Sharat_Sripada_HW9.R

ssharat

2020-03-15

```
#
#      Course: IST-687
#      Name: Sharat Sripada
#      Homework #9
#      Due Date: 3/15/2020
#      Date Submitted: 3/15/2020
#      Topic: SVMs, Comparing different models - Classification, Regression.
#
# install.packages("kernlab")
# install.packages("gridExtra")
# For KSVM
library(kernlab)
# For SVM
library(e1071)
# For plotttting multiple graphs in one
library(gridExtra)

aq <- data.frame(airquality)

# Replace NAs with mean
ozone_mean <- mean(na.omit(aq$Ozone))
solar_mean <- mean(na.omit(aq$Solar.R))
aq$Ozone[is.na(aq$Ozone)] <- ozone_mean
aq$Solar.R[is.na(aq$Solar.R)] <- solar_mean

dim(aq)
```

```
## [1] 153   6
```

```
randindex <- sample(1:dim(aq)[1])

# By theory, we use 2/3rd data for trainData & 1/3rd
# data for testData.
cutpoint2_3 <- floor(2 * length(randindex) /3)
trainData <- aq[randindex[1:cutpoint2_3],]
testData <- aq[randindex[(cutpoint2_3 + 1):length(randindex)],]

# Build a model using kernel SVM
ksvmoutput <- ksvm(Ozone~., data=trainData,
                  kernel="rbfdot", #kernel function that projects the low
dimensional problem into higher dimensional space
                  kpar="automatic", #params used to control radial function
```

```r
kernel(rbfdot)
                  C=10, #C -> cost of constraints
                  cross=10, #use 10 fold cross-validation in this model
                  prob.model=TRUE)
ksvmoutput
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 10
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.139794036766887
##
## Number of Support Vectors : 91
##
## Objective Function Value : -224.6402
## Training error : 0.176371
## Cross validation error : 484.9765
## Laplace distr. width : 36.97081
```

```r
# Predict data based on data from the model/svmoutput
# & testData
ksvmpredict <- predict(ksvmoutput, testData, type="votes")
str(ksvmpredict)
```

```
##  num [1:51, 1] 43.4 47.6 69.8 46.3 54.7 ...
```

```r
str(testData)
```

```
## 'data.frame':    51 obs. of  6 variables:
##  $ Ozone  : num  42.1 49 135 42.1 85 ...
##  $ Solar.R: num  286 248 269 250 175 ...
##  $ Wind   : num  8.6 9.2 4.1 9.2 7.4 6.9 14.9 13.8 12 4.6 ...
##  $ Temp   : int  78 85 84 92 89 91 91 80 86 87 ...
##  $ Month  : int  6 7 7 6 7 9 7 6 7 8 ...
##  $ Day    : int  1 2 1 12 10 1 14 14 27 6 ...
```

```r
# Create a comparison data-frame that contains the testData for Ozone
# & predicted values using the ksvm() function
compTable <- data.frame(testData[,1], ksvmpredict[,1])
colnames(compTable) <- c('Test', 'Pred')
compTable
```

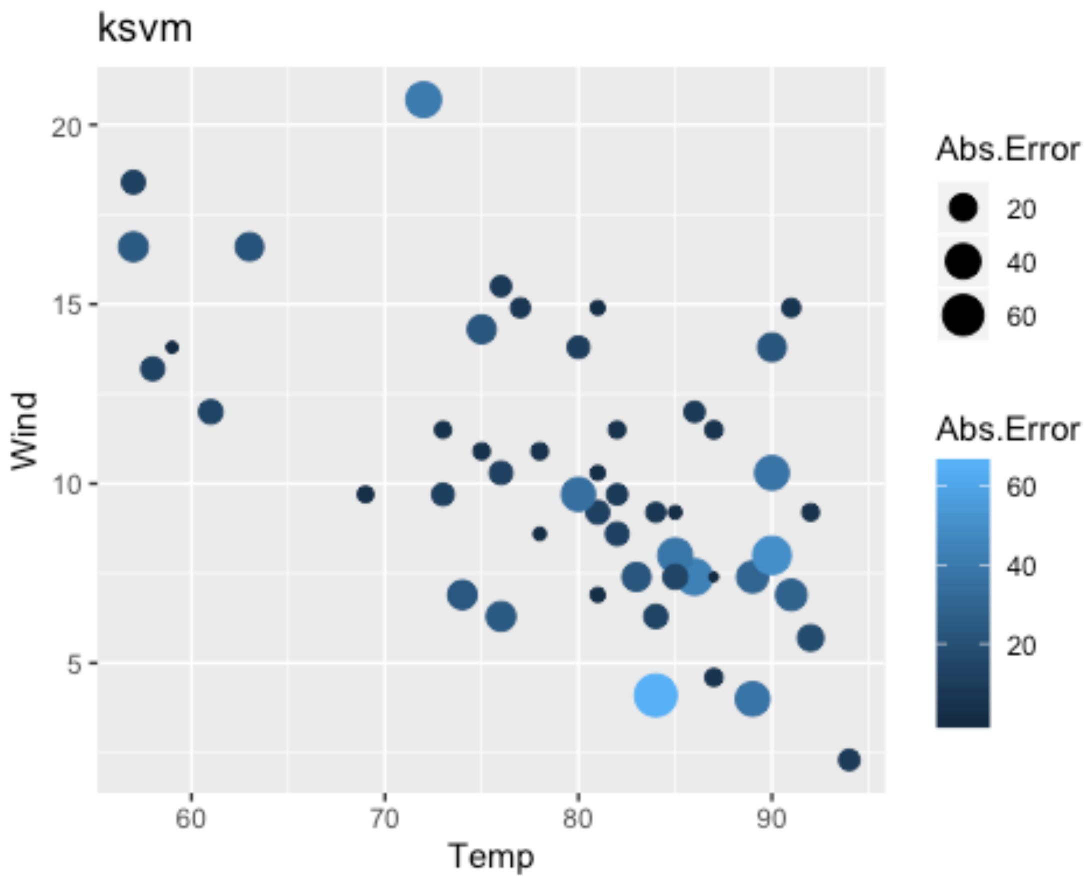
```
