SVM S&P500 Monthly 01-01-2012 to 06-30-2018

Code

Hide

```
#install.packages("quantmod")
#install.packages("e1071")
library(quantmod)
```

```
Loading required package: xts
Loading required package: zoo

Attaching package: <U+393C><U+3E31>zoo<U+393C><U+3E32>

The following objects are masked from <U+393C><U+3E31>package:base<U+393C><U+3E32>:

    as.Date, as.Date.numeric

Loading required package: TTR
Version 0.4-0 included new data defaults. See ?getSymbols.
Learn from a quantmod author: https://www.datacamp.com/courses/importing-and-managing-financial-
data-in-r
```

Hide

```
library(e1071)
# Importing the dataset
startDate = as.Date("2011-01-01")
endDate = as.Date("2018-06-30")
getSymbols("^GSPC",src="yahoo",from=startDate,to=endDate)
```

```
<U+393C><U+3E31>getSymbols<U+393C><U+3E32> currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
<U+393C><U+3E31>loadSymbols<U+393C><U+3E32> to automatically load data. getOption("getSymbols.en
v")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.


WARNING: There have been significant changes to Yahoo Finance data.
Please see the Warning section of <U+393C><U+3E31>?getSymbols.yahoo<U+393C><U+3E32> for details.

This message is shown once per session and may be disabled by setting
options("getSymbols.yahoo.warning"=FALSE).
```

```
[1] "GSPC"
```

Hide

```
dataset=data.frame(to.monthly(GSPC))
dim(dataset)
```

```
[1] 90  6
```

Hide

```
head(dataset)
```

	GSPC.Open <dbl>	GSPC.High <dbl>	GSPC.L... <dbl>	GSPC.Close <dbl>	GSPC.Volume <dbl>	GSPC.Adjusted <dbl>
Jan 2011	1257.62	1302.67	1257.62	1286.12	92164940000	1286.12
Feb 2011	1289.14	1344.07	1289.14	1327.22	59223660000	1327.22
Mar 2011	1328.64	1332.28	1249.05	1325.83	89507640000	1325.83
Apr 2011	1329.48	1364.56	1294.70	1363.61	77364810000	1363.61
May 2011	1365.21	1370.58	1311.80	1345.20	81708980000	1345.20
Jun 2011	1345.20	1345.20	1258.07	1320.64	86122730000	1320.64

6 rows

Hide

```
tail(dataset)
```

	GSPC.Open <dbl>	GSPC.High <dbl>	GSPC.L... <dbl>	GSPC.Close <dbl>	GSPC.Volume <dbl>	GSPC.Adjusted <dbl>
Jan 2018	2683.73	2872.87	2682.36	2823.81	76860120000	2823.81
Feb 2018	2816.45	2835.96	2532.69	2713.83	79579410000	2713.83
Mar 2018	2715.22	2801.90	2585.89	2640.87	76369800000	2640.87
Apr 2018	2633.45	2717.49	2553.80	2648.05	69648590000	2648.05
May 2018	2642.96	2742.24	2594.62	2705.27	75617280000	2705.27
Jun 2018	2718.70	2791.47	2691.99	2718.37	77439710000	2718.37

