Udacity Machine Learning Nanodegree

Project Report

Ashish Rao Mangalore

May 2nd, 2017

Figure 0:[Source: https://techemergence.com/wp-content/uploads/2016/08/Machine-Learning-in-Finance.jpg]

Machine Learning in Finance

Section 1: Definition

1.1 Introduction

The project will be an approach to leverage the power of data to make predictions of a stock to the maximum possible accuracy. Investment funds are hedge funds are always on the lookout for processes to maximize profit. With advances in computational power and data processing techniques coupled with the ubiquitous nature of devices connected to the internet, machine

learning based are extensively used to study the performance of stocks. There are generally two approaches to predicting the price of a stock in finance.

- Qualitative/Fundamental Analysis- Investors study terms sheets, profit and loss statements, company's credibility, recent developments in the market and so on and then execute a trade. This approach to trading requires extensive reasoning and human intelligence and hence is not a good problem for machine learning to solve.

- The second approach is to use quantitative analysis to predict the future price of the stock based on various statistical parameters like rolling mean, daily returns, rolling standard deviation and so on. This approach makes use of data driven methods and would be a good problem to be investigated using machine learning.

1.2 Problem Statement

The objective of the project is to use machine learning algorithms to predict the price of a chosen stock 30 days(value of forecast variable which can be changed) from the last date of the queried date range. After observing the performance of multiple machine learning algorithms, the one giving the best results is selected. Data pertaining to different company stock prices is available online ex. Quandl, Yahoo Finance, Google Finance and Bloomberg to name a few. These sources are processed using a suitable framework like Pandas and ML algorithms are built upon them.

The project is implemented in python 2.7 in the Spyder IDE/iPython notebooks. It is divided into phases like accessing the API , pulling the data, dividing them to data frames, checking their authenticity and so on .It is explained in detail as follows

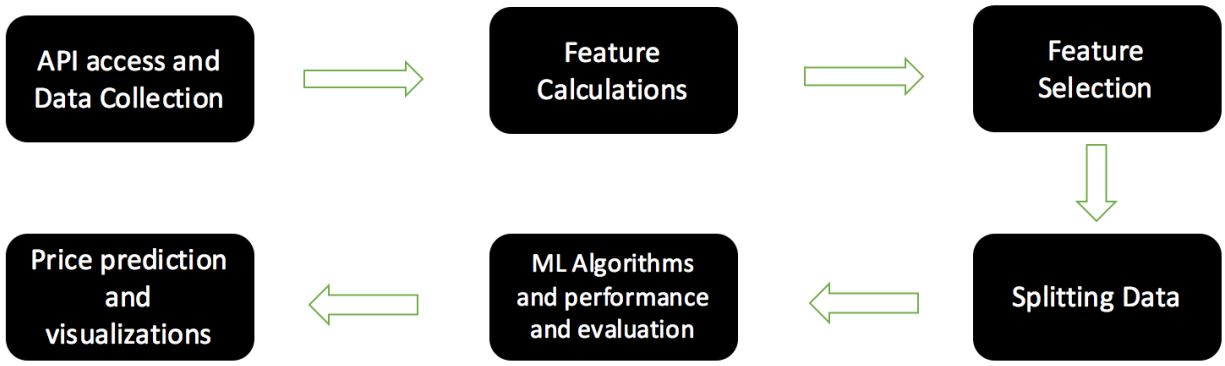

Figure 1.1: Project Design

Stage 1: API access and data collection

Quandl is used to obtain data for the 'AAPL' stock. This is done by importing the Quandl API and setting up the configuration key. The data pertaining to a particular time period is called for and stored in a dataframe. The time period for training is specified and the data during this period is segregated in separate dataframes. Any anomalous data is looked for and removed from the data frame.

Stage 2: Calculating necessary features

While input features like 'Open', 'Close', 'High' , etc are readily available ,for the sake of improving the trainer , it becomes necessary to compute other features which can be incorporated into the training dataset. Features like 10 day rolling mean ,50 day rolling mean ,daily returns are calculated and stored in data frames for training.

Stage 3: Feature Selection

The features which are correlated can be dropped as required as they add no value to the training. This can be done by a PCA or by selecting features manually and testing the performance of the model obtained.

Stage 4: Splitting the training and testing data set

The data is available in the form of time Series data and hence it is split by merely using the first N rows according to the training ratio specified (about 70%) or by using time series train test split methods.

Stage 5: Selection of Algorithm and Finalizing predictor

Since this is a regression problem various ML algorithms like KNN, linearfit , decision tree regressor , static vector regression and multi-layer perceptron networks are run of the test and tested for .Additionally Cross validation can be run for better performance rather than just using one set of train and test data sets. The best performing algorithm is selected using performance metrics like F1 score or the R2 score.

Stage 6: Prediction of prices, Plotting Graphs and visualizations

The price of the stock is predicted for the next 30 days after the date range queried and all necessary graphs are plotted using matplotlib. It is possible to plot the performance of the algorithms used and the predicted prices in the queried date range. This helps in visual summarization of the prediction.

1.3 Evaluation Metrics

Being a regression project, the performance is evaluated by the score generated for the testing data for the queried date. These performance metrics are available as standards and need not be devised by the engineer from scratch. They indicate how well our algorithm is performing evaluation metrics commonly used is the R2 score. Also the error the model developed gives on new data points is a valid metric to finalize on the selection of the model.

Section II: Analysis

2.1 Data Exploration

The dataset for the purpose of this project will be the Stock of 'AAPL' or Apple. This is data is chosen because it has a lot of data points right from the 1980-12-12 and hence is a good problem for a machine learning based approach. The training range selected for this project will be from '2004-01-17' to '2008-03-8'. Data is obtained through the Quandl API in Python 2.7. The data obtained is of the time-series type and it is processed using Pandas in Python. The Quandl API gives information about the opening price, closing price, daily high, daily low ,adjusted close and also the indication of the issue of dividends or stock splits (with a 1 or 0).The very basis of technical analysis is the prediction of stock prices by statistical approaches. However, the amount of insight a human can achieve is much less than what a machine can do given the same amount of data. Hence using a machine learning based predictor would be a very good way to go about with this dataset.

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	\
1980-12-12	28.75	28.87	28.75	28.75	2093900.0	0.0	1.0	
1980-12-15	27.38	27.38	27.25	27.25	785200.0	0.0	1.0	
1980-12-16	25.37	25.37	25.25	25.25	472000.0	0.0	1.0	
1980-12-17	25.87	26.00	25.87	25.87	385900.0	0.0	1.0	
1980-12-18	26.63	26.75	26.63	26.63	327900.0	0.0	1.0	

Date	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
1980-12-12	0.426158	0.427937	0.426158	0.426158	117258400.0
1980-12-15	0.405851	0.405851	0.403924	0.403924	43971200.0
1980-12-16	0.376057	0.376057	0.374278	0.374278	26432000.0
1980-12-17	0.383468	0.385395	0.383468	0.383468	21610400.0
1980-12-18	0.394733	0.396512	0.394733	0.394733	18362400.0

Fig 2.1 Raw data from Quandl

The features used in this project are the high low percentage change, Adj. Open to Adj. Close percentage change, the 50-day rolling mean, the 10-day rolling mean, the 10-day standard deviations the adjusted volume, stock splits, Expected dividends and the Adj. Close price for the day. The target data will be Adj. Close 30 days into the future.