##         Test       Pred
## 1   42.12931  43.375591
## 2   49.00000  47.559827
## 3  135.00000  69.830658
## 4   42.12931  46.269756
## 5   85.00000  54.687693
## 6   96.00000  67.108526
```

```
## 7   42.12931  47.767821
## 8   42.12931  31.631938
## 9   52.00000  43.528537
## 10  66.00000  70.700334
## 11  59.00000  71.414737
## 12   6.00000  18.438487
## 13  42.12931  29.785210
## 14  71.00000  48.820658
## 15  28.00000  32.066891
## 16  39.00000  43.933669
## 17  16.00000  34.841513
## 18  14.00000  -7.590209
## 19  29.00000  20.416271
## 20  42.12931  66.847192
## 21  37.00000  -3.647420
## 22  12.00000  15.768222
## 23  42.12931  35.251958
## 24  27.00000  24.959515
## 25  89.00000  51.526846
## 26  50.00000  93.475228
## 27 108.00000  69.613389
## 28  35.00000  49.211113
## 29  61.00000  74.102091
## 30  21.00000  29.366063
## 31  47.00000  47.371721
## 32 110.00000  59.994114
## 33  18.00000   5.387649
## 34  97.00000  79.554614
## 35  65.00000  28.962253
## 36  39.00000  36.840058
## 37  24.00000  13.334442
## 38  19.00000  18.239346
## 39  36.00000  38.201021
## 40  14.00000  36.701197
## 41  42.12931  17.591625
## 42  16.00000  19.364467
## 43  21.00000  17.136250
## 44   7.00000  30.532234
## 45  21.00000  14.168876
## 46 118.00000 109.197284
## 47 122.00000  84.688440
## 48  13.00000  24.956581
## 49  64.00000  86.728997
## 50  32.00000  18.566006
## 51  31.00000  35.003357
```

```r
# Calculate the root mean square error(RMSE)
sqrt(mean((compTable$Test - compTable$Pred) ^ 2))
```

```
## [1] 21.75754
```

```
# RMSE=17.72

# Compute absolute error
compTable$error <- abs(compTable$Test - compTable$Pred)

# Create a new data-frame with error, temp, wind data
ksvmPlot <- data.frame(compTable$error, testData$Temp, testData$Wind)

# Assign column names
colnames(ksvmPlot) <- c('Abs.Error', 'Temp', 'Wind')

# Plot the data-frame using ggplot
library(ggplot2)

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha

ksvm_ggplot <- ggplot(ksvmPlot, aes(x=Temp, y=Wind)) +
geom_point(aes(size=Abs.Error, color=Abs.Error)) +
  ggtitle("ksvm")

ksvm_ggplot
```




```
# Build a model using SVM
svmoutput <- svm(Ozone~., data=trainData, kernel="linear", cost=10,
scale=FALSE)

svmoutput

##
## Call:
## svm(formula = Ozone ~ ., data = trainData, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.2
##     epsilon:  0.1
##
##
## Number of Support Vectors:  102
```

```r
# Predict data based on data from the model/svmoutput
# & testData
svmpredict <- predict(svmoutput, testData, type="votes")
str(svmpredict)
```

```
##  Named num [1:51] 45.1 52.6 59.7 67.3 60 ...
##  - attr(*, "names")= chr [1:51] "32" "63" "62" "43" ...
```

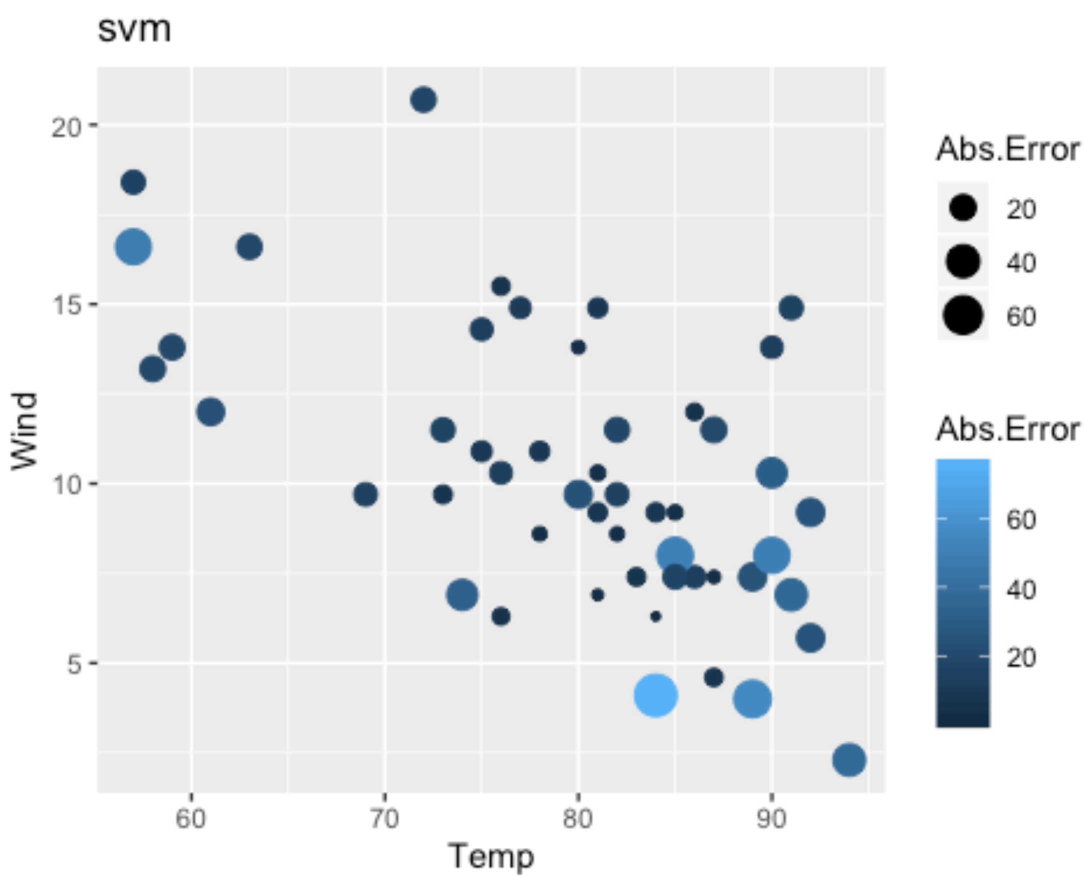
```r
str(testData)
```