6 rows

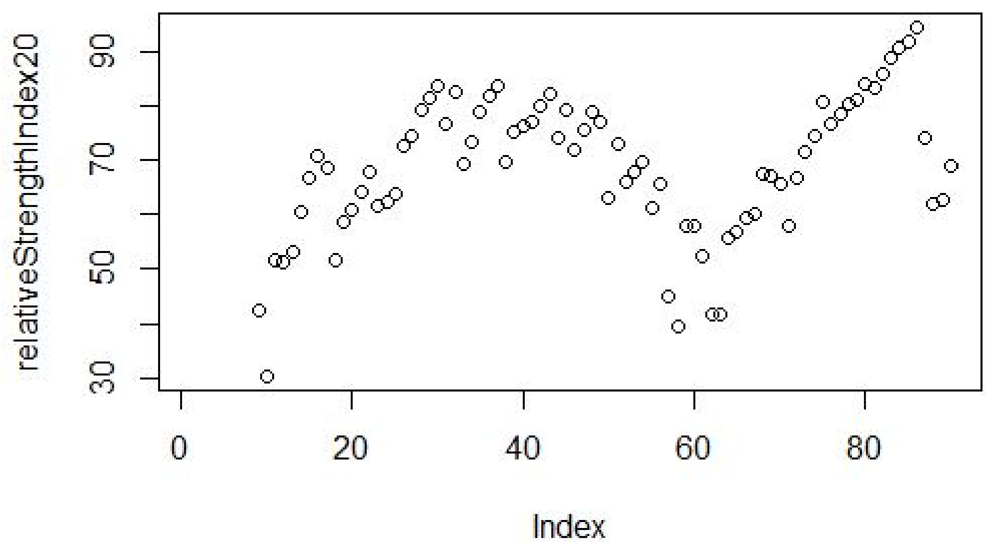
Hide

```
#RSI indicator
relativeStrengthIndex20=RSI(Op(dataset),n=8)
summary(relativeStrengthIndex20)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  30.38   60.81   69.47   68.64   78.78   94.41       8 
```

Hide

```
plot(relativeStrengthIndex20)
```



Hide

```
tail(relativeStrengthIndex20)
```

```
[1] 91.86270 94.41305 74.15401 61.89350 62.71293 68.81637
```

Hide

```
dim(relativeStrengthIndex20)
```

```
NULL
```

Hide

```
ncol(relativeStrengthIndex20)
```

```
NULL
```

Hide

```
nrow(relativeStrengthIndex20)
```

```
NULL
```

Hide

```
# Exponential Moving Average Indicator
exponentialMovingAverage20=EMA(Op(dataset),n=8)
head(exponentialMovingAverage20)
```

```
[1] NA NA NA NA NA NA
```

Hide

```
tail(exponentialMovingAverage20)
```

```
[1] 2540.936 2602.161 2627.285 2628.655 2631.834 2651.138
```

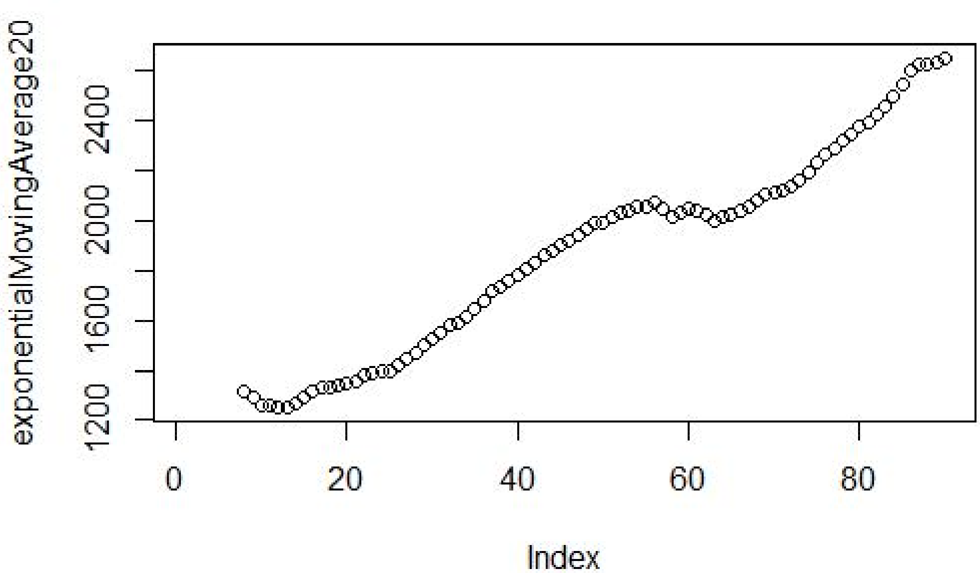
Hide

```
summary(exponentialMovingAverage20)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   1254    1485    1989    1875    2107    2651       7
```

Hide

```
plot(exponentialMovingAverage20)
```



Hide

```
# Difference in Exponential Moving Average
exponentialMovingAverageDiff =  (Op(dataset) - exponentialMovingAverage20)
head(exponentialMovingAverageDiff)
```

```
[1] NA NA NA NA NA NA
```

