SVM S&P500 Weekly MA 01-01-2012 to 06-30-2018

Code ▾

Hide

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

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

Hide

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

```
[1] 391   6
```

Hide

```
head(dataset)
```

	GSPC.Op… <dbl>	GSPC.High <dbl>	GSPC.L… <dbl>	GSPC.Close <dbl>	GSPC.Volume <dbl>	GSPC.Adjusted <dbl>
2011-01-07	1257.62	1278.17	1257.62	1271.50	23655220000	1271.50
2011-01-14	1270.84	1293.24	1262.18	1293.24	21286570000	1293.24
2011-01-21	1293.22	1296.06	1271.26	1283.35	19899340000	1283.35
2011-01-28	1283.29	1302.67	1275.10	1276.34	23156650000	1276.34
2011-02-04	1276.50	1311.00	1276.50	1310.87	21726860000	1310.87
2011-02-11	1311.85	1330.79	1311.74	1329.15	20109950000	1329.15

6 rows

Hide

```
tail(dataset)
```

	GSPC.Op... <dbl>	GSPC.High <dbl>	GSPC.L... <dbl>	GSPC.Close <dbl>	GSPC.Volume <dbl>	GSPC.Adjusted <dbl>
2018-05-25	2735.39	2742.24	2707.38	2721.33	15963780000	2721.33
2018-06-01	2705.11	2736.93	2676.81	2734.62	15217440000	2734.62
2018-06-08	2741.67	2779.90	2739.51	2779.03	17380480000	2779.03
2018-06-15	2780.18	2791.47	2761.73	2779.66	19368250000	2779.66
2018-06-22	2765.79	2774.99	2743.19	2754.88	19026830000	2754.88
2018-06-29	2742.94	2746.09	2691.99	2718.37	17980020000	2718.37

6 rows

Hide

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

```
           GSPC.Close
2011-01-07          1
2011-01-14          1
2011-01-21          0
2011-01-28          0
2011-02-04          1
2011-02-11          1
```

Hide

```
tail(Price)
```

```
           GSPC.Close
2018-05-25          0
2018-06-01          1
2018-06-08          1
2018-06-15          0
2018-06-22          0
2018-06-29          0
```

Hide

```
tail(dataset)
```

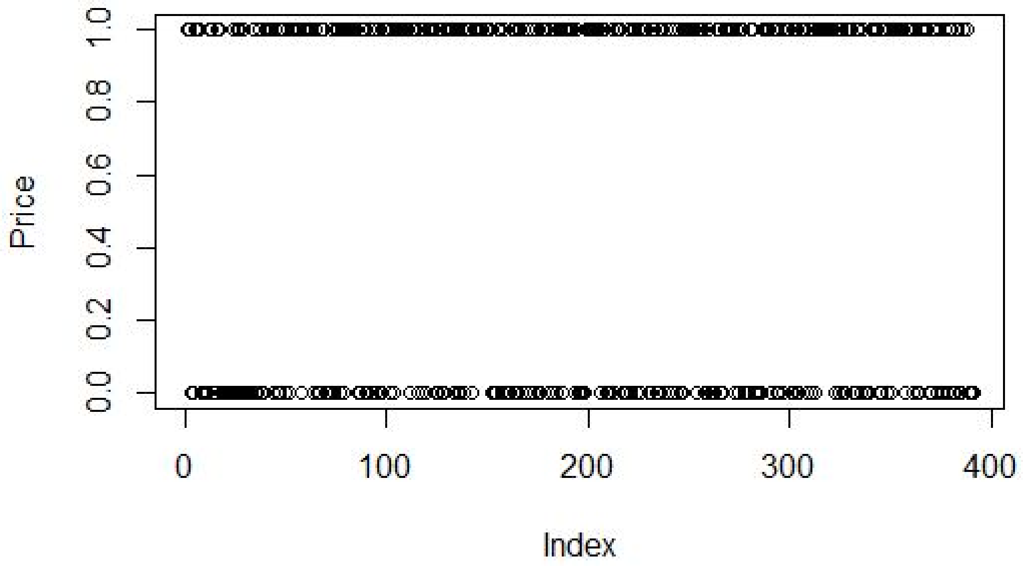
	GSPC.Op... <dbl>	GSPC.High <dbl>	GSPC.L... <dbl>	GSPC.Close <dbl>	GSPC.Volume <dbl>	GSPC.Adjusted <dbl>
2018-05-25	2735.39	2742.24	2707.38	2721.33	15963780000	2721.33
2018-06-01	2705.11	2736.93	2676.81	2734.62	15217440000	2734.62