```
## 'data.frame':    51 obs. of  6 variables:
##  $ Ozone  : num  42.1 49 135 42.1 85 ...
##  $ Solar.R: num  286 248 269 250 175 ...
##  $ Wind   : num  8.6 9.2 4.1 9.2 7.4 6.9 14.9 13.8 12 4.6 ...
##  $ Temp   : int  78 85 84 92 89 91 91 80 86 87 ...
##  $ Month  : int  6 7 7 6 7 9 7 6 7 8 ...
##  $ Day    : int  1 2 1 12 10 1 14 14 27 6 ...
```

```r
# Create a comparison data-frame that contains the testData for Ozone
# & predicted values using the ksvm() function
svm_compTable <- data.frame(testData[,1], svmpredict)
colnames(svm_compTable) <- c('Test', 'Pred')
svm_compTable
```

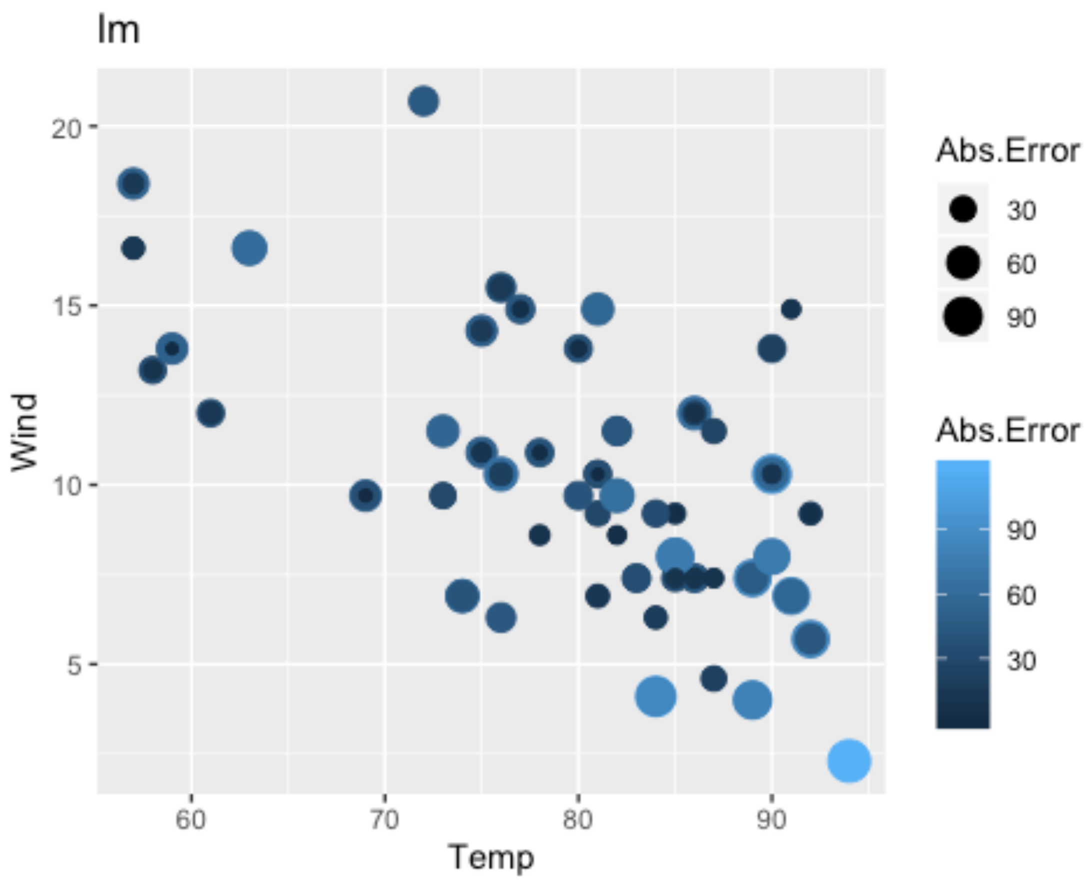
```
##          Test       Pred
## 32   42.12931 45.1162995
## 63   49.00000 52.6085679
## 62  135.00000 59.6650352
## 43   42.12931 67.3179123
## 71   85.00000 59.9760637
## 124  96.00000 59.1940795
## 75   42.12931 57.5915310
## 45   42.12931 44.6387775
## 88   52.00000 46.5951329
## 98   66.00000 59.3965363
## 92   59.00000 51.2791765
## 18    6.00000 -9.5435746
## 72   42.12931 45.1957182
## 40   71.00000 58.2745410
## 105  28.00000 45.6846003
## 41   39.00000 58.5617586
## 82   16.00000 30.4315028
## 148  14.00000 -4.2525130
## 38   29.00000 43.7276148
## 55   42.12931 47.6866285
## 48   37.00000 19.1812656
## 50   12.00000 27.9416380
## 35   42.12931 49.9525243
## 74   27.00000 35.7766872
## 100  89.00000 57.7723400
## 90   50.00000 62.8641110
```

```
## 86  108.00000 57.2466152
## 97   35.00000 51.6148339
## 79   61.00000 59.8800366
## 135  21.00000 27.3967663
## 128  47.00000 49.2755839
## 101 110.00000 60.5048312
## 15   18.00000 -0.9514171
## 70   97.00000 71.7696062
## 106  65.00000 40.0856537
## 93   39.00000 40.4736352
## 133  24.00000 31.1393062
## 8    19.00000  0.1118186
## 146  36.00000 40.0384354
## 151  14.00000 27.2466104
## 25   42.12931 -6.1082287
## 12   16.00000 30.6618164
## 132  21.00000 31.0582289
## 11    7.00000 39.8156187
## 47   21.00000 31.9397794
## 121 118.00000 80.1098041
## 99  122.00000 66.9167709
## 141  13.00000 25.9028414
## 91   64.00000 57.1245987
## 24   32.00000  8.5793971
## 111  31.00000 39.7720197

# Calculate the root mean square error(RMSE)
sqrt(mean((svm_compTable$Test - svm_compTable$Pred) ^ 2))

## [1] 23.90929

# RMSE=19.47

# Compute absolute error
svm_compTable$error <- abs(svm_compTable$Test - svm_compTable$Pred)

# Create a new data-frame with error, temp, wind data
svmPlot <- data.frame(svm_compTable$error, testData$Temp, testData$Wind)

# Assign column names
colnames(svmPlot) <- c('Abs.Error', 'Temp', 'Wind')

# Plot the data-frame using ggplot
svm_ggplot <- ggplot(svmPlot, aes(x=Temp, y=Wind)) +
geom_point(aes(size=Abs.Error, color=Abs.Error)) +
  ggtitle("svm")

