# Homework #9 – Support Vectors Machines Lab

## R Script (Code)

#

# Course: IST687

# Name: Joyce Woznica

# Homework 9 - Support Vector Machines

# Due Date: 3/12/2019

# Date Submitted:

#

# Load the packages

# Packages: kernlab, e1071, gridExtra, ggplot2, caret

#specify the packages of interest

packages=c("kernlab","e1071","gridExtra","ggplot2", "caret")

#use this function to check if each package is on the local machine

#if a package is installed, it will be loaded

#if any are not, the missing package(s) will be installed and loaded

package.check <- lapply(packages, FUN = function(x) {

if (!require(x, character.only = TRUE)) {

install.packages(x, dependencies = TRUE)

library(x, character.only = TRUE)

}

})

#verify they are loaded

search()

#

# Step 1: Load the data

# we will use the airquality data set, which you should already have as part of your R installation

myairQ<-data.frame(airquality)

# There will be NA's - figure out what you are going to do with that

# JLJW NOTE: I elected to use the mean for each column for the NAs as opposed to

# removing all the observations that had an NA for one of the values

# create a function to replace each column NA with mean for that column

replaceNAwMeans<-function(vec)

{

numcols<-length(colnames(vec))

index<-1

while(index<=numcols)

{

theColV <- vec[,index]

theColV[is.na(theColV)]<-mean(theColV,na.rm=TRUE)

vec[,index]<-theColV

index<-index+1

}

return(vec)

}

# Update the dataframe with the modified frame with means applied to NA values

myairQ<-replaceNAwMeans(myairQ)

colnames(myairQ)

# Step 2: Create train and test data sets

# Using the technique discussed in class, created two datasets - one for training, one for test

# create Training Data (2/3) and Test Data set (1/3) of set

randIndex <- sample(1:dim(myairQ)[1])

summary(randIndex)

length(randIndex)

head(randIndex)

cutPoint2\_3 <- floor(2 \* dim(myairQ)[1]/3)

cutPoint2\_3

trainData<- myairQ[randIndex[1:cutPoint2\_3],]

testData<- myairQ[randIndex[(cutPoint2\_3+1):dim(myairQ)[1]],]

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

# 1) Build a model (using ksmv to predict ozone)

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

ksvmOutputH <- ksvm(Ozone~.,data=trainData,kernel="rbfdot",kpar="automatic",C=50,cross=3,prob.model=TRUE)

ksvmOutputM <- ksvm(Ozone~.,data=trainData,kernel="rbfdot",kpar="automatic",C=10,cross=10,prob.model=TRUE)

# use the Middle C model

ksvmOutput <- ksvmOutputM

# 2) Test the model on the testing dataset and compute Root Mean Squared Error (RMSE)

# create prediction

ksvmPred <- predict(ksvmOutput, testData, type="votes")

str(ksvmPred)

compTable <- data.frame(testData[,1],ksvmPred[,1])

colnames(compTable) <- c("test", "Pred")

# this is the RMSE (how low)

RSMEksvm<-sqrt(mean((compTable$test-compTable$Pred)^2))

RSMEksvm

# 3) Plot the results. Use a scatter plot. Have x represent temperature,

# y-axis represent wind, point size and color represented the error

# which is the actual Ozone level minus the predicted Ozone level

# compute absolute error for each case

compTable$error <- abs(compTable$test - compTable$Pred)

# create a new dataframe contains error, tempreture and wind

ksvmPlot <- data.frame(compTable$error, testData$Temp, testData$Wind)

# assign column names

colnames(ksvmPlot) <- c("error","Temp","Wind")

# polt result using ggplot, setting "Temp" as x-axis and "Wind" as y-axis

# use point size and color shade to illustrate how big is the error

plotksvm1 <- ggplot(ksvmPlot, aes(x=Temp,y=Wind)) +

geom\_point(aes(size=error, color=error))+

ggtitle("ksvm")

plotksvm1

# 4) Compute models and plot the result for 'svm' (in the e1071 package) and 'lm'.

# Generate similar charts for each model.

# SVM

svmOutput <- svm(Ozone~.,data=trainData)