Hide

```
tail(exponentialMovingAverageDiff)
```

```
[1] 142.794117 214.288735  87.934588   4.794664  11.125857
[6]  67.562326
```

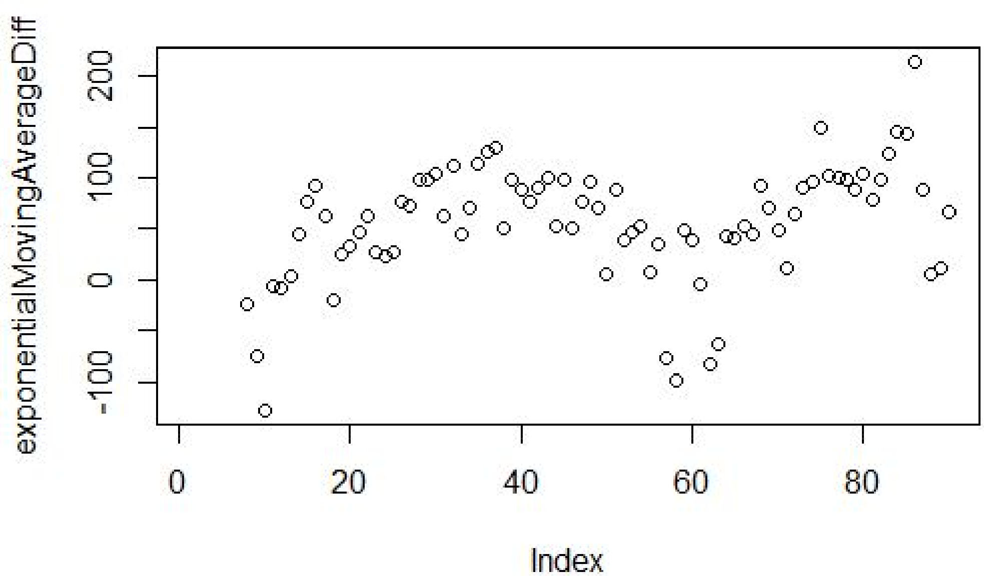
Hide

```
summary(exponentialMovingAverageDiff)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
-127.02   30.22   63.51   56.02   96.41  214.29       7 
```

Hide

```
plot(exponentialMovingAverageDiff)
```



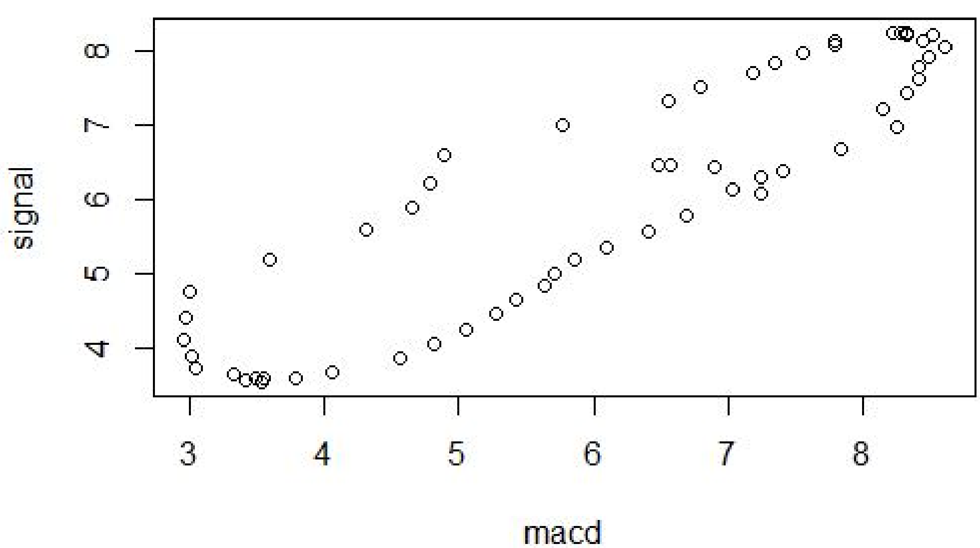
Hide

```
# MACD Indicator
MACD <- MACD(Op(dataset),fast = 3, slow = 8, signal = 5, type = "EMA", histogram = TRUE)
head(MACD)
```

```
     macd signal
[1,]   NA     NA
[2,]   NA     NA
[3,]   NA     NA
[4,]   NA     NA
[5,]   NA     NA
[6,]   NA     NA
```