	Adj. Close	HL_PCT	PCT_change	Adj. Volume	Rolling std	Rolling Mean 50	Rolling Mean 10	Split Ratio	Ex-Dividend	label
count	1011.000000	1011.000000	1011.000000	1.011000e+03	1011.000000	1011.000000	1011.000000	1011.000000	1011.0	1011.000000
mean	8.815968	3.110067	0.022118	1.951845e+08	-889.925393	-4838.415404	-881.477428	1.000989	0.0	9.256005
std	6.011920	1.626634	2.053864	1.160337e+08	9397.572960	21487.332785	9398.375871	0.031450	0.0	6.002347
min	1.411593	0.675788	-7.840806	3.540880e+07	-99999.000000	-99999.000000	-99999.000000	1.000000	0.0	1.549579
25%	4.537311	2.060659	-1.100046	1.134010e+08	0.097208	4.105050	4.410274	1.000000	0.0	4.760808
50%	8.013485	2.736842	0.021097	1.729756e+08	0.196076	7.863177	7.995488	1.000000	0.0	8.397135
75%	11.322547	3.671969	1.201293	2.405680e+08	0.332716	11.235558	11.272858	1.000000	0.0	11.806467
max	25.890658	12.185889	8.697411	8.432424e+08	1.913844	23.760121	25.242064	2.000000	0.0	25.890658

Figure 2.2: Statistics for training dataset from '2004-01-17' to '2008-03-8'

There are 1011 data points for our date range. It Can be seen that the max value of the Adj. Closing Price is 25.89 $ in this period and the minimum is 1.411$. The stock price of AAPL has gone high as high as 700 $ but that is for a later time period as shown in Figure 2.3. The data is drawn from the Quandl API which is also offered as a professional service. It can be seen that the data is clean and does not contain any anomalies

2.2 Exploratory Visualizations

It is known that the objective of the project is to predict the adjusted close price of 'AAPL' given certain input features as required. Hence this mainly falls into the category of a regression problem. However, before a suitable regression algorithm can be applied, it is necessary that only those features be used that are relevant. This helps to reduce the training time and makes the model more robust. Figure 2.3 shows the trend in Apple's stock price over the years since its inception. It can be seen that the price has been taking a rise most of the time except for that drastic fall in 2013.This can be explained with Figure 2.4.

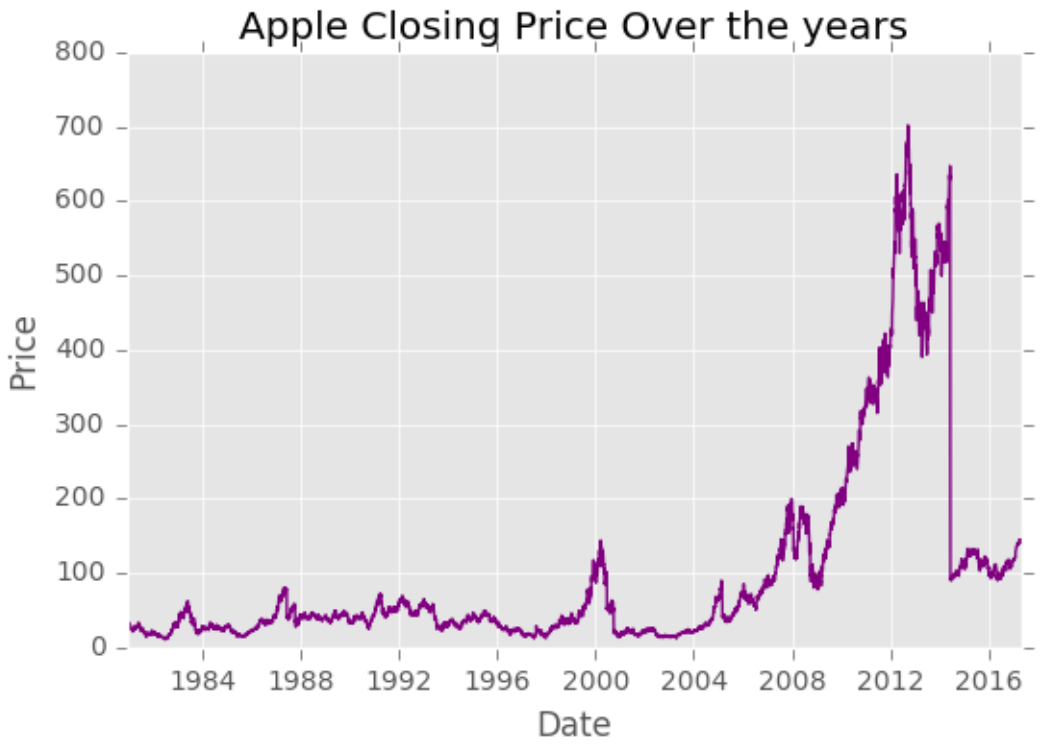

Figure 2.3: AAPL closing price over the years

Figure 2.4 shows the issue of dividends and the stock splits which have taken place in the company since its inception. It can be seen that somewhere in 2013, Apple was valued so highly that the stock was too expensive to buy. As a result, a stock split had to issued hence driving the value of the stock price much lower than its value before although the company was valued the same. Hence, it should be noted that the issue of stock splits or dividends is an important feature for the machine learning model as the stock price goes down even though the company is doing well financially.

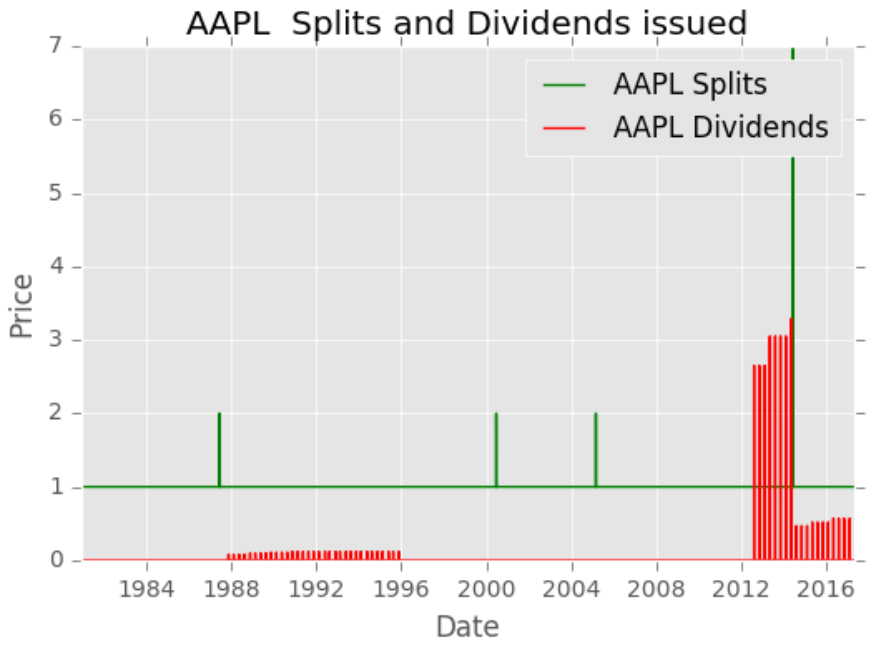

Figure 2.2: AAPL Stock splits and dividends issued over the years

The absolute closing price can give a wrong picture of the stock, hence It is necessary to incorporate the adjusted closing price. The trend of the adjusted closing price is seen to be mostly upwards as shown in figure 2.5.Hence we can assume that even a linear regression algorithm might give decent results.

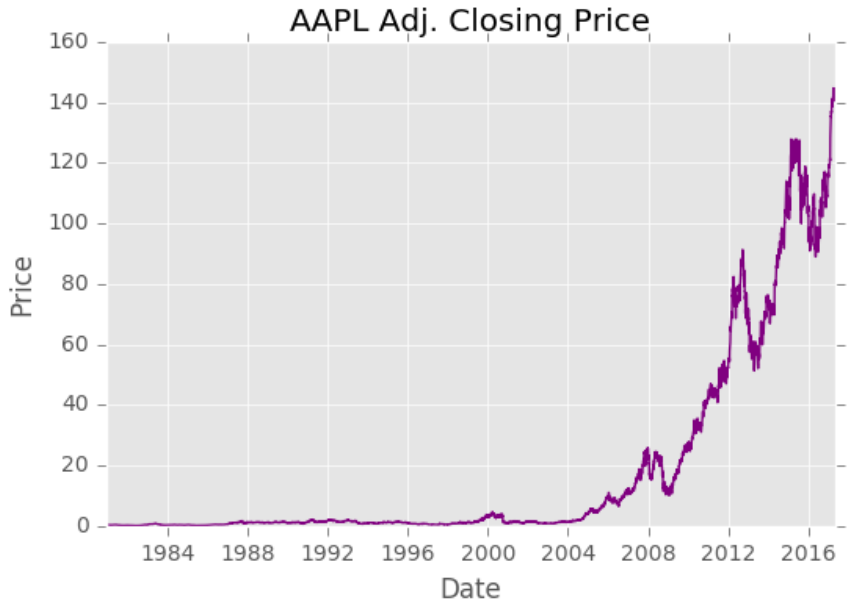

Figure 2.3: AAPL Adj. Closing price

2.3 Benchmark Model

Stock markets are known to be extremely stochastic and thus even with machine learning models the R2 score of the prediction is pretty low(in the range of 40-60%). If the models performed any higher than it would have been a case of an overfit and not a good sign of a predictor(otherwise you would be a billionaire!!). Therefore, getting a performance with predictor accuracies in the abovesaid range would amount to a good training for the stock in question. Another approach would be to fit a very rudimentary model like a linear regressor and compare the performance of the model with this very naïve regressor.

