# Ryan Timbrook

## **Applied Data Science**

## **IST687 Intro to Data Science**, Spring 2019

## **Due Date:** 06/4/2019

## **Homework:** 9

### NetID: RTIMBROO

### SUID: 386792749

## #R Code - unexecuted

## Homework Week 9: Support Vector Machines

#--- Preprocess Steps:----------------------------------------------------------------------

### Clear objects from Memory

rm(list=ls())

### Clear Console:

cat("\014")

### Set Working Directory

setwd("C:\\workspaces\\ms\_datascience\_su\\IST687-IntroDataScience\\R\_workspace\\hw")

#---- Global Variable Assignments --------------------------------------------

#---- Load Required Packages -------------------------------------------------

if(!require("devtools")) {install.packages("devtools")}

devtools::install\_github("dkahle/ggmap")

if(!require("ggplot2")){install.packages("ggplot2")}

if(!require("dplyr")) {install.packages("dplyr")}

if(!require("e1071")) {install.packages("e1071")}

if(!require("arulesViz")) {install.packages("arulesViz")}

if(!require("gridExtra")) {install.packages("gridExtra")}

if(!require("caret")) {install.packages("caret")}

if(!require("kernlab")) {install.packages("kernlab")}

if(!require("arules")) {install.packages("arules")}

#---- Step 1: Load the data -------------------------

## Air Quality dataset

##-- 1.1: Clean the dataset

air <- airquality

#-- 1.2: Clean the data --------------------------------------------------

### Replace NA with column means

na.2.mean <- function(x){

replace(x, is.na(x), mean(x, na.rm = TRUE))

}

cleanDataSet <- function(ds){

#Make all empty cells equal to NA

ds[ds==""] <- NA

#Clean NA Columns from Dataframe

ds <- ds[ ,!apply(ds,2,function(x) all(is.na(x)))]

#Clean empty Rows from Dataframe

ds <- ds[!apply(ds,1,function(x) all(is.na(x))),]

# replace NA's in Ozone col with mean of col (where NA is discarded when calculating the mean)

ds$Ozone[is.na(ds$Ozone)] <- mean(ds$Ozone,na.rm=TRUE)

ds$Ozone <- round(ds$Ozone)

ds$Solar.R[is.na(ds$Solar.R)] <- mean(ds$Solar.R,na.rm=TRUE)

ds$Solar.R <- round(ds$Solar.R)

return(ds)

}

clean.air <- cleanDataSet(air)

#-- 1.3: Understand the data ---------------------------------------------

str(clean.air)

summary(clean.air)

head(clean.air)

#---- Step 2: Create train and test data sets -------------------------

# Set repeatable random seed

set.seed(4)

partitionDataSet <- function(ds, fractionOfTest = 0.3){

randoms <- runif(nrow(ds))

cutoff <- quantile(randoms, fractionOfTest)

testFlag <- randoms <= cutoff

testingData <- ds[testFlag,]

trainingData <- ds[!testFlag,]

dataSetSplit <- list(trainingData=trainingData, testingData=testingData)

return(dataSetSplit)

}

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

dim(clean.air)

clean.air[1:5,]

randIndex <- sample(1:nrow(clean.air))

randIndex

length(randIndex)

## Create a 2/3 cutpoint and round the number

cutPoint <- floor(2\*nrow(clean.air)/3)

cutPoint

## Create train data set, contains the first 2/3 of overall data

train <- clean.air[randIndex[1:cutPoint],]

dim(train)

head(train)

## Create test data set, contains the rest of the 1/3 data that remains

test <- clean.air[randIndex[(cutPoint+1):nrow(clean.air)],]

dim(test)

head(test)

## Test exact split function

airDataSetSplits <- partitionDataSet(clean.air,0.33)

dim(airDataSetSplits$trainingData)

head(airDataSetSplits$trainingData)

dim(airDataSetSplits$testingData)

head(airDataSetSplits$testingData)

#---- Step 2.1: LM Model ---------------------------------------------------------------

airLmModel <- lm(Ozone ~ .,data=train)

summary(airLmModel)

airLmPred <- predict(airLmModel,test)

airLmPred

str(airLmPred)

compTable3 <- data.frame(test[,1],round(airLmPred))

colnames(compTable3) <- c("test","Pred")

# RMSE = 18.8

round(sqrt(mean((compTable3$test-compTable3$Pred)^2)),1)

#lm plot

compTable3$error <- abs(compTable3$test - compTable3$Pred)

plot3 <- data.frame(compTable3$error,test$Temp, test$Wind)

colnames(plot3) <- c("error","Temp","Wind")

plot.lm.Ozone <- ggplot(plot3, aes(x=Temp, y=Wind)) +

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

ggtitle("Linear Model (LM), Airquality, Predict Ozone levels with Error dimension")

ggsave("LM\_Scatter\_Plot\_Prediction\_of\_Ozone.jpg", width = 6, height = 6)

plot.lm.Ozone

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

##-- 3.1: 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.

## Training Step - Ozone is the target predicting variable

# Kernel -> rdfdot: is the kernal function that projects the low-dimensional problem into higher-dimensional space

# i.e., getting the maximum separation of distance between Ozone cases

# results: Training error = 0.081

# Cross validation error = 568.72

# Support Vectors = 91

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

ksvmOzoneOutput

##-- 3.2: Test the model on the testing dataset, and compute the Root Mean Squared Error

ksvmOzonePred <- predict(ksvmOzoneOutput, test, type="votes")

ksvmOzonePred

# Create a comparison dataframe that contains the exact 'Ozone' value and the predicted 'Ozone' value

# use for RMSE calc

ksvmCompTable <- data.frame(test[,1],ksvmOzonePred[,1])

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

head(ksvmCompTable)

# Compute the Root Mean Squared Error - A smaller value indicates better model performance

# RMSE = 21.59642

sqrt(mean((ksvmCompTable$test - ksvmCompTable$Pred)^2))

##-- 3.3: 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)

# Compute the absolute error for each case

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

# Create new dataframe contains error, temperature and wind

ksvmOzonePlotDf <- data.frame(ksvmCompTable$error, test$Temp, test$Wind, test$Ozone)

colnames(ksvmOzonePlotDf) <- c("error","Temp","Wind","Ozone")