Hide

```
plot(MACD)
```



Hide

```
tail(MACD)
```

```
          macd   signal
[85,] 6.684577 5.796747
[86,] 7.231540 6.083705
[87,] 7.236006 6.314165
[88,] 6.893389 6.430010
[89,] 6.572852 6.458579
[90,] 6.476851 6.462233
```

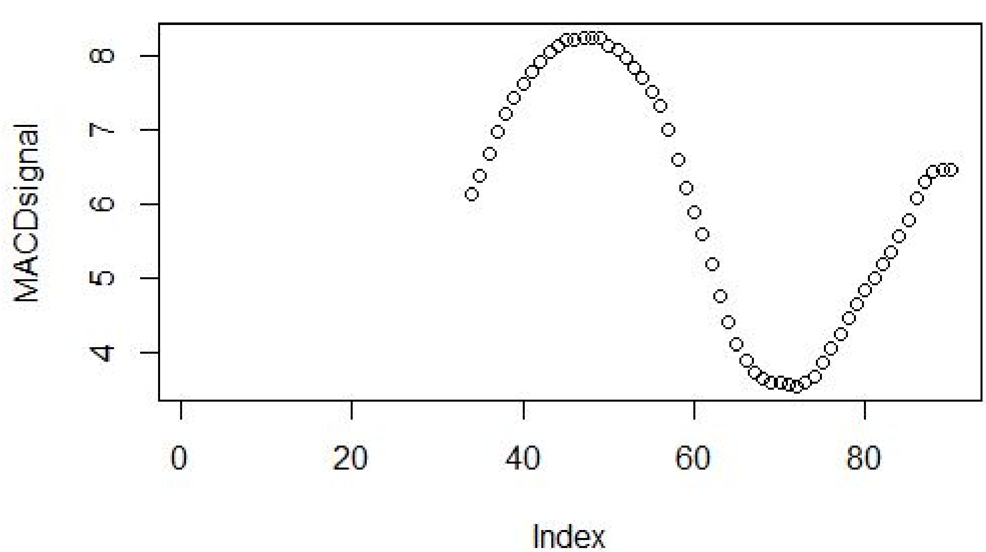
Hide

```
summary(MACD)
```

```
      macd           signal
 Min.   :2.966   Min.   :3.560
 1st Qu.:4.780   1st Qu.:4.461
 Median :6.405   Median :6.231
 Mean   :6.065   Mean   :6.032
 3rd Qu.:7.544   3rd Qu.:7.633
 Max.   :8.601   Max.   :8.258
 NA's   :25      NA's   :33
```

Hide

```
MACDsignal <- MACD[,2]
plot(MACDsignal)
```



Hide

```
# Bollinger Band indicator
BollingerBands <- BBands(Op(dataset),n=8,sd=2)
head(BollingerBands)
```

```
     dn mavg up pctB
[1,] NA   NA NA   NA
[2,] NA   NA NA   NA
[3,] NA   NA NA   NA
[4,] NA   NA NA   NA
[5,] NA   NA NA   NA
[6,] NA   NA NA   NA
```

Hide

```
tail(BollingerBands)
```

```
            dn     mavg       up      pctB
[85,] 2343.950 2528.975 2714.000 0.9181996
[86,] 2335.982 2579.075 2822.167 0.9882400
[87,] 2385.597 2614.554 2843.511 0.7198366
[88,] 2430.075 2634.097 2838.120 0.4984131
[89,] 2490.421 2655.165 2819.909 0.4629577
[90,] 2546.632 2679.852 2813.073 0.6458009
```

