BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

DEPARTMENT OF INDUSTRIAL ENGINEERING

INE4933 SELECTED TOPICS IN INDUSTRIAL ENGINEERING

PROJECT ASSIGNMENT

Name Last Name: Müge Yalçın

Student ID: 2101743

Due : December 23rd, 2024

Contents

1. Problem Definiton

This project aims to use data sets created with the wakefield package used in R language by utilizing machine learning methods within the scope of the INE4933 course. The project will enable us to predict whether patients in the hospital will die or not based on their characteristics such as identity, race, age, sex, hour, IQ and height, with the methods to be employed. To guarantee the consistency of the model's outputs during this process, the set.seed( ) function was used. The random number sequence was determined using my student number, 2101743, as the starting point. This ensured that the results were repeatable because the same random number sequence was generated each time.

2. Solution Methodology

The K-Nearest Neighbors and Decision Tree algorithms, which fall under the supervised learning and classification subcategories of machine learning, were used for this project. In the R Studio environment, separate analyses were performed for both algorithms, and the models' respective performances were assessed. By comparing the two approaches, the goal is to identify the model with the highest accuracy in order to produce the most accurate outcome. Both models were optimized using a variety of parameter changes, and the outcomes were carefully examined.

2.1 Decision Tree Algorithm

The decision tree algorithm is a tree-structured supervised learning method that divides the data set into subgroups according to feature thresholds. It is a representation that visualizes potential solutions to a problem for particular circumstances (Decision Tree Classification Algorithm, 2024). Decision trees were selected due to their ability to handle both numerical and categorical data, model intricate relationships in the data, and offer insights into the significance of individual features.

2.1.1 Implementation

The implementation of the decision tree algorithm, outputs, and definition of the steps are mentioned in this part.

```
install.packages("wakefield")
```

```
library(wakefield)

set.seed(2101743)

data = r_data(1000)
```

# The required packages are downloaded in order to create the data set. While creating the random data set, the set seed function was used with my student number to get the same random result every time the command ran. r_data(1000) creates a random data set consisting of 1000 rows.

```
summary(data)
```

```
     ID               Race          Age          Sex
 Length:1000       White    :654   Min.   :18.0   Male  :508
 Class :character   Hispanic :149   1st Qu.:36.0   Female:492
 Mode  :character   Black    :114   Median :54.0
                    Asian    : 53   Mean   :53.8
                    Bi-Racial: 18   3rd Qu.:72.0
                    Native   :  5   Max.   :89.0
                    (Other)  :  7
     Hour              IQ            Height        Died
 Min.   :00:00:00   Min.   : 67.0   Min.   :56.00   Mode :logical
 1st Qu.:05:30:00   1st Qu.: 93.0   1st Qu.:67.00   FALSE:493
 Median :11:45:00   Median :100.0   Median :69.00   TRUE :507
 Mean   :11:39:13   Mean   :100.3   Mean   :68.96
 3rd Qu.:17:37:30   3rd Qu.:108.0   3rd Qu.:71.00
 Max.   :23:30:00   Max.   :145.0   Max.   :80.00
```

```
class(data$Died)
```

```
[1] "logical"
```

```
data$Died <- as.factor(data$Died)
```

```
class(data$Died)
```

```
[1] "factor"
```

# Categorical and numerical values were determined by looking at the data summary. Our intended variable Died has been converted from numerical value to factor.