svm_ggplot
```

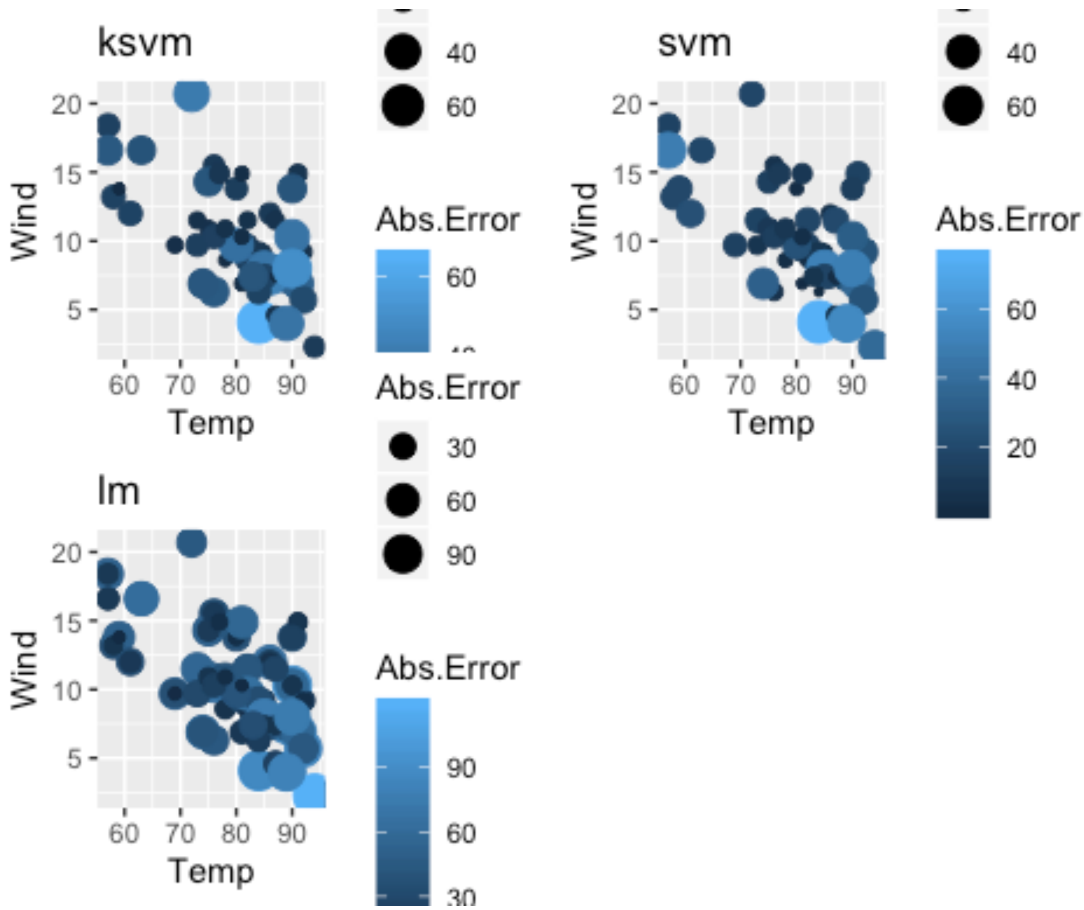



```
# Build a model using liner regression (lm function)
lmoutput <- lm(formula=Ozone~., data=testData)
lm_test <- data.frame(Solar.R=aq$Solar.R, Wind=aq$Wind,
                   Temp=aq$Temp, Month=aq$Month, Day=aq$Day)

lmpredict <- predict(lmoutput, lm_test, type="response")

# Create a comparison data-frame that contains the testData for Ozone
# & predicted values using the lm() function
lm_compTable <- data.frame(testData[,1], lmpredict)
colnames(lm_compTable) <- c('Test', 'Pred')

# Calculate the root mean square error(RMSE)
sqrt(mean((lm_compTable$Test - lm_compTable$Pred) ^ 2))

## [1] 39.9913

# RMSE=29.68

# Compute absolute error
lm_compTable$error <- abs(lm_compTable$Test - lm_compTable$Pred)

# Create a new data-frame with error, temp, wind data
```

```r
lmPlot <- data.frame(lm_compTable$error, testData$Temp, testData$Wind)

# Assign column names
colnames(lmPlot) <- c('Abs.Error', 'Temp', 'Wind')

# Plot the data-frame using ggplot
lm_ggplot <- ggplot(lmPlot, aes(x=Temp, y=Wind)) +
geom_point(aes(size=Abs.Error, color=Abs.Error)) +
  ggtitle("lm")

lm_ggplot
```




```r
# Conclusion:
#   - RMSE for ksvm(17.72) is lower than RMSE for svm(19.47) & lm(29.68)
#   - Plotting the abs. error also showed a higher range & number for lm model
(kvm and svm are comparable)
# For the given data-set, KSVM is a marginally better algorithm than svm &
way better than lm

# Using gridExtra to represent graphs in one plane
grid.arrange(ksvm_ggplot, svm_ggplot, lm_ggplot, nrow=2)
```

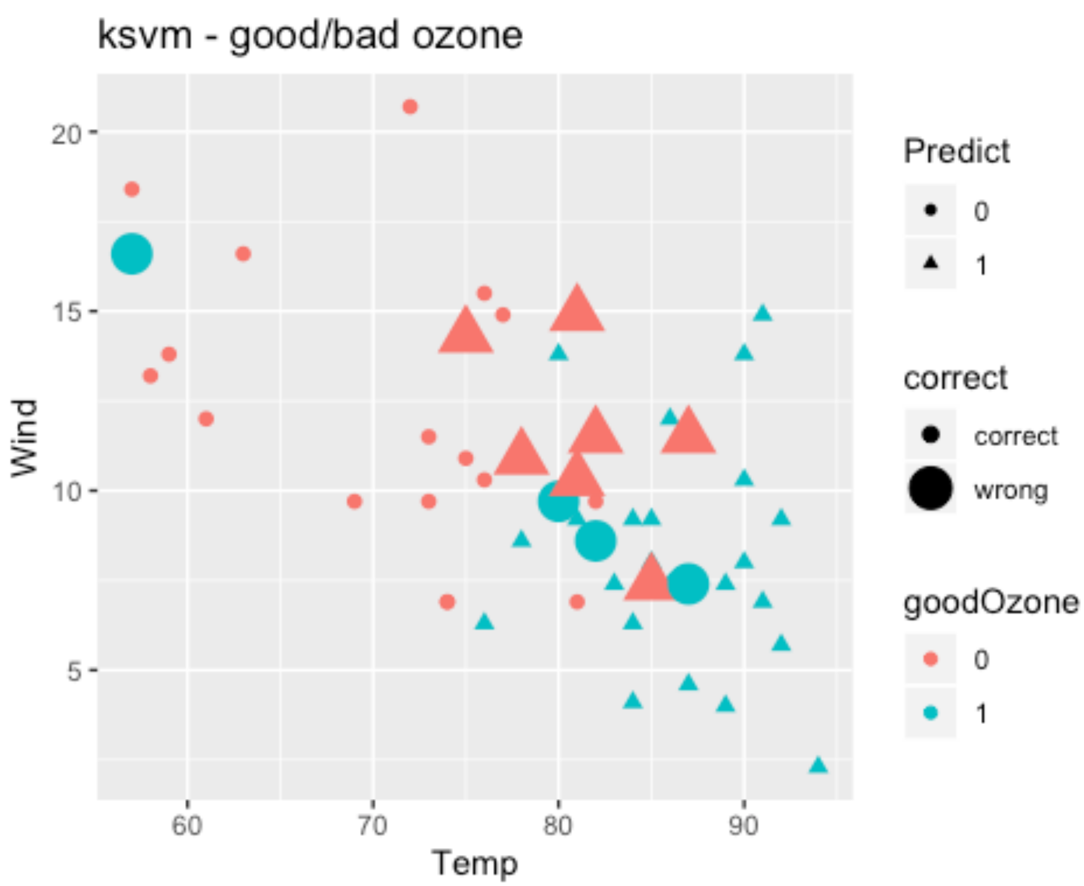



```r
# Moving now to classification based algorithms.
#  - classification based algorithms predict with 0/1
#  - regression/linear based algorithms (previous section) predict a value

# Creating a new var goodOzone: if Ozone >= meanOzone then 1 else 0
trainData$goodOzone <- ifelse(trainData$Ozone < ozone_mean, 0, 1)
testData$goodOzone <- ifelse(testData$Ozone < ozone_mean, 0, 1)

# Remove Ozone from trainData & testData
trainData <- trainData[,-1]
testData <- testData[,-1]
trainData$goodOzone <- as.factor(trainData$goodOzone)
testData$goodOzone <- as.factor(testData$goodOzone)

# Build a model based on ksvm
ksvmgood <- ksvm(goodOzone~., data=trainData,
                   kernel="rbfdot", #kernel function that projects the low
dimensional problem into higher dimensional space
                   kpar="automatic", #params used to control radial function
kernel(rbfdot)
                   C=10, #C -> cost of constraints
                   cross=10, #use 10 fold cross-validation in this model
```

```
                    prob.model=TRUE)
ksvmgood

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 10