2.4 Algorithms and Techniques

The Algorithms used for this project will be discussed and justified in this section. The algorithms this project will be using are Linear Regression, Static Vector Regression, Decision Tree Regression, Multilayer Perceptron, Grid Search and Cross Validation.

2.4.1 Linear Regression

Linear regression is the simplest machine learning Algorithm available. It is often used in cases where the data needs to fit to single straight line. In cases where benchmarking data isn't available or is faulty (like in our case), the model derived from this algorithm can be used as a benchmark model because it is the simplest possible model to obtain. However, linear regression is very susceptible to outliers which can affect the overall accuracy of the model.

2.4.2 Support Vector Machines

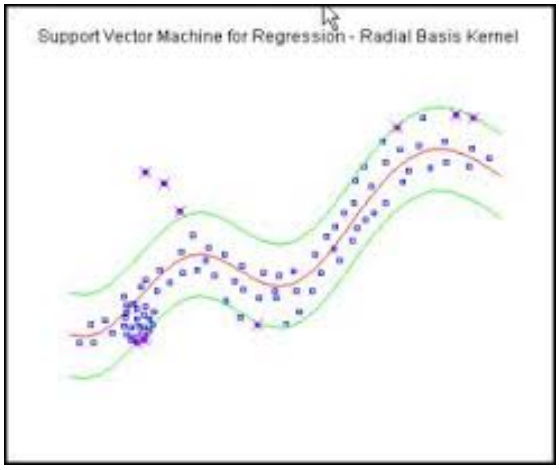

Figure 2.4: Support Vector Machine Source [http://kernelsvm.tripod.com]

For this project, we will be using support vector regressors to fit a model for stock forecasting. This algorithm looks to divide the data into a hyperplane in order to predict the outcome. The hyperplane is defined by what kind of kernels we choose .A nonlinear kernel might perform better but it increases the chances of overfit. The epsilon value controls how close the data points must be to the decision boundary formed by the SVM also known as the tolerance. The disadvantages of SVM are that the choice of the kernel determines the performance of the algorithm and this is pretty arbitrary. In addition to this the time and space required by SVM for training and testing are very high compared to other algorithms. However, SVM provides a robust decision boundary sometimes better than linear or logistic regression.

2.4.3 Multilayer Perceptron

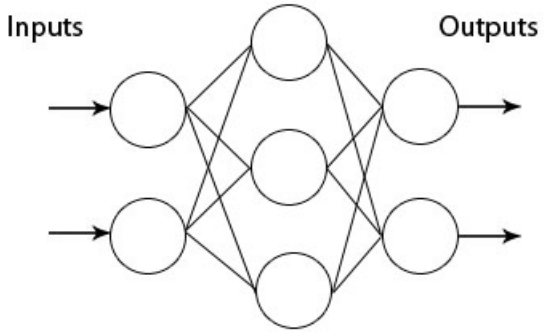

Figure 2.5: A Multilayer Perceptron Source:[http://neuroph.sourceforge.net/tutorials/MultiLayerPerceptron.html]

Multilayer Perceptrons or also called Artificial Neural Networks make use of a basic unit called a perceptron/Neuron. The perceptrons can be arranged in different layers where each layer models a particular feature of the input data. Increasing the number of neurons in each layer also affects the performance of the network. However, care should be taken to not overfit the data by increasing the number of hidden layers and the neurons. Neural Networks can be used in areas where the data is very stochastic so as to get reasonably good models. Hence it is a reasonable choice for the finance dataset.

2.4.4 Decision Tree Regressor

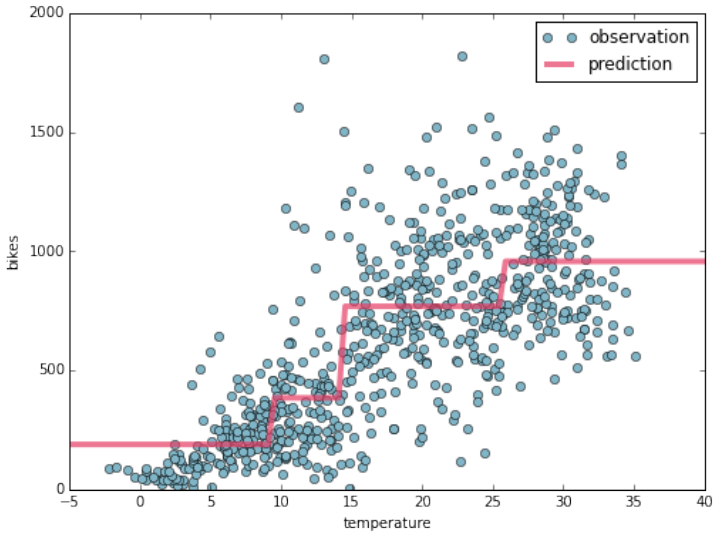

Figure 2.6: Decision Tree on a dataset, Source:[http://online-dev.cambridgecoding.com/notebooks/cca_admin/getting-started-with-regression-and-decision-trees]

Decision trees are applied in various areas like finance ,astronomy ,management and biomedical electronics to name a few. A very good example for a decision tree is the website akinator (http://en.akinator.com).The advantages of decision trees are they do not require much normalization or preprocessing. Also they are easier to explain and understand to a client. Moreover, the importance of features is determined automatically as the more prominent features tend to stay at the upper nodes of the decision tree. A nonlinear relationship between the variables does not affect the output of the decision tree. The weakness of this model is that it is prone to overfit. It is becomes practically useless in extremely large datasets as it doesn't learn the nature of the data. Moreover it cannot predict continuous values or cases where there are more than one output variables. However, since Decision Tree Regressors are used to predict the trend of stocks in finance, it is worth a shot to see how they behave for stock prediction.

2.4.5 Grid Search and Cross Validation

The grid search technique is a method of finding all possible combinations of parameters and constructing models for them. It called a grid search because the best performing model is searched for and selected from the models of different parameters which are arranged like a grid. In this case grid search is performed over max depth parameter. The best performing model from the grid is selected by making use of a cross validation data

In k-fold cross validation training technique, the data is divided into k sets.in the first loop the first set is used for training and the remaining k-1 sets are used for the testing. In the second iteration, the second set is used for training and the remaining k-1 sets are used for testing. This is repeated k times. In the end the optimum parameters are found by averaging the values obtained by the k iterations. This Technique would be very useful for refining the parameters during the model refinement phase.

Section III: Methodology

3.1 Data Preprocessing

The data is pulled from an online service called ‘Quandl’ as a result of which all the data obtained has no missing values or lack of information. All the points are considered as outliers may be a sign of sudden change in market sentiment. For the purposes of training, we select our date range to be '2004-01-17' to '2008-03-8'. It can be seen that for this range the growth rate of the company is fairly stable, hence It can help in modelling AAPL’s normal behavior. the logarithmic growth rate does not depend on the denominator hence it gives a clearer picture of growth irrespective of the company’s absolute stock price.

