HW2

2022-09-07

## Question 3.1

Using the same data set (credit\_card\_data.txt or credit\_card\_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier: #(a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional); and

credit\_data=read.delim("C:\\Users\\aruna\\OneDrive\\Desktop\\HW\\credit\_card\_data-headers.txt")  
set.seed(100)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

credit\_data$R1=as.factor(credit\_data$R1)  
#lets do a 5 fold cross validation  
trControl <- trainControl(method = "cv",number = 5, preProcOptions = c("center", "scale"))  
  
knn\_fit <- train(R1 ~ .,  
 method = "knn",  
 tuneGrid = expand.grid(k = 1:20),  
 trControl = trControl,  
 metric = "Accuracy",  
 data = credit\_data)

Findings: In the above we use traincontrol method to determine the number of K-folds and other preprocess option like scaling the data. Train is used to train the model while selecting different K values to choose from.

#(b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

credit\_data=read.delim("C:\\Users\\aruna\\OneDrive\\Desktop\\HW\\credit\_card\_data-headers.txt")  
  
#To replicate the solution  
set.seed(1)  
  
#Generate a random sample of 70% of the rows  
random\_row<- sample(1:nrow(credit\_data),as.integer(0.7\*nrow(credit\_data)))  
  
#Assign the trainData set to 70% of the original data set.   
trainData = credit\_data[random\_row,]  
  
#Assign the remaining 30% of data to a "Remaining Data" data set.  
remainingData = credit\_data[-random\_row,]  
  
#Generate a random sample of 50% of the remaining rows in "Remaining Data"  
random\_row2<-sample(1:nrow(remainingData),as.integer(0.5\*nrow(remainingData)))  
  
#Assign half the remaining data to validation data  
validateData=remainingData[random\_row2,]  
  
#Assign the remaining data to testData  
testData=remainingData[-random\_row2,]  
  
library(kknn)

## Warning: package 'kknn' was built under R version 4.1.3

##   
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':  
##   
## contr.dummy

knn\_acc2 <-vector("numeric")  
for (i in 1:20) {  
knn.fit <- kknn(R1~., trainData, testData, k = i,kernel = "optimal", scale = TRUE)  
summary(knn.fit)  
pred.knn = round(predict(knn.fit))  
knn\_acc <- sum(pred.knn == testData[,11]) / nrow(testData)  
knn\_acc2 <- c(knn\_acc2,knn\_acc)  
}

##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"  
##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = testData, k = i, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"

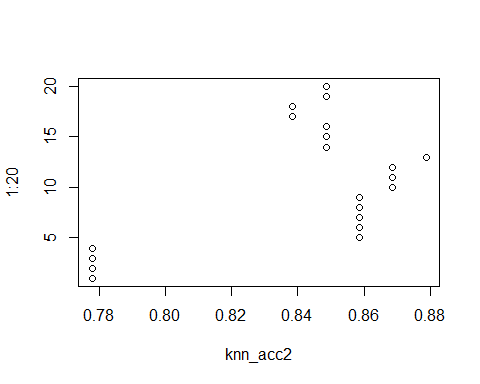
knn\_acc

## [1] 0.8484848

knn\_acc2

## [1] 0.7777778 0.7777778 0.7777778 0.7777778 0.8585859 0.8585859 0.8585859  
## [8] 0.8585859 0.8585859 0.8686869 0.8686869 0.8686869 0.8787879 0.8484848  
## [15] 0.8484848 0.8484848 0.8383838 0.8383838 0.8484848 0.8484848

plot(knn\_acc2,1:20)

 From the we see that the best optimum k value is 13 and the accuracy is 87.8% and then it starts to decrease

# Checking with validation data using the best k value = 13  
  
knn.fit2 <- kknn(R1~., trainData, validateData, k = 13,kernel = "optimal", scale = TRUE)  
summary(knn.fit2)

##   
## Call:  
## kknn(formula = R1 ~ ., train = trainData, test = validateData, k = 13, kernel = "optimal", scale = TRUE)  
##   
## Response: "continuous"

pred.knn = round(predict(knn.fit2))  
knn\_acc <- sum(pred.knn == validateData[,11]) / nrow(validateData)  
knn\_acc

## [1] 0.8571429

Conclusion: We use the validation data with the best Knn model and predict the values. The accuracy for the validation data is 85.7% which is a slight decrease from the test data set which is expected.

## Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

Answer: A clustering model that we tried at our hospital is identifying different clusters of patients who will need a Knee resurgery based on some features: 1.Age 2.Race Different diagnosis in addition to muscluloskeletal ailments(these are to be separated as this would be the main driving diagnosis for a Knee surgery) like 3.Diabetes 4.Hypertension 5.Osteoporosis 6.Hypothyroidism

## Question 4.2

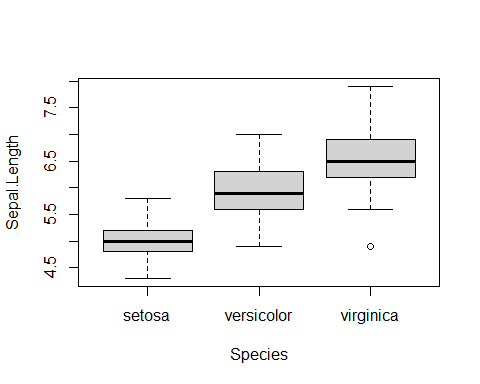
The iris data set iris.txt contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/Iris> ). The response values are only given to see how well a specific method performed and should not be used to build the model.

Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

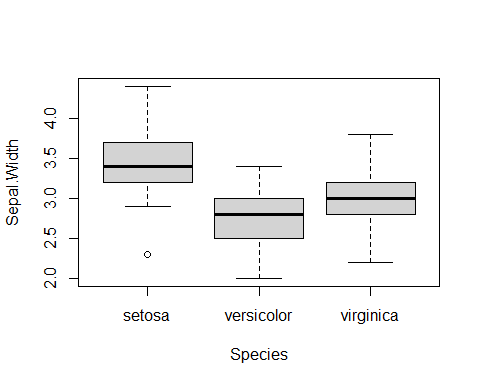
set.seed(111)  
#Loading the data  
iris = iris  
summary

## function (object, ...)   
## UseMethod("summary")  
## <bytecode: 0x000000001db5c040>  
## <environment: namespace:base>

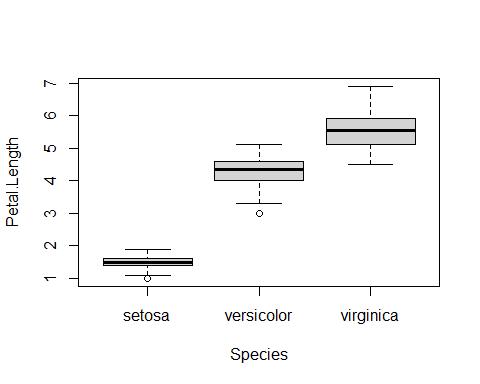
library(ggplot2)  
  
#box plot to see if there are any outliers  
boxplot(Sepal.Length~Species,data=iris)



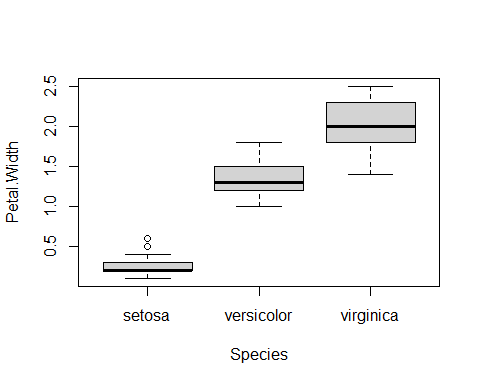
boxplot(Sepal.Width~Species,data=iris)



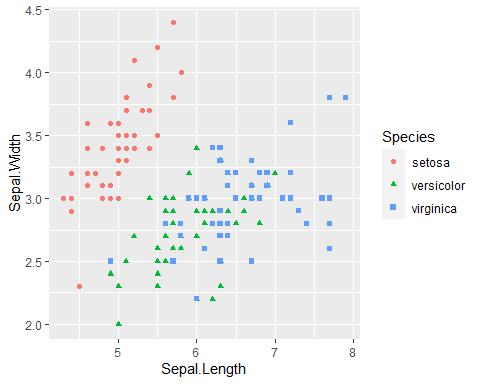
boxplot(Petal.Length~Species,data=iris)



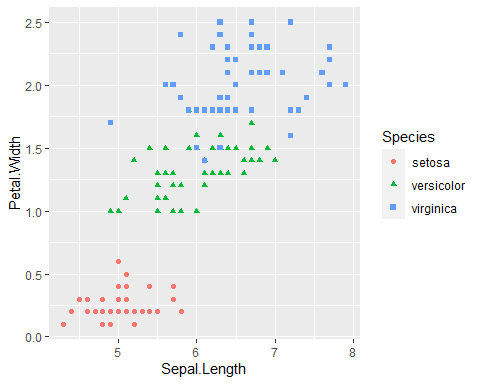
boxplot(Petal.Width~Species,data=iris)



