```
> ####Assignment 2 Part 2 Falkenberg

> ###File Download

> #IRIS

> iris <- read.csv("C:/Users/chaos/Downloads/iris.csv")

> iris <- iris[,-1]

> # abalone dataset from UCI repository

> # reading the dataset from UCI repository URL

> abalone <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-
databases/abalone/abalone.data"), header = FALSE, sep = ",")

> # rename columns

> colnames(abalone) <- c("sex", "length", 'diameter', 'height', 'whole_weight',
'shucked_wieght', 'viscera_wieght', 'shell_weight',

+                    'rings' )

> # add new column abalone$age.group with 3 values based on the number of rings

> abalone$age.group <- cut(abalone$rings, br=c(0,8,11,35), labels = c("young", 'adult',
'old'))

> # drop the sex column (categorical variable)

> abalone.norm <- abalone[,-1]

> # optionally normalize

> #normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x))) }

> #abalone.norm[1:7] <- as.data.frame(lapply(abalone.norm[1:7], normalize))

> ##Exercise 1: Naïve Bayes

> ## Call the NaiveBayes Classifier Package e1071, which auto calls the Class package ##

> #IRIS dataset.

> library("e1071")

> #Train classifier

> classifier<-naiveBayes(iris[,1:4], iris[,5])
> #evaluate classification

> table(predict(classifier, iris[,-5]), iris[,5], dnn=list('predicted','actual'))

          actual

predicted    setosa versicolor virginica

  setosa       50         0        0

  versicolor     0        47        3

  virginica      0         3       47

> #examine class means and standard deviations for petal length

> classifier$tables$Petal.Length

          Petal.Length

iris[, 5]    [,1]      [,2]

  setosa     1.462 0.1736640

  versicolor 4.260 0.4699110

  virginica  5.552 0.5518947

> #plot normal distributions at the means of the classes

> #one class

> plot(function(x) dnorm(x, 1.462, 0.1736640), 0, 8, col="red", main="Petal length
distribution for the 3 different species")

> #another class

> curve(dnorm(x, 4.260, 0.4699110), add=TRUE, col="blue")

> #final class

> curve(dnorm(x, 5.552, 0.5518947 ), add=TRUE, col = "green")

> #Abalone dataset 1: height.

> library("e1071")

> classifier<-naiveBayes(abalone.norm[,1:7], abalone.norm[,9])

> table(predict(classifier, abalone.norm[,1:7]), abalone.norm[,9],
dnn=list('predicted','actual'))
```

```
         actual
predicted young adult  old
    young  1122   423  145
    adult   266   910  437
    old      19   477  378
> classifier$tables$height
              height
abalone.norm[, 9]      [,1]       [,2]
            young 0.1065956 0.04183039
            adult 0.1516906 0.02984784
            old   0.1648125 0.02935998
> #Take information from line 40; first should be mean then standard deviation
> plot(function(x) dnorm(x, 0.1065956 , 0.04183039), 0, 0.4, ylim = c(0,14), col="red",
main="Height for the 3 different ages")
> curve(dnorm(x, 0.1516906, 0.02984784), add=TRUE, col="blue")
> curve(dnorm(x, 0.1648125, 0.02935998), add=TRUE, col = "green")
> #Abalone dataset subset 2: length.
> library("e1071")
> classifier<-naiveBayes(abalone.norm[,1:7], abalone.norm[,9])
> table(predict(classifier, abalone.norm[,1:7]), abalone.norm[,9],
dnn=list('predicted','actual'))
         actual
predicted young adult  old
    young  1122   423  145
    adult   266   910  437
    old      19   477  378
> classifier$tables$length
```

```
          length
abalone.norm[, 9]   [,1]     [,2]
      young 0.4209915 0.11137474
      adult 0.5707182 0.08740980
      old   0.5868542 0.08100644
> plot(function(x) dnorm(x, 0.4209915 , 0.11137474), 0, 1, ylim = c(0,5), col="red",
main="Length for the 3 different ages")
> curve(dnorm(x, 0.5707182 , 0.08740980), add=TRUE, col="blue")
> curve(dnorm(x, 0.5868542 , 0.08100644), add=TRUE, col = "green")
> #Abalone dataset subset 3:whole_weight.
