STOCK MARKET PREDICTION

Sumbitted By

Maneesh M

Register No: 961721106001

PHASE 5

PROJECT

What is Stock Price Prediction?

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.

1 Introduction

1.1 Objective

In the past decades, there is an increasing interest in predicting markets among economists, policymakers, academics and market makers. The objective of the proposed work is to study and improve the supervised learning algorithms to predict the stock price.

Technical Objective

The technical objectives will be implemented in R. The system must be able to access a list of historical prices. It must calculate the estimated price of stock based on the historical data. It must also provide an instantaneous visualization of the market index.

Experimental Objective

Two versions of prediction system will be implemented; one using Decision trees and other using Support Vector Machines. The experimental objective will be to compare the forecasting ability of SVM with Decision Trees. We will test and evaluate both the systems with same test data to find their prediction accuracy.

1.2 What is the problem?

Investors are familiar with the saying, "buy low, sell high" but this does not provide enough context to make proper investment decisions. Before an investor invests in any stock, he needs to be aware how the stock market behaves. Investing in a good stock but at a bad time can have disastrous results, while investment in a mediocre stock at the right time can bear profits. Financial investors of today are facing this problem of trading as they do not properly understand as to which stocks to buy or which stocks to sell in order to get optimum profits. Predicting long term value of the stock is relatively easy than predicting on day-to-day basis as the stocks fluctuate rapidly every hour based on world events.

1.3 Why this is a project related to this class?

The solution to this problem demands the use of tools and technologies related to the field of data mining, pattern recognition, machine learning and data prediction. The application will predict the stock prices for the next trading day. The requirements and the functionality of this application correlates it to the class.

1.4 Why other approach is no good?

The other approach makes use of Neural Networks [1]. Neural networks have the following drawbacks:

1. *Slow Convergence Rate*

The neural network takes a lot of time to train.

2. *Local Minima and Maxima*

Neural Networks are based on gradient descent method to find the local extreme value and they have a tendency to get stuck on the local minima and maxima and therefore it is difficult to find global minima and maxima. In the approach previously discussed, the author has used pattern matching to overcome this problem.

1.5 WHY YOU THINK YOUR APPROACH IS BETTER?

The proposed approach makes use of Support Vector Machines (SVM) and Decision Trees.

The benefit of using Decision trees over Neural Network are:

1. They are easy to program.
2. The top nodes in the tree will give the information about what data affects the prediction.
3. Trees are interpretable and provide visual representation of data.
4. Performs faster than Neural Networks after training.

The benefits of using SVM over neural networks are [2]:

1. SVM has strong founding theory.
2. Global optimum guaranteed.
3. Requires less memory to store the predictive model.
4. Yield more readable results and a geometrical interpretation.

1.6 STATEMENT OF THE PROBLEM

Financial analysts investing in stock market usually are not aware of the stock market behavior. They are facing the problem of trading as they do not properly understand which stocks to buy or which stocks to sell in order to get more profits. In today's world, all the information pertaining to stock market is available. Analyzing all this information individually or manually is tremendously difficult. As such, automation of the process is required. This is where Data mining techniques help.

Understanding that analysis of numerical time series gives close results, intelligent investors use machine learning techniques in predicting the stock market behavior. This will allow financial analysts to foresee the behavior of the stock that they are interested in and thus act accordingly.

The input to our system will be historical data from Yahoo Finance. Appropriate data would be applied to find the stock price trends. Hence the prediction model will notify the up or down of the stock price movement for the next trading day and investors can act upon it so as to maximize their chances of gaining a profit. The entire system would be implemented in

Python/Java and R language using open source libraries. Hence it will effectively be a zero cost system.

1.7 Area or scope of investigation

This project requires investigation in the following areas:

Stock Market [1]

Investigating trends in stock market and factors affecting the stock prices.

Data mining techniques

Investigating the available tools and techniques for data mining and then selecting those that are best fit to solve the problem.

2 Theoretical Bases and Literature Review

2.1 Definition of the problem

Stock market attracts thousands of investors' hearts from all around the world. The risk and profit of it has great charm and every investor wants to book profit from that. People use various methods to predict market volatility, such as K-line diagram analysis method, Point Data Diagram, Moving Average Convergence Divergence, even coin tossing, fortune telling, and so on.

Now, all the financial data is stored digitally and is easily accessible. Availability of this huge amount of financial data in digital media creates appropriate conditions for a data mining research. The important problem in this area is to make effective use of the available data.

2.2 Theoretical background of the problem

Stock market is highly volatile. At the most fundamental level, it is said that supply and demand in the market determines stock price. But, it does not follow any fixed pattern and is also affected by a large number of highly varying factors

The investors on the Wall Street are split in two largest factions of adherents; those who believe the market cannot be predicted and those who believe the market can be beat.

2.3 Related research to solve the problem

Recently, a lot of interesting work has been done in the area of applying Machine Learning Algorithms for analyzing price patterns and predicting stock price. Most stock traders nowadays depend on Intelligent Trading Systems which help them in predicting prices based on various situations and conditions.

Recent researches uses input data from various sources and multiple forms. Some systems use historical stock data, some use financial news articles, some use expert reviews while some use a hybrid system which takes multiple inputs to predict the market.

Also, a wide range of machine learning algorithms are available that can be used to design the system. These systems have different approaches to solve the problem. Some systems perform mathematical analysis on historic data for prediction while some perform sentiment analysis on financial news articles and expert reviews for prediction.

However, because of the volatility of the stock market, no system has a perfect or accurate prediction.

2.4 Advantage/disadvantage of those research

Advantages

The research helps a lot of new investors in deciding when to buy or sell a particular stock. It also helps in understanding the sentiments of experienced financial analysts and financial news data more quickly than doing the same manually.

Disadvantages

Current research makes use of neural networks which have the drawback of slow convergence rate and local optimum. To overcome the problem of slow convergence the author uses a pattern matching algorithm to select the input data to train the network which is an increased overhead [2].

2.5 Our solution to solve this problem

We will implement the system using two different machine learning techniques. One using Support Vector Machines and the second implementation using Decision Trees.

We will train both the systems using 75% of 2 years of historic data and then test our model to check which systems yields better output using the remaining 25% of historic data.

2.6 Why our solution is different from others?

Our solution uses a different algorithm and different technique to perform the prediction. We are using Support Vector Machines with C type classification and Radial Basis Function(RBF) kernel.

2.7 Why our solution is better?

It uses SVM and Decision Trees which have better performance than Neural Network. Moreover, using SVM will takes away the burden of matching the present price pattern with historic patterns and also SVM trains faster than a NN and has a lower computational cost.

Also, other solution uses the financial data as it is without using any indicators, whereas our solution uses many indicators such as EMA, RSI, MACD, SMI, CCI, ROC, CMO, WPR and ADX to get better results.