		GSPC.High				GSPC.Adjusted
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
2018-06-08	2741.67	2779.90	2739.51	2779.03	17380480000	2779.03
2018-06-15	2780.18	2791.47	2761.73	2779.66	19368250000	2779.66
2018-06-22	2765.79	2774.99	2743.19	2754.88	19026830000	2754.88
2018-06-29	2742.94	2746.09	2691.99	2718.37	17980020000	2718.37

6 rows

Hide

```
plot(Price)
```



Hide

```
j=3
##---------------------------##
# Exponential Moving Average Indicator
i = j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff1 =  (Op(dataset) - exponentialMovingAverage20)
##---------------------------##
##---------------------------##
# Exponential Moving Average Indicator
i = 2*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff2 =  (Op(dataset) - exponentialMovingAverage20)
##---------------------------##
##---------------------------##
# Exponential Moving Average Indicator
i = 3*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff3 =  (Op(dataset) - exponentialMovingAverage20)
##---------------------------##
##---------------------------##
# Exponential Moving Average Indicator
i = 4*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff4 =  (Op(dataset) - exponentialMovingAverage20)
##---------------------------##
##---------------------------##
# Exponential Moving Average Indicator
i = 5*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff5 =  (Op(dataset) - exponentialMovingAverage20)
##---------------------------##
##---------------------------##
# Exponential Moving Average Indicator
i = 6*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff6 =  (Op(dataset) - exponentialMovingAverage20)
##---------------------------##
##---------------------------##
# Exponential Moving Average Indicator
i = 7*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff7 =  (Op(dataset) - exponentialMovingAverage20)
##---------------------------##
##---------------------------##
# Exponential Moving Average Indicator
i = 8*j
```

```
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff8 =  (Op(dataset) - exponentialMovingAverage20)
##--------------------------##
##--------------------------##
# Exponential Moving Average Indicator
i = 9*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff9 =  (Op(dataset) - exponentialMovingAverage20)
##--------------------------##
##--------------------------##
# Exponential Moving Average Indicator
i = 10*j
exponentialMovingAverage20=EMA(Op(dataset),n=i)
# Difference in Exponential Moving Average
exponentialMovingAverageDiff10 =  (Op(dataset) - exponentialMovingAverage20)
##--------------------------##
dataset1 = data.frame(exponentialMovingAverageDiff1,
                    exponentialMovingAverageDiff2,
                    exponentialMovingAverageDiff3,
                    exponentialMovingAverageDiff4,
                    exponentialMovingAverageDiff5,
                    exponentialMovingAverageDiff6,
                    exponentialMovingAverageDiff7,
                    exponentialMovingAverageDiff8,
                    exponentialMovingAverageDiff9,
                    exponentialMovingAverageDiff10,
                    Price)
tail(dataset1)
```

	exponentialMovingAverageDiff1 <dbl>	**exponentialMovingAverageDiff2** <dbl>
2018-05-25	13.839521	30.0642684
2018-06-01	-8.220133	-0.1539411
2018-06-08	14.169841	26.0041956
2018-06-15	26.339926	46.0815754
2018-06-22	5.975016	22.6369160
2018-06-29	-8.437541	-0.1522728

