Bing-Je_Wu_HW9

Bing-Je Wu

6/9/2019

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.

```
airdf <- airquality
colSums(is.na(airdf))
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day
##      37       7       0       0       0       0
```

```
  # there NAs on Ozone and Solar.R fields
  # replace NA with mean values
airdf$Ozone[is.na(airdf$Ozone)] <- round(mean(airdf$Ozone, na.rm = TRUE))
airdf$Solar.R[is.na(airdf$Solar.R)] <- round(mean(airdf$Solar.R, na.rm = TRUE))
head(airdf)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    42     186 14.3   56     5   5
## 6    28     186 14.9   66     5   6
```

Step 2: Create train and test data sets

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

```
nrow(airdf)
```

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

```
  # Get the number of records
randomindex <- sample(1:nrow(airdf))
  # Create a index vecotr with number of records without order
cutpoint2_3 <- floor(2*nrow(airdf)/3)
  # Set up a cut point for training data and testing data
training <- airdf[randomindex[1:cutpoint2_3],]
  # Create a training data by masking the 2/3 of data
testing <- airdf[randomindex[(cutpoint2_3+1):nrow(airdf)],]
  # Create a training data by masking the 1/3 of data
```

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

1) Build a model (using the 'ksvm' function, trying to predict ozone). You can use all the possible attributes, or select the attributes that you think would be the most helpful.

```
library(kernlab)
KSVMmodel <- ksvm(Ozone ~., data=training, kernel="rbfdot", kpar="automatic", C=5,
                  cross=3, prob.model=TRUE)
summary(KSVMmodel)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
KSVMmodel
```

```
## 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 =  0.205842545165143
##
## Number of Support Vectors : 91
##
## Objective Function Value : -114.1718
## Training error : 0.181693
## Cross validation error : 448.9428
## Laplace distr. width : 42.44763
```

```
#     ~. : specifies the model we want to test
# kernel : project the low-dimenstional problem into higher-dimensional space
# rbfdot : the desigation refers to the radial basis function
# kpar   : be used to control of the radial basis function kernel
# C      : cost of constraints
# cross  : the algorithm used for cross-validation
# prob.model=TRUE : generate the probabilities with the result is true or not
```

2) Test the model on the testing dataset, and compute the Root Mean Squared Error

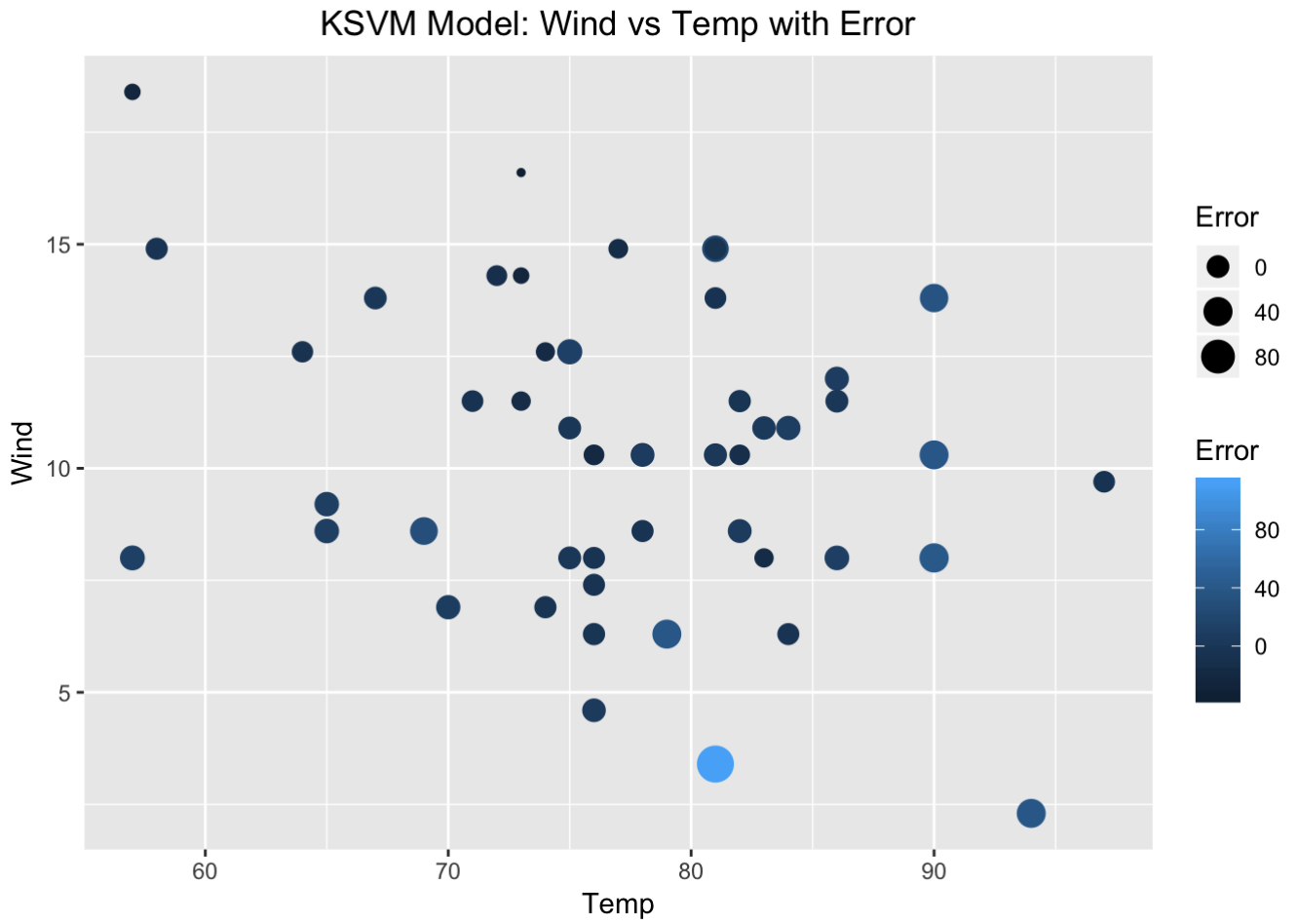
```
KSVMpred<- predict(KSVMmodel,testing, type="votes")
   # Get the predicted result
KSVMpred_actual <- testing$Ozone
   # Get the actual result
RMSE<- function(predict_result, actual_result){
  model_error <- actual_result - predict_result
  N <- nrow(predict_result)
  result <- sqrt(sum(model_error^2)/(N-1))
  sprintf("Root Mean Squared Error: %s", result)
}  # Create RMSE function

RMSE(KSVMpred, KSVMpred_actual)
```

```
## [1] "Root Mean Squared Error: 23.1067077504178"
```

```
   # Calculate the RMSE
```

