Support Vector Machines Lab

John Fields

6/3/2019

‘‘‘

Step 1: Load the data

Let go back and analyze the air quality dataset (if you remember, we used that previously, in the visualization lab). Remember to think about how to deal with the NAs in the data.

```
data(airquality)
str(airquality)
```

```
## 'data.frame':    153 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
##  $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
##  $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
##  $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
aq <-na.omit(airquality)
str(aq)
```

```
## 'data.frame':    111 obs. of  6 variables:
##  $ Ozone  : int  41 36 12 18 23 19 8 16 11 14 ...
##  $ Solar.R: int  190 118 149 313 299 99 19 256 290 274 ...
##  $ Wind   : num  7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...
##  $ Temp   : int  67 72 74 62 65 59 61 69 66 68 ...
##  $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ Day    : int  1 2 3 4 7 8 9 12 13 14 ...
##  - attr(*, "na.action")= 'omit' Named int  5 6 10 11 25 26 27 32 33 34 ...
##   ..- attr(*, "names")= chr  "5" "6" "10" "11" ...
```

Step 2: Create train and test data sets

Using techniques discussed in class, create two datasets – one for training and one for testing.

```
#install.packages("kernlab")
library(kernlab)
randIndex <- sample(1:dim(aq)[1])#create list/vector variable-random index
summary(randIndex)#verify indicies in randIndex
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    28.5    56.0    56.0    83.5   111.0
```

```
length(randIndex)#verify indicies in randIndex
```

```
## [1] 111
```

```
head(randIndex)#look at first few cases
```

```
## [1]  80  97  50 111  85  23
```

```
cutPoint2_3 <- floor(2*dim(aq)[1]/3)#create 2/3 cutpoint
cutPoint2_3#verify 2/3 cutpoint
```

```
## [1] 74
```

```
trainData <- aq[randIndex[1:cutPoint2_3],]#create training data set
testData <- aq[randIndex[(cutPoint2_3+1):dim(aq)[1]],]#create test data set
```

Step 3: Build a Model using KSVM & visualize the results

1) Build a model (using the 'ksvm' function, trying to predict onzone). You can use all the possible attributes, or select the attributes that you think would be the most helpful.
2) Test the model on the testing dataset, and compute the Root Mean Squared Error
3) Plot the results. Use a scatter plot. Have the x-axis represent temperature, the y-axis represent wind, the point size and color represent the error, as defined by the actual ozone level minus the predicted ozone level).
4) Compute models and plot the results for 'svm' (in the e1071 package) and 'lm'. Generate similar charts for each model
5) Show all three results (charts) in one window, using the grid.arrange function

```
#model ksvm
svmOutput <- ksvm(Ozone ~.,data=trainData,kernel="rbfdot",kpar="automatic",C=5,cross=3,prob.model=TRUE)
svmOutput
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.204438020463247
##
## Number of Support Vectors : 55
##
## Objective Function Value : -54.963
## Training error : 0.102088
## Cross validation error : 465.7656
## Laplace distr. width : 42.18049
```

```
svmPred <- predict(svmOutput, testData,type="votes")

#copy predicted and actual to new df
results <- data.frame(svmPred,testData$Ozone)
results_col_names <- c("Predicted","Actual")
colnames(results) <- results_col_names

#calculate the root mean squared error (help from Courtney Smith with this section)
svmPred.error <- (results$Actual - results$Predicted)
library(Metrics)
rmse(results$Actual,results$Predicted)
```

```
## [1] 25.4795
```