# create prediction

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

str(svmPred)

svmPred <- (data.frame(svmPred))

compTable <- data.frame(testData[,1],svmPred[,1])

colnames(compTable) <- c("test", "Pred")

# this is the RMSE (how low)

RSMEsvm<-sqrt(mean((compTable$test-compTable$Pred)^2))

RSMEsvm

# compute absolute error for each case

compTable$error <- abs(compTable$test - compTable$Pred)

# create a new dataframe contains error, tempreture and wind

svmPlot <- data.frame(compTable$error, testData$Temp, testData$Wind)

# assign column names

colnames(svmPlot) <- c("error","Temp","Wind")

# polt result using ggplot, setting "Temp" as x-axis and "Wind" as y-axis

# use point size and color shade to illustrate how big is the error

plotsvm1 <- ggplot(svmPlot, aes(x=Temp,y=Wind)) +

geom\_point(aes(size=error, color=error))+

ggtitle("svm")

plotsvm1

# LM

lmOutput <- lm(formula=Ozone~.,data=trainData)

AICModels<-step(lmOutput,data=trainData, direction="backward")

# AIC removes Month and Day, but keeps Wind, Temp and Solar.R

# create prediction

lmPred <- predict(lmOutput, testData, type="response")

str(lmPred)

lmPred <- data.frame(lmPred)

compTable <- data.frame(testData[,1],lmPred[,1])

colnames(compTable) <- c("test", "Pred")

# this is the RMSE (how low)

RSMElm<-sqrt(mean((compTable$test-compTable$Pred)^2))

RSMElm

# compute absolute error for each case

compTable$error <- abs(compTable$test - compTable$Pred)

# create a new dataframe contains error, tempreture and wind

lmPlot <- data.frame(compTable$error, testData$Temp, testData$Wind)

# assign column names

colnames(lmPlot) <- c("error","Temp","Wind")

# polt result using ggplot, setting "Temp" as x-axis and "Wind" as y-axis

# use point size and color shade to illustrate how big is the error

plotlm1 <- ggplot(lmPlot, aes(x=Temp,y=Wind)) +

geom\_point(aes(size=error, color=error))+

ggtitle("lm")

plotlm1

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

grid.arrange(plotksvm1, plotsvm1, plotlm1, ncol=2, nrow=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.

# reset Train and Test data

randIndex <- sample(1:dim(myairQ)[1])

summary(randIndex)

length(randIndex)

head(randIndex)

cutPoint2\_3 <- floor(2 \* dim(myairQ)[1]/3)

cutPoint2\_3

trainData<- myairQ[randIndex[1:cutPoint2\_3],]

testData<- myairQ[randIndex[(cutPoint2\_3+1):dim(myairQ)[1]],]

# calculate average Ozone

meanOzone <- mean(myairQ$Ozone,na.rm=TRUE)

# create a new variable named "goodOzone" in train data set

# goodOzone = 0 if Ozone is below average Ozone

# googOzone = 1 if Ozone is eaqual or above the average ozone

trainData$goodOzone <- ifelse(trainData$Ozone<meanOzone, 0, 1)

# do the same thing for test dataset

testData$goodOzone <- ifelse(testData$Ozone<meanOzone, 0, 1)

# remove "Ozone" from train data

trainData <- trainData[,-1]

# remove "Ozone" from test data

testData <- testData[,-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 attributes tht yuo think would be most helpful.