> library("e1071")
> classifier<-naiveBayes(abalone.norm[,1:7], abalone.norm[,9])
> table(predict(classifier, abalone.norm[,1:7]), abalone.norm[,9],
dnn=list('predicted','actual'))
     actual
predicted young adult  old
  young  1122   423  145
  adult   266   910  437
  old      19   477  378
> classifier$tables$whole_weight
          whole_weight
abalone.norm[, 9]   [,1]     [,2]
      young 0.4323742 0.3060074
      adult 0.9850878 0.4264315
      old   1.1148922 0.4563715
> plot(function(x) dnorm(x, 0.4323742  , 0.3060074), 0,2.5, ylim = c(0,1.5), col="red",
main="Whole_Weight for the 3 different ages")
```

```
> curve(dnorm(x, 0.9850878  , 0.4264315), add=TRUE, col="blue")

> curve(dnorm(x, 1.1148922  , 0.4563715), add=TRUE, col = "green")

> ##Exercise 2: K-Nearest Neighbors (kNN)

> #Abalone Dataset

> # sample 2924 from 4177 (~70%)

> s_abalone <- sample(4177,2924)

> #Abalone.norm.train <-abalone.norm[s_abalone,]

> #abalone.norm.test <-abalone.norm[-s_abalone,]

> ## create train & test sets based on sampled indexes

> abalone.train <-abalone[s_abalone,]

> abalone.test <-abalone[-s_abalone,]

> sqrt(2924)

[1] 54.07402

> k = 55

> # k = 80

> # train model & predict

> library(class)

> KNNpred <- knn(train = abalone.train[2:7], test = abalone.test[2:7], cl =
abalone.train$age.group, k = k)

> # create contingency table/ confusion matrix

> contingency.table <- table(KNNpred,abalone.test$age.group)

> contingency.matrix = as.matrix(contingency.table)

> sum(diag(contingency.matrix))/length(abalone.test$age.group)

[1] 0.6823623

> accuracy <- c()

> ks <- c(35,45,55,65,75,85,95,105)
> for (k in ks) {
+   KNNpred <- knn(train = abalone.train[2:7], test = abalone.test[2:7], cl =
abalone.train$age.group, k = k)
+   cm = as.matrix(table(Actual=KNNpred, Predicted = abalone.test$age.group,
dnn=list('predicted','actual')))
+   accuracy <- c(accuracy,sum(diag(cm))/length(abalone.test$age.group))
+ }
> plot(ks,accuracy,type = "b", ylim = c(0.66,0.69))
> plot(ks,accuracy,type = "b", ylim = c(0.66,0.75))
> plot(ks,accuracy,type = "b", ylim = c(0.66,0.70))
> #Iris Subset 1.
> s_iris <- sample(150, 105)  # (~70% training data)
> iris.train <- iris[s_iris, ]
> iris.test <- iris[-s_iris, ]
> # Verify the sample size
> sqrt(105)
[1] 10.24695
> k = 10
> # Load required library
> library(class)
> # Implement KNN classification
> KNNpred <- knn(train = iris.train[, 1:4], test = iris.test[, 1:4], cl = iris.train$Species, k = k)
> # Create a contingency table (confusion matrix)
> contingency.table <- table(KNNpred, iris.test$Species)
> # Accuracy calculation
> contingency.matrix <- as.matrix(contingency.table)
> accuracy_single <- sum(diag(contingency.matrix)) / length(iris.test$Species)
> # Initialize accuracy vector and try different k-values

> accuracy <- c()

> ks <- c(9, 11, 13, 15, 17, 19)

> for(k in ks) {

+   KNNpred <- knn(train = iris.train[, 1:4], test = iris.test[, 1:4], cl = iris.train$Species, k = k)

+   cm <- as.matrix(table(Actual = iris.test$Species, Predicted = KNNpred))

+   accuracy <- c(accuracy, sum(diag(cm)) / length(iris.test$Species))

+ }

> plot(ks, accuracy, type = "b", ylim = c(0.93, 0.96), main = "Accuracy vs. k-values", xlab =
"k", ylab = "Accuracy")

> plot(ks, accuracy, type = "b", ylim = c(0.80, 0.96), main = "Accuracy vs. k-values", xlab =
"k", ylab = "Accuracy")

> plot(ks, accuracy, type = "b", ylim = c(0.84, 0.96), main = "Accuracy vs. k-values", xlab =
"k", ylab = "Accuracy")

> plot(ks, accuracy, type = "b", ylim = c(0.86, 0.96), main = "Accuracy vs. k-values", xlab =
"k", ylab = "Accuracy")

> #Iris Subset 2.