3 HYPOTHESIS

3.1 POSITIVE/NEGATIVE HYPOTHESIS

The performance of the stock depends on various factors. If indicator variables are chosen properly than we can obtain satisfactory prediction results.

4 METHODOLOGY

4.1 HOW TO COLLECT INPUT DATA?

Input data is taken from Yahoo Finance using following steps:

1. For our project, we are considering S&P 500 Companies. The list of companies in S&P 500 can be obtained from Wikipedia [3].
2. Use stock's ticker symbol from step a to get data from Yahoo Finance.
3. System will take last 2 years' stock data of the company using *quantmod* package in R.
4. Further we divide the data into two parts, training data and testing data, where 75% of the data will be used for training and 25% of the data will be used for testing.

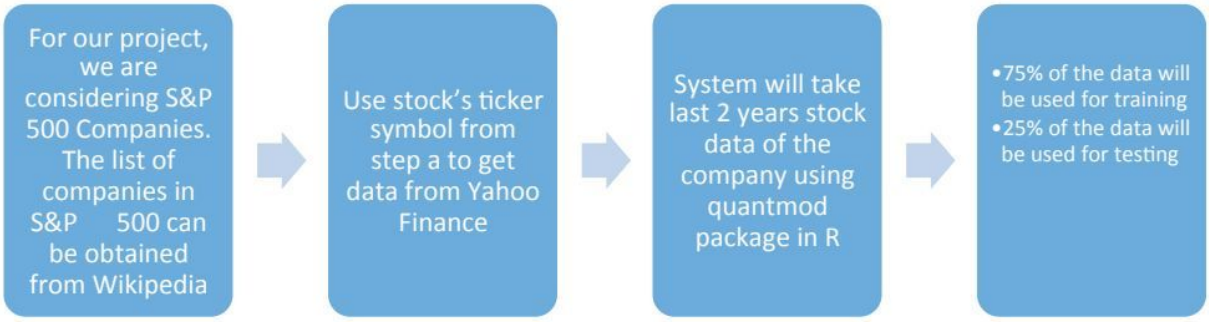

Figure 1 : Steps to collect Input Data

4.2 HOW TO SOLVE THE PROBLEM?

We will solve the problem using below supervised learning techniques to build our model -

- Decision Trees with Technical Indicators.
- Support Vector Machine with Technical Indicators.

To solve the problem, we will follow below steps -

1. Fetch the data of a stock from Yahoo Finance of last 2 years.
2. Calculate the values of technical indicators RSI, EMA, MACD, SMI, etc.
3. Train the model using these indicators and training data.
4. Test the model using testing data.
5. Evaluate our system using various evaluation techniques.

Details of Technical Indicators -

- **Relative Strength Index - RSI**
  The relative strength index (RSI) is a technical momentum indicator that compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an asset. It is calculated using the following formula:

  $$RSI = 100 - 100 / (1 + RS^*)$$

  Where, RS = Average of x days' up closes / Average of x days' down closes.

- **Exponential Moving Average - EMA**
  An exponential moving average (EMA) is a type of moving average that is similar to a simple moving average, except that more weight is given to the latest data. The exponential moving average is also known as "exponentially weighted moving average".

- **Moving Average Convergence Divergence - MACD**
  Moving average convergence divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of prices. The MACD is calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA. A nine-day EMA of the MACD, called the "signal line", is then plotted on top of the MACD, functioning as a trigger for buy and sell signals.

- **Stochastic Momentum Index - SMI**
  The Stochastic oscillator is a technical momentum indicator that compares a security's closing price to its price range over a given time period. The oscillator's sensitivity to market movements can be reduced by adjusting the time period or by taking a moving average of the result. This indicator is calculated with the following formula:
  $\%K = 100 [(C - L14) / (H14 - L14)]$
  Where C = the most recent closing price
  L14 = the low of the 14 previous trading sessions
  H14 = the highest price traded during the same 14-day period.
  %D = 3-period moving average of %K

- **Commodity Channel Index - CCI**
  An oscillator used in technical analysis to help determine when an investment vehicle has been overbought and oversold. The Commodity Channel Index, first developed by Donald Lambert, quantifies the relationship between the asset's price, a moving average (MA) of the asset's price, and normal deviations (D) from that average. It is computed with the following formula:

  $$CCI = \frac{price - MA}{0.015 \times D}$$

- **Collateralized Mortgage Obligation - CMO**
  A collateralized mortgage obligation (CMO) is a type of mortgage-backed security in which principal repayments are organized according to their maturities and into different classes based on risk. A collateralized mortgage obligation is a special purpose entity that receives the mortgage repayments and owns the mortgages it receives cash flows from (called a pool). The mortgages serve as collateral, and are organized into classes based on their risk profile. Income received from the mortgages is passed to investors based on a predetermined set of rules, and investors receive money based on the specific slice of mortgages invested in (called a tranche).

- **Rate of Change - ROC**
  The price rate of change (ROC) is a technical indicator that measures the percentage change between the most recent price and the price "n" periods in the past. It is calculated by using the following formula:

  (Closing Price Today - Closing Price "n" Periods Ago) / Closing Price "n" Periods Ago

  ROC is classed as a price momentum indicator or a velocity indicator because it measures the rate of change or the strength of momentum of change.

- **Average Directional Index - ADX**
  The average directional index (ADX) is an indicator used in technical analysis as an objective value for the strength of trend. ADX is non-directional so it will quantify a trend's strength regardless of whether it is up or down. ADX is usually plotted in a chart window along with two lines known as the DMI (Directional Movement Indicators). ADX is derived from the relationship of the DMI lines.

- **Williams % R - WPR**
  Williams %R, in technical analysis, is a momentum indicator measuring overbought and oversold levels, similar to a stochastic oscillator. It was developed by Larry Williams and compares a stock's close to the high-low range over a certain period of time, usually 14 days.

4.2.1 ALGORITHM DESIGN

Using Decision Trees

Step 1: Get the required data [Date, Open, High, Low, Close, Volume, Adjusted]

Step 2: Calculate all the indicator required indicators

1. RSI (Relative Strength Index): Assuming a 5-period RSI, a zero RSI value means prices moved lower all 5-periods. There were no gains to measure. RSI is 100 when the Average Loss equals zero. This means prices moved higher all 5-periods. There were no losses to measure.

2. EMA Crossover: It is calculating the movement of EMA-ShortPeriod(Opening price) - EMA-LargePeriod(EMA over 5 days).
3. CCI
4. ROC
5. CMO
6. WPR
7. ADX

Step 3: Calculate the prediction variable (Up/Down)

Step 4: Build the decision tree from the data calculated in above steps.

Step 5: Prune the tree to remove any overfitting of the data.

Step 6: Give test data to the trees.

Using Support Vector Machines

We will be using C-classification Support Vector Machine with RBF Kernel.