# Plot results - using point size and color shade to illustrate how big the error is

plot.ksvm.Ozone <- ggplot(ksvmOzonePlotDf, aes(x=Temp, y=Wind)) +

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

ggtitle("KSVM Scatter Plot, Prediction of Ozone with Error dimensions")

ggsave("KSVM\_Scatter\_Plot\_Prediction\_of\_Ozone\_With\_Error.jpg", width = 6, height = 6)

plot.ksvm.Ozone

##-- 3.4: Compute models and plot the results for 'svm' (in the e1071 package)

#

## Training Step - Ozone is the target predicting variable

# Kernel -> rdfdot: is the kernal function that projects the low-dimensional problem into higher-dimensional space

# i.e., getting the maximum separation of distance between Ozone cases

svmOzoneOutput <- svm(Ozone ~ ., data=train, kernel="radial", C=10, cross=10, prob.model=TRUE )

svmOzoneOutput

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

svmOzonePred <- predict(svmOzoneOutput, test)

svmOzonePred

str(svmOzonePred)

# Create a comparison dataframe that contains the exact 'Ozone' value and the predicted 'Ozone' value

# use for RMSE calc

svmCompTable <- data.frame(select(test,'Ozone'),svmOzonePred)

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

head(svmCompTable)

# Compute the Root Mean Squared Error - A smaller value indicates better model performance

# RMSE = 16.54

sqrt(mean((svmCompTable$test - svmCompTable$Pred)^2))

##-- 3.3: 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)

# Compute the absolute error for each case

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

# Create new dataframe contains error, temperature and wind

svmOzonePlotDf <- data.frame(round(svmCompTable$error,2), test$Temp, test$Wind, test$Ozone)

colnames(svmOzonePlotDf) <- c("error","Temp","Wind","Ozone")

# Plot results - using point size and color shade to illustrate how big the error is

plot.svm.Ozone <- ggplot(svmOzonePlotDf, aes(x=Temp, y=Wind)) +

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

ggtitle("SVM Scatter Plot, Prediction of Ozone with Error dimensions")

ggsave("SVM\_Scatter\_Plot\_Prediction\_of\_Ozone\_With\_Error.jpg", width = 6, height = 6)

plot.svm.Ozone

##-- 3.5: Show all three results (charts) in one window, using the grid.arrange function

ga3 <- grid.arrange(plot.ksvm.Ozone, plot.svm.Ozone, plot.lm.Ozone, ncol=3, nrow=2, top="Step 3 Model Comparisions")

ggsave(file="Grid\_Arrange\_KSVM-SVM-LM.jpg", ga3, width = 24, height = 12)

#---- 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.

avgOzone <- round(mean(clean.air$Ozone))

avgOzone

train$goodOzone <- ifelse(train$Ozone<avgOzone,0,1)

test$goodOzone <- ifelse(test$Ozone<avgOzone,0,1)

head(train)

head(test)

#---- Step 5: See if we can do a better job predicting 'good' and 'bad' days -------------------------

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

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

train <- select(train,-'Ozone')

test <- select(test,-'Ozone')

str(train)

str(test)

##-- 5.1: 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.

# Output Results:

# Training error: 0.098

# Cross Validation error: 0.354

# Support Vectors: 61

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

ksvmOzoneOutputGood

##-- 5.2: Test the model on the testing dataset, and compute the percent of 'goodOzone' that was correctly predicted.

ksvmOzonePredGood <- predict(ksvmOzoneOutputGood, test)

ksvmOzoneCompGood1 <- data.frame(select(test,'goodOzone'), ksvmOzonePredGood)

colnames(ksvmOzoneCompGood1) <- c('test','Pred')

head(ksvmOzoneCompGood1)

# Percent of goodOzone that was correctly predicted

# Output: 0.705 or 70%

percKSVMCorrect <- length(which(ksvmOzoneCompGood1$test==ksvmOzoneCompGood1$Pred))/dim(ksvmOzoneCompGood1)[1]

percKSVMCorrect

# Confusion Matrix

# result output: 0 class, 18 identified correctly, 5 identified incorrectly

# result output: 1 class, 10 identified incorrectly, 18 identified correctly

ksvmResults <- table(test=ksvmOzoneCompGood1$test, pred=ksvmOzoneCompGood1$Pred)

print(ksvmResults)

length(ksvmOzoneCompGood1$test)

# Error & Accuracy Score Rate:

ksvmErrorRate <- round((ksvmResults[1,][[2]] + ksvmResults[2,][[1]]) / nrow(ksvmOzoneCompGood1) \*100,2)

ksvmErrorRate

ksvmAccuracyRate <- 100-ksvmErrorRate

ksvmAccuracyRate

##-- 5.3: 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)

ksvmOzoneCompGood1$correct <- ifelse(ksvmOzoneCompGood1$test==ksvmOzoneCompGood1$Pred,'correct','wrong')

plotksvmOzoneCompGood1 <- data.frame(ksvmOzoneCompGood1$correct,test$Temp,test$Wind,test$goodOzone,ksvmOzoneCompGood1$Pred)

colnames(plotksvmOzoneCompGood1) <- c("correct","Temp","Wind","goodOzone","Predict")

plot.ksvm.good <- ggplot(plotksvmOzoneCompGood1, aes(x=Temp,y=Wind)) +

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

ggtitle("KSVM - Good/Bad Ozone Prediction")

ggsave("KSVM\_Good\_Bad\_Ozone\_Prediction.jpg", width = 6, height = 6)

plot.ksvm.good

##-- 5.3: Compute models and plot the results for 'svm' (in the e1071 package) and 'nb' (Naive Bayses)

#

## 5.3.1: Models & Plots for svm

svmOzoneOutputGood <- svm(goodOzone ~ ., data=train, kernel='radial', C=10, cross=10, prob.model=TRUE)

svmOzoneOutputGood

# Test the svm model

svmOzonePredGood <- predict(svmOzoneOutputGood, test)

svmOzoneCompGood1 <- data.frame(select(test,'goodOzone'), svmOzonePredGood)

colnames(svmOzoneCompGood1) <- c('test','Pred')

head(svmOzoneCompGood1)