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.176126877372317
##
## Number of Support Vectors : 58
##
## Objective Function Value : -321.4427
## Training error : 0.107843
## Cross validation error : 0.344545
## Probability model included.

# Predict data based on data from the model/svmoutput
# & testData
ksvm_goodPred <- predict(ksvmgood, testData)
ksvm_goodPred # This should yield a 0/1

##  [1] 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 0 1 0
0 0 0
## [39] 1 1 0 0 0 0 0 1 1 0 1 0 1
## Levels: 0 1

# Create a comparison data-frame that contains the testData for Ozone
# & predicted values using the ksvm() function
ksvm_goodcompTable <- data.frame(testData[,6], ksvm_goodPred)
colnames(ksvm_goodcompTable) <- c('Test', 'Pred')
ksvm_goodcompTable

##    Test Pred
## 1     1    1
## 2     1    1
## 3     1    1
## 4     1    1
## 5     1    1
## 6     1    1
## 7     1    1
## 8     1    1
## 9     1    1
## 10    1    1
## 11    1    1
## 12    0    0
## 13    1    0
## 14    1    1
## 15    0    1
## 16    0    1
```

```
## 17    0    0
## 18    0    0
## 19    0    0
## 20    1    1
## 21    0    0
## 22    0    0
## 23    1    1
## 24    0    1
## 25    1    1
## 26    1    1
## 27    1    1
## 28    0    1
## 29    1    1
## 30    0    0
## 31    1    0
## 32    1    1
## 33    0    0
## 34    1    1
## 35    1    0
## 36    0    0
## 37    0    0
## 38    0    0
## 39    0    1
## 40    0    1
## 41    1    0
## 42    0    0
## 43    0    0
## 44    0    0
## 45    0    0
## 46    1    1
## 47    1    1
## 48    0    0
## 49    1    1
## 50    0    0
## 51    0    1

# Calculate the percentage of correct values (this is different from the
# linear/regression models where we calculate RMSE)
percentage_ksvm <- length(which(ksvm_goodcompTable$Test ==
ksvm_goodcompTable$Pred))/dim(ksvm_goodcompTable)[1]
percentage_ksvm

## [1] 0.7843137

# Pecentage = 0.6862

# Confusion matrix
results <- table(Test=ksvm_goodcompTable$Test, Pred=ksvm_goodcompTable$Pred)
print(results)
```

```
##      Pred
## Test  0  1
##    0 17  7
##    1  4 23

# Plot the results
ksvm_goodcompTable$correct <-
ifelse(ksvm_goodcompTable$Test==ksvm_goodcompTable$Pred,"correct","wrong")
plot_ksvm <- data.frame(ksvm_goodcompTable$correct,
                        testData$Temp,
                        testData$Wind,
                        testData$goodOzone,
                        ksvm_goodcompTable$Pred)

colnames(plot_ksvm) <- c("correct","Temp","Wind","goodOzone","Predict")
ksvm_ggplot <- ggplot(plot_ksvm, aes(x=Temp,y=Wind)) +
  geom_point(aes(size=correct,color=goodOzone,shape = Predict))+
  ggtitle("ksvm - good/bad ozone")
ksvm_ggplot