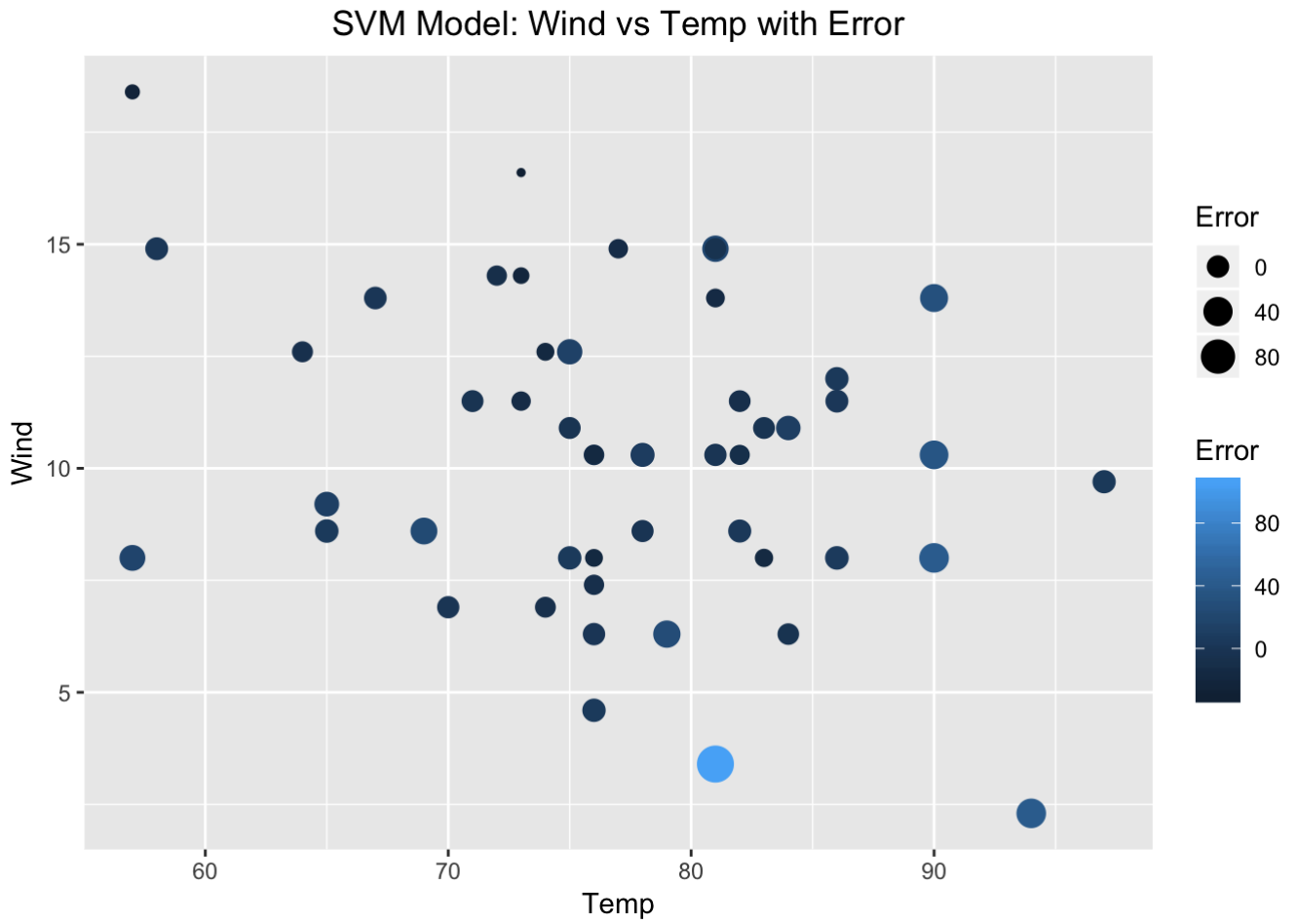
3) Plot the results. Use a scatter plot. Have the x-axis represent temperature, the y-axis represent wind, the point size and color represent the error, as defined by the actual ozone level minus the predicted ozone level).

```
library(ggplot2)
# Create a new dataframe to plot a scatter plot
model1_error <- testing$Ozone - KSVMpred
KSVMresultPlot <- data.frame(KSVMpred, model1_error,testing$Wind, testing$Temp)
colnames(KSVMresultPlot) <- c("Pred", "Error", "Wind", "Temp")
PlotKSVM <- ggplot(KSVMresultPlot, aes(x=Temp, y=Wind)) +
  geom_point(aes(col=Error, size=Error)) +
  ggtitle("KSVM Model: Wind vs Temp with Error") +
  theme(plot.title = element_text(hjust = 0.5))
PlotKSVM
```