Step 1: Read the required data [Date, Open, High, Low, Close, Volume, Adjusted]

Step 2: Calculate all the required indicators -

1. RSI
2. EMA Crossover
3. CCI
4. ROC
5. CMO
6. WPR
7. ADX

Step 3: Calculate the prediction variable (Up/Down)

Step 4: Provide the data from above steps to train SVM (RBF, C = 1, gamma = ½)

Step 5: Provide test data and display the results

Step 6: Compare the output from step 5 and 6 and show the observations.

4.2.2 LANGUAGE USED

R (Programming Language) [4] –

R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

Packages used -

- quantmod
- lubridate
- e1071
- rpart
- ROCR

4.2.3 TOOLS USED

RStudio Desktop [5] –

RStudio is a free and open source integrated development environment (IDE) for R, a programming language for statistical computing and graphics. RStudio was founded by JJ Allaire, creator of the programming language ColdFusion. Hadley Wickham is the Chief Scientist at RStudio.

RStudio is available in two editions: RStudio Desktop, where the program is run locally as a regular desktop application; and RStudio Server, which allows accessing RStudio using a web browser while it is running on a remote Linux server. Prepackaged distributions of RStudio Desktop are available for Microsoft Windows, Mac OS X, and Linux.

RStudio is available in open source and commercial editions and runs on the desktop (Windows, Mac, and Linux) or in a browser connected to RStudio Server or RStudio Server Pro (Debian/Ubuntu, RedHat/CentOS, and SUSE Linux).

RStudio is written in the C++ programming language and uses the Qt framework for its graphical user interface.

4.3 HOW TO GENERATE OUTPUT?

Perform following steps to generate output:

a) Using stock's ticker symbol, get the last 3 years' data.
b) Provide the data to the system.
c) Train the system.
d) System will predict the output.

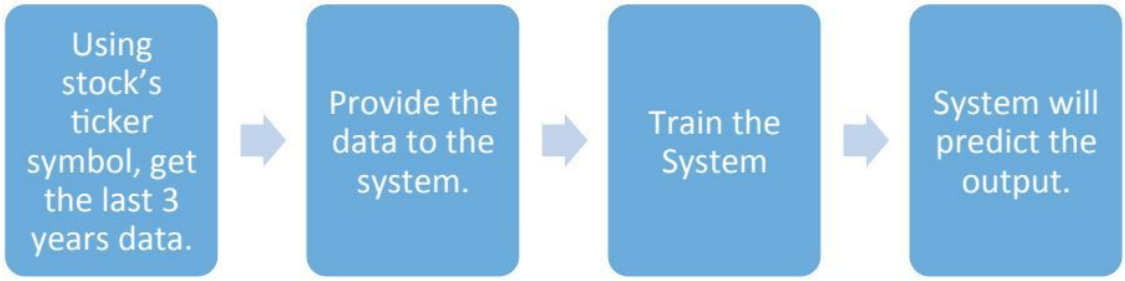

Figure 2 : Steps to generate output

4.4 How to prove correctness?

Once we acquire a dataset, we intend to divide it into two subsets:

Training set is a subset of the dataset used to build predictive models.

Test set or unseen examples is a subset of the dataset to assess the likely future performance of a model. If a model fit to the training set much better than it fits the test set, over fitting is probably the cause.

Confusion Matrix

A confusion matrix shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes (target value) in the data. The matrix is NxN, where N is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix. The following table displays a 2x2 confusion matrix for two classes (Positive and Negative).

		Actual	
		Positive	**Negative**
Prediction	**Positive**	True Positive	False Positive
	Negative	False Negative	True Negative
	Accuracy (ACC) = (Σ True positive + Σ True negative) / Σ Total population	True positive rate (TPR) = Σ True positive / Σ Condition positive	False positive rate (FPR) = Σ False positive / Σ Condition negative

Table 1: Confusion Matrix

Accuracy: The proportion of the total number of predictions that were correct.
Positive Predictive Value or Precision: the proportion of positive cases that were correctly identified.

Negative Predictive Value: the proportion of negative cases that were correctly identified.
Sensitivity or Recall: the proportion of actual positive cases which are correctly identified.
Specificity: The proportion of actual negative cases which are correctly identified.

ROC Chart

The ROC chart is similar to the gain or lift charts in that they provide a means of comparison between classification models. The ROC chart shows false positive rate (1-specificity) on X-axis, the probability of target=1 when its true value is 0, against true positive rate (sensitivity) on Y-axis, the probability of target=1 when its true value is 1. Ideally, the curve will climb quickly toward the top-left meaning the model correctly predicted the cases. The diagonal red line is for a random model (ROC101).

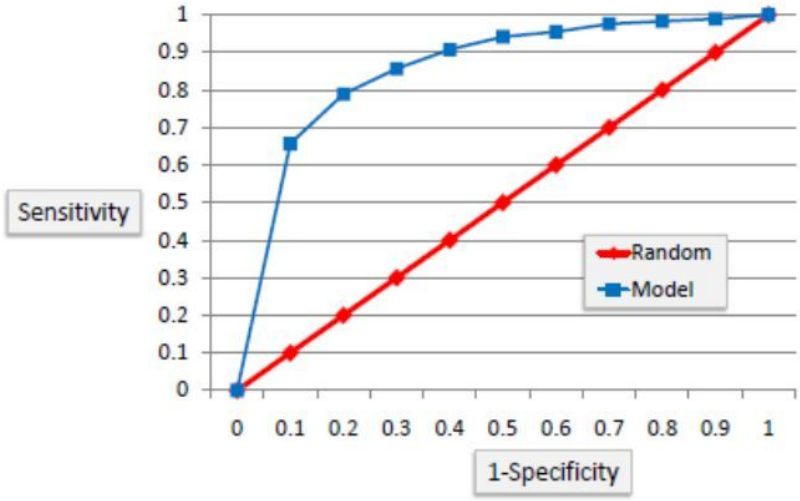

Figure 3: Receiver Operating Curve

Area Under the Curve (AUC)

Area under ROC curve is often used as a measure of quality of the classification models. A random classifier has an area under the curve of 0.5, while AUC for a perfect classifier is equal to 1. In practice, most of the classification models have an AUC between 0.5 and 1.

An area under the ROC curve of 0.8, for example, means that a randomly selected case from the group with the target equals 1 has a score larger than that for a randomly chosen case from the group with the target equals 0 in 80% of the time. When a classifier cannot distinguish between the two groups, the area will be equal to 0.5 (the ROC curve will coincide with the diagonal). When there is a perfect separation of the two groups, i.e., no overlapping of the distributions, the area under the ROC curve reaches to 1 (the ROC curve will reach the upper left corner of the plot).