## Warning: Using size for a discrete variable is not advised.
```




```
# Build a model based on svm
svmgood <- svm(goodOzone~., data=trainData, kernel="linear", cost=10,
```

```
scale=FALSE)
svmgood

##
## Call:
## svm(formula = goodOzone ~ ., data = trainData, kernel = "linear",
##     cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  52

# Predict data based on data from the model/svmoutput
# & testData
svm_goodPred <- predict(svmgood, testData)
svm_goodPred # This should yield a 0/1

##  32  63  62  43  71 124  75  45  88  98  92  18  72  40 105  41  82 148
38  55
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
##  48  50  35  74 100  90  86  97  79 135 128 101  15  70 106  93 133   8
146 151
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
##  25  12 132  11  47 121  99 141  91  24 111
##   0   0   0   0   0   0   0   0   0   0   0
## Levels: 0 1

# Create a comparison data-frame that contains the testData for Ozone
# & predicted values using the ksvm() function
svm_goodcompTable <- data.frame(testData[,6], svm_goodPred)
colnames(svm_goodcompTable) <- c('Test', 'Pred')
svm_goodcompTable

##     Test Pred
## 32     1    0
## 63     1    0
## 62     1    0
## 43     1    0
## 71     1    0
## 124    1    0
## 75     1    0
## 45     1    0
## 88     1    0
## 98     1    0
## 92     1    0
```

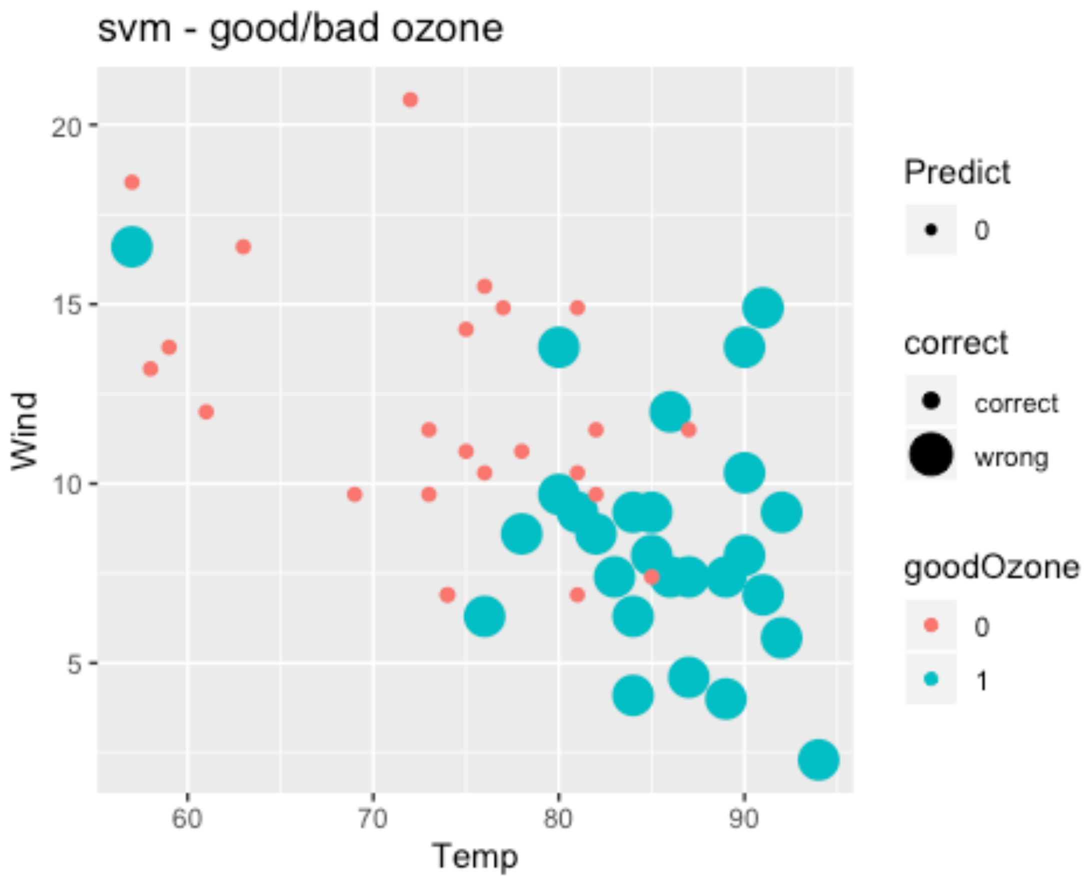
```
## 18      0    0
## 72      1    0
## 40      1    0
## 105     0    0
## 41      0    0
## 82      0    0
## 148     0    0
## 38      0    0
## 55      1    0
## 48      0    0
## 50      0    0
## 35      1    0
## 74      0    0
## 100     1    0
## 90      1    0
## 86      1    0
## 97      0    0
## 79      1    0
## 135     0    0
## 128     1    0
## 101     1    0
## 15      0    0
## 70      1    0
## 106     1    0
## 93      0    0
## 133     0    0
## 8       0    0
## 146     0    0
## 151     0    0
## 25      1    0
## 12      0    0
## 132     0    0
## 11      0    0
## 47      0    0
## 121     1    0
## 99      1    0
## 141     0    0
## 91      1    0
## 24      0    0
## 111     0    0

# Calculate the percentage of correct values (this is different from the
# linear/regression models where we calculate RMSE)
percentage_svm <- length(which(svm_goodcompTable$Test ==
svm_goodcompTable$Pred))/dim(svm_goodcompTable)[1]
percentage_svm

## [1] 0.4705882
```

```r
# Percentage = 0.80392

# Confusion matrix
results <- table(Test=svm_goodcompTable$Test, Pred=svm_goodcompTable$Pred)
print(results)
```

```
##     Pred
## Test  0  1
##    0 24  0
##    1 27  0
```

```r
# Plot the results
svm_goodcompTable$correct <-
ifelse(svm_goodcompTable$Test==svm_goodcompTable$Pred,"correct","wrong")
plot_svm <- data.frame(svm_goodcompTable$correct,
                        testData$Temp,
                        testData$Wind,
                        testData$goodOzone,
                        svm_goodcompTable$Pred)

colnames(plot_svm) <- c("correct","Temp","Wind","goodOzone","Predict")
svm_ggplot <- ggplot(plot_svm, aes(x=Temp,y=Wind)) +
  geom_point(aes(size=correct,color=goodOzone,shape = Predict))+
  ggtitle("svm - good/bad ozone")
svm_ggplot
```