> s_iris <- sample(150, 105)  # (~70% training data)

> iris.train <- iris[s_iris, ]

> iris.test <- iris[-s_iris, ]

> # Verify the sample size

> sqrt(105)

[1] 10.24695

> k = 10

> # Load required library

> library(class)

> # Implement KNN classification
> KNNpred <- knn(train = iris.train[, 1:4], test = iris.test[, 1:4], cl = iris.train$Species, k = k)

> # Create a contingency table (confusion matrix)

> contingency.table <- table(KNNpred, iris.test$Species)

> # Accuracy calculation

> contingency.matrix <- as.matrix(contingency.table)

> accuracy_single <- sum(diag(contingency.matrix)) / length(iris.test$Species)

> print(paste("Accuracy for k =", k, ":", accuracy_single))

[1] "Accuracy for k = 10 : 0.955555555555556"

> # Initialize accuracy vector and try different k-values

> accuracy <- c()

> ks <- c(7, 21, 33, 45, 57, 69, 71, 83, 95, 105)

> for(k in ks) {

+   KNNpred <- knn(train = iris.train[, 1:4], test = iris.test[, 1:4], cl = iris.train$Species, k = k)

+   cm <- as.matrix(table(Actual = iris.test$Species, Predicted = KNNpred))

+   accuracy <- c(accuracy, sum(diag(cm)) / length(iris.test$Species))

+ }

> plot(ks, accuracy, type = "b", ylim = c(0.2, 1), main = "Accuracy vs. k-values", xlab = "k",
ylab = "Accuracy")

> ##Exercise 3: K-Means

> #Iris Dataset

> # Plot iris petal length vs. petal width, color by species

> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +

+   geom_point()

Error in ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) :

  could not find function "ggplot"

> library(ggplot2)
> ##Exercise 3: K-Means

> #Iris Dataset

> # Plot iris petal length vs. petal width, color by species

> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +

+   geom_point()

> # set seed for random number generator

> set.seed(123)

> # run k-means

> iris.km <- kmeans(iris[,-5], centers = 3)

> assigned.clusters <- as.factor(iris.km$cluster)

> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = assigned.clusters)) +

+   geom_point()

> wss <- c()

> ks <- c(2,3,4,5)

> for (k in ks) {

+   iris.km <- kmeans(iris[,-5], centers = k)

+   wss <- c(wss,iris.km$tot.withinss)

+ }

> plot(ks,wss,type = "b")

> labeled.clusters <- as.character(assigned.clusters)

> labeled.clusters[labeled.clusters==1] <- "setosa"

> labeled.clusters[labeled.clusters==2] <- "versivolor"

> labeled.clusters[labeled.clusters==3] <- "virginica"

> table(labeled.clusters, iris[,5])


labeled.clusters setosa versicolor virginica
```

```
    setosa       50        0       0
    versivolor     0       48       14
    virginica      0        2       36
> #Iris with different values of k
> # Plot iris petal length vs. petal width, color by species
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
+   geom_point()
> # set seed for random number generator
> set.seed(234)
> # run k-means
> iris.km <- kmeans(iris[,-5], centers = 3)
> assigned.clusters <- as.factor(iris.km$cluster)
> ggplot(iris, aes(x = Petal.Length, y = Petal.Width, colour = assigned.clusters)) +
+   geom_point()
> wss <- c()
> ks <- c(1,3,5,7,9)
> for (k in ks) {
+   iris.km <- kmeans(iris[,-5], centers = k)
+   wss <- c(wss,iris.km$tot.withinss)
+ }
> plot(ks,wss,type = "b")
> labeled.clusters <- as.character(assigned.clusters)
> labeled.clusters[labeled.clusters==1] <- "setosa"
> labeled.clusters[labeled.clusters==2] <- "versivolor"
> labeled.clusters[labeled.clusters==3] <- "virginica"
> table(labeled.clusters, iris[,5])