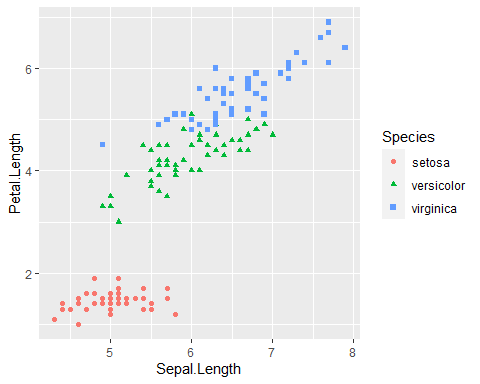
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, shape=Species, color=Species)) +geom\_point()



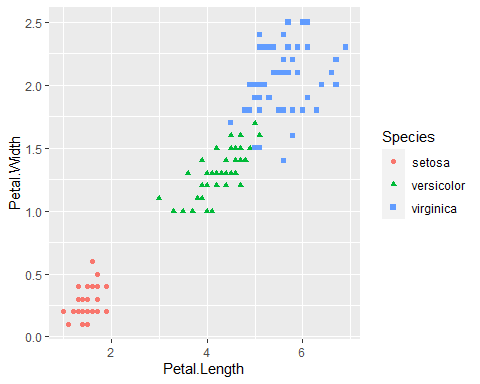
ggplot(iris, aes(x=Sepal.Length, y=Petal.Width, shape=Species, color=Species)) +geom\_point()



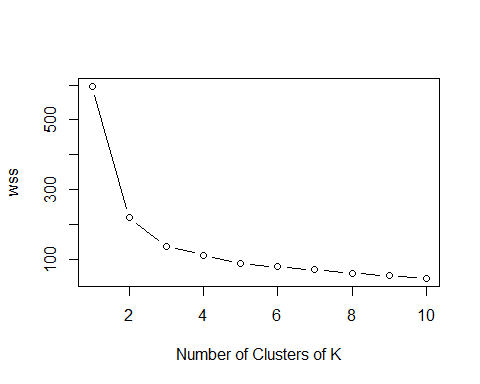
ggplot(iris, aes(x=Sepal.Length, y=Petal.Length, shape=Species, color=Species)) +geom\_point()



ggplot(iris, aes(x=Petal.Length, y=Petal.Width, shape=Species, color=Species)) +geom\_point()



#Findings:  
#From the scatter plot between different predictors we can see that the combination of (Sepal.Length, Petal.Length) and (Petal.Length,Petal.Width) are best at distinguishing different species with slight overlap of versicolor and virginica species where as with first 2 two combinations all the data is scattered without clear distinction of species.  
  
  
#Standardize the data as avoid sensitivity towards the difference in distance  
iris\_std=data.frame(scale(iris[,c(1:4)], center = TRUE, scale = TRUE))  
#What K to use: For this we try to use Elbow method where we plot the Total within clusters sum of squares and K  
k=10  
wss = sapply(1:k,function(K){kmean = kmeans(iris\_std,K,nstart=20)$tot.withinss})  
plot(1:k,wss,type="b",xlab="Number of Clusters of K",ylab="wss")



According to the elbow method where we plot Within clusters sum of squares and K values. As the number clusters increase the WSS decrease and form an elbow and start to follow a linear trend. #In this case the optimal K value would be 3

set.seed(9)  
kmean1=kmeans(iris\_std[,c(1,2)],3,nstart=20)  
fit1=kmean1$betweenss/kmean1$totss  
fit1

## [1] 0.6579488

table(kmean1$cluster,iris$Species)

##   
## setosa versicolor virginica  
## 1 49 0 0  
## 2 1 36 19  
## 3 0 14 31

kmean2=kmeans(iris\_std[,c(1,3,4)],3,nstart=20)  
fit2=kmean2$betweenss/kmean2$totss  
  
fit2

## [1] 0.8599084

table(kmean2$cluster,iris$Species)

##   
## setosa versicolor virginica  
## 1 50 1 0  
## 2 0 5 36  
## 3 0 44 14

best\_model = kmeans(iris\_std,3,nstart=10)  
fit\_best=best\_model$betweenss/best\_model$totss  
  
fit\_best

## [1] 0.7669658

table(best\_model$cluster,iris$Species)

##   
## setosa versicolor virginica  
## 1 50 0 0  
## 2 0 39 14  
## 3 0 11 36

#NOn standardized model with best parameters  
  
kmean3 = kmeans(iris[,c(1:4)],3,nstart=20)  
  
fit3=kmean3$betweenss/kmean3$totss  
fit3

## [1] 0.8842753

table(kmean3$cluster,iris$Species)

##   
## setosa versicolor virginica  
## 1 0 2 36  
## 2 0 48 14  
## 3 50 0 0

Findings and conclusion:

We can see that the best model is with k=3 and including all the predictors

Though other combination of predictors give better fit of the model.They are unable to classify versicolor,virginica correctly.

#References: 1.https://cran.r-project.org/web/packages/kknn/kknn.pdf 2.https://www.statmethods.net/graphs/boxplot.html