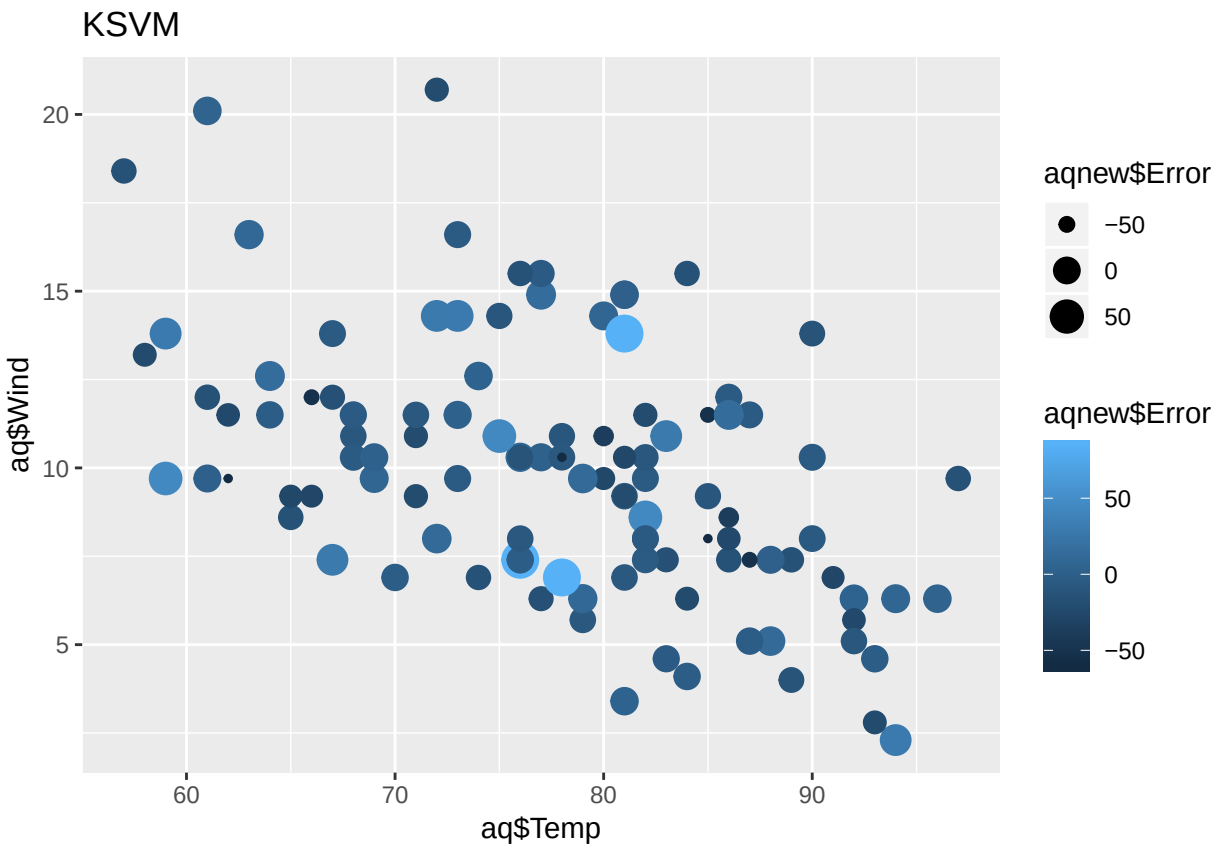
```
#plot the results for ksvm
aqnew <- data.frame(aq$Wind,aq$Temp,svmPred.error)
colnames(aqnew) <-c("Wind","Temp","Error")
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
##
## Attaching package: 'ggplot2'
```

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

```
plotaq <- ggplot(data=aqnew,aes(x=aq$Temp,y=aq$Wind))+ geom_point(aes(size=aqnew$Error, color=aqnew$Erro
plotaq
```

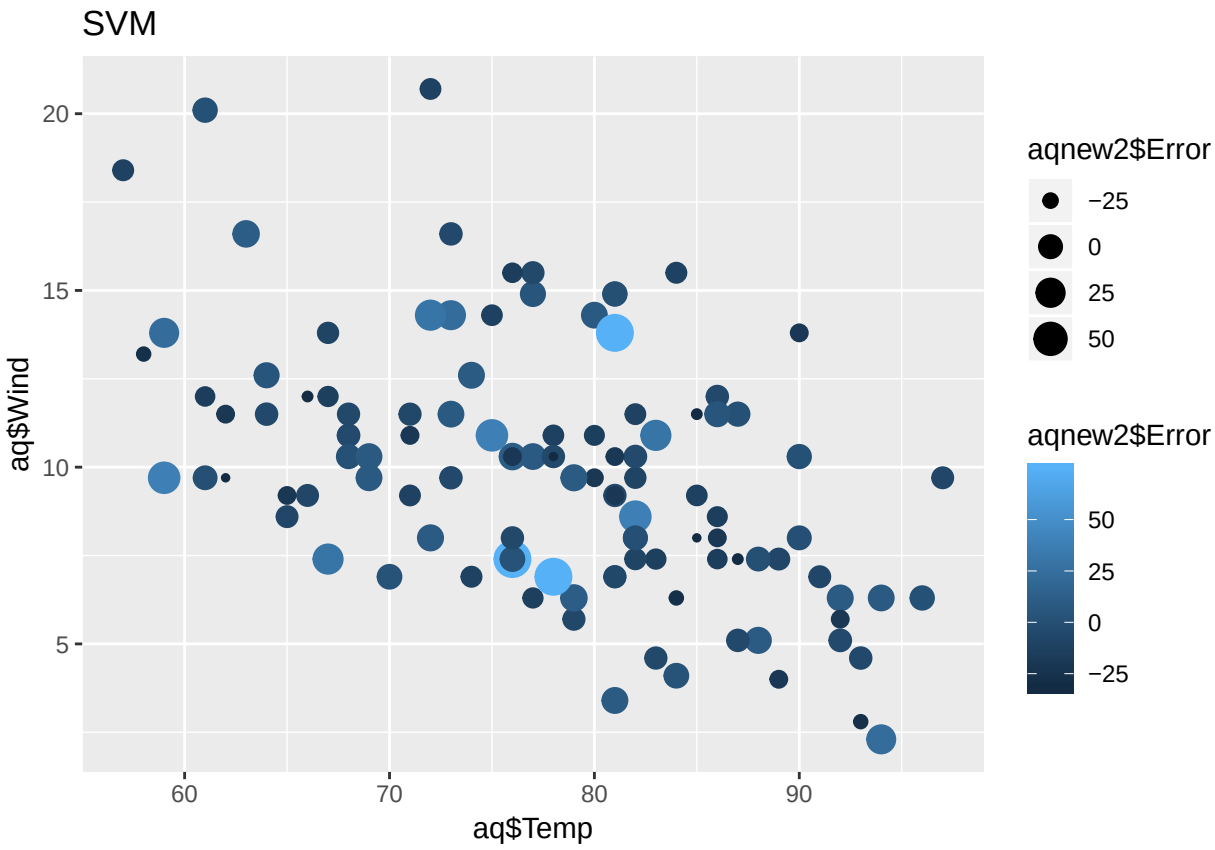



```
#compute models using e1071 svm
library(e1071)
svmOutput2 <- svm(Ozone ~.,data=trainData)
svmPred2 <- predict(svmOutput2, testData,type="votes")

#copy predicted and actual to new df for e1071
results2 <- data.frame(svmPred2,testData$Ozone)
results_col_names2 <- c("Predicted","Actual")
```

```
colnames(results2) <- results_col_names2

#calculate the root mean squared error (help from Courtney Smith with this section) for e1071
svmPred.error2 <- (results2$Actual - results2$Predicted)
library(Metrics)
rmse(results2$Actual,results2$Predicted)
```

```
## [1] 20.00264
```

```
#plot the results for e1071
aqnew2 <- data.frame(aq$Wind,aq$Temp,svmPred.error2)
colnames(aqnew2) <-c("Wind","Temp","Error")
library(ggplot2)
plotaq2 <- ggplot(data=aqnew2,aes(x=aq$Temp,y=aq$Wind))+ geom_point(aes(size=aqnew2$Error, color=aqnew2$
plotaq2
```