6 rows | 1-3 of 11 columns

Hide

```
# Size of Data
str(dataset1)
```

```
'data.frame':	391 obs. of  11 variables:
 $ exponentialMovingAverageDiff1 : num  NA NA 19.33 4.7 -1.05 ...
 $ exponentialMovingAverageDiff2 : num  NA NA NA NA NA ...
 $ exponentialMovingAverageDiff3 : num  NA NA NA NA NA ...
 $ exponentialMovingAverageDiff4 : num  NA NA NA NA NA NA NA NA NA NA ...
 $ exponentialMovingAverageDiff5 : num  NA NA NA NA NA NA NA NA NA NA ...
 $ exponentialMovingAverageDiff6 : num  NA NA NA NA NA NA NA NA NA NA ...
 $ exponentialMovingAverageDiff7 : num  NA NA NA NA NA NA NA NA NA NA ...
 $ exponentialMovingAverageDiff8 : num  NA NA NA NA NA NA NA NA NA NA ...
 $ exponentialMovingAverageDiff9 : num  NA NA NA NA NA NA NA NA NA NA ...
 $ exponentialMovingAverageDiff10: num  NA NA NA NA NA NA NA NA NA NA ...
 $ GSPC.Close                    : num  1 1 0 0 1 1 1 0 0 0 ...
```

Hide

```
dim(dataset1)
```

```
[1] 391  11
```

Hide

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

```
[1] "exponentialMovingAverageDiff1"
[1] 2
[1] "exponentialMovingAverageDiff2"
[1] 5
[1] "exponentialMovingAverageDiff3"
[1] 8
[1] "exponentialMovingAverageDiff4"
[1] 11
[1] "exponentialMovingAverageDiff5"
[1] 14
[1] "exponentialMovingAverageDiff6"
[1] 17
[1] "exponentialMovingAverageDiff7"
[1] 20
[1] "exponentialMovingAverageDiff8"
[1] 23
[1] "exponentialMovingAverageDiff9"
[1] 26
[1] "exponentialMovingAverageDiff10"
[1] 29
[1] "GSPC.Close"
[1] 0
```

Hide

```
# Removing all rows of missing data
dataset1 = na.omit(dataset1)
#Checking for missing data again
dim(dataset1)
```

```
[1] 362  11
```

Hide

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

```
[1] "exponentialMovingAverageDiff1"
[1] 0
[1] "exponentialMovingAverageDiff2"
[1] 0
[1] "exponentialMovingAverageDiff3"
[1] 0
[1] "exponentialMovingAverageDiff4"
[1] 0
[1] "exponentialMovingAverageDiff5"
[1] 0
[1] "exponentialMovingAverageDiff6"
[1] 0
[1] "exponentialMovingAverageDiff7"
[1] 0
[1] "exponentialMovingAverageDiff8"
[1] 0
[1] "exponentialMovingAverageDiff9"
[1] 0
[1] "exponentialMovingAverageDiff10"
[1] 0
[1] "GSPC.Close"
[1] 0
```

Hide

```
colnames(dataset1)=c ("EMA1", "EMA2", "EMA3", "EMA4", "EMA5", "EMA6", "EMA7", "EMA8", "EMA9", "E
MA10", "Price")
# Encoding the target feature as factor
dataset1$Price=as.factor(dataset1$Price)
# Exploring the data set components
#str(dataset1)
# 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[-11] = scale(training_set[-11])
test_set[-11] = scale(test_set[-11])
# Applying Kernel SVM Model on the Training set
library(e1071)
classifier = svm(formula = Price ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')
summary(classifier)
```

```


Call:
svm(formula = Price ~ ., data = training_set, type = "C-classification",
    kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.1

Number of Support Vectors:  255

 ( 119 136 )


Number of Classes:  2

Levels:
 0 1
```

Hide

```
classifier
```

```

Call:
svm(formula = Price ~ ., data = training_set, type = "C-classification", 
    kernel = "radial")


Parameters:
   SVM-Type:  C-classification 
 SVM-Kernel:  radial 
       cost:  1 
      gamma:  0.1 

Number of Support Vectors:  255
```

Hide

```
# Predicting the Test set results
predict_val = predict(classifier, newdata = test_set[-11])
# Confusion Matrix
cm = table(test_set[, 11], predict_val)
print(cm)
```

```
   predict_val
     0  1
  0  0 30
  1  0 43
```

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] 0.5890411
```