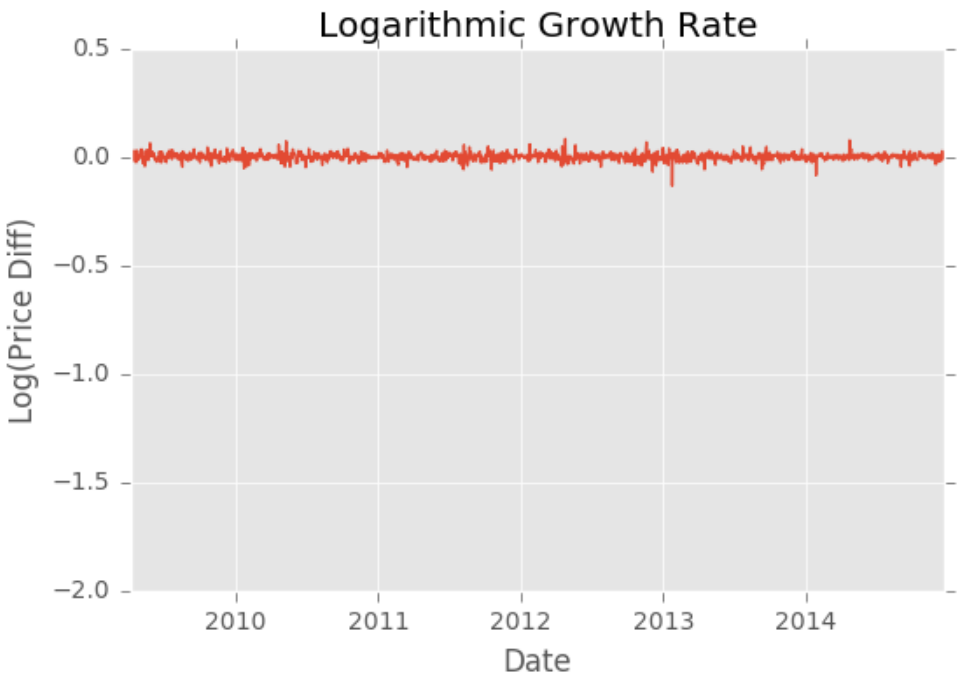

Figure 3.1: Logarithmic growth rate for the AAPL stock from the time period '2004-01-17' to '2008-03-8'

However, it is necessary to engineer a few features for the purpose of this project. The high-low percentage change and the open-close percentage change are calculated as shown below.

```
df["HL_PCT"] = (df["Adj. High"] - df["Adj. Low"])/df["Adj. Low"] * 100.0 #The High-Low percentage change
df["PCT_change"] = (df["Adj. Close"] - df["Adj. Open"])/df["Adj. Open"]* 100.0 #The Close-Open  percentage change
```

Further, the 10-day rolling mean, the 50-day rolling mean and the 10-day rolling standard deviation are calculated as shown below.

```
df["Rolling Mean 10"]=pd.rolling_mean(df["Adj. Close"],window=10) #The 10 day rolling mean
df["Rolling Mean 50"]=pd.rolling_mean(df["Adj. Close"],window=50) #The 50 day rolling mean
df["Rolling std"]=pd.rolling_std(df["Adj. Close"],window=10) #The 10 day rolling standard deviation
```

Since the rolling mean for the first 50 days of the data set has no value (NaN), they are dropped as setting them to any other value (like 0) would have a very drastic influence of sensitive datasets like those of finance. Additionally, the Adj. Closing price of 30 days in the future is the variable which has to be predicted on the input of those features. These features are added by shifting the dataframe up by 30 rows and adding it to the existing dataframe. The rows which do not have any values for this new column are now dropped.

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

new=df.copy()

forecast_out=30 # number of days into the future
df['label']=df[forecast_col].shift(-forecast_out)
df = df[["Adj. Close","HL_PCT", "PCT_change", "Adj. Volume","Rolling std","Rolling Mean 50","Rolling Mean 10","Split Ra
df.dropna(inplace=True)
```

It is now necessary to preprocess the data and spit it into the X and y components for training. The preprocessing is done by making use of the preprocessing.scale() function in the preprocessing library. This ensures that the scale of all the data points remains the same in the X dataset. Other types of processing are not really necessary since this the data is obtained from a repository which contains clean data. Also, some of the data is stored in X_recent in order to use them for data points to validate the data later on. The data points have to be split into training data and testing data which is accomplished by the train_test_split( ) function available in the cross_validation library. The testing size is set to 30 percent of the total data left in X.

```
from sklearn import preprocessing
X=np.array(df.drop(['label'],axis=1))
X_init=X.copy()
#X=X[:,1:] #cooment this line to exclude Adj. Close as a feature
X=preprocessing.scale(X)
X_recent=X[-forecast_out:]
X=X[:-forecast_out]
```