# convert "goodOzone" in train data from numeric to factor

trainData$goodOzone <- as.factor(trainData$goodOzone)

# convert "goodOzone" in test data from numeric to factor

testData$goodOzone <- as.factor(testData$goodOzone)

# build a model using ksvm function,and use all other variables to predict

ksvmGoodO <- ksvm(goodOzone~.,data=trainData,kernel="rbfdot",kpar="automatic",C=10,cross=10,prob.model=TRUE)

# 2) Test the model on the resting dataset, and compute the precent of 'goodOzone' that was

# correctly predicted.

goodPredO <- predict(ksvmGoodO, testData)

# create a dataframe that contains the exact "goodOzone" value and the predicted "goodOzone"

compGood1 <- data.frame(testData[,6], goodPredO)

# change column names

colnames(compGood1) <- c("test","Pred")

# Compute the percentage of correct cases

perc\_ksvm <- length(which(compGood1$test==compGood1$Pred))/dim(compGood1)[1]

perc\_ksvm

# Confusion Matrix

results <- table(test=compGood1$test, pred=compGood1$Pred)

print(results)

# 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 that the model got wrong).

# determine the prediction is "correct" or "wrong" for each case

compGood1$correct <- ifelse(compGood1$test==compGood1$Pred,"correct","wrong")

# create a new dataframe contains correct, temperature and wind, and goodZone

Plot\_ksvm <- data.frame(compGood1$correct,testData$Temp,testData$Wind,testData$goodOzone,compGood1$Pred)

# change column names

colnames(Plot\_ksvm) <- c("correct","Temp","Wind","goodOzone","Predict")

# plot result using ggplot

# size representing correct/wrong; color representing actual good/bad day; shape representing predicted good/bad day.

ksvmplot <- ggplot(Plot\_ksvm, aes(x=Temp,y=Wind)) +

geom\_point(aes(size=correct,color=goodOzone,shape = Predict))+

ggtitle("ksvm - good/bad ozone")

ksvmplot

# 4) Compute the models and plot the results for 'svm' (in the e1071 package) and 'nb'

# (Naive Bayes, also in the e1071 package).

# SVM

svmGoodO <- svm(goodOzone~.,data=trainData)

# create prediction

goodPredO <- predict(svmGoodO, testData)

# create a dataframe that contains the exact "goodOzone" value and the predicted "goodOzone"

compGood1 <- data.frame(testData[,6], goodPredO)

# change column names

colnames(compGood1) <- c("test","Pred")

# Compute the percentage of correct cases

perc\_svm <- length(which(compGood1$test==compGood1$Pred))/dim(compGood1)[1]

perc\_svm

# Confusion Matrix

results <- table(test=compGood1$test, pred=compGood1$Pred)

print(results)

# determine the prediction is "correct" or "wrong" for each case

compGood1$correct <- ifelse(compGood1$test==compGood1$Pred,"correct","wrong")

# create a new dataframe contains correct, tempreture and wind, and goodZone

Plot\_svm <- data.frame(compGood1$correct,testData$Temp,testData$Wind,testData$goodOzone,compGood1$Pred)

# change column names

colnames(Plot\_svm) <- c("correct","Temp","Wind","goodOzone","Predict")

# plot result using ggplot

# size representing correct/wrong; color representing actual good/bad day; shape representing predicted good/bad day.

svmplot <- ggplot(Plot\_svm, aes(x=Temp,y=Wind)) +

geom\_point(aes(size=correct,color=goodOzone,shape = Predict))+

ggtitle("svm - good/bad ozone")

svmplot

# Naive Bayes

nbGoodO <- naiveBayes(goodOzone~.,data=trainData)

# create prediction

goodPredO <- predict(nbGoodO, testData)

# create a dataframe that contains the exact "goodOzone" value and the predicted "goodOzone"

compGood1 <- data.frame(testData[,6], goodPredO)

# change column names

colnames(compGood1) <- c("test","Pred")

# Compute the percentage of correct cases

perc\_nb <- length(which(compGood1$test==compGood1$Pred))/dim(compGood1)[1]

perc\_nb

# Confusion Matrix

results <- table(test=compGood1$test, pred=compGood1$Pred)

print(results)

# determine the prediction is "correct" or "wrong" for each case

compGood1$correct <- ifelse(compGood1$test==compGood1$Pred,"correct","wrong")

# create a new dataframe contains correct, tempreture and wind, and goodZone

Plot\_nb <- data.frame(compGood1$correct,testData$Temp,testData$Wind,testData$goodOzone,compGood1$Pred)