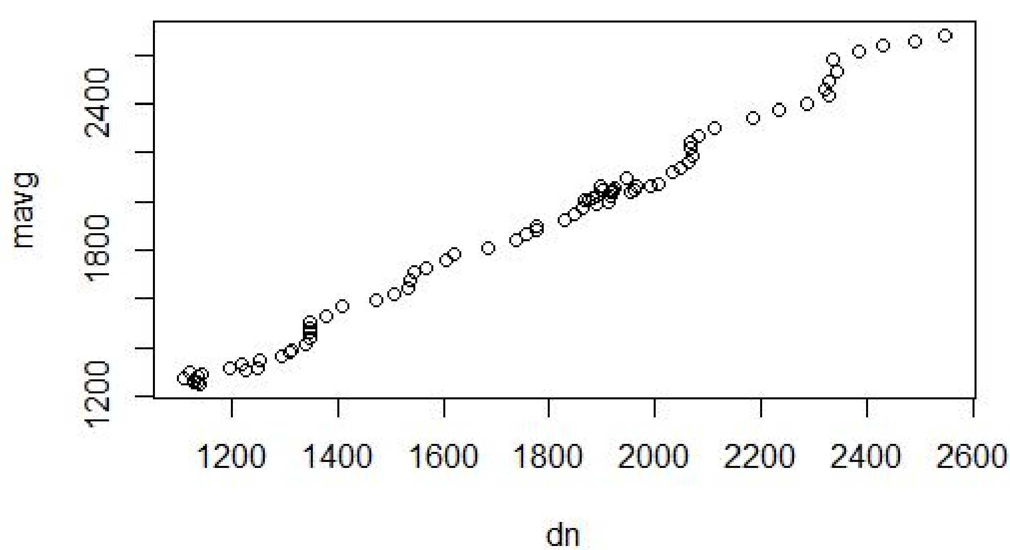
Hide

```
summary(BollingerBands)
```

```
       dn            mavg             up      
 Min.   :1112   Min.   :1254   Min.   :1368  
 1st Qu.:1349   1st Qu.:1472   1st Qu.:1596  
 Median :1867   Median :1992   Median :2088  
 Mean   :1748   Mean   :1874   Mean   :2000  
 3rd Qu.:2005   3rd Qu.:2085   3rd Qu.:2216  
 Max.   :2547   Max.   :2680   Max.   :2844  
 NA's   :7      NA's   :7      NA's   :7     
      pctB        
 Min.   :-0.0484  
 1st Qu.: 0.6486  
 Median : 0.7843  
 Mean   : 0.7189  
 3rd Qu.: 0.9027  
 Max.   : 1.0238  
 NA's   :7        
```

Hide

```
plot(BollingerBands)
```



Hide

```
# % Change BB
PercentageChngpctB <- BollingerBands[,4]
head(PercentageChngpctB)
```

```
[1] NA NA NA NA NA NA
```

Hide

```
tail(PercentageChngpctB)
```

```
[1] 0.9181996 0.9882400 0.7198366 0.4984131 0.4629577 0.6458009
```

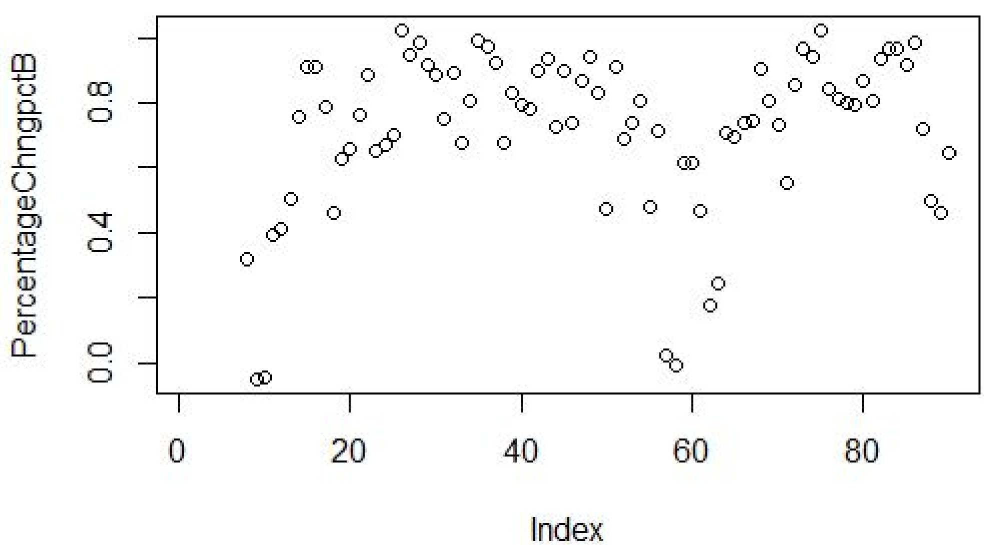
Hide

```
summary(PercentageChngpctB)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
-0.0484  0.6486  0.7843  0.7189  0.9027  1.0238       7 
```