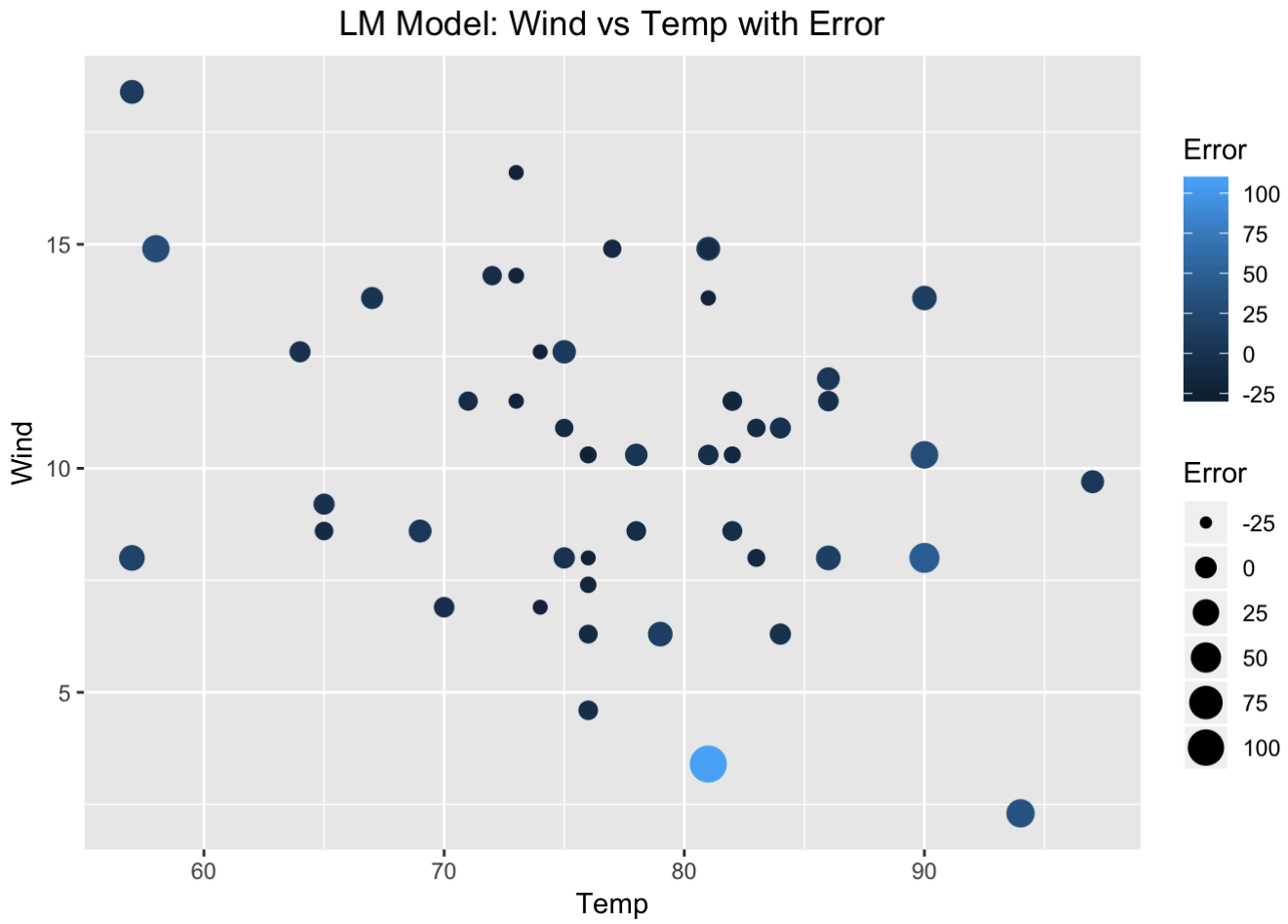


4) Compute models and plot the results for ‘svm’ (in the e1071 package) and ‘lm’. Generate similar charts for each model

```
# --svm--
library(e1071)
SVMmodel <- svm(Ozone~.,training)
SVMpred <- predict(SVMmodel, testing)
SVMpred_actual <- testing$Ozone
model2_error <- SVMpred_actual - SVMpred
SVMresultPlot<- data.frame(SVMpred, model2_error,testing$Wind, testing$Temp)
colnames(SVMresultPlot) <- c("Pred", "Error", "Wind", "Temp")
PlotSVM <- ggplot(SVMresultPlot, aes(x=Temp, y=Wind)) +
  geom_point(aes(col=Error, size=Error)) +
  ggtitle("SVM Model: Wind vs Temp with Error") +
  theme(plot.title = element_text(hjust = 0.5))
PlotSVM
```



