Stock Market Prediction

Anurag Gujarathi
3134

Literature Survey

Regression-

’’Global Nonlinear Kernel Prediction for Large Data Set With a Particle Swarm-Optimized Interval Support Vector Regression”

Motivation

- Volatile nature of stock market.
- Correct decision can make an investor a billionaire.
- Wrong decision can make an investor bankrupt.
- Success depends on the future worth of purchased stocks and fundamentals of the firm.
- Notion for profitability –
  - Buy low
  - Sell high

Motivation

- A prediction model can assist investors in making informed and intelligent decisions.
- Intelligent decisions can potentially earn high return on investments for investors.
- Thus a stock prediction model has been proposed for forecasting prices on the basis of historical data.

Problem Statement

- Prediction of Stock Market using historical data.
- 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.

Regression Analysis

- In statistical modeling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). source : https://en.wikipedia.org/wiki/Regression_analysis

Features

- Dataset – Quandl GOOGL

```
  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
```

Adjusting Features

```
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'])
```

Algorithm Logic

```
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()
```

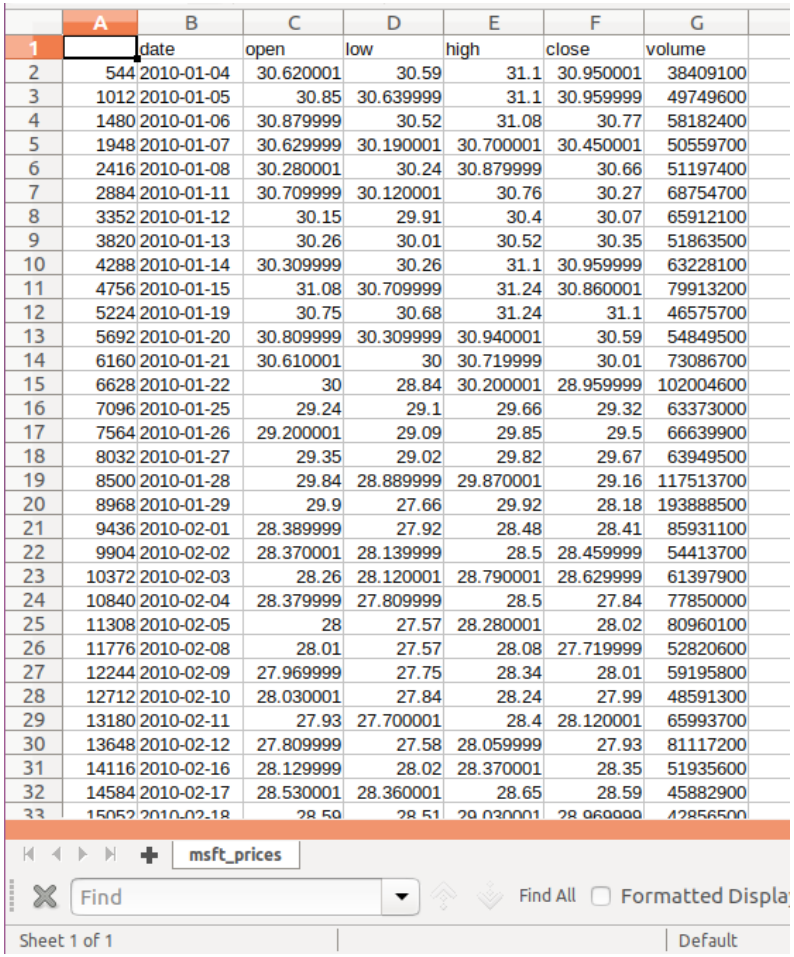
Result


1200
1000
800
600
400
200
0
Price
2006
2008
2010
2012
2014
2016
Date
Adj. Close
Forecast

Support Vector Machine

- In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.
- Source- https://en.wikipedia.org/wiki/Support-vector_machine

Dataset



	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

Algorithm Logic

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)
```

Result

```
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
```

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.
- Source: https://en.wikipedia.org/wiki/Particle_swarm_optimization

Conclusion

- Using polynomial regression and support vector regression, accuracy of prediction model varies between 50% and 70% for the dataset used.
- The accuracy is dependent on training data.
- Although stock prices can be predicted to a certain extent using only historic data, other factors also influence stock prices.

Conclusion

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

Conclusion

Therefore, a prediction model must take into account all factors which may affect the stock market.