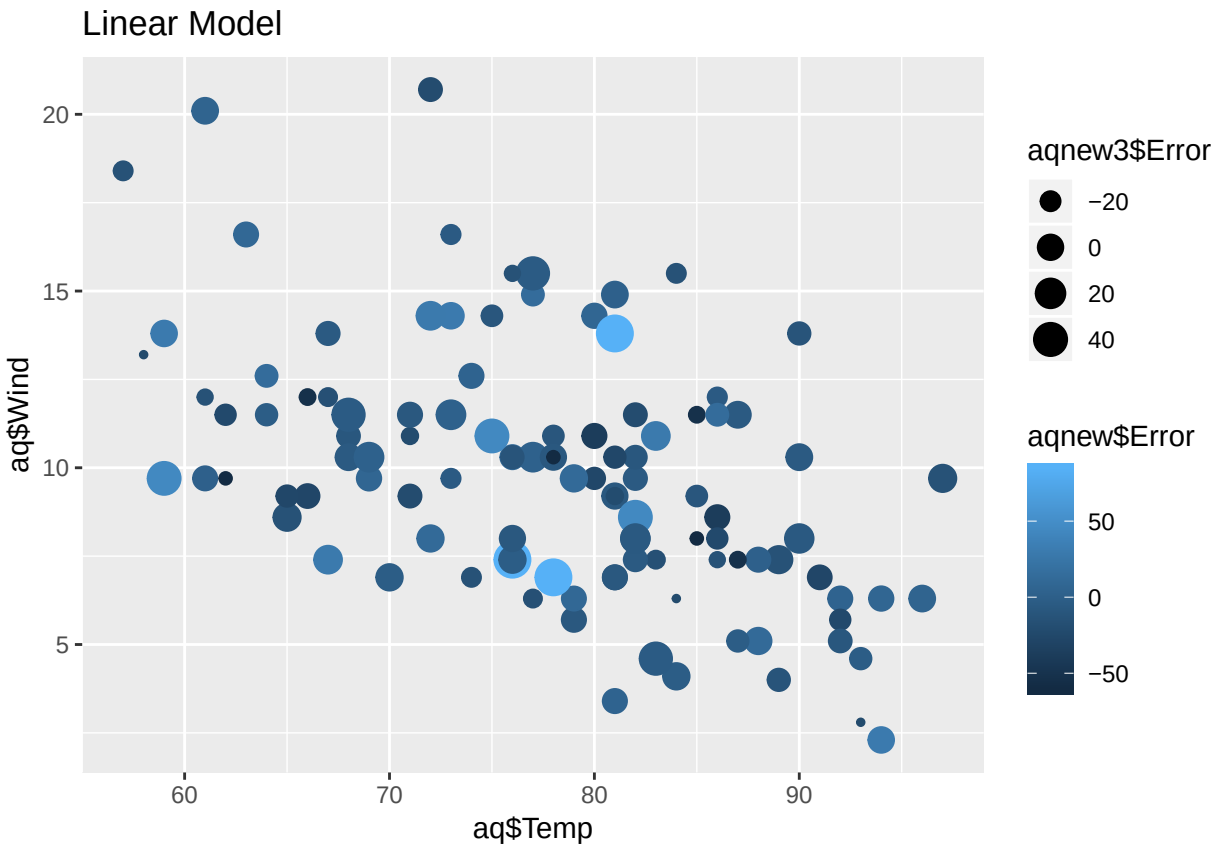



```
#compute models using lm
svmOutput3 <- lm(Ozone ~.,data=trainData)
svmPred3 <- predict(svmOutput3, testData)

#copy predicted and actual to new df for lm
results3 <- data.frame(svmPred3,testData$Ozone)
results_col_names3 <- c("Predicted","Actual")
colnames(results3) <- results_col_names3

#calculate the root mean squared error (help from Courtney Smith with this section) for lm
svmPred.error3 <- (results3$Actual - results3$Predicted)
library(Metrics)
```