```
## Warning: Using size for a discrete variable is not advised.
```




```
# Build a model based on Naive Bayes algorithm
nbgood <- svm(goodOzone~., data=trainData)
nbgood

##
## Call:
## svm(formula = goodOzone ~ ., data = trainData)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  73

# Predict data based on data from the model/svmoutput
# & testData
nb_goodPred <- predict(nbgood, testData)
nb_goodPred # This should yield a 0/1

##  32  63  62  43  71 124  75  45  88  98  92  18  72  40 105  41  82 148
38  55
##   0   1   1   1   1   1   1   0   1   1   1   0   1   1   1   1   0   0
```

```
1   1
##  48  50  35  74 100  90  86  97  79 135 128 101  15  70 106  93 133   8
146 151
##   0   0   1   1   1   1   1   1   1   0   1   1   0   1   1   0   0   0
0   0
##  25  12 132  11  47 121  99 141  91  24 111
##   0   0   0   0   0   1   1   0   1   0   1
## Levels: 0 1

# Create a comparison data-frame that contains the testData for Ozone
# & predicted values using the ksvm() function
nb_goodcompTable <- data.frame(testData[,6], nb_goodPred)
colnames(nb_goodcompTable) <- c('Test', 'Pred')
nb_goodcompTable

##     Test Pred
## 32     1    0
## 63     1    1
## 62     1    1
## 43     1    1
## 71     1    1
## 124    1    1
## 75     1    1
## 45     1    0
## 88     1    1
## 98     1    1
## 92     1    1
## 18     0    0
## 72     1    1
## 40     1    1
## 105    0    1
## 41     0    1
## 82     0    0
## 148    0    0
## 38     0    1
## 55     1    1
## 48     0    0
## 50     0    0
## 35     1    1
## 74     0    1
## 100    1    1
## 90     1    1
## 86     1    1
## 97     0    1
## 79     1    1
## 135    0    0
## 128    1    1
## 101    1    1
## 15     0    0
## 70     1    1
```

```
## 106    1    1
## 93     0    0
## 133    0    0
## 8      0    0
## 146    0    0
## 151    0    0
## 25     1    0
## 12     0    0
## 132    0    0
## 11     0    0
## 47     0    0
## 121    1    1
## 99     1    1
## 141    0    0
## 91     1    1
## 24     0    0
## 111    0    1
```