Hide

```
plot(PercentageChngpctB)
```



Hide

```
# Price (Closes above Open = 1, Closes below Open = 0)
Price=ifelse(dataset[4]>dataset[1], 1,0)
head(Price)
```

```
         GSPC.Close
Jan 2011          1
Feb 2011          1
Mar 2011          0
Apr 2011          1
May 2011          0
Jun 2011          0
```

Hide

```
tail(Price)
```

```
         GSPC.Close
Jan 2018          1
Feb 2018          0
Mar 2018          0
Apr 2018          1
May 2018          1
Jun 2018          0
```

Hide

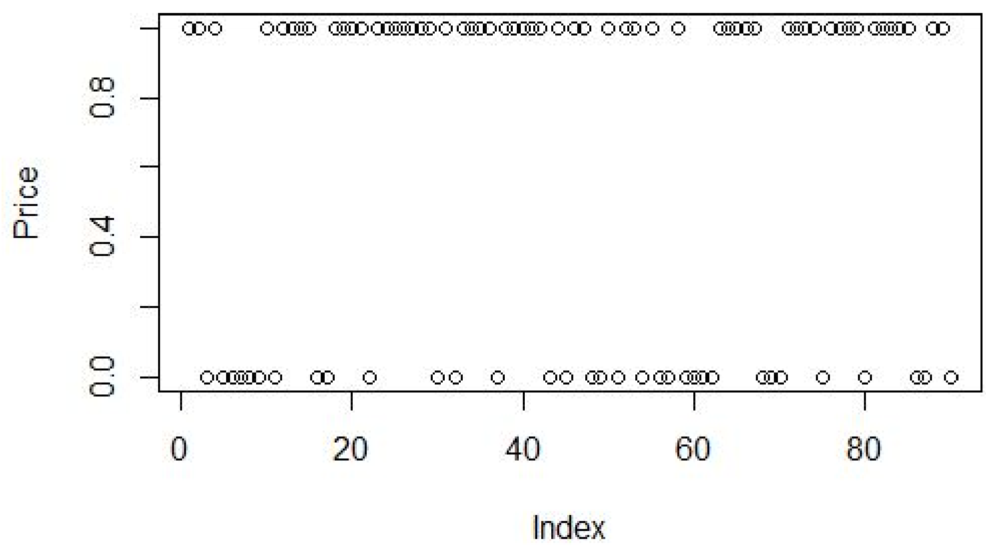
```
tail(dataset)
```

	GSPC.Open <dbl>	GSPC.High <dbl>	GSPC.L... <dbl>	GSPC.Close <dbl>	GSPC.Volume <dbl>	GSPC.Adjusted <dbl>
Jan 2018	2683.73	2872.87	2682.36	2823.81	76860120000	2823.81
Feb 2018	2816.45	2835.96	2532.69	2713.83	79579410000	2713.83
Mar 2018	2715.22	2801.90	2585.89	2640.87	76369800000	2640.87
Apr 2018	2633.45	2717.49	2553.80	2648.05	69648590000	2648.05
May 2018	2642.96	2742.24	2594.62	2705.27	75617280000	2705.27
Jun 2018	2718.70	2791.47	2691.99	2718.37	77439710000	2718.37