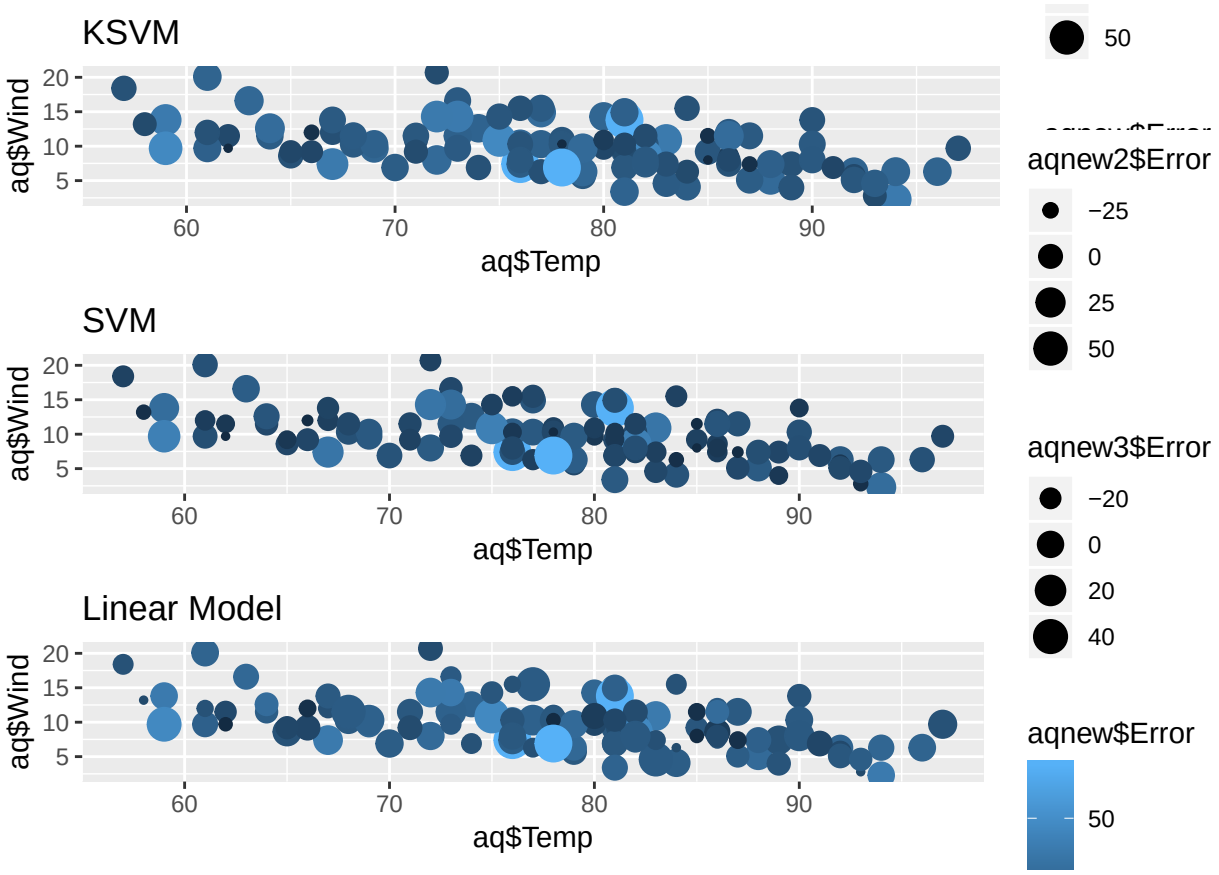
```
rmse(results3$Actual,results3$Predicted)
```

```
## [1] 20.72248
```

```
#plot the results for lm
aqnew3 <- data.frame(aq$Wind,aq$Temp,svmPred.error3)
colnames(aqnew3) <-c("Wind","Temp","Error")
library(ggplot2)
plotaq3 <- ggplot(data=aqnew3,aes(x=aq$Temp,y=aq$Wind))+ geom_point(aes(size=aqnew3$Error, color=aqnew$
plotaq3
```




```
#plot all three results in one window, using the grid arrange function
#install.packages("gridExtra")
library(gridExtra)
grid.arrange(plotaq,plotaq2,plotaq3)
```



Step 4: Create a 'goodOzone' variable This variable should be either 0 or 1. It should be 0 if the ozone is below the average for all the data observations, and 1 if it is equal to or above the average ozone observed.

```
#Thanks to Ryan Fisher and Courtney Smith for help with this section
aq$goodOzone <- ifelse(aq$Ozone >= mean(aq$Ozone),1,0)

#create a new train and test data set with the new variable
library(kernlab)
randIndex2 <- sample(1:dim(aq)[1])#create list/vector variable-random index
summary(randIndex2)#verify indicies in randIndex
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    28.5    56.0    56.0    83.5   111.0
```

```
length(randIndex2)#verify indicies in randIndex
```

```
## [1] 111
```

```
head(randIndex2)#look at first few cases
```

```
## [1] 23 56 20 72 68 82
```

```
cut2Point2_3 <- floor(2*dim(aq)[1]/3)#create 2/3 cutpoint
cut2Point2_3#verify 2/3 cutpoint
```

```
## [1] 74
```

```
trainData2 <- aq[randIndex2[1:cut2Point2_3],]#create training data set
testData2 <- aq[randIndex2[(cut2Point2_3+1):dim(aq)[1]],]#create test data set
```