```
from sklearn import cross_validation
df.dropna(inplace= True)
y=np.array(df['label'])
y=y[:-forecast_out]
X_train, X_test,y_train,y_test=cross_validation.train_test_split(X,y,test_size=0.3)
```

3.2 Implementation

The implementation of four different algorithms in will be discussed here in different sections. All the graphs below depict the price of the stock against the days from the end of the queried date. The y-axis denotes the stock price and the x-axis denotes the days elapsed. The implementation is for the date range **'2004-01-17' to '2008-03-8'.**

3.2.1 The Benchmark – Linear Regression

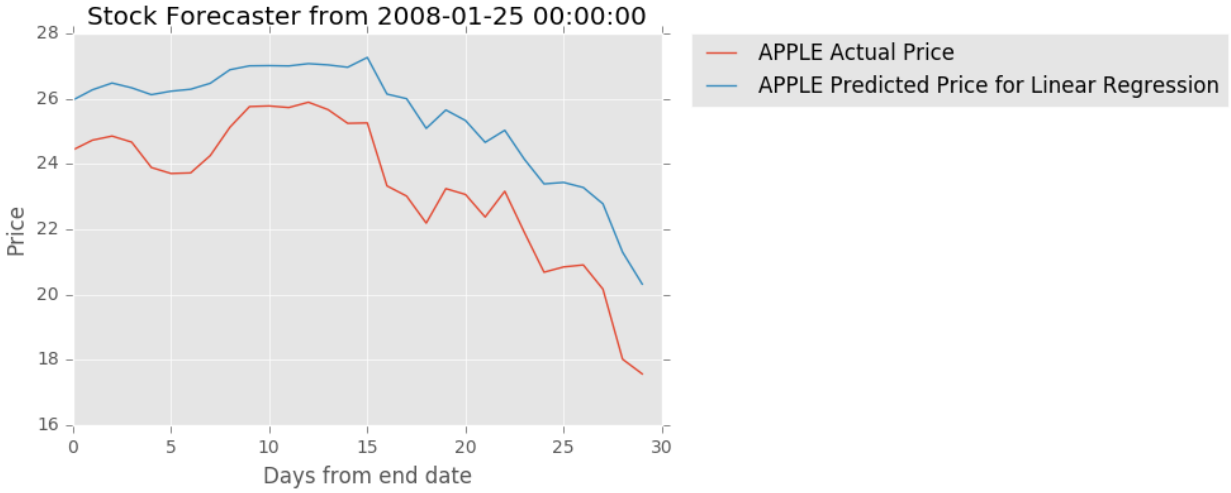

Figure 3.2: The Benchmark

The linear regressor made use of multivariate regression here. Default parameters were used because a very optimized algorithm was not the need. We set n_jobs=-1 and carry out the linear regression with the simple call to the linear regression Class from the sklearn library LinearRegression. The traing data set is used for the .fit() function and the testing dataset it used

for the .score() function for the R2 score .The X_recent dataset is used to forecast the values by employing the .predict() function. The The error used here is the Mean Absolute Error and it is about 9.4% here which will be the benchmark for all the future models.

3.2.2 The Support Vector Regressor(SVR)

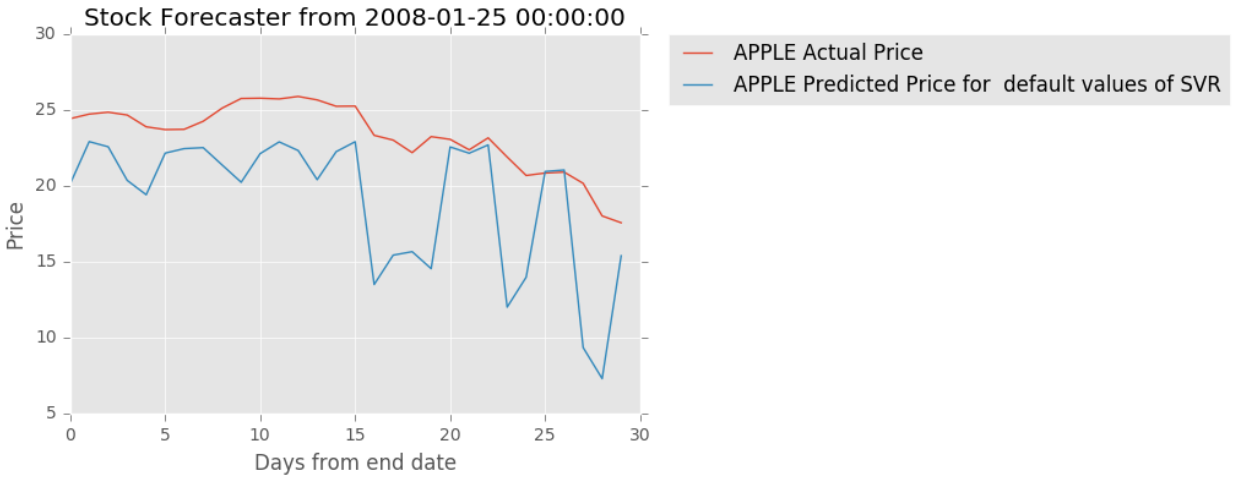

Figure 3.3: The default SVR implementation

It is important to select an appropriate basis function for the algorithm to work properly. Here we will be using the radial basis function. As a result, two parameters namely C and Gamma have to varied. For the initial run we go with the default values of Gamma and C. The values will be refined using gridSearch in the refinement part. The svm.SVR() class is used to initialize the regressor.

3.2.3 The Multilayer Perceptron(MLP)

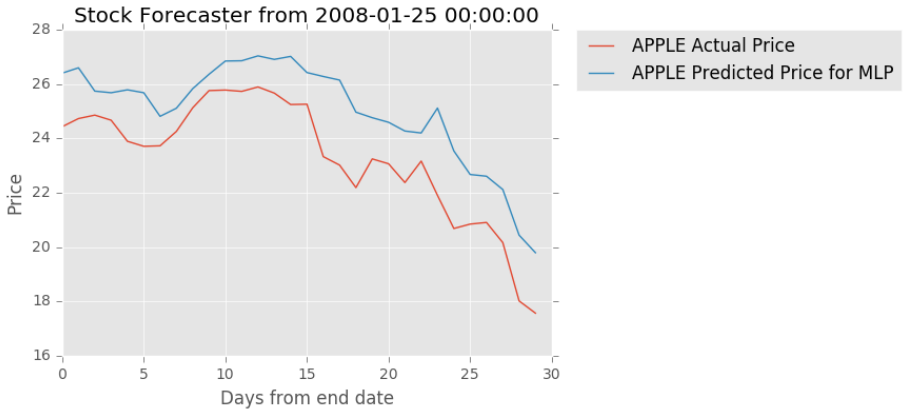

Figure 3.4: The default MLP implementation

The accuracy of the MLP depends on the number of hidden layer and number of neurons in each layer. Each layer helps model a non-linear feature of the model and adds to the complexity. Hence increasing the layer can increase the accuracy. For the initial runs, the default values of hidden_layer_sizes will be used.The model is refined by increasing the number of hidden layers of neurons. Doing so may affect the convergence of the algorithm as more compuations need to be done. This is taken care of by varying the max_iter parameter. The regressor is initialized using neural_network.MLPRegressor().It is necessary to import the neural network library from sklearn for this as this is a relatively new feature.

3.2.4 The Decision Tree Regressor

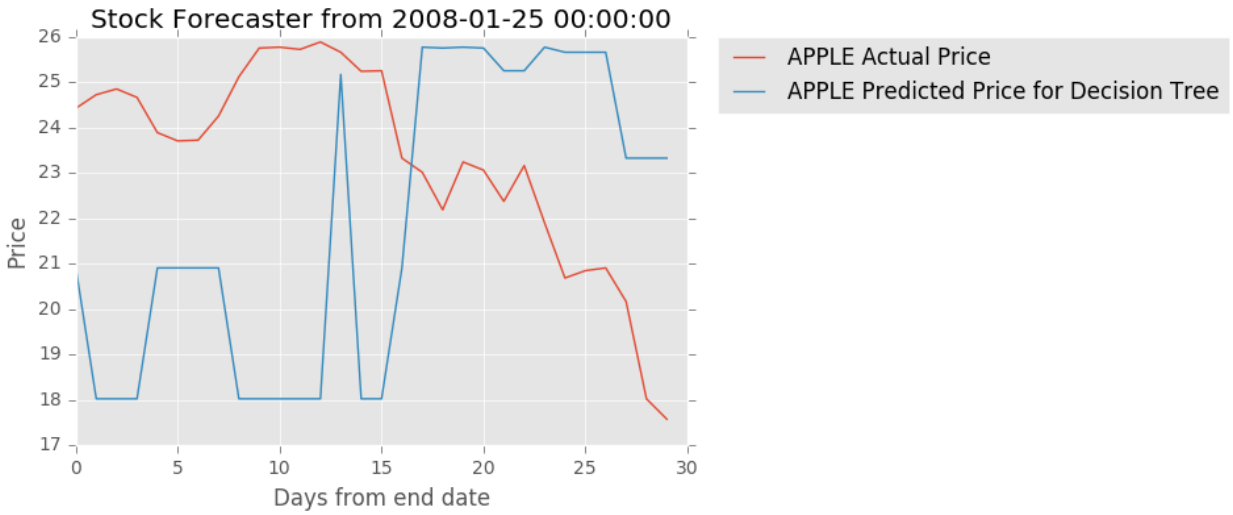

Figure 3.5: The default Decision Tree Regressor implementation

In the decision tree regressor , the complexity increases as the depth of the decision tree increases. This is controlled by varying the max depth parameter. The decision tree regressor is initialized using the DecisionTreeRegressor() class from the sklearn library. Fitting, scoring, predicting are all done just as in the case of the other algorithms used above.

3.3 Refinement

3.3.1 The Support Vector Regressor(SVR)

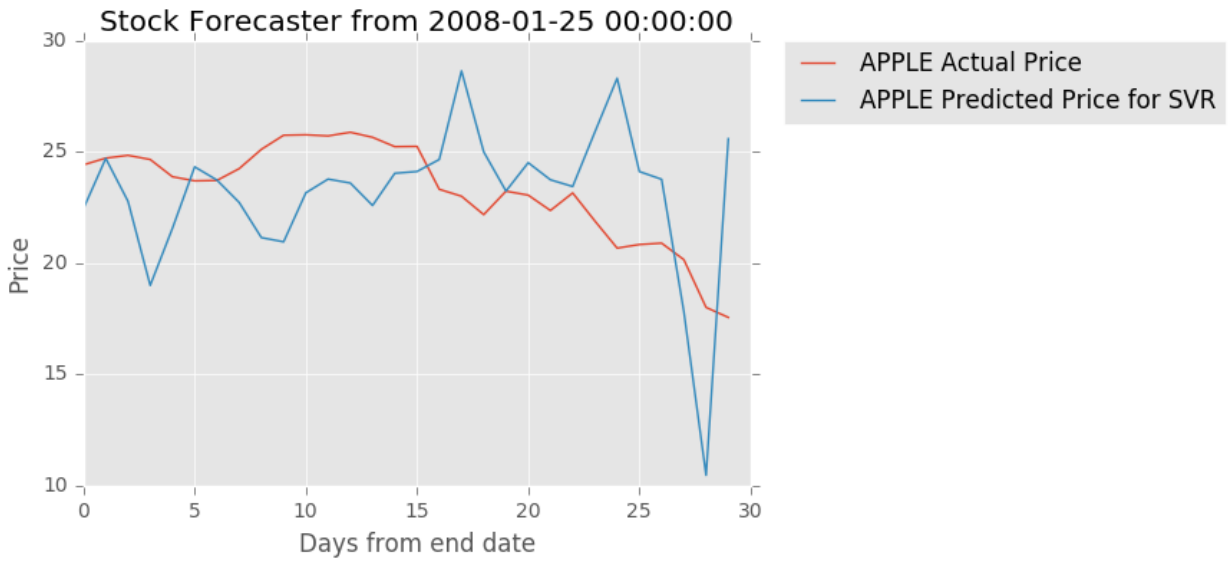

Figure 3.6: The Refinement of SVR implementation

The refining is done here by calling out the grid search function for the SVR algorithm. The parameter C is made to vary with a range of 1 to 1000 and Gamma is made to vary from 0.01 to 0.1. After the grid search the most optimum value is seen to be C=217 and Gamma=0.11