# change column names

colnames(Plot\_nb) <- c("correct","Temp","Wind","goodOzone","Predict")

# plot result using ggplot

# size representing correct/wrong; color representing actual good/bad day; shape representing predicted good/bad day.

nbplot <- ggplot(Plot\_nb, aes(x=Temp,y=Wind)) +

geom\_point(aes(size=correct,color=goodOzone,shape = Predict))+

ggtitle("nb - good/bad ozone")

nbplot

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

# two charts in one row).

grid.arrange(ksvmplot,svmplot,nbplot, ncol=2, nrow=2)

# Step 6: Which are the best models for this data?

# Review what you have done and state which is the best and why

# ANSWER: If you just review the Root Mean Squared Error (RMSE) for predicting Ozone from

# the other variables: Wind, Temp, Solar.R, you see that depending on the training data and the

# test data - the "best model" varies. I ran this code multiple times and I found that

# the usually the svm had the lowest RMSE, so I would think it would be the best model; however,

# this was not always the case. It depended on the random training and test data.

# For example, in the run that I turned in for homework - these are the values.

# for ksvm, RSME = 16.99584

# for svm, RSME = 13.63764

# for lm, RSME = 14.15575

# we would conclude that svm is the best model for this set of data since it has the smallest

# Root Mean Squared Error.

#

# The same occurred when reviewing the Good/Bad Ozone prediction and look at the models

# and the "Percent Good" with varying values for the percent that was properly predicted.

# For example, in the run that I turned in for homework - you would see the following:

# perc\_ksvm = 72.54902%

# perc\_svm = 74.5098%

# perc\_nb = 80.39216%

# We would again conclude that the Naive Bayes (nb) is the best model for this set of data since

# it carries the highest percent of correct predictions

#

# The plots also show the same, the svm plot shows smaller (less error) size and darker color

# points in the plot. The plot for good/bad ozone, the nb plot shows many smaller (correct)

# triangles in the nb plot.

## Console Log (Executed Code)

|  |
| --- |
| #  > # Course: IST687  > # Name: Joyce Woznica  > # Homework 9 - Support Vector Machines  > # Due Date: 3/12/2019  > # Date Submitted:  > #  > # Load the packages  > # Packages: kernlab, e1071, gridExtra, ggplot2, caret  >  > #specify the packages of interest  > packages=c("kernlab","e1071","gridExtra","ggplot2", "caret")  >  > #use this function to check if each package is on the local machine  > #if a package is installed, it will be loaded  > #if any are not, the missing package(s) will be installed and loaded  > package.check <- lapply(packages, FUN = function(x) {  + if (!require(x, character.only = TRUE)) {  + install.packages(x, dependencies = TRUE)  + library(x, character.only = TRUE)  + }  + })  >  > #verify they are loaded  > search()  [1] ".GlobalEnv" "package:caret" "package:lattice" "package:ggplot2" "package:gridExtra" "package:e1071"  [7] "package:kernlab" "tools:rstudio" "package:stats" "package:graphics" "package:grDevices" "package:utils"  [13] "package:datasets" "package:methods" "Autoloads" "package:base"  > #  > # Step 1: Load the data  > # we will use the airquality data set, which you should already have as part of your R installation  > myairQ<-data.frame(airquality)  >  > # There will be NA's - figure out what you are going to do with that  > # JLJW NOTE: I elected to use the mean for each column for the NAs as opposed to  > # removing all the observations that had an NA for one of the values  >  > # create a function to replace each column NA with mean for that column  > replaceNAwMeans<-function(vec)  + {  + numcols<-length(colnames(vec))  + index<-1  + while(index<=numcols)  + {  + theColV <- vec[,index]  + theColV[is.na(theColV)]<-mean(theColV,na.rm=TRUE)  + vec[,index]<-theColV  + index<-index+1  + }  + return(vec)  + }  >  > # Update the dataframe with the modified frame with means applied to NA values  > myairQ<-replaceNAwMeans(myairQ)  > colnames(myairQ)  [1] "Ozone" "Solar.R" "Wind" "Temp" "Month" "Day"  > # Step 2: Create train and test data sets  > # Using the technique discussed in class, created two datasets - one for training, one for test  > # create Training Data (2/3) and Test Data set (1/3) of set  > randIndex <- sample(1:dim(myairQ)[1])  > summary(randIndex)  Min. 1st Qu. Median Mean 3rd Qu. Max.  1 39 77 77 115 153  > length(randIndex)  [1] 153  > head(randIndex)  [1] 13 91 54 151 72 64  > cutPoint2\_3 <- floor(2 \* dim(myairQ)[1]/3)  > cutPoint2\_3  [1] 102  > trainData<- myairQ[randIndex[1:cutPoint2\_3],]  > testData<- myairQ[randIndex[(cutPoint2\_3+1):dim(myairQ)[1]],]  > # Step 3: Build a Model using KSVM & visualize the results  > # 1) Build a model (using ksmv to predict ozone)  > ksvmOutputL <- ksvm(Ozone~.,data=trainData,kernel="rbfdot",kpar="automatic",C=5,cross=3,prob.model=TRUE)  > ksvmOutputH <- ksvm(Ozone~.,data=trainData,kernel="rbfdot",kpar="automatic",C=50,cross=3,prob.model=TRUE)  > ksvmOutputM <- ksvm(Ozone~.,data=trainData,kernel="rbfdot",kpar="automatic",C=10,cross=10,prob.model=TRUE)  >  > # use the Middle C model  > ksvmOutput <- ksvmOutputM  > # 2) Test the model on the testing dataset and compute Root Mean Squared Error (RMSE)  > # create prediction  > ksvmPred <- predict(ksvmOutput, testData, type="votes")  > str(ksvmPred)  num [1:51, 1] 17.3 92.2 48.2 48.8 35.1 ...  > compTable <- data.frame(testData[,1],ksvmPred[,1])  > colnames(compTable) <- c("test", "Pred")  > # this is the RMSE (how low)  > RSMEksvm<-sqrt(mean((compTable$test-compTable$Pred)^2))  > RSMEksvm  [1] 16.99584  > # 3) Plot the results. Use a scatter plot. Have x represent temperature,  > # y-axis represent wind, point size and color represented the error  > # which is the actual Ozone level minus the predicted Ozone level  >  > # compute absolute error for each case  > compTable$error <- abs(compTable$test - compTable$Pred)  > # create a new dataframe contains error, tempreture and wind  > ksvmPlot <- data.frame(compTable$error, testData$Temp, testData$Wind)  > # assign column names  > colnames(ksvmPlot) <- c("error","Temp","Wind")  > # polt result using ggplot, setting "Temp" as x-axis and "Wind" as y-axis  > # use point size and color shade to illustrate how big is the error  > plotksvm1 <- ggplot(ksvmPlot, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=error, color=error))+  + ggtitle("ksvm")  > plotksvm1  > # 4) Compute models and plot the result for 'svm' (in the e1071 package) and 'lm'.  > # Generate similar charts for each model.  > # SVM  > svmOutput <- svm(Ozone~.,data=trainData)  > # create prediction  > svmPred <- predict(svmOutput, testData, type="votes")  > str(svmPred)  Named num [1:51] 13 81.9 55.9 46.1 33.3 ...  - attr(\*, "names")= chr [1:51] "138" "80" "43" "81" ...  > svmPred <- (data.frame(svmPred))  > compTable <- data.frame(testData[,1],svmPred[,1])  > colnames(compTable) <- c("test", "Pred")  > # this is the RMSE (how low)  > RSMEsvm<-sqrt(mean((compTable$test-compTable$Pred)^2))  > RSMEsvm  [1] 13.63764  > # compute absolute error for each case  > compTable$error <- abs(compTable$test - compTable$Pred)  > # create a new dataframe contains error, tempreture and wind  > svmPlot <- data.frame(compTable$error, testData$Temp, testData$Wind)  > # assign column names  > colnames(svmPlot) <- c("error","Temp","Wind")  > # polt result using ggplot, setting "Temp" as x-axis and "Wind" as y-axis  > # use point size and color shade to illustrate how big is the error  > plotsvm1 <- ggplot(svmPlot, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=error, color=error))+  + ggtitle("svm")  > plotsvm1  > # LM  > lmOutput <- lm(formula=Ozone~.,data=trainData)  > AICModels<-step(lmOutput,data=trainData, direction="backward")  Start: AIC=653.11  Ozone ~ Solar.R + Wind + Temp + Month + Day  Df Sum of Sq RSS AIC  - Month 1 40.0 54782 651.19  - Day 1 637.6 55380 652.29  <none> 54742 653.11  - Solar.R 1 4624.2 59366 659.38  - Temp 1 7140.7 61883 663.62  - Wind 1 9650.3 64392 667.67  Step: AIC=651.19  Ozone ~ Solar.R + Wind + Temp + Day  Df Sum of Sq RSS AIC  - Day 1 607.5 55390 650.31  <none> 54782 651.19  - Solar.R 1 5346.6 60129 658.69  - Temp 1 8242.1 63024 663.48  - Wind 1 9781.6 64564 665.94  Step: AIC=650.31  Ozone ~ Solar.R + Wind + Temp  Df Sum of Sq RSS AIC  <none> 55390 650.31  - Solar.R 1 5346.0 60736 657.71  - Temp 1 7647.5 63037 661.50  - Wind 1 10505.3 65895 666.03  > # AIC removes Month and Day, but keeps Wind, Temp and Solar.R  > # create prediction  > lmPred <- predict(lmOutput, testData, type="response")  > str(lmPred)  Named num [1:51] 21.3 70 67.2 50.5 45.2 ...  - attr(\*, "names")= chr [1:51] "138" "80" "43" "81" ...  > lmPred <- data.frame(lmPred)  > compTable <- data.frame(testData[,1],lmPred[,1])  > colnames(compTable) <- c("test", "Pred")  > # this is the RMSE (how low)  > RSMElm<-sqrt(mean((compTable$test-compTable$Pred)^2))  > RSMElm  [1] 14.15575  > # compute absolute error for each case  > compTable$error <- abs(compTable$test - compTable$Pred)  > # create a new dataframe contains error, tempreture and wind  > lmPlot <- data.frame(compTable$error, testData$Temp, testData$Wind)  > # assign column names  > colnames(lmPlot) <- c("error","Temp","Wind")  > # polt result using ggplot, setting "Temp" as x-axis and "Wind" as y-axis  > # use point size and color shade to illustrate how big is the error  > plotlm1 <- ggplot(lmPlot, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=error, color=error))+  + ggtitle("lm")  > plotlm1  > # 5) Show all three results (charts) in one window, using grid.arrange function  > grid.arrange(plotksvm1, plotsvm1, plotlm1, ncol=2, nrow=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.  > # reset Train and Test data  > randIndex <- sample(1:dim(myairQ)[1])  > summary(randIndex)  Min. 1st Qu. Median Mean 3rd Qu. Max.  1 39 77 77 115 153  > length(randIndex)  [1] 153  > head(randIndex)  [1] 134 107 92 149 139 3  > cutPoint2\_3 <- floor(2 \* dim(myairQ)[1]/3)  > cutPoint2\_3  [1] 102  > trainData<- myairQ[randIndex[1:cutPoint2\_3],]  > testData<- myairQ[randIndex[(cutPoint2\_3+1):dim(myairQ)[1]],]  > # calculate average Ozone  > meanOzone <- mean(myairQ$Ozone,na.rm=TRUE)  > # create a new variable named "goodOzone" in train data set  > # goodOzone = 0 if Ozone is below average Ozone  > # googOzone = 1 if Ozone is eaqual or above the average ozone  > trainData$goodOzone <- ifelse(trainData$Ozone<meanOzone, 0, 1)  > # do the same thing for test dataset  > testData$goodOzone <- ifelse(testData$Ozone<meanOzone, 0, 1)  > # remove "Ozone" from train data  > trainData <- trainData[,-1]  > # remove "Ozone" from test data  > testData <- testData[,-1]  > # Step 5: See if we can do a better job predicting 'good' and 'bad' days  > # 1) Build a model (using the 'ksvm' function, trying to predict 'goodOzone'). You can use  > # all the possible attributes, or select attributes tht yuo think would be most helpful.  > # convert "goodOzone" in train data from numeric to factor  > trainData$goodOzone <- as.factor(trainData$goodOzone)  > # convert "goodOzone" in test data from numeric to factor  > testData$goodOzone <- as.factor(testData$goodOzone)  > # build a model using ksvm function,and use all other variables to predict  > ksvmGoodO <- ksvm(goodOzone~.,data=trainData,kernel="rbfdot",kpar="automatic",C=10,cross=10,prob.model=TRUE)  >  > # 2) Test the model on the resting dataset, and compute the precent of 'goodOzone' that was  > # correctly predicted.  > goodPredO <- predict(ksvmGoodO, testData)  > # create a dataframe that contains the exact "goodOzone" value and the predicted "goodOzone"  > compGood1 <- data.frame(testData[,6], goodPredO)  > # change column names  > colnames(compGood1) <- c("test","Pred")  > # Compute the percentage of correct cases  > perc\_ksvm <- length(which(compGood1$test==compGood1$Pred))/dim(compGood1)[1]  > perc\_ksvm  [1] 0.7254902  > # Confusion Matrix  > results <- table(test=compGood1$test, pred=compGood1$Pred)  > print(results)  pred  test 0 1  0 14 9  1 5 23  > # 3) Plot the results. Use a scatter plot. Have the x-axis represent temperature, the y-axis  > # represent wind, the shape representing what was predicted (good or bad day), the  > # color representing the actual value of 'goodOzone' (i.e. if the actual ozone level was  > # good) and the size reprsent if the prediction was correct (larger symbols should be the  > # observations that the model got wrong).  > # determine the prediction is "correct" or "wrong" for each case  > compGood1$correct <- ifelse(compGood1$test==compGood1$Pred,"correct","wrong")  > # create a new dataframe contains correct, tempreture and wind, and goodZone  > Plot\_ksvm <- data.frame(compGood1$correct,testData$Temp,testData$Wind,testData$goodOzone,compGood1$Pred)  > # change column names  > colnames(Plot\_ksvm) <- c("correct","Temp","Wind","goodOzone","Predict")  > # plot result using ggplot  > # size representing correct/wrong; color representing actual good/bad day; shape representing predicted good/bad day.  > ksvmplot <- ggplot(Plot\_ksvm, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=correct,color=goodOzone,shape = Predict))+  + ggtitle("ksvm - good/bad ozone")  > ksvmplot  Warning message:  Using size for a discrete variable is not advised.  > # 4) Compute the models and plot the results for 'svm' (in the e1071 package) and 'nb'  > # (Naive Bayes, also in the e1071 package).  > # SVM  > svmGoodO <- svm(goodOzone~.,data=trainData)  > # create prediction  > goodPredO <- predict(svmGoodO, testData)  > # create a dataframe that contains the exact "goodOzone" value and the predicted "goodOzone"  > compGood1 <- data.frame(testData[,6], goodPredO)  > # change column names  > colnames(compGood1) <- c("test","Pred")  > # Compute the percentage of correct cases  > perc\_svm <- length(which(compGood1$test==compGood1$Pred))/dim(compGood1)[1]  > perc\_svm  [1] 0.745098  > # Confusion Matrix  > results <- table(test=compGood1$test, pred=compGood1$Pred)  > print(results)  pred  test 0 1  0 17 6  1 7 21  > # determine the prediction is "correct" or "wrong" for each case  > compGood1$correct <- ifelse(compGood1$test==compGood1$Pred,"correct","wrong")  > # create a new dataframe contains correct, temperature and wind, and goodZone  > Plot\_svm <- data.frame(compGood1$correct,testData$Temp,testData$Wind,testData$goodOzone,compGood1$Pred)  > # change column names  > colnames(Plot\_svm) <- c("correct","Temp","Wind","goodOzone","Predict")  > # plot result using ggplot  > # size representing correct/wrong; color representing actual good/bad day; shape representing predicted good/bad day.  > svmplot <- ggplot(Plot\_svm, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=correct,color=goodOzone,shape = Predict))+  + ggtitle("svm - good/bad ozone")  > svmplot  Warning message:  Using size for a discrete variable is not advised.  > # Naive Bayes  > nbGoodO <- naiveBayes(goodOzone~.,data=trainData)  > # create prediction  > goodPredO <- predict(nbGoodO, testData)  > # create a dataframe that contains the exact "goodOzone" value and the predicted "goodOzone"  > compGood1 <- data.frame(testData[,6], goodPredO)  > # change column names  > colnames(compGood1) <- c("test","Pred")  > # Compute the percentage of correct cases  > perc\_nb <- length(which(compGood1$test==compGood1$Pred))/dim(compGood1)[1]  > perc\_nb  [1] 0.8039216  > # Confusion Matrix  > results <- table(test=compGood1$test, pred=compGood1$Pred)  > print(results)  pred  test 0 1  0 18 5  1 5 23  > # determine the prediction is "correct" or "wrong" for each case  > compGood1$correct <- ifelse(compGood1$test==compGood1$Pred,"correct","wrong")  > # create a new dataframe contains correct, tempreture and wind, and goodZone  > Plot\_nb <- data.frame(compGood1$correct,testData$Temp,testData$Wind,testData$goodOzone,compGood1$Pred)  > # change column names  > colnames(Plot\_nb) <- c("correct","Temp","Wind","goodOzone","Predict")  > # plot result using ggplot  > # size representing correct/wrong; color representing actual good/bad day; shape representing predicted good/bad day.  > nbplot <- ggplot(Plot\_nb, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=correct,color=goodOzone,shape = Predict))+  + ggtitle("nb - good/bad ozone")  > nbplot  Warning message:  Using size for a discrete variable is not advised.  > # 5) Show all three results (charts) in one window, using the grid.arrange function (have  > # two charts in one row).  > grid.arrange(ksvmplot,svmplot,nbplot, ncol=2, nrow=2)  Warning messages:  1: Using size for a discrete variable is not advised.  2: Using size for a discrete variable is not advised.  3: Using size for a discrete variable is not advised.  > # Step 6: Which are the best models for this data?  > # Review what you have done and state which is the best and why  > # ANSWER: If you just review the Root Mean Squared Error (RMSE) for predicting Ozone from  > # the other variables: Wind, Temp, Solar.R, you see that depending on the training data and the  > # test data - the "best model" varies. I ran this code multiple times and I found that  > # the usually the svm had the lowest RMSE, so I would think it would be the best model; however,  > # this was not always the case. It depended on the random training and test data.  > # For example, in the run that I turned in for homework - these are the values.  > # for ksvm, RSME = 16.99584  > # for svm, RSME = 13.63764  > # for lm, RSME = 14.15575  > # we would conclude that svm is the best model for this set of data since it has the smallest  > # Root Mean Squared Error.  > #  > # The same occurred when reviewing the Good/Bad Ozone prediction and look at the models  > # and the "Percent Good" with varying values for the percent that was properly predicted.  > # For example, in the run that I turned in for homework - you would see the following:  > # perc\_ksvm = 72.54902%  > # perc\_svm = 74.5098%  > # perc\_nb = 80.39216%  > # We would again conclude that the Naive Bayes (nb) is the best model for this set of data since  > # it carries the highest percent of correct predictions  > #  > # The plots also show the same, the svm plot shows smaller (less error) size and darker color  > # points in the plot. The plot for good/bad ozone, the nb plot shows many smaller (correct)  > # triangles in the nb plot. |
|  |

## Visualizations