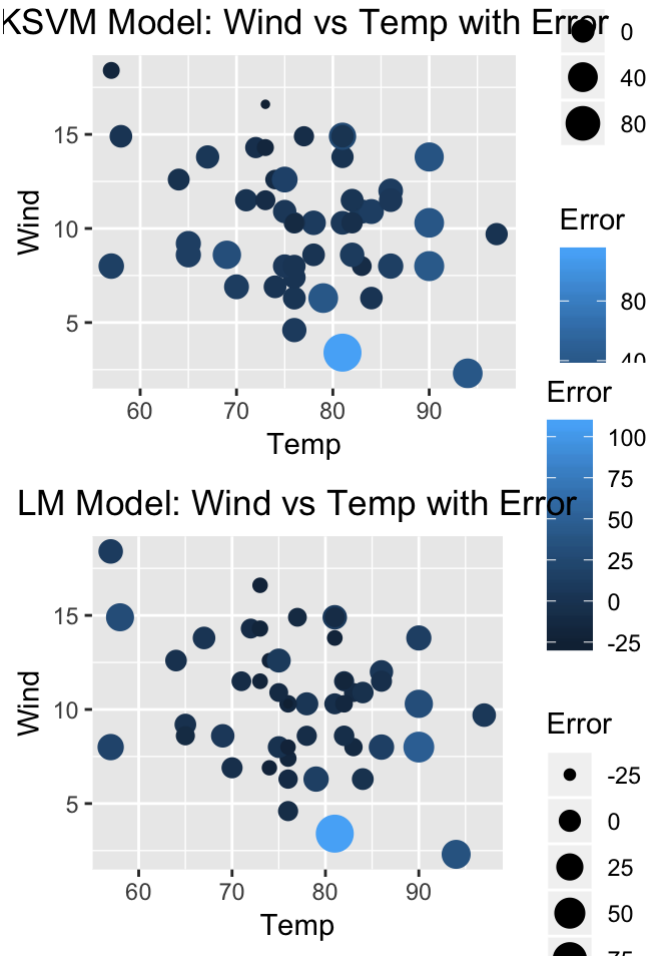

```
# --lm--
LMmodel <- lm(Ozone~.,training)
LMpred <- predict(LMmodel, testing)
LMpred_actual <- testing$Ozone
model3_error <- LMpred_actual - LMpred
LMresultPlot<- data.frame(LMpred, model3_error,testing$Wind, testing$Temp)
colnames(LMresultPlot) <- c("Pred", "Error", "Wind", "Temp")
PlotLM <- ggplot(LMresultPlot, aes(x=Temp, y=Wind)) +
  geom_point(aes(col=Error, size=Error)) +
  ggtitle("LM Model: Wind vs Temp with Error") +
  theme(plot.title = element_text(hjust = 0.5))
PlotLM
```



5) Show all three results (charts) in one window, using the grid.arrange function

Reference: https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html (https://cran.r-project.org/web/packages/gridExtra/vignettes/arrangeGrob.html)

```
library(gridExtra)
grid.arrange(PlotKSVM,PlotSVM,PlotLM, ncol=2)
```



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.

Reference: https://stackoverflow.com/questions/12125980/how-to-create-a-new-variable-in-a-data-frame-based-on-a-condition (https://stackoverflow.com/questions/12125980/how-to-create-a-new-variable-in-a-data-frame-based-on-a-condition)

```
airdf$goodOzone <- ifelse(airdf$Ozone<mean(airdf$Ozone),0,1)
```

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.

```
randomindex <- sample(1:nrow(airdf))
  # Create a index vecotr with number of records without order
cutpoint2_3 <- floor(2*nrow(airdf)/3)
  # Set up a cut point for training data and testing data
training <- airdf[randomindex[1:cutpoint2_3],]
  # Create a training data by masking the 2/3 of data
testing <- airdf[randomindex[(cutpoint2_3+1):nrow(airdf)],]
library(kernlab)
KSVMmodel_GnB <- ksvm(goodOzone~., data=training, kernel="rbfdot", kpar="automatic", C=5
,
                  cross=3, prob.model=TRUE)
summary(KSVMmodel_GnB)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```
KSVMmodel_GnB
```

```
## 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 =  0.137715377201388
##
## Number of Support Vectors : 79
##
## Objective Function Value : -76.5625
## Training error : 0.102592
## Cross validation error : 0.077407
## Laplace distr. width : 0.291832
```

2) Test the model on the testing dataset, and compute the percent of ‘goodOzone’ that was correctly predicted.