Step 5: See if we can do a better job predicting 'good' and 'bad' days 1) Build a model (using the 'ksvm' function, trying to predict 'goodOzone'). You can use all the possible attributes, or select the attributes that you think would be the most helpful. 2) Test the model on the testing dataset, and compute the percent of 'goodOzone' that was correctly predicted. 3) Plot the results. Use a scatter plot. Have the x-axis represent temperature, the y-axis represent wind, the shape representing what was predicted (good or bad day), the color representing the actual value of 'goodOzone' (i.e. if the actual ozone level was good) and the size represent if the prediction was correct (larger symbols should be the observations the model got wrong). 4) Compute models and plot the results for 'svm' (in the e1071 package) and 'nb' (Naive Bayes, also in the e1071 package). 5) Show all three results (charts) in one window, using the grid.arrange function (have two charts in one row).

```
library(kernlab)

#model ksvm for goodOzone
svmOutput4 <- ksvm(goodOzone ~.,data=trainData2)
svmOutput4
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.242181990954208
##
## Number of Support Vectors : 52
##
## Objective Function Value : -16.5031
## Training error : 0.105741
```

```
svmPred4 <- predict(svmOutput4,testData2, type="votes")
svmPred4 <- round(svmPred4)

results4 <- data.frame(as.factor(testData2$goodOzone),as.factor(svmPred4))

library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:Metrics':
##
##     precision, recall
```

```
confusionMatrix(results4[,2],results4[,1])
```

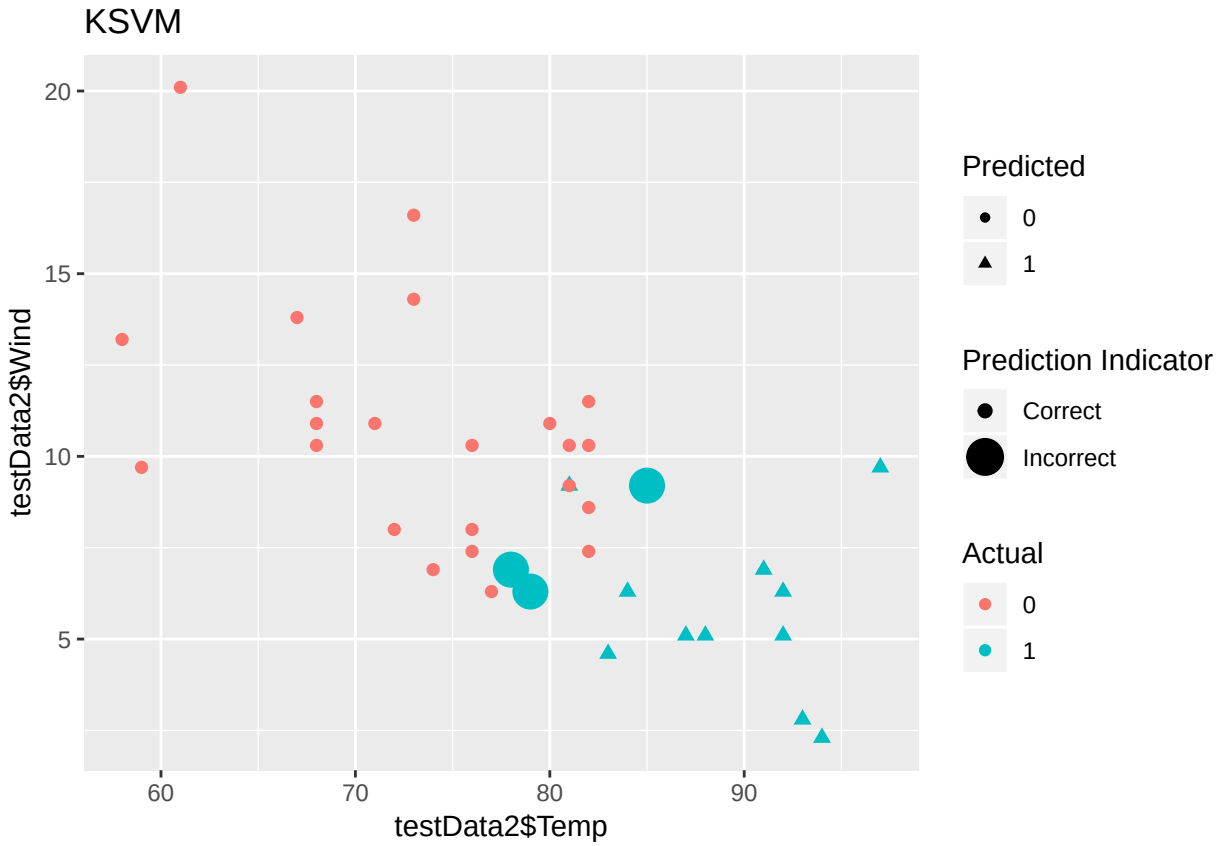
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23  3
##          1  0 11
##
##                Accuracy : 0.9189
##                  95% CI : (0.7809, 0.983)
```

