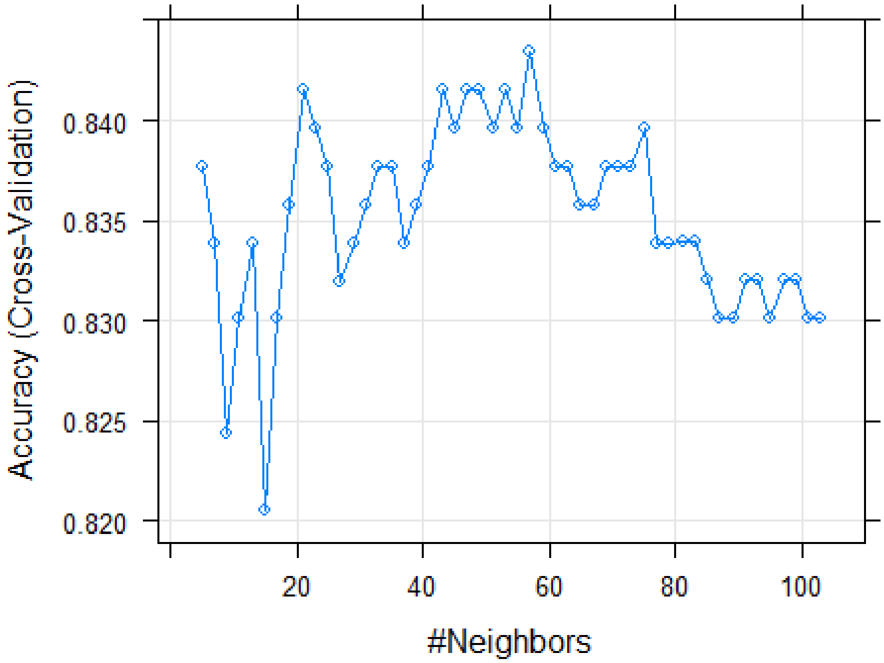
# Hw2.R

2021-09-08

Question 3.1.a, 3.1.b and 4.3 lies in the R script below:

```
library(tidyverse)

library(caret)

library(kernlab)

setwd("C:/Users/Muhammad/ISYE/hw3")
data<-read.table("credit_card_data-headers.txt", header = TRUE)
data<-as.data.frame(data)

#Q3.1.a
set.seed(100)
#splitting data for training/validation and testing. 80% training data set
indxTrain <- createDataPartition(y = data$R1,p = 0.8,list = FALSE)
training <- data[indxTrain,]
testing <- data[-indxTrain,]

#Pre-processing data i.e scaling for Knn
trainX <- training[,names(training) != "R1"]
Proccessed_Values <- preProcess(x = trainX,method = c("center", "scale"))

#I am choosing k-fold value of 5 beacause our dataset is too small

control <- trainControl(method="cv",number=5)
#training data set
knnFit <- train(as.factor(R1) ~ ., data = training, method = "knn", trControl
= control, preProcess = c("center","scale"), tuneLength = 50)

#finding the value of kappa where accuracy maximizes;
index_where_max<-which.max(knnFit$results$Accuracy)
knnFit$results$Accuracy[index_where_max]

## [1] 0.8434249

k=knnFit$results$k[index_where_max]
k

## [1] 57

plot(knnFit)
```




```
cat("For final model Accuracy =", knnFit$results$Accuracy[index_where_max],
"and k =",k)

## For final model Accuracy = 0.8434249 and k = 57


#Testing our model on test dataset
knnPredict <- predict(knnFit, newdata = testing)
con<-confusionMatrix(knnPredict, as.factor(testing$R1))
con

## Confusion Matrix and Statistics
##
##             Reference
## Prediction  0  1
##          0 63 16
##          1  6 45
##
##                Accuracy : 0.8308
##                  95% CI : (0.7551, 0.8908)
##     No Information Rate : 0.5308
##     P-Value [Acc > NIR] : 6.438e-13
##
##                   Kappa : 0.657
##
##  Mcnemar's Test P-Value : 0.05501
##
```

```
##             Sensitivity : 0.9130
##             Specificity : 0.7377
##          Pos Pred Value : 0.7975
##          Neg Pred Value : 0.8824
##              Prevalence : 0.5308
##          Detection Rate : 0.4846
##    Detection Prevalence : 0.6077
##       Balanced Accuracy : 0.8254
##
##        'Positive' Class : 0
##
```

```
overall_accuracy<-con$overall["Accuracy"]
cat("Accuracy on test data set =", overall_accuracy*100, "%")
```

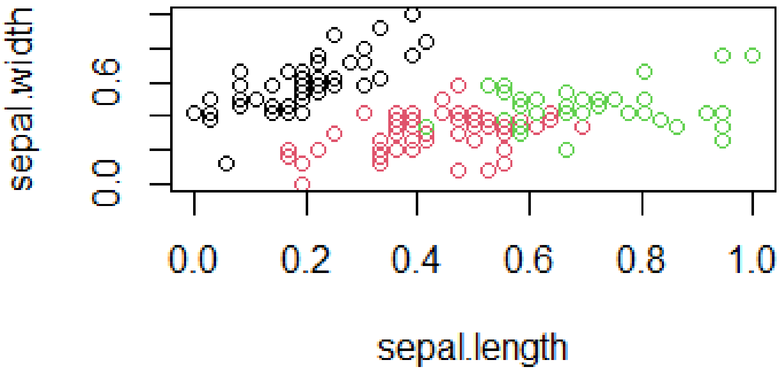
```
## Accuracy on test data set = 83.07692 %
```