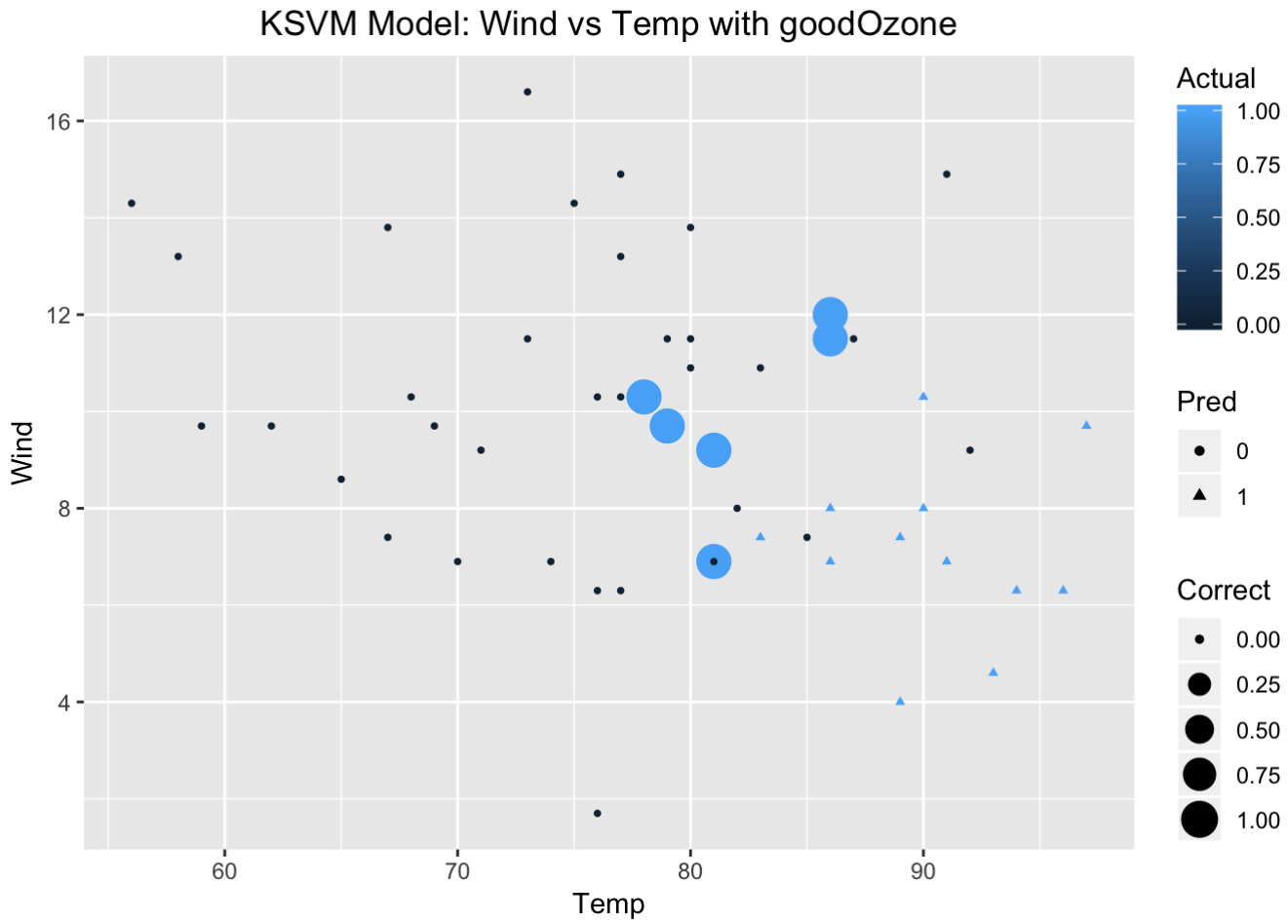
```
KSVMpredGnB <- round(predict(KSVMmodel_GnB, testing))
#CorrectPercent
cPercent <- function(predicted, actual){
  confMatrix<- table(predicted, actual, dnn=c("Prediction","Actual"))
  Result <- (confMatrix[1,1]+confMatrix[2,2])/sum(colSums(confMatrix))*100
  print(confMatrix)
  return(sprintf("Correct Percentage: %1.2f%% ", Result))
}     #Create correctPercent function
cPercent(KSVMpredGnB,testing$goodOzone)
```

```
##           Actual
## Prediction  0  1
##          0 33  6
##          1  0 12
```

```
## [1] "Correct Percentage: 88.24% "
```

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).

```
correct <- abs(testing$goodOzone-KSVMpredGnB)
GnBresultPlot <-
  data.frame(KSVMpredGnB,testing$goodOzone,correct,testing$Wind,testing$Temp)
colnames(GnBresultPlot) <- c("Pred", "Actual","Correct", "Wind", "Temp")
GnBresultPlot$Pred <- as.factor(GnBresultPlot$Pred)
PlotGnB <- ggplot(GnBresultPlot, aes(x=Temp, y=Wind)) +
  geom_point(aes(col=Actual, shape = Pred, size=Correct)) +
  ggtitle("KSVM Model: Wind vs Temp with goodOzone") +
  theme(plot.title = element_text(hjust = 0.5))
PlotGnB
```

