logic

Following code snippet gives the implementation of the support vector regression.

```
In [7]: def train_and_test(price,window_length,accurarys,reports):
            x,y = get_x_and_y(msft_prices,window_length=window_length)
            y = y.flatten()
            scaler = preprocessing.StandardScaler()
            scaler.fit_transform(x)
            x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=233)
            for kernel_arg in ['rbf','poly','linear']:
                clf = svm.SVC(kernel=kernel_arg,max_iter=5000)
                clf.fit(x_train,y_train)
                y_predict = clf.predict(x_test)

                accurary = clf.score(x_test,y_test)
                report = classification_report(y_test,y_predict,target_names = ['drop','up'])
                if window_length in accurarys:
                    accurarys[window_length].append(accurary)
                    reports[window_length].append(report)
                else:
                    accurarys[window_length] = [accurary]
                    reports[window_length] = [report]
                print('The Accurary of %s : %f'%(kernel_arg,clf.score(x_test,y_test)))
                print(report)
```

Figure 7. Code snippet of SVR

3.3 Result

3 kernels - radial basis function, linear and polynomial were implemented. Following is the result of the implementation.

```
The Accurary of rbf : 0.471526
              precision    recall  f1-score   support

        drop       0.47      1.00      0.64       207
          up       0.00      0.00      0.00       232

   micro avg       0.47      0.47      0.47       439
   macro avg       0.24      0.50      0.32       439
weighted avg       0.22      0.47      0.30       439

The Accurary of poly : 0.528474
              precision    recall  f1-score   support

        drop       0.00      0.00      0.00       207
          up       0.53      1.00      0.69       232

   micro avg       0.53      0.53      0.53       439
   macro avg       0.26      0.50      0.35       439
weighted avg       0.28      0.53      0.37       439

The Accurary of linear : 0.571754
              precision    recall  f1-score   support

        drop       0.60      0.28      0.38       207
          up       0.56      0.83      0.67       232

   micro avg       0.57      0.57      0.57       439
   macro avg       0.58      0.56      0.53       439
weighted avg       0.58      0.57      0.54       439
```

Figure 8. Result of support vector regression

4 Particle Swarm Optimization

In computational science, Particle Swarm Optimization (PSO) is a method that optimizes the problem by iteratively trying to improve a candidate solution with regard to a given measure of quality.

It solves a problem by having a population of candidate solutions, here called particles, and moving these particles around in the search space over the particle's position and velocity.

5 Conclusion and Future Enhancement

4.1 Conclusion

1) Using polynomial regression and support vector regression, accuracy of prediction model varies between 50% and 70% for the datasets used.

2) The accuracy is dependent on training data.

4.2 Future Enhancements

- Although stock prices can be predicted to a certain extent using only historic data, other factors also influence stock prices.
- Following factors also affect stock prices -
  - Economics
    - –Interest rates
    - –Inflation
  - Politics
    - –Government policy
    - –Elections
  - Natural and Man-made disasters
    - –Japan in 2011 tsunami
    - –World War II
  - Market Psychology
    - –Hype created by economists

The predictive model must account for these factors in order to give a comprehensive prediction. These factors can be taken as input through a natural language classifier which processes current happenings through media reports.

References

[1] Yongsheng Ding, Lijun Cheng, Witold Pedrycz and Kuangrong Hao, "Global Nonlinear Kernel Prediction for Large Data Set With a Particle Swarm-Optimized Interval Support

Vector Regression" IEEE Transactions on Neural Networks and Learning Systems, Vol. 26, No. 10, October 2015

[2] Chen, L., Qiao, Z., Wang, M., Wanga, C., Du, R., & Stanley, H. E. (2018). "Which artificial intelligence algorithm better predicts the Chinese stock market?" IEEE Access, 1–1.

[3] *https://www.kaggle.com/rosand/fork-of-predict-stock-prices-with-svm*

APPENDIX – D

Log Book

Roll No. **:- 3134**

Name of the Student **:- Anurag Gujarathi**

Name of the Guide **:- Prof. A.G. Phakatkar**

Seminar Title **:- Stock Market Prediction**

Sr. No.	**Date**	**Details of Discussion/ Remarks**	**Signature of guide / Seminar Incharge**
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			

Student Signature **Guide Signature**