ISYE6501 – Fall 2021

Homework #2

3.1] a)

In order to select the appropriate sorting methodologies, I opted to first check the distribution of the datapoints response. An histogram was plotted for the same with a resulting breakdown of 54% (0's) to 45% (1's). Since the responses are fairly equally distributed, I opted for randomized data point selection for train+validation set (70%) and testing set (30%). If the response was unequally divided, I would have opted to distribute the datapoints by % values of each response in each set to reduce randomization effect.

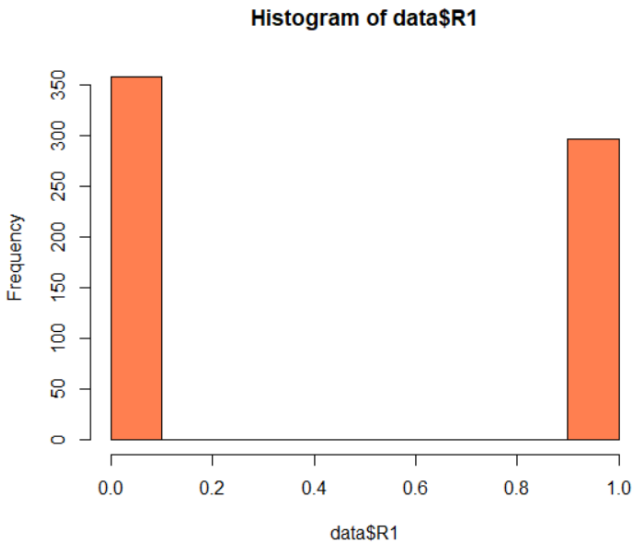
Performing cross validation technique using train.kknn function on the randomized training and validation data set (tv_data) the best value of k was determined as 12 (this is the number of neighboring datapoints considered which predicting the response of a datapoint)

Using the k value of 12, a new model was generated for the entire tv_data set along with validation using knn method with a prediction accuracy of 86.2%

Using the k value of 12, a final test_data set was trained and validated using knn method with a prediction accuracy of 84.7%

The lower accuracy on the test_data set was an expected outcome since the best k value to be used for knn method was modelled using tv_data set that yielded a higher accuracy rate due to added randomization effect.

Histogram Distribution:



Code attached for reference for 3.1.a:

```
rm(list=ls())
set.seed(100)

#Read data
data=read.csv("M:/OMSA/ISYE6501/HW1/credit_card_data.csv")

#Select Library
library(kknn)

#Checking if data is balanced to select appropriate sorting methodology
hist(data$R1,col="coral")
prop.table(table(data$R1))


#Select random set of data points to create training and validation dataset
random_rows=sample(1:nrow(data),as.integer(0.7*nrow(data)))
tv_data=data[random_rows,]
test_data=data[-random_rows,]

#train model
train.kknn(as.factor(R1)~.,tv_data,kmax=100,scale=TRUE)

#Checking model accuracy on train data using K=12

predicted_value_train=rep(0,nrow(tv_data))

for (i in 1:nrow(tv_data))
  {
  model_train=kknn(R1~.,tv_data[-i,],tv_data[i,],k=12,distance=2,kernel="optimal",scale=TRUE)
predicted_value_train[i]=round(fitted.values(model_train))
  }

#Accuracy of the model for k=12
train_accuracy=sum(predicted_value_train==tv_data[,11])/nrow(tv_data)


#Testing Data Accuracy on testing model
predicted_value_test=rep(0,nrow(test_data))

for (i in 1:nrow(test_data))
{
  model_test=kknn(R1~.,test_data[-i,],test_data[i,],k=12,distance=2,kernel="optimal",scale=TRUE)
  predicted_value_test[i]=round(fitted.values(model_test))
}

#Accuracy of the model for k=12
test_accuracy=sum(predicted_value_test==test_data[,11])/nrow(test_data)


train_accuracy
test_accuracy
```

3.1] b)

The credit data dataset was divided into 3 categories: Training, Validation and Test data sets

Using the knn method on the training data set (similar to question 3.1.a) a manual looping was done for k values between 1 to 20. The best data accuracy was observed for k value of 12 ~ 86.21% accurte.

The value of k=12 was then used to validate the model on a validation set with an accuracy of 85.71% and for a final verification, the test data was predicted with an accuracy of 82.82%

This replicated the earlier finding where the model based on training dataset yields a lower accuracy on its validation and testing dataset due to randomization effect. However, this can also be linked with the testing data set having a smaller data sample.