```
from sklearn.grid_search import GridSearchCV
from sklearn.cross_validation import ShuffleSplit
def fit_model(X, y):
    """ Performs grid search over the 'max_depth' parameter for a
        decision tree regressor trained on the input data [X, y]. """

    # Create cross-validation sets from the training data
    cv_sets = ShuffleSplit(X.shape[0], n_iter = 10, test_size = 0.20, random_state = 0)

    # TODO: Create a decision tree regressor object
    regressor = svm.SVR()
    gamma=[]
    for i in range(1,20):
        gamma.append(0.01+0.05*i)
    C=[]
    for i in range(1,500):
        C.append(1+2*i)


    # TODO: Create a dictionary for the parameter 'C' and 'Gamma' to vary w
    params = {'C': C, 'gamma': gamma}

    # TODO: Create the grid search object
    grid = GridSearchCV(regressor, params, cv=cv_sets)

    # Fit the grid search object to the data to compute the optimal model
    grid = grid.fit(X, y)

    # Return the optimal model after fitting the data
    print('Best score for data1:', grid.best_score_)
    print('Best C:',grid.best_estimator_.C)
    print('Best Gamma:',grid.best_estimator_.gamma)
    return grid.best_estimator_
```

3.3.2 The Multilayer Perceptron(MLP)

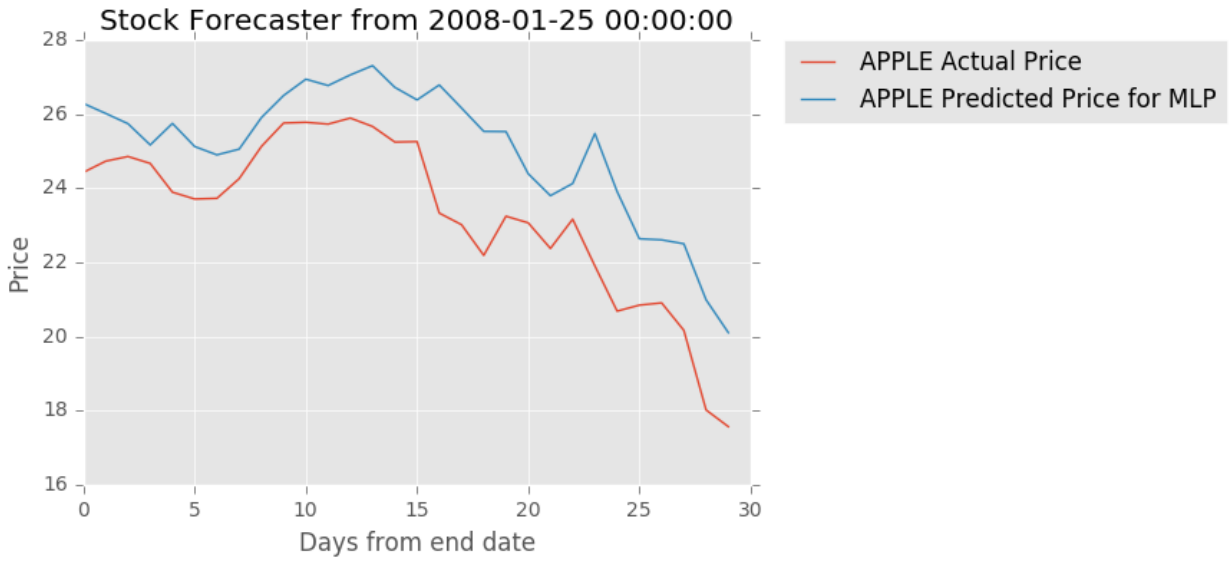

Figure 3.7: The refinement of MLP implementation

The accuracy of the MLP is refined by varying the hidden_layer_sizes parameter. After a lot of trial and error based approaches, the configuration of 100 hidden layers and 200 neurons was found to work in the case of this project.

3.3.3 The Decision Tree Regressor

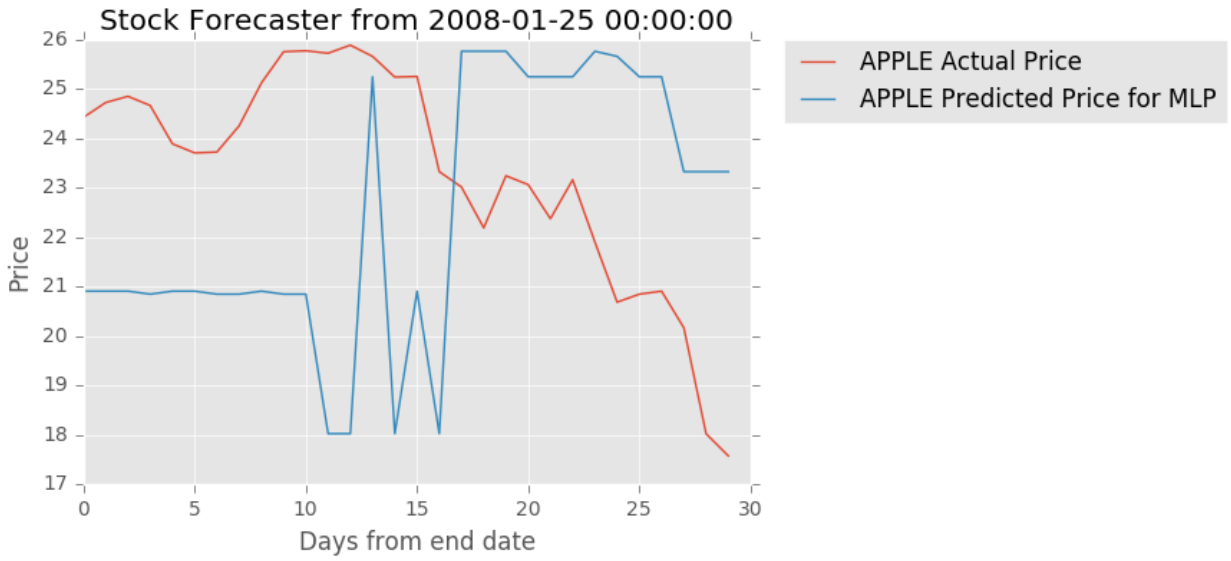

Figure 3.8: The refinement of the Decision Tree Regressor implementation

As in the case of SVR grid search was used to refine the regressor here but on the max_depth parameter. The range was kept from 1 to 11. The optimal max depth was found to be 8 here.

3.4 Prediction for other Date Ranges

For the other two date ranges the same steps were run by keeping the parameters of the SVR, MLP, Decision Tree Regression and the Linear Regression models the same. The performance of these algorithms are shown below for a few examples. The graphs of the other algorithms have been plotted in the jupyter notebook attached along with this project and have been avoided here.

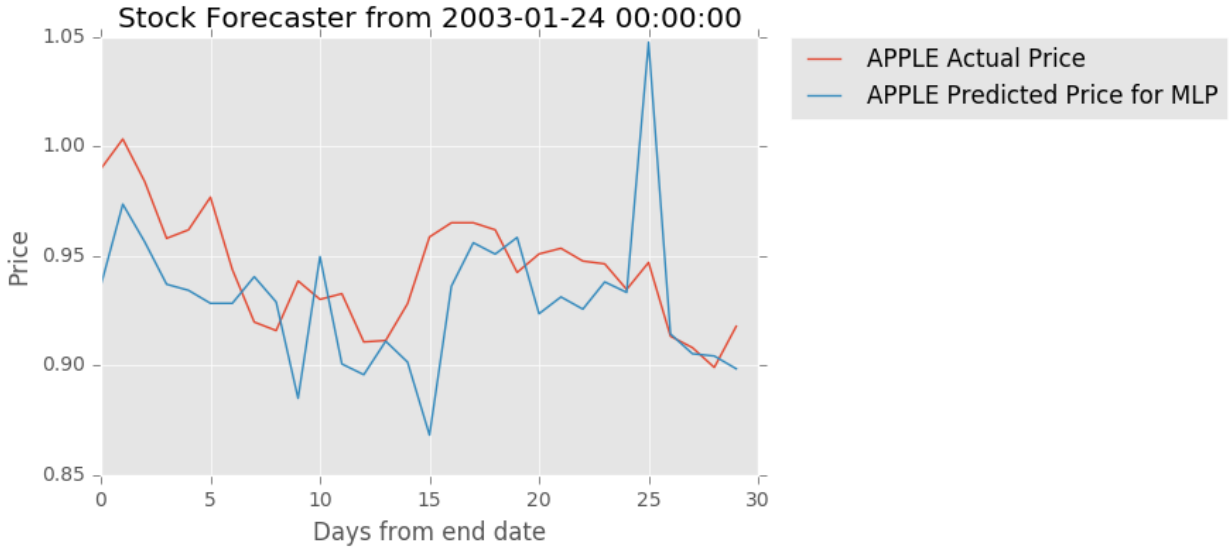

Figure 3.9: Modelled MLP for the date range '1999-05-9' to '2003-03-7'

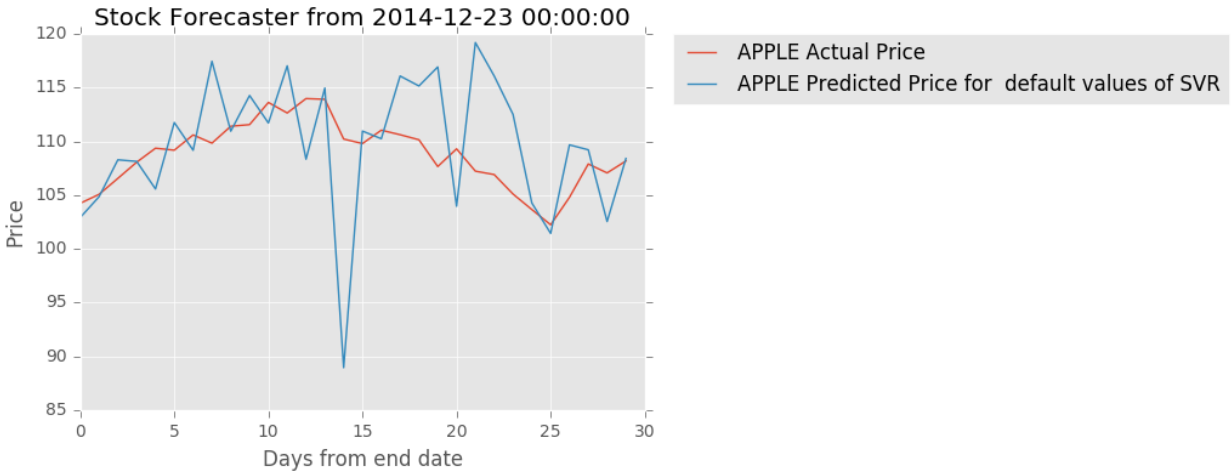

Figure 3.10: Modelled SVR for the date range '2009-04-4' to '2015-02-5'

Section IV: Results

Table 4.1: For the date range '2004-01-17' to '2008-03-8'

Algorithm	R2 Score	Error%
Linear Regression	0.941	9.41
SVR	0.945	12.68
MLP	0.935	7.89
Decision Tree Regressor	0.987	17.638

Table 4.2: For the date range '1999-05-9' to '2003-03-7'

Algorithm	R2 Score	Error%
Linear Regression	0.675	11.4
SVR	0.703	9.99
MLP	0.703	2.69
Decision Tree Regressor	0.815	8.82

Table 4.3: For the date range '2009-04-4' to '2015-02-5'

Algorithm	R2 Score	Error%
Linear Regression	0.95	2.17

Algorithm	R2 Score	Error%
SVR	0.95	3.8
MLP	0.828	4.62
Decision Tree Regressor	0.985	2.98