```
##     No Information Rate : 0.6216
##     P-Value [Acc > NIR] : 4.639e-05
##
##                   Kappa : 0.8201
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.7857
##          Pos Pred Value : 0.8846
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6216
##          Detection Rate : 0.6216
##    Detection Prevalence : 0.7027
##       Balanced Accuracy : 0.8929
##
##        'Positive' Class : 0
##
```

```
results4$correct <- ifelse(results4[,2]==results4[,1],"Correct","Incorrect")

#plot the results for ksmv goodOzone
library(ggplot2)
svmPred4 <- as.factor(svmPred4)
plotaq4 <- ggplot(data=testData2,aes(x=testData2$Temp,y=testData2$Wind))+ geom_point(aes(size=results4$
```

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

```
plotaq4
```




```
#model svm in e1071 for goodOzone
library(e1071)
svmOutput5 <- svm(goodOzone ~.,data=trainData2)
svmOutput5
```

```
##
## Call:
## svm(formula = goodOzone ~ ., data = trainData2)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1666667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  55
```

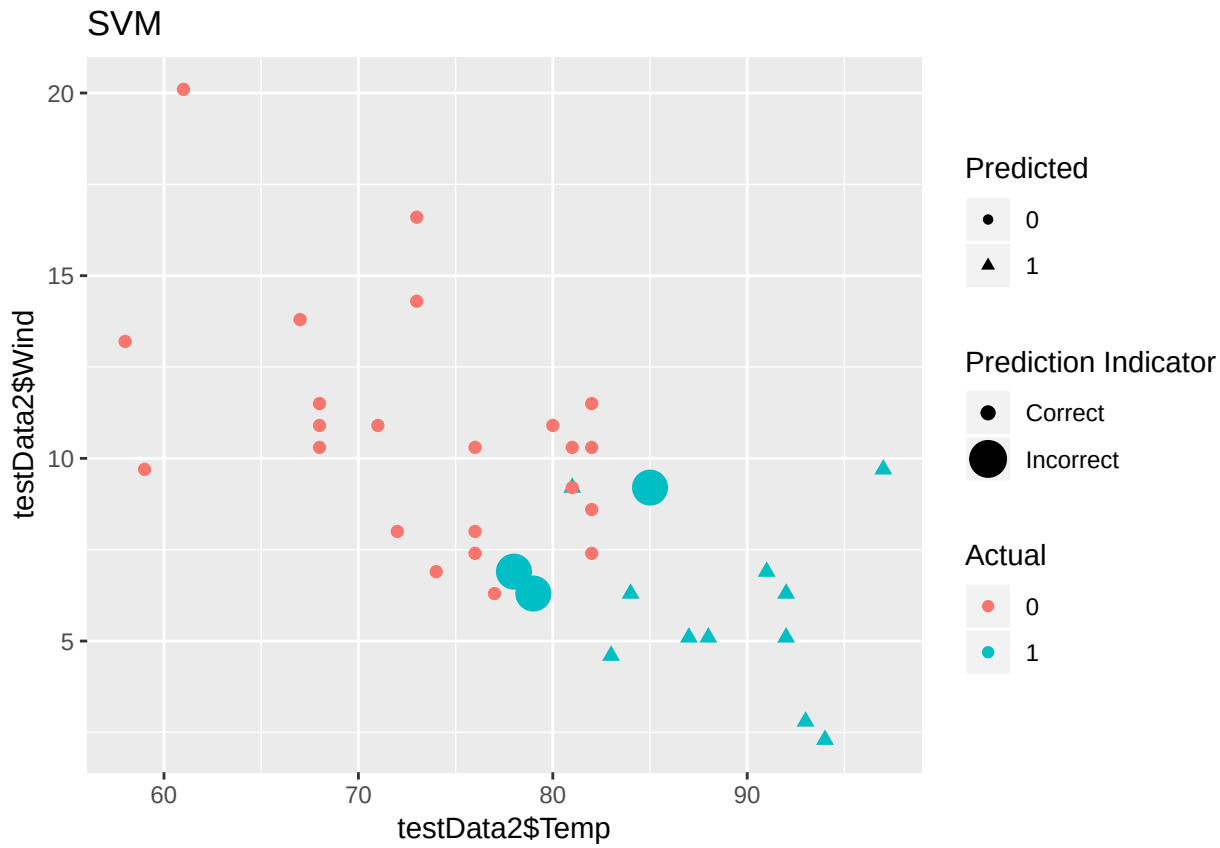
```
svmPred5 <- predict(svmOutput5,testData2, type="votes")
svmPred5 <- round(svmPred5)

results5 <- data.frame(as.factor(testData2$goodOzone),as.factor(svmPred5))

library(caret)
confusionMatrix(results5[,2],results5[,1])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23  3
##          1  0 11
##
##                Accuracy : 0.9189
##                  95% CI : (0.7809, 0.983)
##     No Information Rate : 0.6216
##     P-Value [Acc > NIR] : 4.639e-05
##
##                   Kappa : 0.8201
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.7857
##          Pos Pred Value : 0.8846
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6216
##          Detection Rate : 0.6216
##    Detection Prevalence : 0.7027
##       Balanced Accuracy : 0.8929
##
##        'Positive' Class : 0
##
```