```
# Calculate the percentage of correct values (this is different from the
# linear/regression models where we calculate RMSE)
percentage_nb <- length(which(nb_goodcompTable$Test ==
nb_goodcompTable$Pred))/dim(nb_goodcompTable)[1]
percentage_nb
```

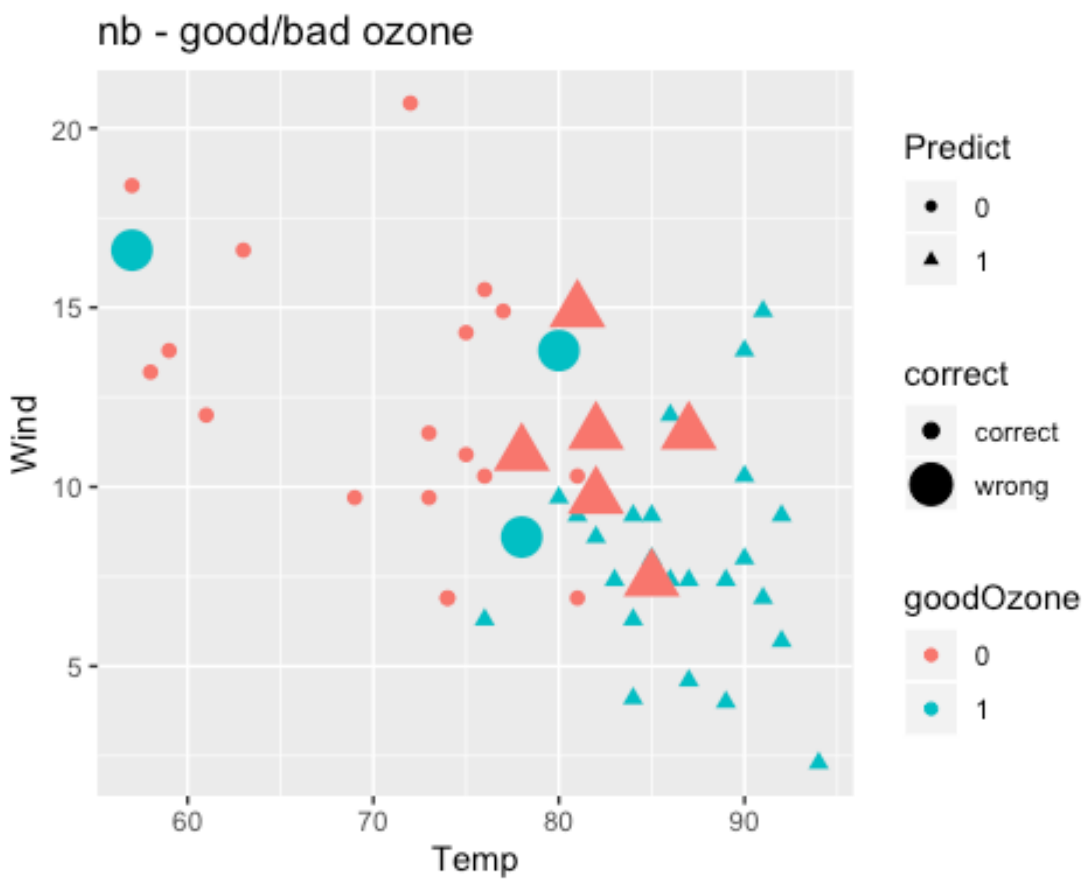
```
## [1] 0.8235294
```

```
# Percentage = 0.7843

# Confusion matrix
results <- table(Test=nb_goodcompTable$Test, Pred=nb_goodcompTable$Pred)
print(results)
```

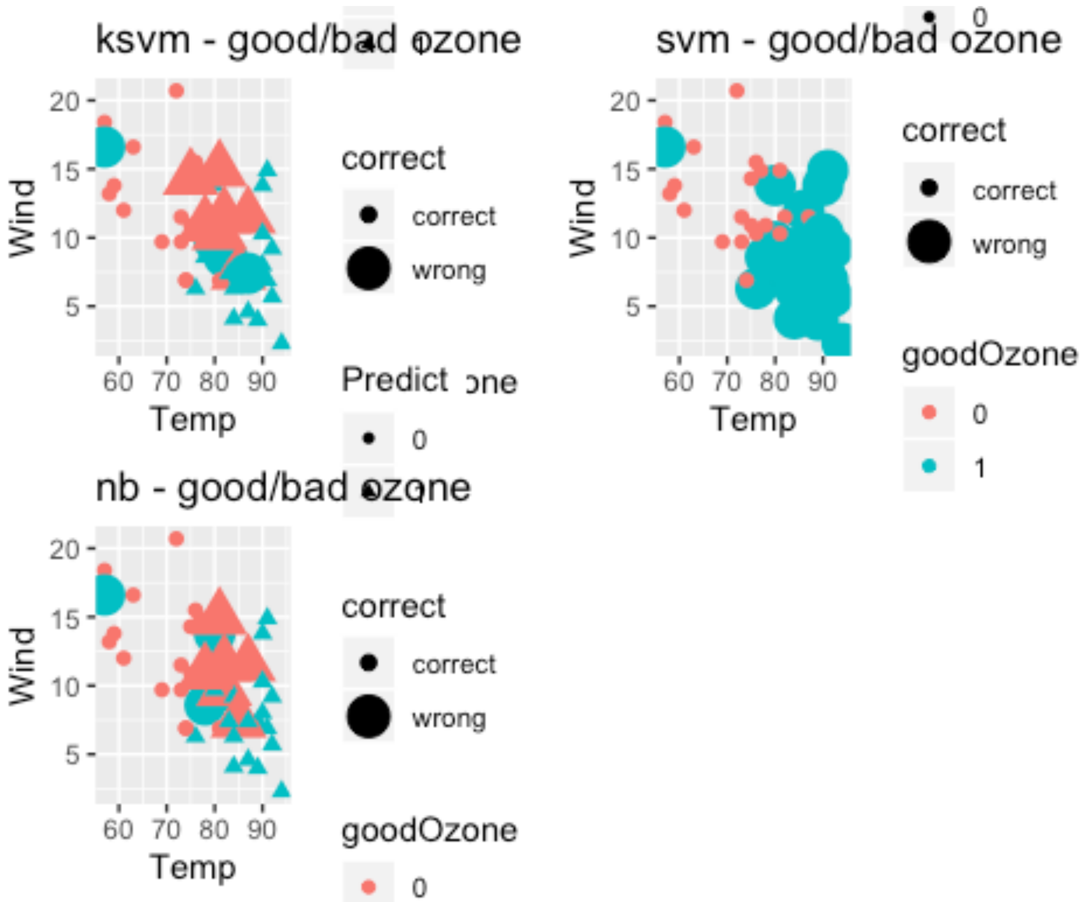
```
##      Pred
## Test  0  1
##    0 18  6
##    1  3 24
```

```
# Plot the results
nb_goodcompTable$correct <-
ifelse(nb_goodcompTable$Test==nb_goodcompTable$Pred,"correct","wrong")
plot_nb <- data.frame(nb_goodcompTable$correct,
                       testData$Temp,
                       testData$Wind,
                       testData$goodOzone,
                       nb_goodcompTable$Pred)

colnames(plot_nb) <- c("correct","Temp","Wind","goodOzone","Predict")
nb_ggplot <- ggplot(plot_nb, aes(x=Temp,y=Wind)) +
  geom_point(aes(size=correct,color=goodOzone,shape = Predict))+
  ggtitle("nb - good/bad ozone")
nb_ggplot
```

```
## Warning: Using size for a discrete variable is not advised.
```




```
# Conclusion:
  #  - Percentage of accuracy for svm(80%) is higher than ksvm(68%) & nb(78%)
  # For the given data-set, SVM is a better algorithm than KSVM & Naive Bayes

# Using gridExtra to represent graphs in one plane
grid.arrange(ksvm_ggplot, svm_ggplot, nb_ggplot, nrow=2)

## Warning: Using size for a discrete variable is not advised.

## Warning: Using size for a discrete variable is not advised.

## Warning: Using size for a discrete variable is not advised.
```


ksvm - good/bad ozone
Wind
Temp
correct
correct
wrong
svm - good/bad ozone
Wind
Temp
correct
correct
wrong
goodOzone
0
1
nb - good/bad ozone
Wind
Temp
correct
correct
wrong
goodOzone
0