```

```
labeled.clusters setosa versicolor virginica
    setosa       17       4       0
    versivolor    33       0       0
    virginica      0      46      50
> #Abalone Dataset
> # Plot abalone height vs. whole_weight, colored by age.group
> ggplot(abalone.norm, aes(x = height, y = whole_weight, colour = age.group)) +
+   geom_point() +
+   ggtitle("Height vs. Whole Weight by Age Group")
> # Set seed
> set.seed(345)
> # Run K-means clustering
> abalone.km <- kmeans(abalone.norm[,-c(9)], centers = 3)
> # Assign clusters
> assigned.clusters <- as.factor(abalone.km$cluster)
> # Plot height vs whole_weight colored by the clusters assigned by K-means
> ggplot(abalone.norm, aes(x = height, y = whole_weight, colour = assigned.clusters)) +
+   geom_point() +
+   ggtitle("Height vs. Whole Weight by K-means Clusters")
> # Try different values of k to calculate total within-cluster sum of squares (wss)
> wss <- c()  # Within Sum of Squares
> ks <- c(2, 3, 4, 5)
> for (k in ks) {
+   abalone.km <- kmeans(abalone.norm[,-c(9)], centers = k)
+   wss <- c(wss, abalone.km$tot.withinss)  # Add wss for each k
+ }

> # Plot the total within-cluster sum of squares (Elbow plot)

> plot(ks, wss, type = "b", xlab = "Number of clusters (k)", ylab = "Total within-cluster sum of
squares",

+      main = "Elbow Method for Optimal k")

> # Map cluster numbers to age groups (young, adult, old)

> labeled.clusters <- as.character(assigned.clusters)

> labeled.clusters[labeled.clusters == "1"] <- "young"

> labeled.clusters[labeled.clusters == "2"] <- "adult"

> labeled.clusters[labeled.clusters == "3"] <- "old"

> table(labeled.clusters, abalone.norm[,9])


labeled.clusters young adult  old

         adult  1407    0    0

         old      0    0  693

         young    0  1810  267

> #Abalone with different values of k

> # Plot abalone diameter vs. length, colored by age.group

> ggplot(abalone.norm, aes(x = diameter, y = length, colour = age.group)) +

+   geom_point() +

+   ggtitle("Diameter vs. Length by Age Group")

> # Set seed

> set.seed(345)

> # Run K-means clustering

> abalone.km <- kmeans(abalone.norm[,-c(9)], centers = 3)

> # Assign clusters
> assigned.clusters <- as.factor(abalone.km$cluster)

> # Plot length vs diameter colored by the clusters assigned by K-means

> ggplot(abalone.norm, aes(x = diameter, y = length, colour = assigned.clusters)) +

+   geom_point() +

+   ggtitle("Diameter vs. Length by K-means Clusters")

> # Try different values of k to calculate total within-cluster sum of squares (wss)

> wss <- c()  # Within Sum of Squares

> ks <- c(1, 3, 5, 7, 9)

> for (k in ks) {

+   abalone.km <- kmeans(abalone.norm[,-c(9)], centers = k)

+   wss <- c(wss, abalone.km$tot.withinss)  # Add wss for each k

+ }

> # Plot the total within-cluster sum of squares (Elbow plot)

> plot(ks, wss, type = "b", xlab = "Number of clusters (k)", ylab = "Total within-cluster sum of
squares",

+     main = "Elbow Method for Optimal k")

> # Map cluster numbers to age groups (young, adult, old)

> labeled.clusters <- as.character(assigned.clusters)

> labeled.clusters[labeled.clusters == "1"] <- "young"

> labeled.clusters[labeled.clusters == "2"] <- "adult"

> labeled.clusters[labeled.clusters == "3"] <- "old"

> table(labeled.clusters, abalone.norm[,9])


labeled.clusters young adult  old

       adult  1407    0    0

       old      0    0  693
```

```
    young    0  1810  267
>
```