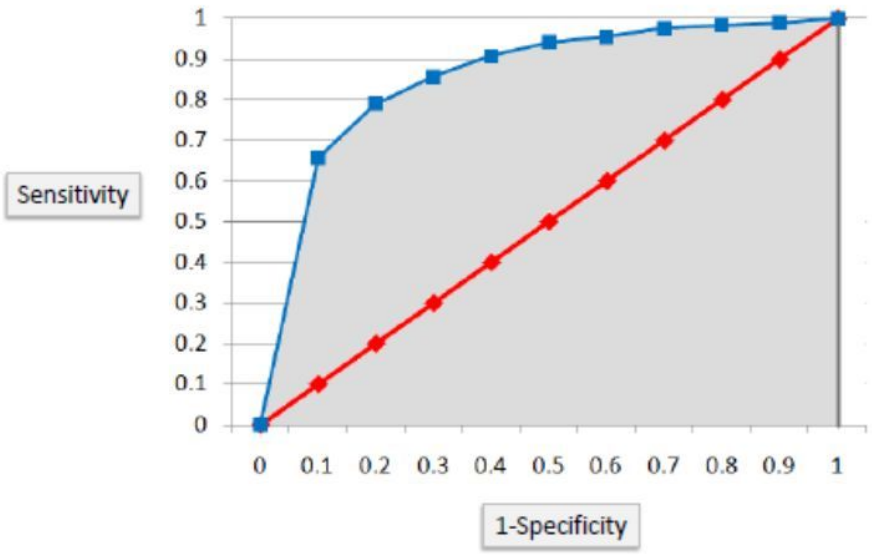

Figure 4: Area Under Curve

RMS Error

The regression line predicts the average y value associated with a given x value. Note that is also necessary to get a measure of the spread of the y values around that average. To do this, we use the root-mean-square error (RMS error).

To construct the RMS error, we first need to determine the residuals. Residuals are the difference between the actual values and the predicted values. We denoted them by *(p - a)*, where *a* is the observed value for the *i*th observation and *p* is the predicted value.

They can be positive or negative as the predicted value under or overestimates the actual value. Squaring the residuals, averaging the squares, and taking the square root gives us the RMS error. You then use the RMS error as a measure of the spread of the y values about the predicted y value.

$$RMS\ Error = \sqrt{\frac{\sum_{i=1}^{n}(p-a)^2}{n}}$$

5 Methodology

5.1 Code

5.1.1 Decision Tree Implementation Code

```
library(quantmod)

library(lubridate)
library(e1071)
library(rpart)
library(rpart.plot)
library(ROCR)

options(warn = -1)
tryCatch({
  print('--------------------------------------------------------------------------------')
  SYM <- 'FB'
  print('--------------------------------------------------------------------------------')
  print(paste('Predicting the output for', SYM, sep = ' '))
  trainPerc <- 0.75
  #Percent of data to be used as Training Data and remaining will be used as Testing data
  date <- as.Date(Sys.Date() - 1)
  endDate <- date#as.Date("2016-01-01")
  d <- as.POSIXlt(endDate)
  d$year <- d$year - 2
  #To take last 2 years of data
  startDate <- as.Date(d)
  STOCK <- getSymbols(
    SYM,
    env = NULL,
    src = "yahoo",
    from = startDate,
    to = endDate
  )
  RSI <- RSI(Op(STOCK), n = 3)
  #Calculate a 3-period relative strength index (RSI) off the open price
  EMA <- EMA(Op(STOCK), n = 5)
  #Calculate a 5-period exponential moving average (EMA)
  EMAcross <- Op(STOCK) - EMA
  #Let us explore the difference between the open price and our 5-period EMA
  MACD <- MACD(Op(STOCK),
               fast = 12,
               slow = 26,
               signal = 9)
  #Calculate a MACD with standard parameters
  MACD <- MACD[, 2]
```

```r
  #Grab just the signal line to use as our indicator.
  SMI <- SMI(
    Op(STOCK),
    n = 13,
    slow = 25,
    fast = 2,
    signal = 9
  )
  #Stochastic Oscillator with standard parameters
  SMI <- SMI[, 1]
  #Grab just the oscillator to use as our indicator
  WPR <- WPR(Cl(STOCK), n = 14)
  WPR <- WPR[, 1]
  #Williams %R with standard parameters
  ADX <- ADX(STOCK, n = 14)
  ADX <- ADX[, 1]
  #Average Directional Index with standard parameters
  CCI <- CCI(Cl(STOCK), n = 14)
  CCI <- CCI[, 1]
  #Commodity Channel Index with standard parameters
  CMO <- CMO(Cl(STOCK), n = 14)
  CMO <- CMO[, 1]
  #Collateralized Mortgage Obligation with standard parameters
  ROC <- ROC(Cl(STOCK), n = 2)
  ROC <- ROC[, 1]
  #Price Rate Of Change with standard parameters
  PriceChange <- Cl(STOCK) - Op(STOCK)
  #Calculate the difference between the close price and open price
  Class <- ifelse(PriceChange > 0, "UP", "DOWN")
  #Create a binary classification variable, the variable we are trying to pre
dict.
  DataSet <-
    data.frame(Class, RSI, EMAcross, MACD, SMI, WPR, ADX, CCI, CMO, ROC)
  #Create our data set
  colnames(DataSet) <-
    c("Class",
      "RSI",
      "EMAcross",
      "MACD",
      "SMI",
      "WPR",
      "ADX",
      "CCI",
      "CMO",
      "ROC")
  #Name the columns
  #DataSet <- DataSet[-c(1:33),]
  #Get rid of the data where the indicators are being calculated
  TrainingSet <- DataSet[1:floor(nrow(DataSet) * trainPerc), ]
  #Use 2/3 of the data to build the tree
```

```
  TestSet <-
    DataSet[(floor(nrow(DataSet) * trainPerc) + 1):nrow(DataSet), ]
  #And leave out 1/3 data to test our strategy
  DecisionTree <-
    rpart(
      Class ~ RSI + EMAcross + WPR + ADX + CMO + CCI + ROC,
      data = TrainingSet,
      na.action = na.omit,
      cp = .001
    )
  #Specifying the indicators to we want to use to predict the class and contr
olling the growth of the tree by setting the minimum amount of information ga
ined (cp) needed to justify a split.
  prp(DecisionTree, type = 2, extra = 8)
  #Plotting tool with a couple parameters to make it look good.
  fit <- printcp(DecisionTree)
  #Shows the minimal cp for each trees of each size.
  mincp <- fit[which.min(fit[, 'xerror']), 'CP']
  #Get the lowest cross-validated error (xerror)
  plotcp(DecisionTree, upper = "splits")
  #plots the average geometric mean for trees of each size.
  PrunedDecisionTree <- prune(DecisionTree, cp = mincp)
  #Selecting the complexity parameter (cp) that has the lowest cross-validate
d error (xerror)
  t <- prp(PrunedDecisionTree, type = 2, extra = 8)
  confmat <-
    table(
      predict(PrunedDecisionTree, TestSet, type = "class"),
      TestSet[, 1],
      dnn = list('predicted', 'actual')
    )
  #Building confusion matrix
  print(confmat)
  acc <-
    (confmat[1, "DOWN"] + confmat[2, "UP"]) * 100 / (confmat[2, "DOWN"] + con
fmat[1, "UP"] + confmat[1, "DOWN"] + confmat[2, "UP"])
  #Calculating accuracy
  xy <-
    paste('Decision Tree : Considering the output for', SYM, sep = ' ')
  yz <-
    paste('Accuracy =',
          acc,
          sep = ' ')
  print(xy)
  print(yz)
  predout <- data.frame(predict(PrunedDecisionTree, TestSet))
  predval <- predout['UP'] - predout['DOWN']
  predclass <- ifelse(predout['UP'] >= predout['DOWN'], 1, 0)
  predds <- data.frame(predclass, TestSet$Class)
  colnames(predds) <- c("pred", "truth")
```

```
  predds[, 2] <- ifelse(predds[, 2] == 'UP', 1, 0)
  pred <- prediction(predds$pred, predds$truth)
  perf = performance(pred, measure = "tpr", x.measure = "fpr")
  auc.perf = performance(pred, measure = 'auc')
  #Calculating the AUC
  rmse.perf = performance(pred, measure = 'rmse')
  #Calculating the RMSE
  RMSE <- paste('RMSE =', rmse.perf@y.values, sep = ' ')
  AUC <- paste('AUC =', auc.perf@y.values, sep = ' ')
  print(AUC)
  print(RMSE)
  plot(perf, col = 1:10)
  abline(a = 0, b = 1, col = "red")
  #Plotting ROC curve
  print('-------------------------------------------------------------------
-----')
}, error = function(e) {
  print(e)
})
```