Code attached for reference for 3.1.b:

```
rm(list=ls())
set.seed(100)

#Read data
data=read.csv("M:/OMSA/ISYE6501/HW1/credit_card_data.csv")

#Select Library
library(kknn)

#Select random set of data points to create training,validation and testing dataset
random_rows=sample(1:nrow(data),as.integer(0.7*nrow(data)))
training_data=data[random_rows,]
rest_data=data[-random_rows,]
random_rows_2=sample(1:nrow(rest_data),as.integer(0.5*nrow(rest_data)))
validation_data=rest_data[random_rows_2,]
testing_data=rest_data[-random_rows_2,]

#finding best fit k using knn method

predicted_value_training=rep(0,nrow(training_data))

#Define a function to try multiple K Values
k_value_best_fit=function(X){

  #Define a loop to create training data and testing data for each datapoint
  for (i in 1:nrow(training_data))
  {
    training_model=kknn(R1~.,training_data[-i,],training_data[i,],k=X,distance=2,kernel="optimal",scale=TRUE)
    predicted_value_training[i]=round(fitted.values(training_model))
  }

  #Accuracy of then model for k value X
  sum(predicted_value_training==training_data[,11])/nrow(training_data)
}

#define range of k values to test
range=rep(0,20)
for(X in 1:20)
{
  range[X]=k_value_best_fit(X)
}

#Save Accuracy model as a matrix
knn_training_model_accuracy=as.matrix(range*100)
knn_training_model_accuracy  #prints accuracy by k value as rownumber

#Find Max Accuracy Model
max(knn_training_model_accuracy)
```

```r


#Validate using validation_data
predicted_value_validation=rep(0,nrow(validation_data))

#Run model
  for (i in 1:nrow(validation_data))
  {
    validation_model=kknn(R1~.,validation_data[-i,],validation_data[i,],k=12,distance=2,kernel="optimal",scale=TRUE)
    predicted_value_validation[i]=round(fitted.values(validation_model))
  }

  #Accuracy of then model in validation
  validation_model_accuracy=sum(predicted_value_validation==validation_data[,11])/nrow(validation_data)

validation_model_accuracy  #prints accuracy for validation model


#Validate using test_data
predicted_value_test=rep(0,nrow(testing_data))

#Run model
for (i in 1:nrow(testing_data))
{
  test_model=kknn(R1~.,testing_data[-i,],testing_data[i,],k=12,distance=2,kernel="optimal",scale=TRUE)
  predicted_value_test[i]=round(fitted.values(test_model))
}

#Accuracy of then model in validation
test_model_accuracy=sum(predicted_value_test==testing_data[,11])/nrow(testing_data)

test_model_accuracy  #prints accuracy for validation model
validation_model_accuracy
max(knn_training_model_accuracy)
```

4.1]

In the semiconductor industry the manufacturing process is done on a nanoscale level making the process extremely prone to defects. In order to have contamination free manufacturing, a clustering model can be used to identify signs of misprocessing that could lead to specific defects. The variable inputs used could be:

a) Metal Flaking
b) Pattern Misalignment
c) Increased surface particles
d) Incomplete etching

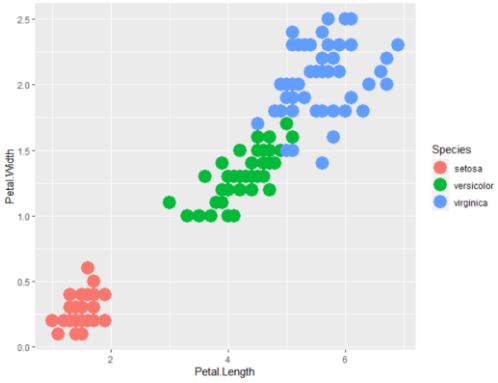
These specific variables could then be clustered using multiple models to identify if there will be a wafer yield impact, a chip performance degradation, dead chip or any other clusters that may arise as a result of the model and its analysis.

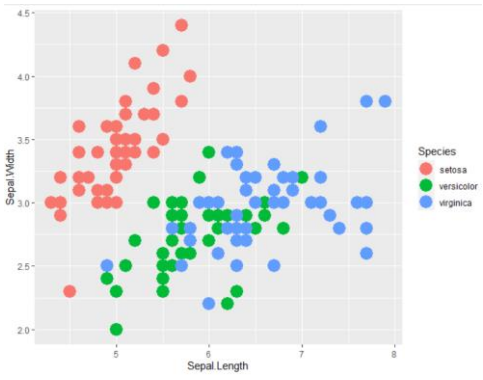
4.2]

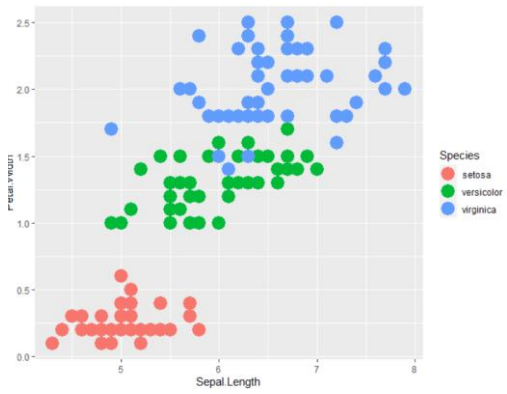
The iris data was plotted for identifying the potential best combinations of predictors. Based on the graphical distribution the Petal length and Petal Width were the potential best combination of predictors. Where there was a distinct distribution between the three species in the petal sizes.