# Percent of goodOzone that was correctly predicted

percSVMCorrect <- length(which(svmOzoneCompGood1$test==svmOzoneCompGood1$Pred))/dim(svmOzoneCompGood1)[1]

percSVMCorrect

# Confusion Matrix

# result output: 0 class, 21 identified correctly, 5 identified incorrectly

# result output: 1 class, 7 identified incorrectly, 18 identified correctly

svmResults <- table(test=svmOzoneCompGood1$test, pred=svmOzoneCompGood1$Pred)

print(svmResults)

# Error & Accuracy Score Rate:

svmErrorRate <- round((svmResults[1,][[2]] + svmResults[2,][[1]]) / nrow(svmOzoneCompGood1) \*100,2)

svmErrorRate

svmAccuracyRate <- 100-svmErrorRate

svmAccuracyRate

## 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)

svmOzoneCompGood1$correct <- ifelse(svmOzoneCompGood1$test==svmOzoneCompGood1$Pred,'correct','wrong')

plotSvmOzoneCompGood1 <- data.frame(svmOzoneCompGood1$correct,test$Temp,test$Wind,test$goodOzone,svmOzoneCompGood1$Pred)

colnames(plotSvmOzoneCompGood1) <- c("correct","Temp","Wind","goodOzone","Predict")

plot.svm.good <- ggplot(plotSvmOzoneCompGood1, aes(x=Temp,y=Wind)) +

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

ggtitle("SVM - Good/Bad Ozone Prediction")

ggsave("SVM\_Good\_Bad\_Ozone\_Prediction.jpg", width = 6, height = 6)

plot.svm.good

## 5.3.2: Models & Plots for 'nb'

nbOzoneOutputGood <- naiveBayes(goodOzone ~ ., data=train)

nbOzoneOutputGood

# Test the naiveBase model

nbOzonePredGood <- predict(nbOzoneOutputGood, test)

nbOzoneCompGood1 <- data.frame(select(test,'goodOzone'), nbOzonePredGood)

colnames(nbOzoneCompGood1) <- c('test','Pred')

head(nbOzoneCompGood1)

# Percent of goodOzone that was correctly predicted

percNBCorrect <- length(which(nbOzoneCompGood1$test==nbOzoneCompGood1$Pred))/dim(nbOzoneCompGood1)[1]

percNBCorrect

# Confusion Matrix

# result output: 0 class, 19 identified correctly, 3 identified incorrectly

# result output: 1 class, 9 identified incorrectly, 20 identified correctly

nbResults <- table(test=nbOzoneCompGood1$test, pred=nbOzoneCompGood1$Pred)

print(nbResults)

nbErrorRate <- round((nbResults[1,][[2]] + nbResults[2,][[1]]) / nrow(nbOzoneCompGood1) \*100,2)

nbErrorRate

nbAccuracyRate <- 100-nbErrorRate

nbAccuracyRate

## 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)

nbOzoneCompGood1$correct <- ifelse(nbOzoneCompGood1$test==nbOzoneCompGood1$Pred,'correct','wrong')

plotNBOzoneCompGood1 <- data.frame(nbOzoneCompGood1$correct,test$Temp,test$Wind,test$goodOzone,nbOzoneCompGood1$Pred)

colnames(plotNBOzoneCompGood1) <- c("correct","Temp","Wind","goodOzone","Predict")

plot.nb.good <- ggplot(plotNBOzoneCompGood1, aes(x=Temp,y=Wind)) +

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

ggtitle("Niave Baise - Good/Bad Ozone Prediction")

ggsave("NB\_Good\_Bad\_Ozone\_Prediction.jpg", width = 6, height = 6)

plot.nb.good

##-- 5.5: Show all three results (charts) in one window, using the grid.array function (have two charts in one row)

ga5 <- grid.arrange(plot.ksvm.good, plot.svm.good, plot.nb.good, ncol=3, nrow=2, top="Step 5 Model Comparisions")

ggsave(file="Grid\_Arrange\_Good\_KSVM-SVM-NB.jpg", ga5, width = 24, height = 12)

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

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

#

# Answer: It's observed that the SVM and Naive Baise models have the heighest accuracy ratings, measured at 76.5%,

# for predicting the goodOzone Class. The result output calculation of Accuracy Rating for each model is shown below.

#

## Output Results ##

# Step 3 results:

# LM: RMSE = 18.8

# KSVM: RMSE = 21.59642

# SVM: RMSE = 16.54

#

# Step 5 results:

# KSVM:

# Accuracy Rate: 70.59%

# Error Rate: 29.41%

# SVM:

# Accuracy Rate: 76.47%

# Error Rate: 23.53%

# Naive Baise:

# Accuracy Rate: 76.47%

# Error Rate: 23.53%

#--------------------------------------------------------------------------------