5.1.2 SVM Implementation Code

```
library(quantmod)

library(lubridate)
library(e1071)
library(rpart)
library(rpart.plot)
library(ROCR)

options(warn = -1)
tryCatch({
  SYM <- 'FB'
  print('-------------------------------------------------------------------
-----')
  print(paste('Predicting the output for', SYM, sep = ' '))
  trainPerc <- 0.75
  #Percent of data to be used as Training Data and remaining will be used as
Testing data
  date <- as.Date(Sys.Date() - 1)
  endDate <- date#as.Date("2016-01-01")
  d <- as.POSIXlt(endDate)
  d$year <- d$year - 2
  #To take last 2 years of data
  startDate <- as.Date(d)
  STOCK <- getSymbols(
    SYM,
    env = NULL,
    src = "yahoo",
    from = startDate,
```

```
    to = endDate
  )
  RSI <- RSI(Op(STOCK), n = 3)
  #Calculate a 3-period relative strength index (RSI) off the open price
  EMA5 <- EMA(Op(STOCK), n = 5)
  #Calculate a 5-period exponential moving average (EMA)
  EMAcross <- Op(STOCK) - EMA5
  #Let us explore the difference between the open price and our 5-period EMA
  MACD <- MACD(Op(STOCK),
               fast = 12,
               slow = 26,
               signal = 9)
  #Calculate a MACD with standard parameters
  MACD <- MACD[, 2]
  #Grab just the signal line to use as our indicator.
  SMI <- SMI(
    Op(STOCK),
    n = 13,
    slow = 25,
    fast = 2,
    signal = 9
  )
  #Stochastic Oscillator with standard parameters
  SMI <- SMI[, 1]
  #Grab just the oscillator to use as our indicator
  WPR <- WPR(Cl(STOCK), n = 14)
  WPR <- WPR[, 1]
  #Williams %R with standard parameters
  ADX <- ADX(STOCK, n = 14)
  ADX <- ADX[, 1]
  #Average Directional Index with standard parameters
  CCI <- CCI(Cl(STOCK), n = 14)
  CCI <- CCI[, 1]
  #Commodity Channel Index with standard parameters
  CMO <- CMO(Cl(STOCK), n = 14)
  CMO <- CMO[, 1]
  #Collateralized Mortgage Obligation with standard parameters
  ROC <- ROC(Cl(STOCK), n = 2)
  ROC <- ROC[, 1]
  #Price Rate Of Change with standard parameters
  PriceChange <- Cl(STOCK) - Op(STOCK)
  #Calculate the difference between the close price and open price
  Class <- ifelse(PriceChange > 0, 'UP', 'DOWN')
  #Create a binary classification variable, the variable we are trying to pre
dict.
  DataSet <-
    data.frame(Class, RSI, EMAcross, MACD, SMI, WPR, ADX, CCI, CMO, ROC)
  #Create our data set
  colnames(DataSet) <-
    c("Class",
```

```
      "RSI",
      "EMAcross",
      "MACD",
      "SMI",
      "WPR",
      "ADX",
      "CCI",
      "CMO",
      "ROC")
  #Name the columns
  #DataSet <- DataSet[-c(1:33), ]
  #Get rid of the data where the indicators are being calculated
  TrainingSet <- DataSet[1:floor(nrow(DataSet) * trainPerc), ]
  #Use 2/3 of the data to build the tree
  TestSet <-
    DataSet[(floor(nrow(DataSet) * trainPerc) + 1):nrow(DataSet), ]
  #And leave out 1/3 data to test our strategy
  SVM <-
    svm(
      Class ~ RSI + EMAcross + WPR + ADX + CMO + CCI + ROC,
      data = TrainingSet,
      kernel = "radial",
      type = "C-classification",
      na.action = na.omit,
      cost = 1,
      gamma = 1 / 5
    )
  #Specifying the indicators to we want to use to predict the class.
  print(SVM)
  confmat <-
    table(predict(SVM, TestSet, type = "class"),
          TestSet[, 1],
          dnn = list('predicted', 'actual'))
  #Building confusion matrix
  print(confmat)
  acc <-
    (confmat[1, "DOWN"] + confmat[2, "UP"]) * 100 / (confmat[2, "DOWN"] + con
fmat[1, "UP"] + confmat[1, "DOWN"] + confmat[2, "UP"])
  #Calculating accuracy
  xy <- paste('SVM : Considering the output for', SYM, sep = ' ')
  yz <-
    paste('Accuracy =',
          acc,
          sep = ' ')
  print(xy)
  print(yz)
  predds <- data.frame(predict(SVM, TestSet), TestSet$Class)
  colnames(predds) <- c("pred", "truth")
  predds[, 1] <- ifelse(predds[, 1] == 'UP', 1, 0)
  predds[, 2] <- ifelse(predds[, 2] == 'UP', 1, 0)
```

```
  pred <- prediction(predds$pred, predds$truth)
  perf = performance(pred, measure = "tpr", x.measure = "fpr")
  auc.perf = performance(pred, measure = 'auc', col = "red")
  #Calculating the AUC
  rmse.perf = performance(pred, measure = 'rmse')
  #Calculating the RMSE
  RMSE <- paste('RMSE =', rmse.perf@y.values, sep = ' ')
  AUC <- paste('AUC =', auc.perf@y.values, sep = ' ')
  print(AUC)
  print(RMSE)
  plot(perf, col = 1:10)
  abline(a = 0, b = 1, col = "red")
  #Plotting ROC curve
  print('-------------------------------------------------------------------
-----')
}, error = function(e) {
  print(e)
})
```