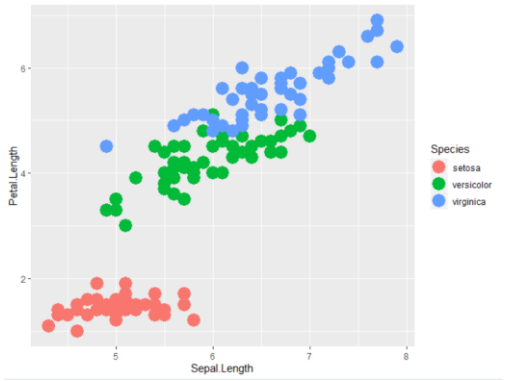
The iris data was first prepared for analysis by dropping the response column. The function *fviz_nbclust* from the *factoextra* library was used to identify the best number of clusters (k value). Based on the function output, k=3 was determined to be the most optimal value for analysis using kmeans method. As visible in the elbow plot, that adding another cluster above 3 does not provide a smaller total within sum of square. Which essentially means that the compactness of the clusters does not improve enough to justify the need of an addition of another cluster above 3.

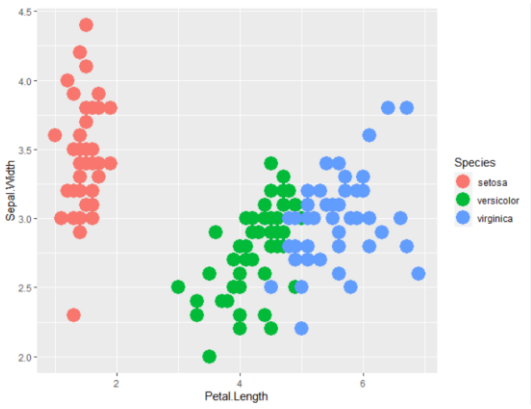
Using the k value of 3 the prediction model accuracy based on *kmeans* function with a *nstart* value of 10 was 89.33%. This exercise was also repeated using *nstart* values between 1 through 50 with no change in prediction accuracy.

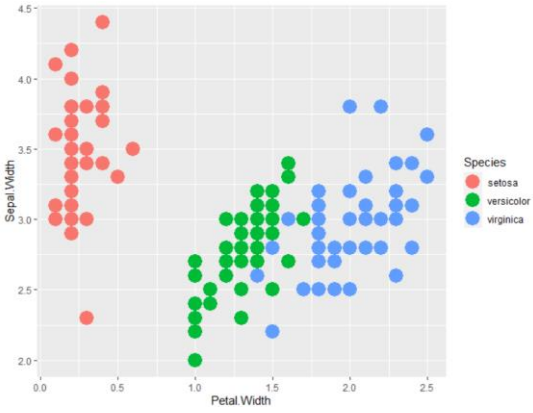













```r
rm(list=ls())
set.seed(100)

library(kknn)
library(factoextra)
library(ggplot2)

data=read.table("M:/OMSA/ISYE6501/HW2/iris.txt", stringsAsFactor=FALSE,header=TRUE)


#Plot to look for best combination of predictors
ggplot(data,aes(Sepal.Length,Sepal.Width,color=Species))+geom_point(size=6)
ggplot(data,aes(Petal.Length,Petal.Width,color=Species))+geom_point(size=6)
ggplot(data,aes(Sepal.Length,Petal.Length,color=Species))+geom_point(size=6)
ggplot(data,aes(Sepal.Length,Petal.Width,color=Species))+geom_point(size=6)
ggplot(data,aes(Petal.Length,Sepal.Width,color=Species))+geom_point(size=6)

#create database without response column
iris_data=data[,1:4]

#convert species in numbers
species=c("setosa"=1,"versicolor"=2,"virginica"=3)
data$Species=species[data$Species]

#Finding Best number of Clusters
fviz_nbclust(iris_data,kmeans,method="wss")

#perform clustering
clustering=kmeans(iris_data,centers=3,nstart=10)

#saving predicted cluster values
predicted_species=clustering$cluster

#Checking Accuracy
accuracy=sum(predicted_species==data[,5])/nrow(data)
accuracy


#Plot to look for best combination of predictors
ggplot(data,aes(Sepal.Length,Sepal.Width,color=Species))
```