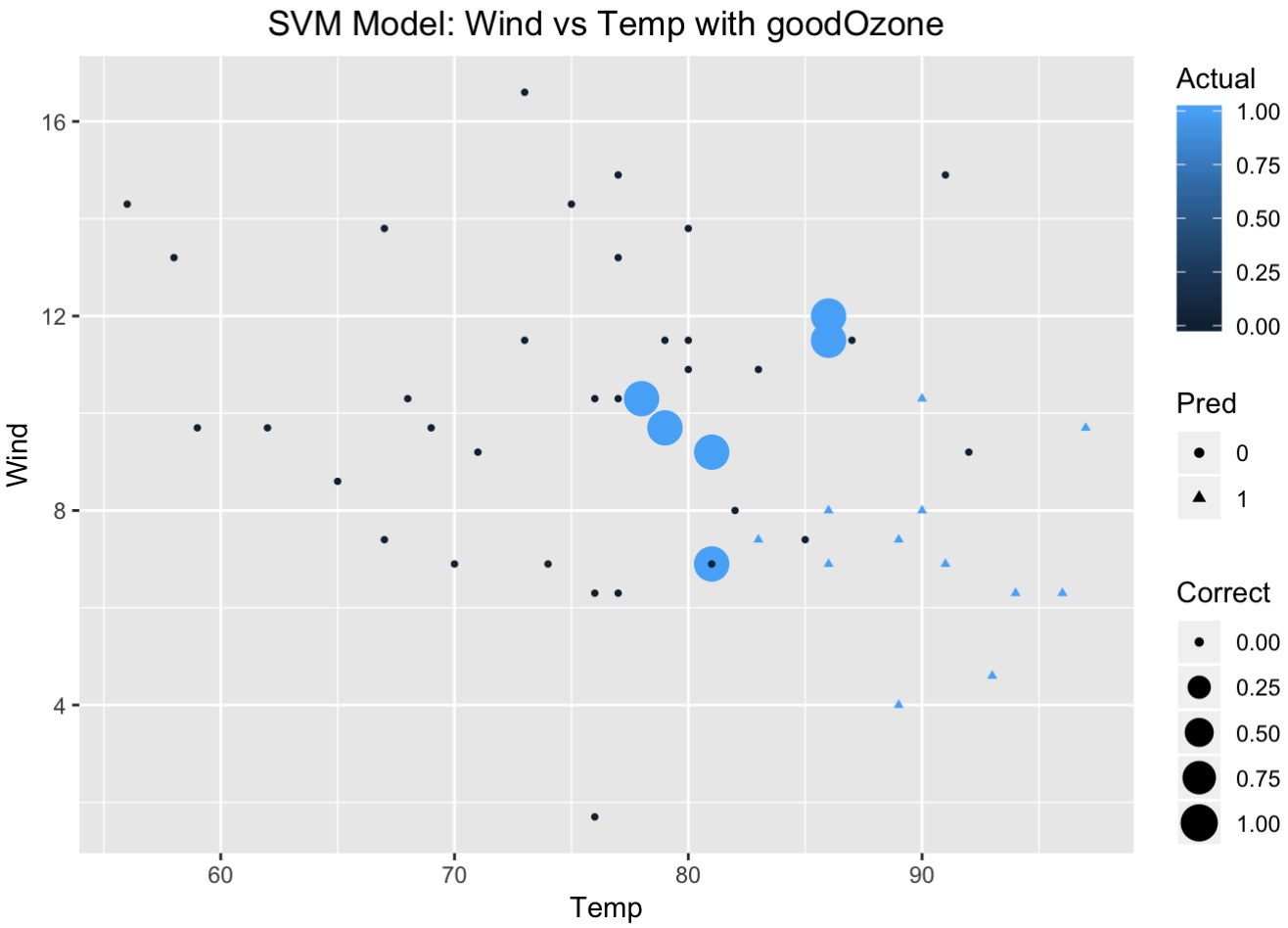


4) Compute models and plot the results for ‘svm’ (in the e1071 package) and ‘nb’ (Naive Bayes, also in the e1071 package).

```
# --svm--
library(e1071)
SVMmodelGnB <- svm(goodOzone~.,training)
SVMmodelGnB
```

```
##
## Call:
## svm(formula = goodOzone ~ ., data = training)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1666667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  64
```

```
SVMpredGnB <- round(predict(SVMmodelGnB, testing))
SVMcorrect <- abs(testing$goodOzone-SVMpredGnB)
SVMresultPlotGnB <-
  data.frame(SVMpredGnB, testing$goodOzone, SVMcorrect, testing$Wind, testing$Temp)
colnames(SVMresultPlotGnB) <- c("Pred", "Actual","Correct", "Wind", "Temp")
SVMresultPlotGnB$Pred <- as.factor(SVMresultPlotGnB$Pred)
PlotSVMGnB <- ggplot(SVMresultPlotGnB, aes(x=Temp, y=Wind)) +
  geom_point(aes(col=Actual, shape = Pred, size=Correct)) +
  ggtitle("SVM Model: Wind vs Temp with goodOzone") +
  theme(plot.title = element_text(hjust = 0.5))
PlotSVMGnB
```




```
cPercent(SVMpredGnB,testing$goodOzone)
```