5.2 Design Document and Flowchart

5.2.1 Design Document

Methods used for indicators

RSI(): To calculate RSI

EMA (): To calculate EMA

EMAcross = Open price - EMA

WPR(): To calculate WPR

CCI(): To calculate CCI

CMO(): To calculate CMO

ADX(): To calculate ADX

ROC(): To calculate ROC

Method used for Decision Tree

rpart(): To prepare decision tree model

Method used for pruning Decision Tree

prune(): To prune the decision tree

Method used for predicting the output

predict(): To predict the output

Methods used for evaluating the model

performance(): To calculate data for ROC graph, AUC, and RMSE

plot(): To plot ROC graph

Method used for SVM

svm(): To prepare decision tree model

Method used for predicting the output

predict(): To predict the output

Methods used for evaluating the model

performance(): To calculate data for ROC graph, AUC, and RMSE

plot(): To plot ROC graph

5.2.2 Flowchart

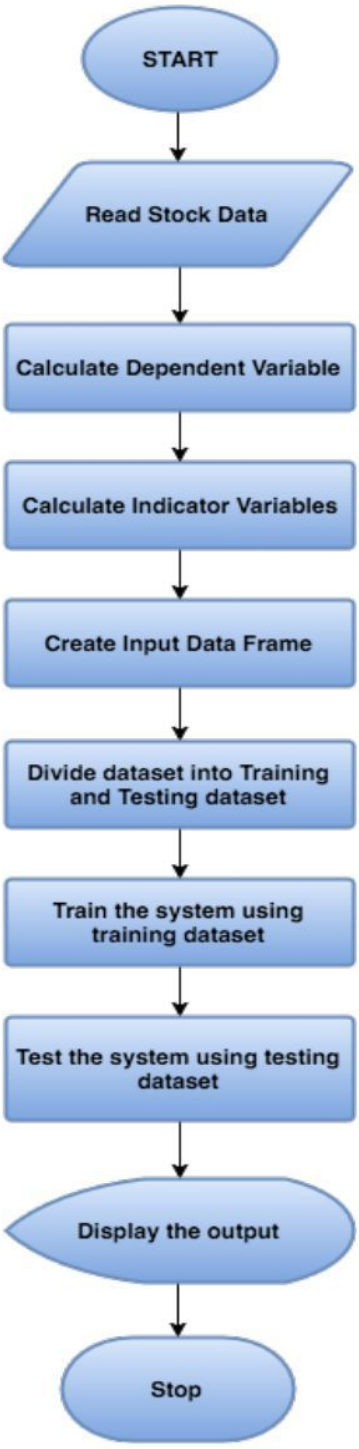

Figure 5: Program Flowchart

6 Data Analysis and Discussion

6.1 Output Generation

The application reads the input and applies the prediction algorithm to it to generate the output. The output consists of *confusion matrix, accuracy, ROC curve, area under curve* and *root mean square error*. This output is generated for all the selected stocks.

6.2 Output Analysis

The close analysis of the output of Decision trees and SVM algorithm reveals that the SVM gives better results than decision trees. Table 2 displays a comparison of the output of both the algorithms.

Stock Name	Parameters	Decision Tree	SVM
Apple	RSI	48.81	46.45
	WPR	60.62	59.84
	ADX	55.11	50.39
	CMO	58.26	53.54
	CCI	59.05	62.99
	ROC	65.35	63.77
	EMA Cross	50.39	50.39
	Combined Accuracy	**72.44**	**80.31**
	RMSE	**0.525**	**0.444**
Microsoft	RSI	57.48	49.60
	WPR	57.48	60.62
	ADX	51.96	49.60
	CMO	60.62	57.48
	CCI	61.41	64.56
	ROC	69.29	66.92
	EMA Cross	47.24	49.60
	Combined Accuracy	**81.10**	**82.67**
	RMSE	**0.435**	**0.416**
IBM	RSI	52.75	51.96
	WPR	62.99	66.92
	ADX	51.96	51.96
	CMO	51.18	59.05
	CCI	61.41	62.99
	ROC	70.86	71.65
	EMA Cross	51.96	51.18
	Combined Accuracy	**76.37**	**85.03**
	RMSE	**0.486**	**0.387**

General Motors	RSI	53.54	53.54
	WPR	58.26	66.14
	ADX	49.60	48.81
	CMO	62.99	61.41
	CCI	59.05	66.14
	ROC	70.07	70.07
	EMA Cross	48.81	48.81
	Combined Accuracy	**74.01**	**82.67**
	RMSE	**0.509**	**0.416**
General Electric	RSI	51.96	40.94
	WPR	66.92	66.92
	ADX	40.15	39.37
	CMO	49.6	66.92
	CCI	62.20	66.92
	ROC	65.35	64.56
	EMA Cross	44.88	40.94
	Combined Accuracy	**73.22**	**82.67**
	RMSE	**0.517**	**0.416**
Facebook	RSI	58.26	48.81
	WPR	65.35	62.99
	ADX	54.33	51.18
	CMO	56.69	55.9
	CCI	63.77	65.35
	ROC	67.71	70.07
	EMA Cross	58.18	58.26
	Combined Accuracy	**78.74**	**87.4**
	RMSE	**0.461**	**0.355**
Google	RSI	47.24	48.03
	WPR	60.62	66.14
	ADX	50.39	48.81
	CMO	56.69	56.69
	CCI	64.56	63.77
	ROC	70.86	67.7
	EMA Cross	48.03	48.03
	Combined Accuracy	**80.31**	**81.88**
	RMSE	**0.444**	**0.425**