The above tables show us the accuracies and errors we obtain for the same machine learning algorithms for different date ranges.

The the accuracies obtained during training can be considered a metric for model evaluation. Higher accuracies can mean that we have a good model, however care should be taken to see that there isn't a case of overfit as can be seen in the decision tree regressor. We shall discuss the parameters and the significance they play on the model.

In the MLP, the parameter *hidden_layer_sizes* controls the number of neurons and hidden layers in the neural network. This helps to increase the complexity of the model and increase accuracies as it adds more nodes into the network.

The decision tree regressor needs branches to vary the accuracy of the model. By increasing the *max_depth* parameter it is possible to achieve this. However, care should be taken as to not to overfit on the model. This is the reason why we use cross validation and grid search to find the most optimum depth.

The SVR algorithm uses a decision boundary to fit a regressor. Here we will be using the radial basis function to fit a boundary. The parameters to change thus will be *C* and *Gamma*. *C* is the cost for misclassification and *Gamm*a is the tolerance form the boundary region. Again, like the decision tree regressor this is found by using a grid search and cross validation for *C* and *Gamma* parameters.

We can see that MLP model we obtained performs better than the rest of the models and can usually outperform the model obtained by linear regression. It can be seen that SVR models also perform well but they give out lesser accuracies during training. It is also a common belief in the finance circles that SVR models are good for financial analytics. The findings in this project further

reinforces this belief. The models have been tested with different date ranges over different periods of time , hence the findings can be considered more robust.

Section V: Conclusion

5.1 Freeform Visualizations

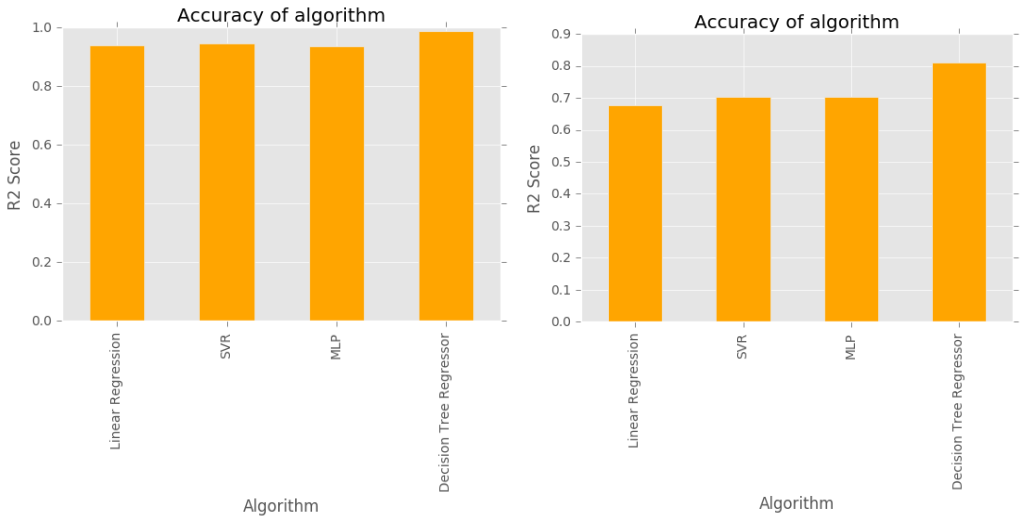

Figure 5.1 & 5.2: R2 Scores for '2004-01-17' to '2008-03-8' and '1999-05-9' to '2003-03-7'

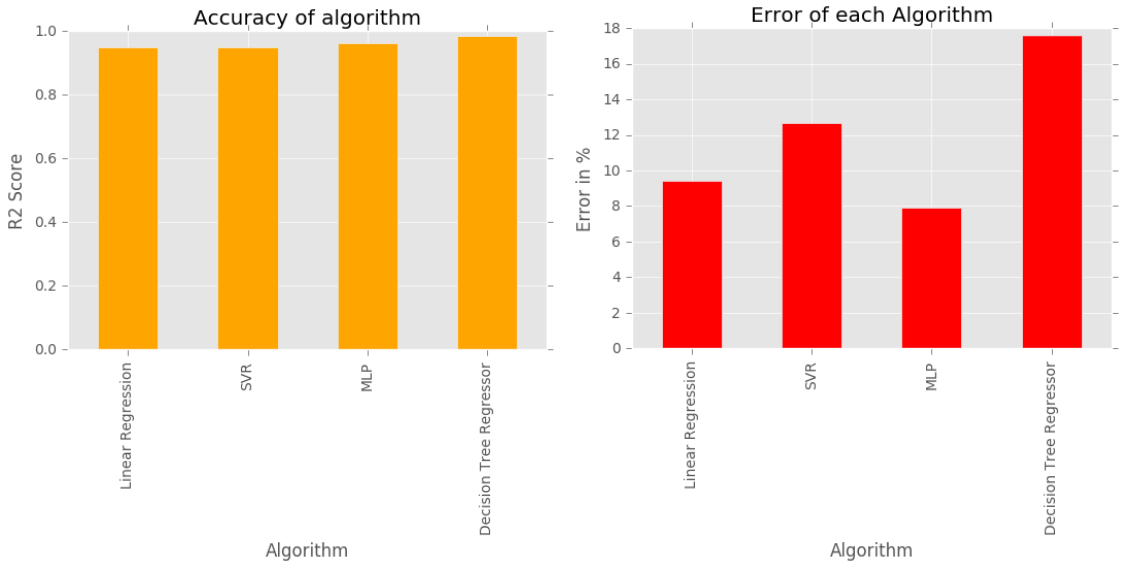

Figure 5.3 & 5.4: R2 Score for 2009-04-4' to '2015-02-5'to '2008-03-8' and Error % for '2004-01-17' to '2008-03-8'

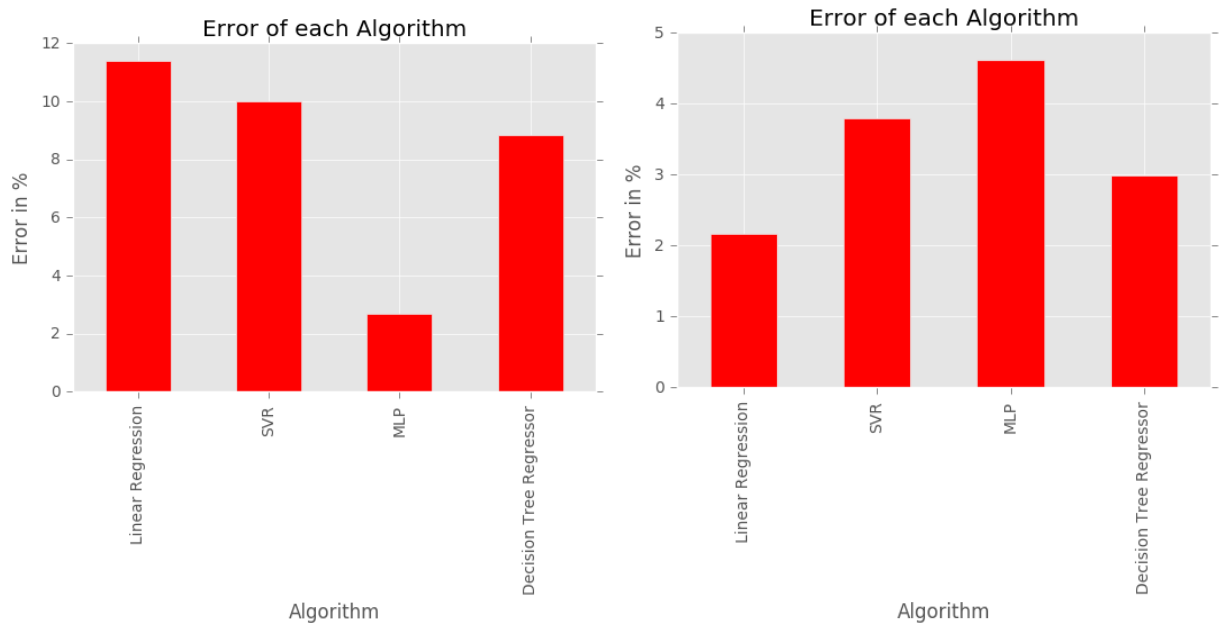

Figure 5.5 & 5.6: R2 Scores for '2004-01-17' to '2008-03-8' and '1999-05-9' to '2003-03-7'

From the above visualizations it can be seen that the SVR and the MLP algorithms gave high R2 scores and lower errors than our benchmark linear regressor in 2 out of the 3 cases. Even in the 3rd case, the error was pretty much on par with the benchmark model. All three date ranges are random in different periods of time. Hence the performance of the algorithms is pretty robust. Hence for AAPL stock, we can go with an SVR model of radial basis function with C=217 and gamma=0.11.

5.2 Reflection