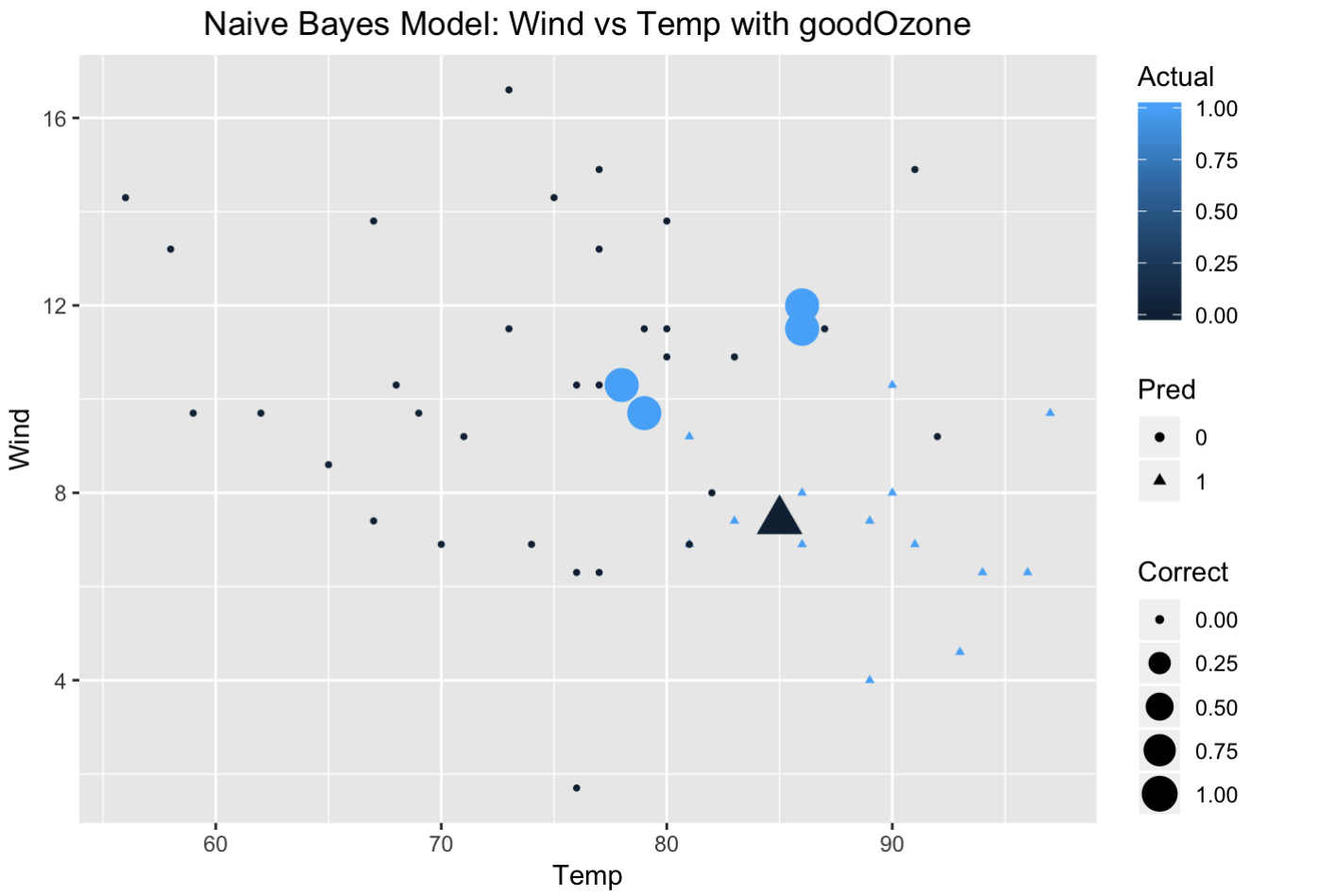
```
##            Actual
## Prediction  0  1
##          0 33  6
##          1  0 12
```

```
## [1] "Correct Percentage: 88.24% "
```

```
# --nb--
NBmodelGnB <- naiveBayes(as.factor(goodOzone)~.,training)
NBmodelGnB
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        0        1
## 0.745098 0.254902
##
## Conditional probabilities:
##    Ozone
## Y       [,1]     [,2]
##   0 27.94737 12.73514
##   1 78.34615 30.51025
##
##    Solar.R
## Y       [,1]     [,2]
##   0 170.3947 95.55718
##   1 222.1154 58.37556
##
##    Wind
## Y        [,1]     [,2]
##   0 11.176316 3.260669
##   1  7.196154 3.433888
##
##    Temp
## Y       [,1]     [,2]
##   0 74.35526 8.586352
##   1 85.65385 4.638468
##
##    Month
## Y       [,1]     [,2]
##   0 6.789474 1.490555
##   1 7.384615 1.098250
##
##    Day
## Y       [,1]     [,2]
##   0 15.43421 8.448014
##   1 15.03846 9.957834
```

```
NBpredGnB <- predict(NBmodelGnB, testing)
NBcorrect <- ifelse(testing$goodOzone==NBpredGnB,0, 1)
NBresultPlotGnB <-
  data.frame(NBpredGnB, testing$goodOzone, NBcorrect, testing$Wind, testing$Temp)
colnames(NBresultPlotGnB) <- c("Pred", "Actual","Correct", "Wind", "Temp")
NBresultPlotGnB$Pred <- as.factor(NBresultPlotGnB$Pred)
PlotNBGnB <- ggplot(NBresultPlotGnB, aes(x=Temp, y=Wind)) +
  geom_point(aes(col=Actual, shape = Pred, size=Correct)) +
  ggtitle("Naive Bayes Model: Wind vs Temp with goodOzone") +
  theme(plot.title = element_text(hjust = 0.5))
PlotNBGnB
```