Table 2: Effect of Indicators on prediction accuracy

PROGRAM FLOWCHART

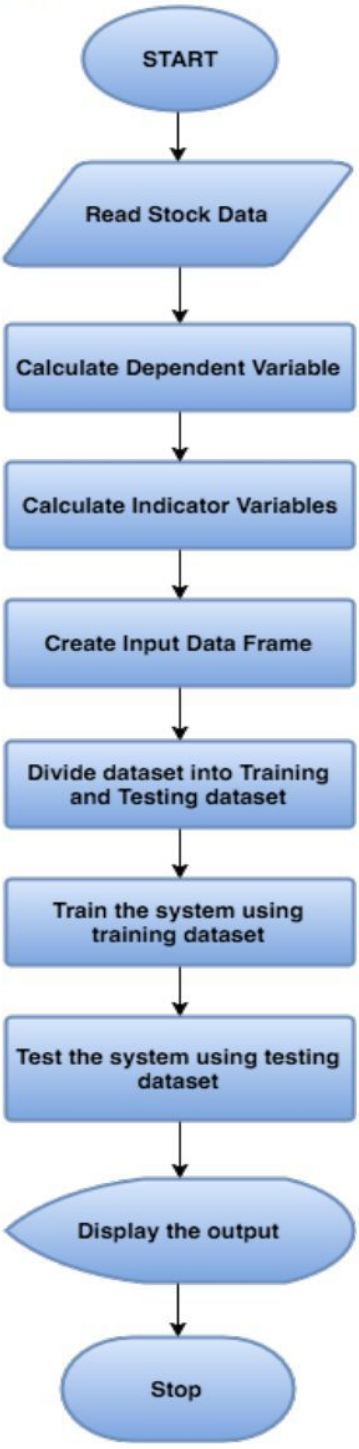

Figure 6: Program Flow

Program Source Code and Documentation

```
library(quantmod)

library(lubridate)
library(e1071)
library(rpart)
library(rpart.plot)
library(ROCR)

options(warn = -1)
a <- c('AAPL', 'FB', 'GE', 'GOOG', 'GM', 'IBM', 'MSFT')
for (i in 1:length(a))
{
  SYM <- a[i]
  print('-------------------------------------------------------------------
-----')
  print(paste('Predicting the output for', SYM, sep = ' '))
  trainPerc <- 0.75
  date <- as.Date(Sys.Date() - 1)
  endDate <- date#as.Date("2016-01-01")
  d <- as.POSIXlt(endDate)
  d$year <- d$year - 2
  startDate <- as.Date(d)
  STOCK <- getSymbols(
    SYM,
    env = NULL,
    src = "yahoo",
    from = startDate,
    to = endDate
  )
  RSI3 <- RSI(Op(STOCK), n = 3)
  #Calculate a 3-period relative strength index (RSI) off the open price
  EMA5 <- EMA(Op(STOCK), n = 5)
  #Calculate a 5-period exponential moving average (EMA)
  EMAcross <- Op(STOCK) - EMA5
  #Let us explore the difference between the open price and our 5-period EMA
  MACD <- MACD(Op(STOCK),
               fast = 12,
               slow = 26,
               signal = 9)
  #Calculate a MACD with standard parameters
  MACDsignal <- MACD[, 2]
  #Grab just the signal line to use as our indicator.
  SMI <- SMI(
    Op(STOCK),
    n = 13,
    slow = 25,
    fast = 2,
```

```
    signal = 9
  )
  #Stochastic Oscillator with standard parameters
  SMI <- SMI[, 1]
  #Grab just the oscillator to use as our indicator
  WPR <- WPR(Cl(STOCK), n = 14)
  WPR <- WPR[, 1]
  ADX <- ADX(STOCK, n = 14)
  ADX <- ADX[, 1]
  CCI <- CCI(Cl(STOCK), n = 14)
  CCI <- CCI[, 1]
  CMO <- CMO(Cl(STOCK), n = 14)
  CMO <- CMO[, 1]
  ROC <- ROC(Cl(STOCK), n = 2)
  ROC <- ROC[, 1]
  PriceChange <- Cl(STOCK) - Op(STOCK)
  #Calculate the difference between the close price and open price
  Class <- ifelse(PriceChange > 0, "UP", "DOWN")
  #Create a binary classification variable, the variable we are trying to pre
dict.
  DataSet <-
    data.frame(Class, RSI3, EMAcross, MACDsignal, SMI, WPR, ADX, CCI, CMO, RO
C)
  #Create our data set
  colnames(DataSet) <-
    c(
      "Class",
      "RSI3",
      "EMAcross",
      "MACDsignal",
      "Stochastic",
      "WPR",
      "ADX",
      "CCI",
      "CMO",
      "ROC"
    )
  #Name the columns
  #DataSet <- DataSet[-c(1:33),]#33
  #Get rid of the data where the indicators are being calculated
  TrainingSet <- DataSet[1:floor(nrow(DataSet) * trainPerc), ]
  #Use 2/3 of the data to build the tree
  TestSet <-
    DataSet[(floor(nrow(DataSet) * trainPerc) + 1):nrow(DataSet), ]
  #And leave out 1/3 data to test our strategy
  DecisionTree <-
    rpart(
      Class ~ RSI3 + EMAcross + WPR + ADX + CMO + CCI + ROC,
      #+ MACDsignal + Stochastic,
      data = TrainingSet,
```

```
      na.action = na.omit,
      cp = .001
    )
  #Specifying the indicators to we want to use to predict the class and contr
olling the growth of the tree by setting the minimum amount of information ga
ined (cp) needed to justify a split.
  prp(DecisionTree, type = 2, extra = 8)
  #Nice plotting tool with a couple parameters to make it look good. If you w
ant to play around with the visualization yourself, here is a great resource.
  fit <- printcp(DecisionTree)
  #shows the minimal cp for each trees of each size.
  mincp <- fit[which.min(fit[, 'xerror']), 'CP']
  #Get the lowest cross-validated error (xerror)
  plotcp(DecisionTree, upper = "splits")
  #plots the average geometric mean for trees of each size.
  PrunedDecisionTree <- prune(DecisionTree, cp = mincp)
  #We are selecting the complexity parameter (cp) that has the lowest cross-v
alidated error (xerror)
  t <- prp(PrunedDecisionTree, type = 2, extra = 8)
  confmat <-
    table(
      predict(PrunedDecisionTree, TestSet, type = "class"),
      TestSet[, 1],
      dnn = list('predicted', 'actual')
    )
  print(confmat)
  tryCatch({
    acc <-
      (confmat[1, "DOWN"] + confmat[2, "UP"]) * 100 / (confmat[2, "DOWN"] + c
onfmat[1, "UP"] + confmat[1, "DOWN"] + confmat[2, "UP"])
    #if (acc > 60) {
    xy <-
      paste('Decision Tree : Considering the output for', SYM, sep = ' ')
    yz <-
      paste('Accuracy',
            acc,
            sep = ' ')
    out <- paste(xy, yz, sep = '\n')
    print(out)
    write(out,
          file = "out",
          append = TRUE,
          sep = "\n\n")
    #}
  }, error = function(e) {

  })
  predout <- data.frame(predict(PrunedDecisionTree, TestSet))
  predval <- predout['UP'] - predout['DOWN']
  predclass <- ifelse(predout['UP'] >= predout['DOWN'], 1, 0)
```

```
  predds <- data.frame(predclass, TestSet$Class)
  colnames(predds) <- c("pred", "truth")
  predds[, 2] <- ifelse(predds[, 2] == 'UP', 1, 0)
  #print(predds)
  pred <- prediction(predds$pred, predds$truth)
  perf = performance(pred, measure = "tpr", x.measure = "fpr")
  plot(perf, col = 1:10)
  auc.perf = performance(pred, measure = 'auc')
  rmse.perf = performance(pred, measure = 'rmse')
  print(paste('RMSE =', rmse.perf@y.values), sep = ' ')
  print(paste('AUC =', auc.perf@y.values), sep = ' ')
  abline(a = 0, b = 1, col = "red")
  print('-----------------------------------------------------------------------
-----')
}
```