## #R Code – executed

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  | | --- | | > ### Set Working Directory  > setwd("C:\\workspaces\\ms\_datascience\_su\\IST687-IntroDataScience\\R\_workspace\\hw")  >  > #---- Global Variable Assignments --------------------------------------------  >  >  > #---- Load Required Packages -------------------------------------------------  > if(!require("devtools")) {install.packages("devtools")}  > devtools::install\_github("dkahle/ggmap")  > if(!require("ggplot2")){install.packages("ggplot2")}  > if(!require("dplyr")) {install.packages("dplyr")}  > if(!require("e1071")) {install.packages("e1071")}  > if(!require("arulesViz")) {install.packages("arulesViz")}  > if(!require("gridExtra")) {install.packages("gridExtra")}  > if(!require("caret")) {install.packages("caret")}  > if(!require("kernlab")) {install.packages("kernlab")}  > if(!require("arules")) {install.packages("arules")}  >  >  > #---- Step 1: Load the data -------------------------  > ## Air Quality dataset  >  > ##-- 1.1: Clean the dataset  > air <- airquality  >  > #-- 1.2: Clean the data --------------------------------------------------  >  > ### Replace NA with column means  > na.2.mean <- function(x){  + replace(x, is.na(x), mean(x, na.rm = TRUE))  + }  >  > cleanDataSet <- function(ds){  + #Make all empty cells equal to NA  + ds[ds==""] <- NA  +  + #Clean NA Columns from Dataframe  + ds <- ds[ ,!apply(ds,2,function(x) all(is.na(x)))]  +  + #Clean empty Rows from Dataframe  + ds <- ds[!apply(ds,1,function(x) all(is.na(x))),]  +  + # replace NA's in Ozone col with mean of col (where NA is discarded when calculating the mean)  + ds$Ozone[is.na(ds$Ozone)] <- mean(ds$Ozone,na.rm=TRUE)  + ds$Ozone <- round(ds$Ozone)  + ds$Solar.R[is.na(ds$Solar.R)] <- mean(ds$Solar.R,na.rm=TRUE)  + ds$Solar.R <- round(ds$Solar.R)  +  + return(ds)  + }  >  > clean.air <- cleanDataSet(air)  >  > #-- 1.3: Understand the data ---------------------------------------------  > str(clean.air)  'data.frame': 153 obs. of 6 variables:  $ Ozone : num 41 36 12 18 42 28 23 19 8 42 ...  $ Solar.R: num 190 118 149 313 186 186 299 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 ...  > summary(clean.air)  Ozone Solar.R Wind Temp Month Day  Min. : 1.0 Min. : 7.0 Min. : 1.700 Min. :56.00 Min. :5.000 Min. : 1.0  1st Qu.: 21.0 1st Qu.:120.0 1st Qu.: 7.400 1st Qu.:72.00 1st Qu.:6.000 1st Qu.: 8.0  Median : 42.0 Median :194.0 Median : 9.700 Median :79.00 Median :7.000 Median :16.0  Mean : 42.1 Mean :185.9 Mean : 9.958 Mean :77.88 Mean :6.993 Mean :15.8  3rd Qu.: 46.0 3rd Qu.:256.0 3rd Qu.:11.500 3rd Qu.:85.00 3rd Qu.:8.000 3rd Qu.:23.0  Max. :168.0 Max. :334.0 Max. :20.700 Max. :97.00 Max. :9.000 Max. :31.0  > head(clean.air)  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 -------------------------  > # Set repeatable random seed  > set.seed(4)  > partitionDataSet <- function(ds, fractionOfTest = 0.3){  + randoms <- runif(nrow(ds))  + cutoff <- quantile(randoms, fractionOfTest)  + testFlag <- randoms <= cutoff  + testingData <- ds[testFlag,]  + trainingData <- ds[!testFlag,]  + dataSetSplit <- list(trainingData=trainingData, testingData=testingData)  + return(dataSetSplit)  + }  >  >  > ## Using techniques discussed in class, create two datasets - one for training and one for testing.  > dim(clean.air)  [1] 153 6  > clean.air[1:5,]  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  > randIndex <- sample(1:nrow(clean.air))  > randIndex  [1] 90 2 45 42 122 39 107 133 138 11 108 41 15 134 58 63 140 80 130 103 96 132 67 64 84 147 62 128 65 66 70 30 126 79 139  [36] 115 54 73 151 1 121 28 145 20 99 10 97 95 76 59 142 77 91 81 82 150 18 17 85 123 53 7 78 83 21 56 92 118 69 109  [71] 29 52 34 112 124 117 25 100 110 143 131 129 137 125 111 4 68 120 14 60 13 8 33 146 89 5 44 102 152 38 94 61 93 27 23  [106] 136 32 74 57 36 12 135 43 75 149 40 98 153 47 3 119 106 50 127 105 104 31 6 51 9 46 72 148 116 16 19 71 22 87 114  [141] 48 24 101 49 55 144 113 26 35 37 141 88 86  > length(randIndex)  [1] 153  >  > ## Create a 2/3 cutpoint and round the number  > cutPoint <- floor(2\*nrow(clean.air)/3)  > cutPoint  [1] 102  >  > ## Create train data set, contains the first 2/3 of overall data  > train <- clean.air[randIndex[1:cutPoint],]  > dim(train)  [1] 102 6  > head(train)  Ozone Solar.R Wind Temp Month Day  90 50 275 7.4 86 7 29  2 36 118 8.0 72 5 2  45 42 332 13.8 80 6 14  42 42 259 10.9 93 6 11  122 84 237 6.3 96 8 30  39 42 273 6.9 87 6 8  >  > ## Create test data set, contains the rest of the 1/3 data that remains  > test <- clean.air[randIndex[(cutPoint+1):nrow(clean.air)],]  > dim(test)  [1] 51 6  > head(test)  Ozone Solar.R Wind Temp Month Day  93 39 83 6.9 81 8 1  27 42 186 8.0 57 5 27  23 4 25 9.7 61 5 23  136 28 238 6.3 77 9 13  32 42 286 8.6 78 6 1  74 27 175 14.9 81 7 13  >  > ## Test exact split function  > airDataSetSplits <- partitionDataSet(clean.air,0.33)  > dim(airDataSetSplits$trainingData)  [1] 102 6  > head(airDataSetSplits$trainingData)  Ozone Solar.R Wind Temp Month Day  1 41 190 7.4 67 5 1  3 12 149 12.6 74 5 3  5 42 186 14.3 56 5 5  6 28 186 14.9 66 5 6  7 23 299 8.6 65 5 7  8 19 99 13.8 59 5 8  >  > dim(airDataSetSplits$testingData)  [1] 51 6  > head(airDataSetSplits$testingData)  Ozone Solar.R Wind Temp Month Day  2 36 118 8.0 72 5 2  4 18 313 11.5 62 5 4  13 11 290 9.2 66 5 13  14 14 274 10.9 68 5 14  15 18 65 13.2 58 5 15  16 14 334 11.5 64 5 16  >  > #---- Step 2.1: LM Model ---------------------------------------------------------------  > airLmModel <- lm(Ozone ~ .,data=train)  > summary(airLmModel)  Call:  lm(formula = Ozone ~ ., data = train)  Residuals:  Min 1Q Median 3Q Max  -43.127 -13.609 -2.886 9.752 95.716  Coefficients:  Estimate Std. Error t value Pr(>|t|)  (Intercept) -32.72841 25.60172 -1.278 0.2042  Solar.R 0.06502 0.02694 2.413 0.0177 \*  Wind -3.19521 0.75902 -4.210 5.76e-05 \*\*\*  Temp 1.35679 0.31707 4.279 4.44e-05 \*\*\*  Month -2.30716 1.78393 -1.293 0.1990  Day 0.35835 0.24086 1.488 0.1401  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 21.92 on 96 degrees of freedom  Multiple R-squared: 0.5078, Adjusted R-squared: 0.4822  F-statistic: 19.81 on 5 and 96 DF, p-value: 1.615e-13  > airLmPred <- predict(airLmModel,test)  > airLmPred  93 27 23 136 32 74 57 36 12 135 43 75 149  42.422477 29.280078 17.374036 50.983276 50.733316 29.449796 51.271145 57.373034 39.305852 21.237579 69.412436 50.918221 41.301254  40 98 153 47 3 119 106 50 127 105 104 31 6  53.591597 66.400856 27.273631 28.445118 26.641619 69.622565 41.588951 30.340388 71.712818 45.734955 45.537288 64.456385 11.919004  51 9 46 72 148 116 16 19 71 22 87 114 48  39.708682 -21.263113 50.181500 47.879066 -10.796395 47.391629 33.275326 38.997316 63.193195 30.430706 49.483178 11.036158 9.533987  24 101 49 55 144 113 26 35 37 141 88 86  14.739633 62.047943 21.080146 61.269998 16.081890 28.126647 13.433045 51.530140 34.238725 24.918465 44.469998 64.344964  >  > str(airLmPred)  Named num [1:51] 42.4 29.3 17.4 51 50.7 ...  - attr(\*, "names")= chr [1:51] "93" "27" "23" "136" ...  > compTable3 <- data.frame(test[,1],round(airLmPred))  > colnames(compTable3) <- c("test","Pred")  >  > # RMSE = 18.8  > round(sqrt(mean((compTable3$test-compTable3$Pred)^2)),1)  [1] 18.8  >  > #lm plot  > compTable3$error <- abs(compTable3$test - compTable3$Pred)  > plot3 <- data.frame(compTable3$error,test$Temp, test$Wind)  > colnames(plot3) <- c("error","Temp","Wind")  > plot.lm.Ozone <- ggplot(plot3, aes(x=Temp, y=Wind)) +  + geom\_point(aes(size=error, color=error)) +  + ggtitle("Linear Model (LM), Airquality, Predict Ozone levels with Error dimension")  > ggsave("LM\_Scatter\_Plot\_Prediction\_of\_Ozone.jpg", width = 6, height = 6)  > plot.lm.Ozone  >  > #---- Step 3: Build a Model using KSVM & visualize the results -------------------------  >  > ##-- 3.1: 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.  >  > ## Training Step - Ozone is the target predicting variable  > # Kernel -> rdfdot: is the kernal function that projects the low-dimensional problem into higher-dimensional space  > # i.e., getting the maximum separation of distance between Ozone cases  > # results: Training error = 0.081  > # Cross validation error = 568.72  > # Support Vectors = 91  > ksvmOzoneOutput <- ksvm(Ozone ~ ., data=train, kernel="rbfdot", kpar="automatic", C=10, cross=10, prob.model=TRUE )  > ksvmOzoneOutput  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 = 0.220439906185414  Number of Support Vectors : 89  Objective Function Value : -149.9922  Training error : 0.073994  Cross validation error : 549.7396  Laplace distr. width : 46.05834  >  > ##-- 3.2: Test the model on the testing dataset, and compute the Root Mean Squared Error  > ksvmOzonePred <- predict(ksvmOzoneOutput, test, type="votes")  > ksvmOzonePred  [,1]  [1,] 18.166951  [2,] 64.455104  [3,] 10.950205  [4,] 51.717998  [5,] 50.727273  [6,] 29.994896  [7,] 43.561524  [8,] 31.858064  [9,] 12.435743  [10,] 28.666047  [11,] 41.849001  [12,] 62.588736  [13,] 35.126635  [14,] 49.318547  [15,] 83.397686  [16,] 18.111359  [17,] 25.381528  [18,] 47.930125  [19,] 85.677968  [20,] 25.732833  [21,] 34.860444  [22,] 74.933268  [23,] 28.909551  [24,] 46.429706  [25,] 98.285913  [26,] 35.394158  [27,] 37.294241  [28,] -2.640910  [29,] 23.822402  [30,] 23.763612  [31,] 22.429212  [32,] 49.031397  [33,] 28.316769  [34,] 36.086531  [35,] 50.183269  [36,] 55.290535  [37,] 44.862748  [38,] 24.706120  [39,] 35.732445  [40,] 40.480157  [41,] 57.484816  [42,] 5.916529  [43,] 72.844627  [44,] 23.595645  [45,] 33.838382  [46,] 69.411286  [47,] 33.658825  [48,] 34.473067  [49,] 30.457873  [50,] 57.916715  [51,] 67.121088  >  > # Create a comparison dataframe that contains the exact 'Ozone' value and the predicted 'Ozone' value  > # use for RMSE calc  > ksvmCompTable <- data.frame(test[,1],ksvmOzonePred[,1])  > colnames(ksvmCompTable) <- c("test","Pred")  > head(ksvmCompTable)  test Pred  1 39 18.16695  2 42 64.45510  3 4 10.95020  4 28 51.71800  5 42 50.72727  6 27 29.99490  >  > # Compute the Root Mean Squared Error - A smaller value indicates better model performance  > # RMSE = 21.59642  > sqrt(mean((ksvmCompTable$test - ksvmCompTable$Pred)^2))  [1] 22.26252  >  > ##-- 3.3: 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)  >  > # Compute the absolute error for each case  > ksvmCompTable$error <- abs(ksvmCompTable$test - ksvmCompTable$Pred)  >  > # Create new dataframe contains error, temperature and wind  > ksvmOzonePlotDf <- data.frame(ksvmCompTable$error, test$Temp, test$Wind, test$Ozone)  > colnames(ksvmOzonePlotDf) <- c("error","Temp","Wind","Ozone")  >  > # Plot results - using point size and color shade to illustrate how big the error is  > plot.ksvm.Ozone <- ggplot(ksvmOzonePlotDf, aes(x=Temp, y=Wind)) +  + geom\_point(aes(size=error, color=error)) +  + ggtitle("KSVM Scatter Plot, Prediction of Ozone with Error dimensions")  > ggsave("KSVM\_Scatter\_Plot\_Prediction\_of\_Ozone\_With\_Error.jpg", width = 6, height = 6)  > plot.ksvm.Ozone  >  > ##-- 3.4: Compute models and plot the results for 'svm' (in the e1071 package)  > #  > ## Training Step - Ozone is the target predicting variable  > # Kernel -> rdfdot: is the kernal function that projects the low-dimensional problem into higher-dimensional space  > # i.e., getting the maximum separation of distance between Ozone cases  > svmOzoneOutput <- svm(Ozone ~ ., data=train, kernel="radial", C=10, cross=10, prob.model=TRUE )  Warning message:  In cret$cresults \* scale.factor :  Recycling array of length 1 in vector-array arithmetic is deprecated.  Use c() or as.vector() instead.  > svmOzoneOutput  Call:  svm(formula = Ozone ~ ., data = train, kernel = "radial", C = 10, cross = 10, prob.model = TRUE)  Parameters:  SVM-Type: eps-regression  SVM-Kernel: radial  cost: 1  gamma: 0.2  epsilon: 0.1  Number of Support Vectors: 82  >  > ## Test the model on the testing dataset, and compute the Root Mean Squared Error  > svmOzonePred <- predict(svmOzoneOutput, test)  > svmOzonePred  93 27 23 136 32 74 57 36 12 135 43 75 149 40  33.049942 34.904249 13.695370 43.430237 37.549640 34.098356 44.226325 39.501597 17.253211 34.748474 48.616004 51.163768 32.850975 43.206289  98 153 47 3 119 106 50 127 105 104 31 6 51 9  75.356836 31.218378 32.640250 29.662234 77.516778 29.834708 22.189763 77.718924 36.941442 44.071532 51.688922 32.682183 27.935125 25.465356  46 72 148 116 16 19 71 22 87 114 48 24 101 49  29.382416 31.848861 29.087265 44.566448 24.371959 27.090378 58.732153 46.392391 45.646845 22.748363 45.517584 23.780091 66.941354 8.899384  55 144 113 26 35 37 141 88 86  48.469430 23.204711 41.317757 42.347772 35.325172 35.352311 20.619950 50.217865 64.976495  > str(svmOzonePred)  Named num [1:51] 33 34.9 13.7 43.4 37.5 ...  - attr(\*, "names")= chr [1:51] "93" "27" "23" "136" ...  >  > # Create a comparison dataframe that contains the exact 'Ozone' value and the predicted 'Ozone' value  > # use for RMSE calc  > svmCompTable <- data.frame(select(test,'Ozone'),svmOzonePred)  > colnames(svmCompTable) <- c("test","Pred")  > head(svmCompTable)  test Pred  93 39 33.04994  27 42 34.90425  23 4 13.69537  136 28 43.43024  32 42 37.54964  74 27 34.09836  >  > # Compute the Root Mean Squared Error - A smaller value indicates better model performance  > # RMSE = 16.54  > sqrt(mean((svmCompTable$test - svmCompTable$Pred)^2))  [1] 16.5422  >  > ##-- 3.3: 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)  >  > # Compute the absolute error for each case  > svmCompTable$error <- abs(svmCompTable$test - svmCompTable$Pred)  >  > # Create new dataframe contains error, temperature and wind  > svmOzonePlotDf <- data.frame(round(svmCompTable$error,2), test$Temp, test$Wind, test$Ozone)  > colnames(svmOzonePlotDf) <- c("error","Temp","Wind","Ozone")  >  > # Plot results - using point size and color shade to illustrate how big the error is  > plot.svm.Ozone <- ggplot(svmOzonePlotDf, aes(x=Temp, y=Wind)) +  + geom\_point(aes(size=error, color=error)) +  + ggtitle("SVM Scatter Plot, Prediction of Ozone with Error dimensions")  > ggsave("SVM\_Scatter\_Plot\_Prediction\_of\_Ozone\_With\_Error.jpg", width = 6, height = 6)  > plot.svm.Ozone  >  >  > ##-- 3.5: Show all three results (charts) in one window, using the grid.arrange function  > ga3 <- grid.arrange(plot.ksvm.Ozone, plot.svm.Ozone, plot.lm.Ozone, ncol=3, nrow=2, top="Step 3 Model Comparisions")  > ggsave(file="Grid\_Arrange\_KSVM-SVM-LM.jpg", ga3, width = 24, height = 12)  >  >  > #---- Step 4: Create a 'goodOzone' variable -------------------------  > ##-- This variable should be either 0 or 1. It should be 0 if the ozone is below the average for all  > # the data observations, and 1 if it is equal to or above the average ozone observed.  >  > avgOzone <- round(mean(clean.air$Ozone))  > avgOzone  [1] 42  > train$goodOzone <- ifelse(train$Ozone<avgOzone,0,1)  > test$goodOzone <- ifelse(test$Ozone<avgOzone,0,1)  > head(train)  Ozone Solar.R Wind Temp Month Day goodOzone  90 50 275 7.4 86 7 29 1  2 36 118 8.0 72 5 2 0  45 42 332 13.8 80 6 14 1  42 42 259 10.9 93 6 11 1  122 84 237 6.3 96 8 30 1  39 42 273 6.9 87 6 8 1  > head(test)  Ozone Solar.R Wind Temp Month Day goodOzone  93 39 83 6.9 81 8 1 0  27 42 186 8.0 57 5 27 1  23 4 25 9.7 61 5 23 0  136 28 238 6.3 77 9 13 0  32 42 286 8.6 78 6 1 1  74 27 175 14.9 81 7 13 0  >  > #---- Step 5: See if we can do a better job predicting 'good' and 'bad' days -------------------------  > train$goodOzone <- as.factor(train$goodOzone)  > test$goodOzone <- as.factor(test$goodOzone)  > train <- select(train,-'Ozone')  > test <- select(test,-'Ozone')  > str(train)  'data.frame': 102 obs. of 6 variables:  $ Solar.R : num 275 118 332 259 237 273 64 259 112 186 ...  $ Wind : num 7.4 8 13.8 10.9 6.3 6.9 11.5 9.7 11.5 6.9 ...  $ Temp : int 86 72 80 93 96 87 79 73 71 74 ...  $ Month : int 7 5 6 6 8 6 8 9 9 5 ...  $ Day : int 29 2 14 11 30 8 15 10 15 11 ...  $ goodOzone: Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 1 1 1 ...  > str(test)  'data.frame': 51 obs. of 6 variables:  $ Solar.R : num 83 186 25 238 286 175 127 220 256 259 ...  $ Wind : num 6.9 8 9.7 6.3 8.6 14.9 8 8.6 9.7 15.5 ...  $ Temp : int 81 57 61 77 78 81 78 85 69 76 ...  $ Month : int 8 5 5 9 6 7 6 6 5 9 ...  $ Day : int 1 27 23 13 1 13 26 5 12 12 ...  $ goodOzone: Factor w/ 2 levels "0","1": 1 2 1 1 2 1 2 2 1 1 ...  > ##-- 5.1: 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.  > # Output Results:  > # Training error: 0.098  > # Cross Validation error: 0.354  > # Support Vectors: 61  > ksvmOzoneOutputGood <- ksvm(goodOzone ~ ., data=train, kernel="rbfdot", kpar="automatic", C=10, cross=10, prob.model=TRUE)  > ksvmOzoneOutputGood  Support Vector Machine object of class "ksvm"  SV type: C-svc (classification)  parameter : cost C = 10  Gaussian Radial Basis kernel function.  Hyperparameter : sigma = 0.155172599536458  Number of Support Vectors : 58  Objective Function Value : -339.667  Training error : 0.117647  Cross validation error : 0.331818  Probability model included.  >  > ##-- 5.2: Test the model on the testing dataset, and compute the percent of 'goodOzone' that was correctly predicted.  > ksvmOzonePredGood <- predict(ksvmOzoneOutputGood, test)  > ksvmOzoneCompGood1 <- data.frame(select(test,'goodOzone'), ksvmOzonePredGood)  > colnames(ksvmOzoneCompGood1) <- c('test','Pred')  > head(ksvmOzoneCompGood1)  test Pred  93 0 0  27 1 1  23 0 0  136 0 0  32 1 0  74 0 1  >  > # Percent of goodOzone that was correctly predicted  > # Output: 0.705 or 70%  > percKSVMCorrect <- length(which(ksvmOzoneCompGood1$test==ksvmOzoneCompGood1$Pred))/dim(ksvmOzoneCompGood1)[1]  > percKSVMCorrect  [1] 0.7254902  >  > # Confusion Matrix  > # result output: 0 class, 18 identified correctly, 5 identified incorrectly  > # result output: 1 class, 10 identified incorrectly, 18 identified correctly  > ksvmResults <- table(test=ksvmOzoneCompGood1$test, pred=ksvmOzoneCompGood1$Pred)  > print(ksvmResults)  pred  test 0 1  0 19 9  1 5 18  > length(ksvmOzoneCompGood1$test)  [1] 51  >  > # Error & Accuracy Score Rate:  > ksvmErrorRate <- round((ksvmResults[1,][[2]] + ksvmResults[2,][[1]]) / nrow(ksvmOzoneCompGood1) \*100,2)  > ksvmErrorRate  [1] 27.45  >  > ksvmAccuracyRate <- 100-ksvmErrorRate  > ksvmAccuracyRate  [1] 72.55  >  > ##-- 5.3: Plot the results. Use a scatter plot. Have the x-axis represent temperature,  > # the y-axis represent wind, the shape representing what was predicted (good or bad day),  > # the color representing the actual value of 'goodozone' (i.e., if the actual ozone level was good)  > # and the size represent if the prediction was correct (larger symbols should be the observations the model got wrong)  > ksvmOzoneCompGood1$correct <- ifelse(ksvmOzoneCompGood1$test==ksvmOzoneCompGood1$Pred,'correct','wrong')  > plotksvmOzoneCompGood1 <- data.frame(ksvmOzoneCompGood1$correct,test$Temp,test$Wind,test$goodOzone,ksvmOzoneCompGood1$Pred)  > colnames(plotksvmOzoneCompGood1) <- c("correct","Temp","Wind","goodOzone","Predict")  > plot.ksvm.good <- ggplot(plotksvmOzoneCompGood1, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=correct, color=goodOzone, shape=Predict)) +  + ggtitle("KSVM - Good/Bad Ozone Prediction")  > ggsave("KSVM\_Good\_Bad\_Ozone\_Prediction.jpg", width = 6, height = 6)  Warning message:  Using size for a discrete variable is not advised.  > plot.ksvm.good  >  >  > ##-- 5.3: Compute models and plot the results for 'svm' (in the e1071 package) and 'nb' (Naive Bayses)  > #  > ## 5.3.1: Models & Plots for svm  > svmOzoneOutputGood <- svm(goodOzone ~ ., data=train, kernel='radial', C=10, cross=10, prob.model=TRUE)  > svmOzoneOutputGood  Call:  svm(formula = goodOzone ~ ., data = train, kernel = "radial", C = 10, cross = 10, prob.model = TRUE)  Parameters:  SVM-Type: C-classification  SVM-Kernel: radial  cost: 1  gamma: 0.2  Number of Support Vectors: 73  >  > # Test the svm model  > svmOzonePredGood <- predict(svmOzoneOutputGood, test)  > svmOzoneCompGood1 <- data.frame(select(test,'goodOzone'), svmOzonePredGood)  > colnames(svmOzoneCompGood1) <- c('test','Pred')  > head(svmOzoneCompGood1)  test Pred  93 0 1  27 1 0  23 0 0  136 0 0  32 1 0  74 0 0  >  > # Percent of goodOzone that was correctly predicted  > percSVMCorrect <- length(which(svmOzoneCompGood1$test==svmOzoneCompGood1$Pred))/dim(svmOzoneCompGood1)[1]  > percSVMCorrect  [1] 0.7647059  >  > # Confusion Matrix  > # result output: 0 class, 21 identified correctly, 5 identified incorrectly  > # result output: 1 class, 7 identified incorrectly, 18 identified correctly  > svmResults <- table(test=svmOzoneCompGood1$test, pred=svmOzoneCompGood1$Pred)  > print(svmResults)  pred  test 0 1  0 21 7  1 5 18  >  > # Error & Accuracy Score Rate:  > svmErrorRate <- round((svmResults[1,][[2]] + svmResults[2,][[1]]) / nrow(svmOzoneCompGood1) \*100,2)  > svmErrorRate  [1] 23.53  >  > svmAccuracyRate <- 100-svmErrorRate  > svmAccuracyRate  [1] 76.47  >  >  > ## Plot the results. Use a scatter plot. Have the x-axis represent temperature,  > # the y-axis represent wind, the shape representing what was predicted (good or bad day),  > # the color representing the actual value of 'goodozone' (i.e., if the actual ozone level was good)  > # and the size represent if the prediction was correct (larger symbols should be the observations the model got wrong)  > svmOzoneCompGood1$correct <- ifelse(svmOzoneCompGood1$test==svmOzoneCompGood1$Pred,'correct','wrong')  > plotSvmOzoneCompGood1 <- data.frame(svmOzoneCompGood1$correct,test$Temp,test$Wind,test$goodOzone,svmOzoneCompGood1$Pred)  > colnames(plotSvmOzoneCompGood1) <- c("correct","Temp","Wind","goodOzone","Predict")  > plot.svm.good <- ggplot(plotSvmOzoneCompGood1, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=correct, color=goodOzone, shape=Predict)) +  + ggtitle("SVM - Good/Bad Ozone Prediction")  > ggsave("SVM\_Good\_Bad\_Ozone\_Prediction.jpg", width = 6, height = 6)  Warning message:  Using size for a discrete variable is not advised.  > plot.svm.good  >  > ## 5.3.2: Models & Plots for 'nb'  > nbOzoneOutputGood <- naiveBayes(goodOzone ~ ., data=train)  > nbOzoneOutputGood  Naive Bayes Classifier for Discrete Predictors  Call:  naiveBayes.default(x = X, y = Y, laplace = laplace)  A-priori probabilities:  Y  0 1  0.4313725 0.5686275  Conditional probabilities:  Solar.R  Y [,1] [,2]  0 161.7727 100.59448  1 200.3448 74.10508  Wind  Y [,1] [,2]  0 10.625000 2.548609  1 8.532759 3.577188  Temp  Y [,1] [,2]  0 73.31818 8.083209  1 82.63793 8.313376  Month  Y [,1] [,2]  0 7.113636 1.701106  1 7.103448 1.149976  Day  Y [,1] [,2]  0 13.56818 7.531014  1 17.01724 10.089931  >  > # Test the naiveBase model  > nbOzonePredGood <- predict(nbOzoneOutputGood, test)  > nbOzoneCompGood1 <- data.frame(select(test,'goodOzone'), nbOzonePredGood)  > colnames(nbOzoneCompGood1) <- c('test','Pred')  > head(nbOzoneCompGood1)  test Pred  93 0 1  27 1 0  23 0 0  136 0 1  32 1 1  74 0 1  >  > # Percent of goodOzone that was correctly predicted  > percNBCorrect <- length(which(nbOzoneCompGood1$test==nbOzoneCompGood1$Pred))/dim(nbOzoneCompGood1)[1]  > percNBCorrect  [1] 0.7647059  >  > # Confusion Matrix  > # result output: 0 class, 19 identified correctly, 3 identified incorrectly  > # result output: 1 class, 9 identified incorrectly, 20 identified correctly  > nbResults <- table(test=nbOzoneCompGood1$test, pred=nbOzoneCompGood1$Pred)  > print(nbResults)  pred  test 0 1  0 19 9  1 3 20  >  > nbErrorRate <- round((nbResults[1,][[2]] + nbResults[2,][[1]]) / nrow(nbOzoneCompGood1) \*100,2)  > nbErrorRate  [1] 23.53  >  > nbAccuracyRate <- 100-nbErrorRate  > nbAccuracyRate  [1] 76.47  >  >  > ## Plot the results. Use a scatter plot. Have the x-axis represent temperature,  > # the y-axis represent wind, the shape representing what was predicted (good or bad day),  > # the color representing the actual value of 'goodozone' (i.e., if the actual ozone level was good)  > # and the size represent if the prediction was correct (larger symbols should be the observations the model got wrong)  > nbOzoneCompGood1$correct <- ifelse(nbOzoneCompGood1$test==nbOzoneCompGood1$Pred,'correct','wrong')  > plotNBOzoneCompGood1 <- data.frame(nbOzoneCompGood1$correct,test$Temp,test$Wind,test$goodOzone,nbOzoneCompGood1$Pred)  > colnames(plotNBOzoneCompGood1) <- c("correct","Temp","Wind","goodOzone","Predict")  > plot.nb.good <- ggplot(plotNBOzoneCompGood1, aes(x=Temp,y=Wind)) +  + geom\_point(aes(size=correct, color=goodOzone, shape=Predict)) +  + ggtitle("Niave Baise - Good/Bad Ozone Prediction")  > ggsave("NB\_Good\_Bad\_Ozone\_Prediction.jpg", width = 6, height = 6)  Warning message:  Using size for a discrete variable is not advised.  > plot.nb.good  >  > ##-- 5.5: Show all three results (charts) in one window, using the grid.array function (have two charts in one row)  > ga5 <- grid.arrange(plot.ksvm.good, plot.svm.good, plot.nb.good, ncol=3, nrow=2, top="Step 5 Model Comparisions")  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.  > ggsave(file="Grid\_Arrange\_Good\_KSVM-SVM-NB.jpg", ga5, width = 24, height = 12)  >  > #---- Step 6: Which are the best Models for this data? -------------------------  > ## Review what you have done and state which is the best and why  > #  > # Answer: It's observed that the SVM and Naive Baise models have the heighest accuracy ratings, measured at 76.5%,  > # for predicting the goodOzone Class. The result output calculation of Accuracy Rating for each model is shown below.  > #  > ## Output Results ##  > # Step 3 results:  > # LM: RMSE = 18.8  > # KSVM: RMSE = 21.59642  > # SVM: RMSE = 16.54  > #  > # Step 5 results:  > # KSVM:  > # Accuracy Rate: 70.59%  > # Error Rate: 29.41%  > # SVM:  > # Accuracy Rate: 76.47%  > # Error Rate: 23.53%  > # Naive Baise:  > # Accuracy Rate: 76.47%  > # Error Rate: 23.53%  > #-------------------------------------------------------------------------------- | |  | | |  | | --- | | > | | | |  | |  | |
|  |
|  |