The project started with the objective of trying to predict the stock price of the Apple stock for any queried date range. The trends of the stock were studied and it was seen that even a linear regression model might perform good enough. The reasons for the fluctuation of the stock were also explored and justified. We then proceeded to build our machine learning model. The date range of '2004-01-17' to '2008-03-8' was selected for model building. The required features were built and aggregated along with creating a vector for the target data. The data was then preprocessed to remove NaN values and make all the features of the same scale. We then proceeded to set the Linear Regression as the benchmark model for our project. Algorithms like Decision Trees, MLP's and SVR's were used for prediction. They were further refined using grid search and trial and error techniques for better performance. The model was then implemented on two different date ranges. It was found that MLP's performed the best followed by SVR's. Decision trees performed even worse than the benchmark.

From the project it could be seen that decision trees are not a very good choice for predicting stochastic data as they tend to overfit and give unsatisfactory results. This was the case for all the three cases. It was also seen that multivariate linear regression was a pretty good indicator of the performance of the stock.

Neural Networks also gave encouraging results. Probably more convoluted neural networks or deep learning approaches would give us more accuracy in prediction. It could also be seen that SVR's perform well in these cases.

Even though the errors obtained are in the ranges of 8%-15% which would mean somethings as less as 1$ a share, when dealing in large volumes (millions of units) this amounts to a significant number and can cause huge losses to the trader. Hence, even such errors should not encourage an ML specialist to jump into trading. However financial markets are good subject to understand and model stochastic systems. Finding benchmark models is difficult in this domain and one has to often rely on primitive models as benchmarks to compare models against.

5.3 Improvements

For the SVR model, more combinations of C and gamma were needed to try and find a more optimum model. This could have been achieved by using a more powerful computational unit. However, this couldn't be done at this stage due to lack of resources and knowledge (this project was intended as beginner's project and integrating GPUs requires some advanced knowledge). The same could be done for all parameters of different algorithms. Another alternative would be to use something like AWS (Amazon Web Services) and pickle the model obtained.

In the case of MLP, a grid search could have been used to get the optimum value of hidden_layer_sizes. However, the gridSearch threw an error even when all possible combinations of the tuple was fed as a parameter to the hidden_layer_sizes. Employing a grid search here would have been a better alternative than trial and error used in the project.

Also with knowledge of Deep Neural Networks, it would be easier to design a more robust and accurate model in the future as primitive neural networks were giving reasonably good performance here.