```
results5$correct <- ifelse(results5[,2]==results5[,1],"Correct","Incorrect")

#plot the results for sum goodOzone
library(ggplot2)
svmPred5 <- as.factor(svmPred5)
plotaq5 <- ggplot(data=testData2,aes(x=testData2$Temp,y=testData2$Wind))+ geom_point(aes(size=results5$
```

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

```
plotaq5
```




```
#model Naive Bayes in e1071 for goodOzone
library(e1071)
library(Metrics)
svmOutput6 <- naiveBayes(as.factor(goodOzone) ~.,data=trainData2)
svmOutput6
```

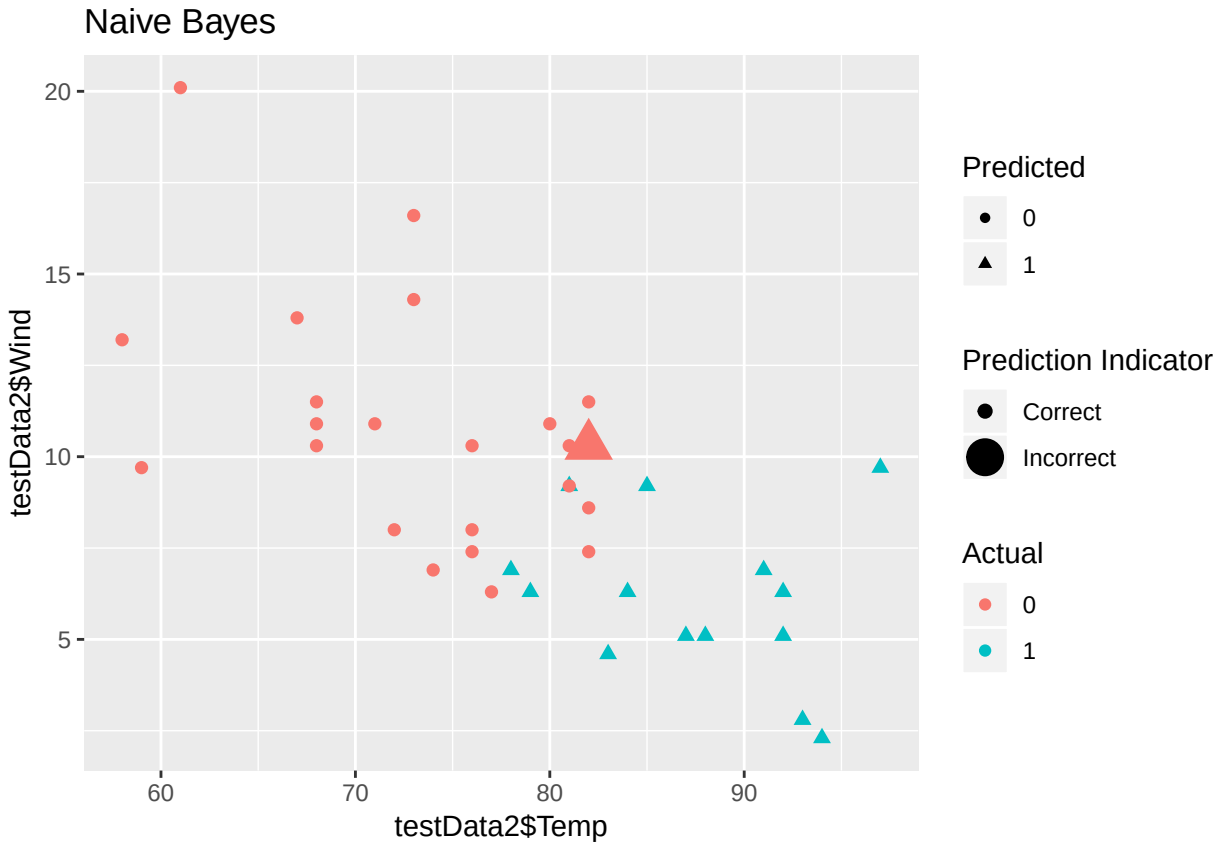
```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.6216216 0.3783784
##
## Conditional probabilities:
##    Ozone
## Y       [,1]     [,2]
##   0 21.32609 10.19598
##   1 78.78571 31.62085
##
##    Solar.R
## Y       [,1]      [,2]
##   0 169.3261 104.03612
##   1 219.3571  50.61855
```