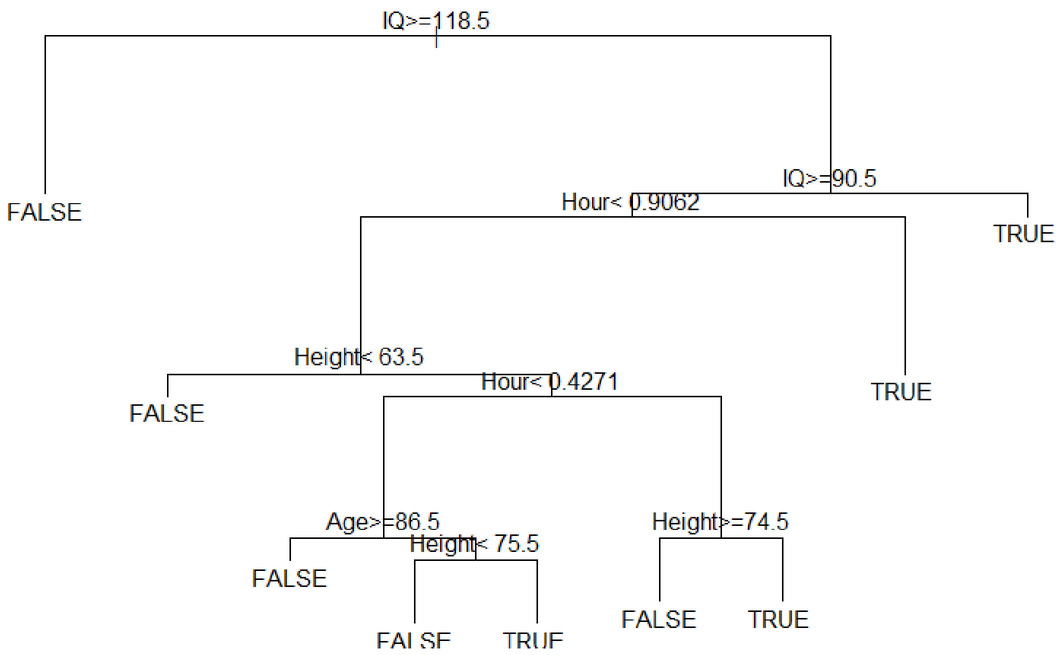
```
install.packages("caTools")

library(caTools)

set.seed(2101743)

sample <- sample.split(data$Died, SplitRatio = 0.8)

trainData <- subset(data, sample == TRUE)

testData <- subset(data, sample == FALSE)
```

# Using the caTools library, the data set was split into 80% training and 20% testing.

```
install.packages("rpart")

library(rpart)

model <- rpart(Died ~ Age + Race + Sex + Hour + IQ + Height, data = trainData, method
= "class",  parms = list(split = "information"))
```

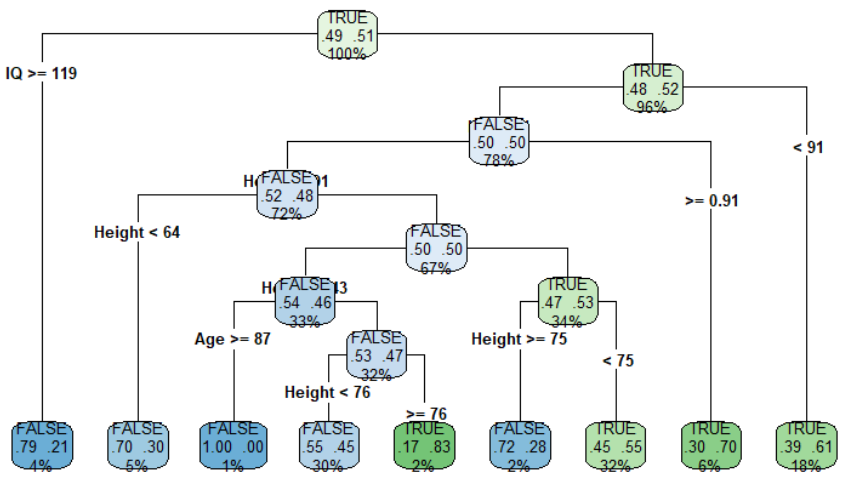
# The Decision Tree model was created using the rpart library.

```
plot(model)

text(model)
```



#The structure of the model is visualized by plot() and text() functions. IQ parameter is selected as a root node. Meaning it has the highest information gain. It can be observed that under height 64 and other sub-criteria, the probability of death not occurring is 79%. In cases where Height more or equal than 76 and Hour smaller than 3, the probability of death is very high (83%).

```
install.packages("rpart.plot")

library(rpart.plot)

rpart.plot(model, type = 4, extra = 104, cex = 0.8)
```



# A better visualization of the decision tree is provided with the rpart.plot library.

```
predicted=predict(model,testData,type="class")

confusion_matrix <- table(predicted, testData$Died)

confusion_matrix
```

```
pred    FALSE TRUE

FALSE    37   39

TRUE     62   62
```

# The model's performance was assessed by comparing the predicted and actual classes. The confusion matrix obtained from this comparison summarizes the correct and incorrect predictions of the model.

```
accuracy <- mean(predicted == testData$Died)

print(paste("Accuracy:", accuracy))
```

```
[1] "Accuracy: 0.495"
```

# 49% of accuracy was maintained

2.2 K- Nearest Neighbors (KNN)

A supervised learning classifier called the k-nearest neighbors (KNN) method classifies or predicts groups of a single data point based on closeness. This methodology was chosen because of its easy interpretability and efficient handling of numerical data. It also offers a standard against which more intricate models may be compared.