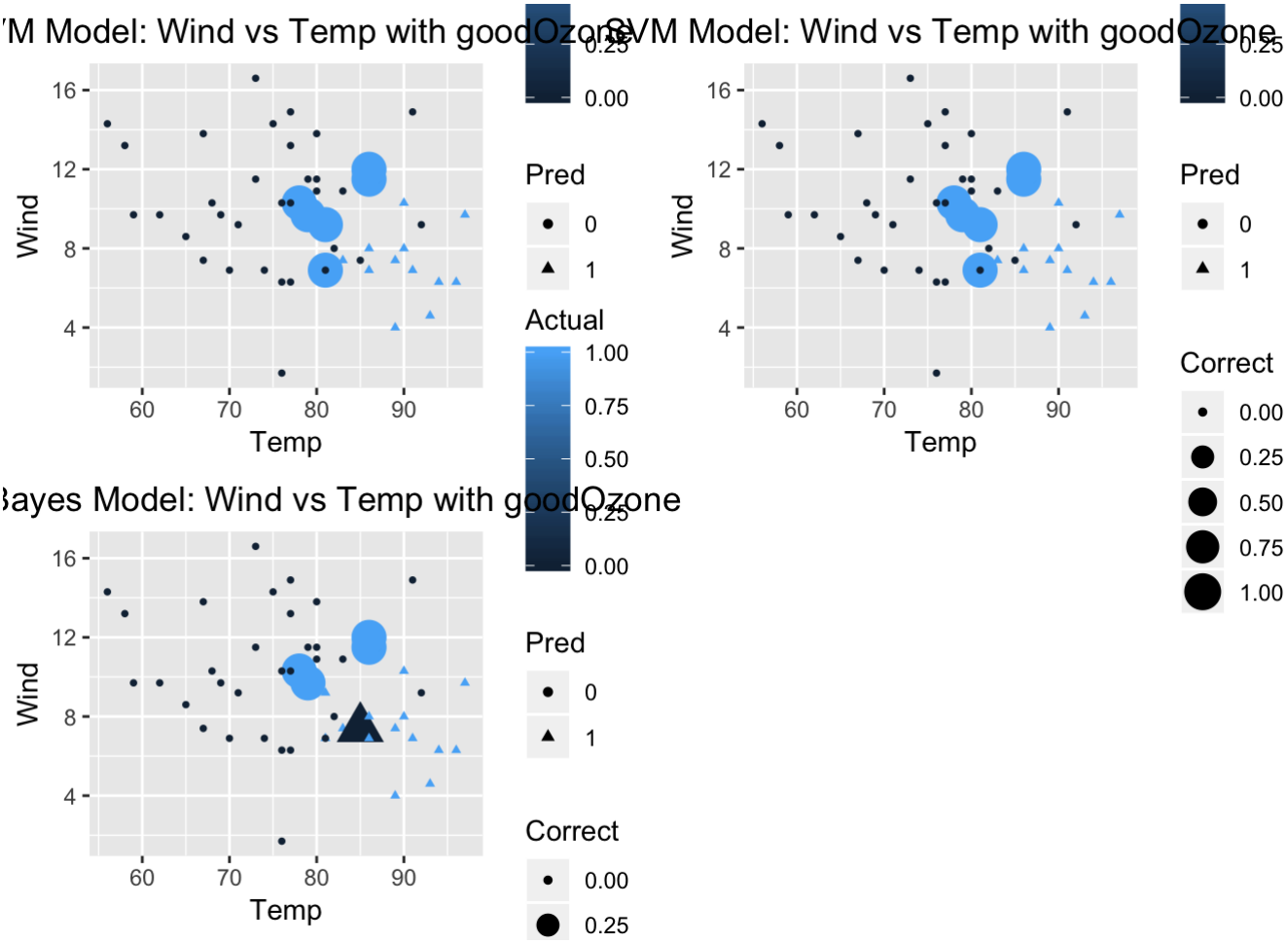

```
cPercent(NBpredGnB,testing$goodOzone)
```

```
##            Actual
## Prediction  0  1
##          0 32  4
##          1  1 14
```

```
## [1] "Correct Percentage: 90.20% "
```

5) Show all three results (charts) in one window, using the grid.arrange function (have two charts in one row).

```
library(gridExtra)
grid.arrange(PlotGnB,PlotSVMGnB,PlotNBGnB, 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.

```
# Naive Bayes model is the best model among the three models because it has the highest
 Correct Percentage rate.
```