```
##
##    Wind
## Y    [,1]     [,2]
##   0 11.65 3.030860
##   1  8.40 3.132269
##
##    Temp
## Y       [,1]     [,2]
##   0 72.15217 7.630254
##   1 85.89286 4.840717
##
##    Month
## Y       [,1]     [,2]
##   0 6.826087 1.690553
##   1 7.428571 0.997351
##
##    Day
## Y       [,1]     [,2]
##   0 15.32609 7.728891
##   1 18.32143 9.718457
```

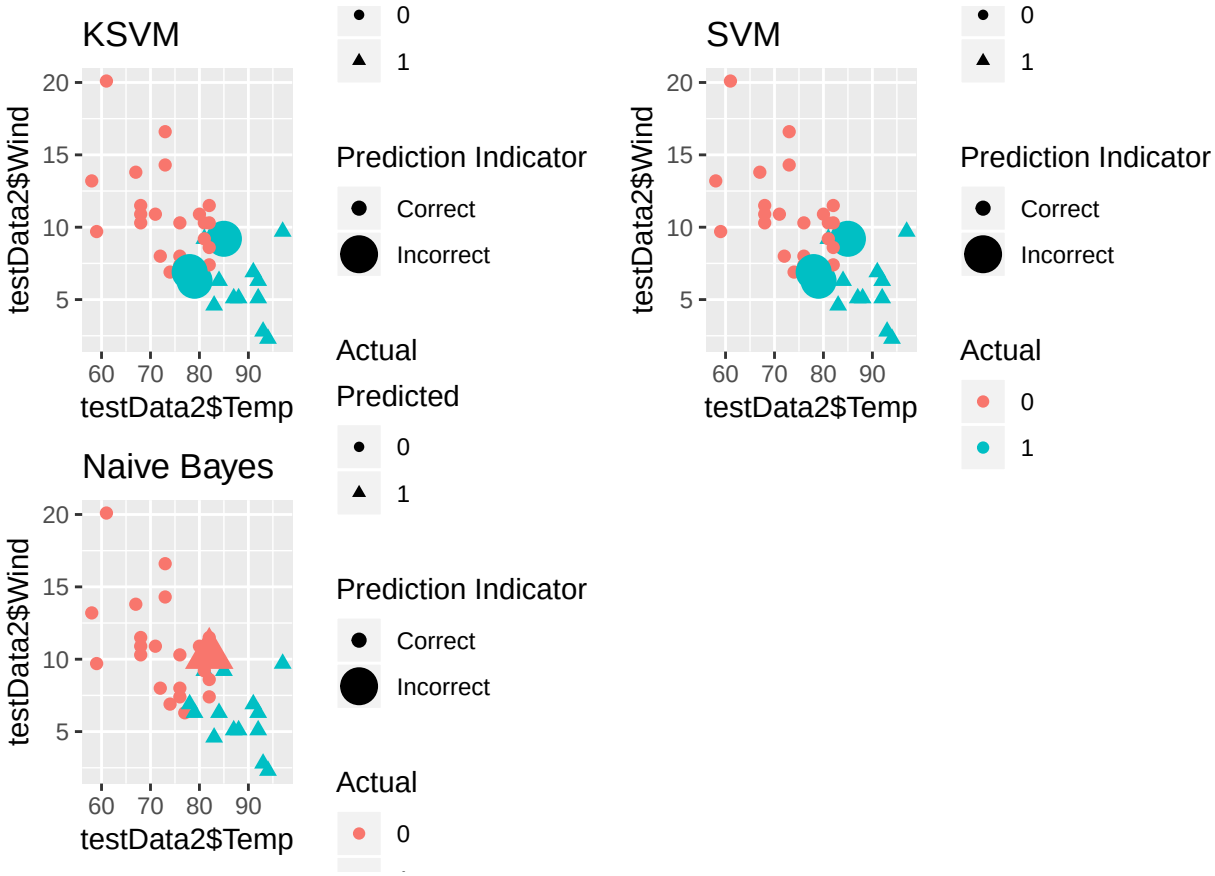
```
svmPred6 <- predict(svmOutput6,testData2, type="class")
#svmPred6 <- round(svmPred6)

results6 <- data.frame(as.factor(testData2$goodOzone),as.factor(svmPred6))

library(caret)
confusionMatrix(results6[,2],results6[,1])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 22  0
##          1  1 14
##
##                Accuracy : 0.973
##                  95% CI : (0.8584, 0.9993)
##     No Information Rate : 0.6216
##     P-Value [Acc > NIR] : 5.394e-07
##
##                   Kappa : 0.9433
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9565
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9333
##              Prevalence : 0.6216
##          Detection Rate : 0.5946
##    Detection Prevalence : 0.5946
##       Balanced Accuracy : 0.9783
##
##        'Positive' Class : 0
```

```
##
results6$correct <- ifelse(results6[,2]==results6[,1],"Correct","Incorrect")

#plot the results for sum goodOzone
library(ggplot2)
svmPred6 <- as.factor(svmPred6)
plotaq6 <- ggplot(data=testData2,aes(x=testData2$Temp,y=testData2$Wind))+ geom_point(aes(size=results6$

## Warning: Using size for a discrete variable is not advised.
plotaq6
```




```
#plot all three results in one window, using the grid arrange function
#install.packages("gridExtra")
library(gridExtra)
grid.arrange(plotaq4,plotaq5,plotaq6,ncol = 2)
```



Step 6: Which are the best Models for this data? Review what you have done and state which is the best and why.

The Naive Bayes has the best results for predicting good ozone with one or two incorrect predictions compared to four or five incorrect for the SVM and KSVM models.