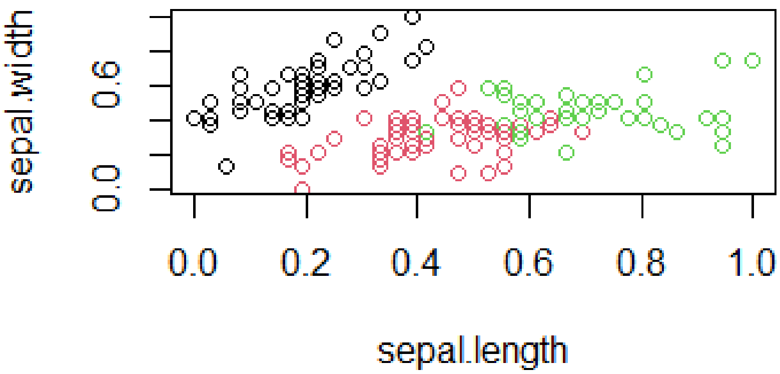
```
#Q3.1.b

#Sampling and splitting data for training and testing

index_sample<- sample(1:nrow(data), as.integer(0.75*nrow(data)))
train_data <- data[index_sample,]

#Assigning validation and test data
valid_test_data <- data[-index_sample,]
index_sample2<- sample(1:nrow(valid_test_data),
as.integer(0.50*nrow(valid_test_data)))
validation_data<-valid_test_data[index_sample2,]
testing_data<- valid_test_data[-index_sample2,]

#c_value <- seq(1, 100, by=5)
c_value= seq(.1,1,by=0.1)
accuracy1= rep(1,10)
#looping for accuracy for svm model for different c value with increment of
0.1, tried 1 to 100 by 5 increment already
for (i in 1:10) {

  #training model on training data
  svm_model = ksvm(as.matrix(train_data[,1:10]),as.factor(train_data[,11]),
type = "C-svc",kernel = "vanilladot",C = c_value[i],scaled=TRUE)
  #testing it on validation data
  prediction1 = predict(svm_model,validation_data[,1:10])
  accuracy1[i] = (sum(prediction1 == validation_data$R1) /
nrow(validation_data))*100
}
```

```
#which model maximizes
max_ind<-which.max(accuracy1)
max(accuracy1)

## [1] 86.58537

c_max<-c_value[max_ind]
cat("C value which maximizes accuracy =",c_max,"\n")

## C value which maximizes accuracy = 0.1

#retraining model on training dataset with parameter c=0.1,
#(in this case it really doesn't matter because we got same accuracy with the
c values in last sequence I tried.
#however just to show that I know generally we have to rerun our model on
training set with the parameter we found for
#maximizing accuracy in validation data set)

svm_model_test =
ksvm(as.matrix(train_data[,1:10]),as.factor(train_data[,11]), type = "C-
svc",kernel = "vanilladot",C =0.1,scaled=TRUE)

##  Setting default kernel parameters

Accuracy_on_test =
sum(predict(svm_model_test,testing_data[,1:10])==testing_data[,11])/nrow(test
ing_data)
cat("Accuracy on test data set =", Accuracy_on_test)

## Accuracy on test data set = 0.8292683

#Accuracy= 82.9%

#Q4.2
library(datasets)
data(iris)
names(iris) <- tolower(names(iris))
iris_category <- unique(iris[,5])

#Scaling all data set
preproc2 <- preProcess(iris[,1:4], method=c("range"))

normalized_data <- predict(preproc2, iris[,1:4])

model_kmean<-kmeans(normalized_data[,1:4], 3, nstart = 20)
model_kmean

## K-means clustering with 3 clusters of sizes 39, 50, 61
##
## Cluster means:
##   sepal.length sepal.width petal.length petal.width
## 1    0.7072650   0.4508547   0.79704476  0.82478632
```

```
## 2    0.1961111   0.5950000   0.07830508  0.06083333
## 3    0.4412568   0.3073770   0.57571548  0.54918033
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3
##  [75] 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1
1 1 1
## [112] 1 1 3 1 1 1 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 3 1 1 1 3 1 1 1 3 1 1
1 3 1
## [149] 1 3
##
## Within cluster sum of squares by cluster:
## [1] 2.073324 1.829062 3.079830
##  (between_SS / total_SS =  83.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
#There is not really a way to tell accuracy  since clustering is unsupervised
learning
#We could look at the compactness which is 76.7% or how similar to members
within a group are
#We can look at purity but I don't think that is expected here
#Moreover anything over 3 dimensions would be really hard to determine
manually
#We could loop over multiple values of K but since we already know that there
are 3 classes it's not necessary,
#however we could gain some extra information but we only have 150
observation so that isn't a good idea since
#it will over-fit and accuracy will probably go down

# We could visualize for different combination of attributes and their graphs
and see how two attributes
# are placed in a category depending on their values, but as we can see after
just 3 dimensions we have
#difficulty visualizing, hence usually we could just look at individual
cluster means for group and try
#understand intuitively or we could look at visualize 2 or 3 features of
interest at a time to study further.

#i.e:
par(mfrow=c(2,1))
plot(normalized_data[c(1,2)], col=model_kmean$cluster)
plot(iris[c(1,2)], col=iris_category)
```




The upper graph is classification with our prediction, the bottom one is actual classification

Question 4.1

Answer: We could use clustering to Identify fraudulent delivery drivers for delivery apps such uber-eats, door dash, skip-the-dishes, etc. by grouping on features like similar behavior on GPS logs, age, number of deliveries, percentage of wrong orders delivered, type of wrong orders reported etc.