2.2.1 Implementation

```
install.packages("wakefield")

library(wakefield)

install.packages(caTools)

library(caTools)

library(class)


set.seed(2101743)

data = r_data(1000)

set.seed(2101743)
```

# Preparing the dataset by downloading the necessary packages

```
data$Died <- as.factor(data$Died)

sample <- sample.split(data$Died, SplitRatio = 0.8)

trainData <- subset(data, sample == TRUE)

testData <- subset(data, sample == FALSE)
```

# Splitting the data into training and testing sets.

```
trainDataY <- trainData$Died

trainDataX <- trainData[, c("Age", "Race", "Sex", "Hour", "IQ", "Height")]

testDataY <- testData$Died

testDataX <- testData[, c("Age", "Race", "Sex", "Hour", "IQ", "Height")]
```

# Separating features and labels for training and testing

```
trainDataX$Race <- as.numeric(as.factor(trainDataX$Race))

trainDataX$Sex <- as.numeric(as.factor(trainDataX$Sex))
```

```
    testDataX$Race <- as.numeric(as.factor(testDataX$Race))

    testDataX$Sex <- as.numeric(as.factor(testDataX$Sex))
```

#Converting categorical variables to numeric

```
    trainDataX <- scale(trainDataX)

    testDataX <- scale(testDataX)
```

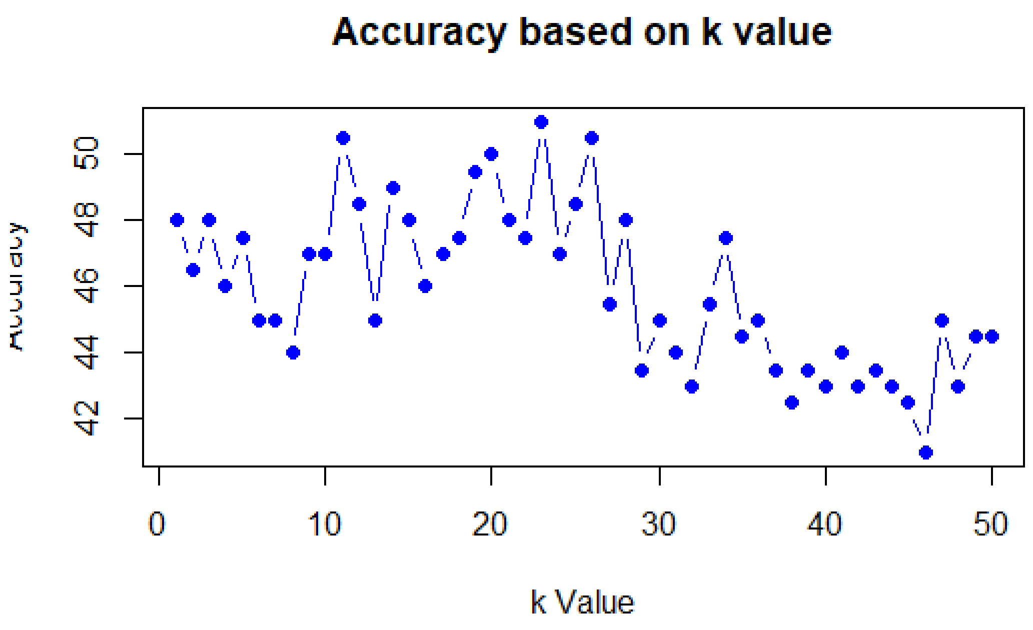
# Normalizing the data

```
    k <- sqrt(NROW(data))

    k.optm <- numeric(50)

    for (i in 1:50) {

      knn.mod <- knn(trainDataX, testDataX, trainDataY, k = i)

      k.optm[i] <- 100 * sum(knn.mod == testDataY) / NROW(testDataY)

      cat(i, '=', k.optm[i], '\n')

    }
```