6 rows

Hide

```
plot(Price)
```



Hide

```
dim(relativeStrengthIndex20)
```

```
NULL
```

Hide

```
dim(exponentialMovingAverage20)
```

```
NULL
```

Hide

```
dim(MACDsignal)
```

```
NULL
```

Hide

```
dim(PercentageChngpctB)
```

```
NULL
```

Hide

```
dim(Price)
```

```
[1] 90  1
```

Hide

```
dataset1 = data.frame(relativeStrengthIndex20, exponentialMovingAverage20, MACDsignal, Percentag
eChngpctB, Price)
# Size of Data
str(dataset1)
```

```
'data.frame':	90 obs. of  5 variables:
 $ relativeStrengthIndex20   : num  NA NA NA NA NA ...
 $ exponentialMovingAverage20: num  NA NA NA NA NA ...
 $ MACDsignal                : num  NA NA NA NA NA NA NA NA NA NA ...
 $ PercentageChngpctB        : num  NA NA NA NA NA ...
 $ GSPC.Close                : num  1 1 0 1 0 0 0 0 0 1 ...
```

Hide

```
dim(dataset1)
```

```
[1] 90  5
```

Hide

```
#Checking for missing data
d3=dataset1
for(i in 1:ncol(d3))
   {
    print(colnames(d3[i]))
    print(sum(is.na(d3[i])))
   }
```

```
[1] "relativeStrengthIndex20"
[1] 8
[1] "exponentialMovingAverage20"
[1] 7
[1] "MACDsignal"
[1] 33
[1] "PercentageChngpctB"
[1] 7
[1] "GSPC.Close"
[1] 0
```

Hide

```
dataset1 = na.omit(dataset1)
#Checking for missing data again
dim(dataset1)
```

```
[1] 57  5
```

Hide

```
d3=dataset1
for(i in 1:ncol(d3))
   {
    print(colnames(d3[i]))
    print(sum(is.na(d3[i])))
   }
```

```
[1] "relativeStrengthIndex20"
[1] 0
[1] "exponentialMovingAverage20"
[1] 0
[1] "MACDsignal"
[1] 0
[1] "PercentageChngpctB"
[1] 0
[1] "GSPC.Close"
[1] 0
```

Hide

```
colnames(dataset1)=c ("RSI20", "EMA20", "MACDsignal", "BB", "Price")
# Exploring the data set components
str(dataset1)
```

```
'data.frame':   57 obs. of  5 variables:
 $ RSI20     : num  73.5 79 81.7 83.7 69.8 ...
 $ EMA20     : num  1611 1644 1680 1717 1732 ...
 $ MACDsignal: num  6.14 6.39 6.68 6.99 7.22 ...
 $ BB        : num  0.806 0.993 0.97 0.925 0.678 ...
 $ Price     : num  1 1 1 0 1 1 1 1 1 0 ...
 - attr(*, "na.action")=Class 'omit'  Named int [1:33] 1 2 3 4 5 6 7 8 9 10 ...
  .. ..- attr(*, "names")= chr [1:33] "Jan 2011" "Feb 2011" "Mar 2011" "Apr 2011" ...
```

Hide

```
# Encoding the target feature as factor
dataset1$Price=as.factor(dataset1$Price)
# Splitting the dataset into the Training set and Test set
library(caTools)
set.seed(123)
split = sample.split(dataset1$Price, SplitRatio = 0.8)
training_set = subset(dataset1, split == TRUE)
test_set = subset(dataset1, split == FALSE)
# Feature Scaling (Normalization and dropping the predicted variable)
training_set[-5] = scale(training_set[-5])
test_set[-5] = scale(test_set[-5])
# Applying Kernel SVM Model on the Training set
library(e1071)
classifier = svm(formula = Price ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')
# Predicting the Test set results
predict_val = predict(classifier, newdata = test_set[-5])
# Confusion Matrix
cm = table(test_set[, 5], predict_val)
print(cm)
```

```
   predict_val
    0 1
  0 0 4
  1 1 6
```

Hide

```
# Evaluating Model Accuracy on test data set using Confusion Matrix
Model_Accuracy=(cm[1,1] + cm[2,2])/ (cm[1,1] + cm[1,2] + cm[2,1] + cm[2,2])
print("Model Accuracy is")
```

```
[1] "Model Accuracy is"
```

Hide

```
print(Model_Accuracy)
```

```
[1] 0.5454545
```