9.2.2 SVM Implementation Code

```
library(quantmod)

library(lubridate)
library(e1071)
library(rpart)
library(rpart.plot)
library(ROCR)

options(warn = -1)
a <- c('AAPL', 'FB', 'GE', 'GOOG', 'GM', 'IBM', 'MSFT')
for (i in 1:length(a))
{
  SYM <- a[i]
  print('-----------------------------------------------------------------------
-----')
  print(paste('Predicting the output for', SYM, sep = ' '))
  trainPerc <- 0.75
  date <- as.Date(Sys.Date() - 1)
  endDate <- date#as.Date("2016-01-01")
  d <- as.POSIXlt(endDate)
  d$year <- d$year - 2
  startDate <- as.Date(d)
  STOCK <- getSymbols(
    SYM,
    env = NULL,
    src = "yahoo",
    from = startDate,
    to = endDate
  )
  RSI3 <- RSI(Op(STOCK), n = 3)
```

```r
  #Calculate a 3-period relative strength index (RSI) off the open price
  EMA5 <- EMA(Op(STOCK), n = 5)
  #Calculate a 5-period exponential moving average (EMA)
  EMAcross <- Op(STOCK) - EMA5
  #Let us explore the difference between the open price and our 5-period EMA
  MACD <- MACD(Op(STOCK),
               fast = 12,
               slow = 26,
               signal = 9)
  #Calculate a MACD with standard parameters
  MACDsignal <- MACD[, 2]
  #Grab just the signal line to use as our indicator.
  SMI <- SMI(
    Op(STOCK),
    n = 13,
    slow = 25,
    fast = 2,
    signal = 9
  )
  #Stochastic Oscillator with standard parameters
  SMI <- SMI[, 1]
  #Grab just the oscillator to use as our indicator
  WPR <- WPR(Cl(STOCK), n = 14)
  WPR <- WPR[, 1]
  ADX <- ADX(STOCK, n = 14)
  ADX <- ADX[, 1]
  CCI <- CCI(Cl(STOCK), n = 14)
  CCI <- CCI[, 1]
  CMO <- CMO(Cl(STOCK), n = 14)
  CMO <- CMO[, 1]
  ROC <- ROC(Cl(STOCK), n=2)
  ROC <- ROC[, 1]
  #DPO <- DPO(Cl(STOCK), n = 10)
  #DPO <- DPO[, 1]
  PriceChange <- Cl(STOCK) - Op(STOCK)
  #Calculate the difference between the close price and open price
  Class <- ifelse(PriceChange > 0, 'UP', 'DOWN')
  #Create a binary classification variable, the variable we are trying to
predict.
  DataSet <-
    data.frame(Class, RSI3, EMAcross, MACDsignal, SMI, WPR, ADX, CCI, CMO,
ROC)
  #Create our data set
  colnames(DataSet) <-
    c(
      "Class",
```

```
      "RSI3",
      "EMAcross",
      "MACDsignal",
      "Stochastic",
      "WPR",
      "ADX",
      "CCI",
      "CMO",
      "ROC"
    )
  #Name the columns
  #DataSet <- DataSet[-c(1:33), ]
  #Get rid of the data where the indicators are being calculated
  TrainingSet <- DataSet[1:floor(nrow(DataSet) * trainPerc),]
  #Use 2/3 of the data to build the tree
  TestSet <-
    DataSet[(floor(nrow(DataSet) * trainPerc) + 1):nrow(DataSet),]
  #And leave out 1/3 data to test our strategy
  SVM <-
    svm(
      Class ~ RSI3 + EMAcross + WPR + ADX + CMO + CCI + ROC,
      # + MACDsignal + Stochastic,
      data = TrainingSet,
      kernel = "radial",
      type = "C-classification",
      na.action = na.omit,
      cost = 1,
      gamma = 1 / 5
    )
  print(SVM)
  confmat <-
    table(predict(SVM, TestSet, type = "class"),
          TestSet[, 1],
          dnn = list('predicted', 'actual'))
  print(confmat)
  tryCatch({
    acc <-
      (confmat[1, "DOWN"] + confmat[2, "UP"]) * 100 / (confmat[2, "DOWN"] +
confmat[1, "UP"] + confmat[1, "DOWN"] + confmat[2, "UP"])
    #if (acc > 60) {
    xy <- paste('SVM : Considering the output for', SYM, sep = ' ')
    yz <-
      paste('Accuracy =',
            acc,
            sep = ' ')
    out <- paste(xy, yz, sep = '\n')
```

```
    print(out)
    write(out,
          file = "out",
          append = TRUE,
          sep = "\n\n")
    #}
  }, error = function(e) {

  })
  predds <- data.frame(predict(SVM, TestSet), TestSet$Class)
  colnames(predds) <- c("pred", "truth")
  predds[, 1] <- ifelse(predds[, 1] == 'UP', 1, 0)
  predds[, 2] <- ifelse(predds[, 2] == 'UP', 1, 0)
  pred <- prediction(predds$pred, predds$truth)
  perf = performance(pred, measure = "tpr", x.measure = "fpr")
  auc.perf = performance(pred, measure = 'auc', col = "red")
  rmse.perf = performance(pred, measure = 'rmse')
  print(paste('RMSE =', rmse.perf@y.values), sep = ' ')
  print(paste('AUC =', auc.perf@y.values), sep = ' ')
  plot(perf, col = 1:10)
  abline(a = 0, b = 1, col = "red")

  print('-------------------------------------------------------------------
-----')
}
```

Input/Output Listing

Input

The input is two years of historic data along with the indicator variables for all the stocks and both the algorithms. A sample of input is shown in Table 3.

```
## Date         Class   RSI        EMAcross       MACD         SMI

## 2014-05-05  UP      71.420701  3.194429e+00   1.181512792  68.2833293
## 2014-05-06  DOWN    88.852547  9.902935e+00   1.522929335  72.7037309
## 2014-05-07  DOWN    58.688940  2.235288e+00   1.806807870  74.7893073
## 2014-05-08  DOWN    38.005966 -3.176475e+00   2.015919827  74.4003458
## 2014-05-09  UP      29.688201 -4.590977e+00   2.152247208  72.5946848
## 2014-05-12  UP      44.242592 -1.094002e+00   2.238940024  70.1038033
## 2014-05-13  UP      62.190599  2.277339e+00   2.297133579  68.4194937
## 2014-05-14  UP      63.854668  1.804908e+00   2.330191625  66.8260825
```

Output

Technically the Output of the system should be the direction of stock for the next day. But for the better understanding of the prediction model we are displaying the *confusion matrix,* root mean squared error and area under curve.