# The optimal is determined by running KNN with different values of k.

```
     plot(k.optm, type = "b", xlab = "k Value", ylab = "Accuracy", main = "Accuracy based on
k value", col="blue", pch =19)
```

#Plotting the Accuracy based on k value



```
optimal_k <- which.max(k.optm)

cat("Optimal k value:", optimal_k, "\n")

Optimal k value: 23
```

#Determining the optimal k value

```
set.seed(2101743)

knn.model <- knn(trainDataX, testDataX, trainDataY, k = optimal_k)
```

# Running KNN with the optimal k value

```
confusion_matrix <- table(knn.model, testDataY)

confusion_matrix
```

```
         testDataY
knn.model FALSE TRUE
    FALSE    46   45
    TRUE     53   56
```

```
accuracy <- mean(knn.model == testDataY)

cat("Accuracy: ", accuracy, "\n")
```

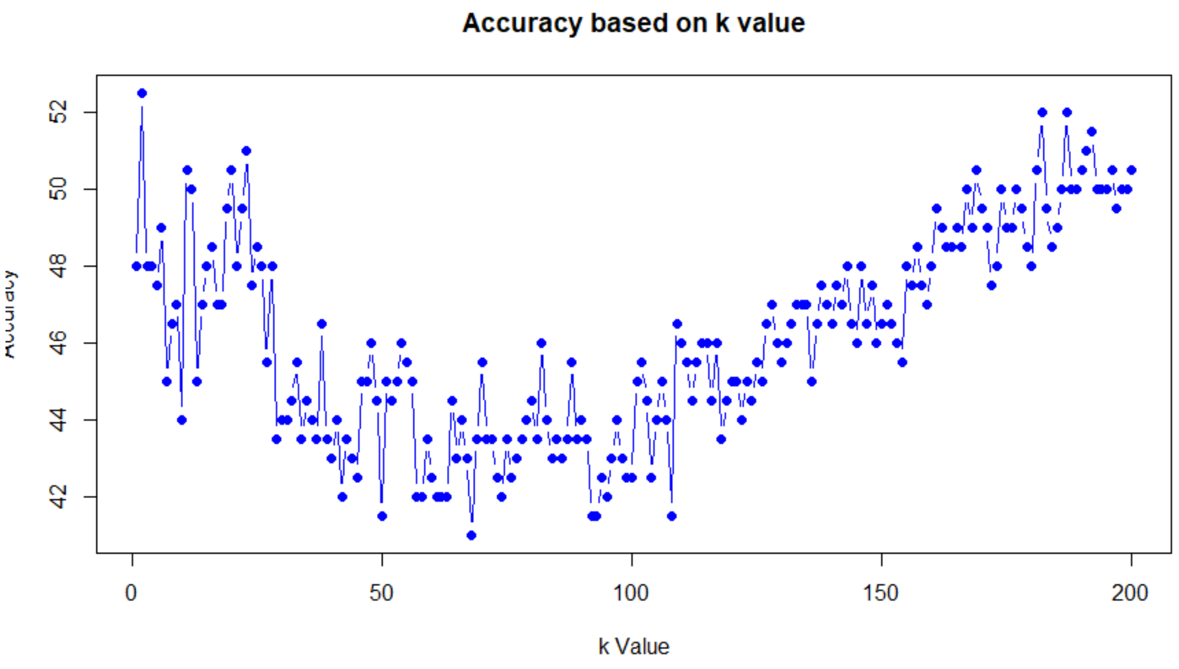
[1] "Accuracy: 0.51

# performance with a confusion matrix and accuracy is evaluated. 51% of accuracy is maintained.

Determination of Optimal k Value

When i = 200

Plot:

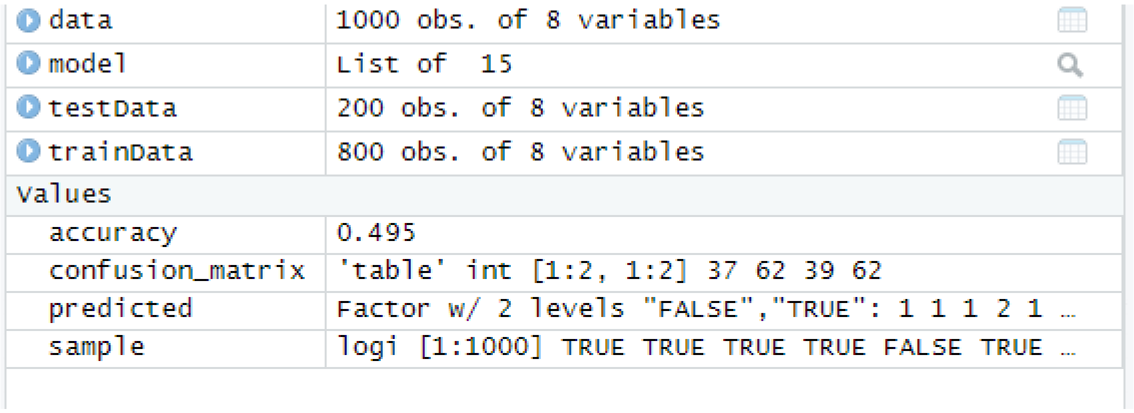


Accuracy: 0.55

When the i value was 200, the accuracy rate was 55%. However, an excessively high value of k in the k-NN algorithm may lead to an overly generalized model. Large k values also increase the computational cost and reduce the precision of the model. Consequently, i = 50 was chosen since it offers the best accuracy rate and a more suitable generalization level.

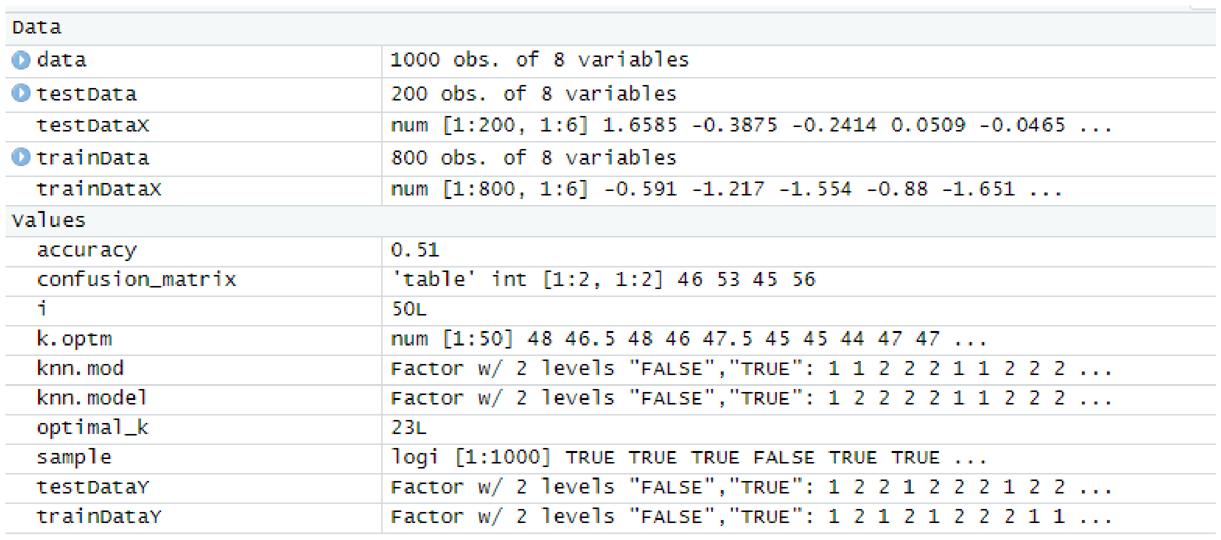
3. Comparasion of the Models

Decision Tree Algorithm:



```
data                1000 obs. of 8 variables
model               List of  15
testData            200 obs. of 8 variables
trainData           800 obs. of 8 variables
Values
  accuracy          0.495
  confusion_matrix  'table' int [1:2, 1:2] 37 62 39 62
  predicted         Factor w/ 2 levels "FALSE","TRUE": 1 1 1 2 1 …
  sample            logi [1:1000] TRUE TRUE TRUE TRUE FALSE TRUE …
```

K- Nearest Neighbors (KNN):



```
Data
data                1000 obs. of 8 variables
testData            200 obs. of 8 variables
  testDataX         num [1:200, 1:6] 1.6585 -0.3875 -0.2414 0.0509 -0.0465 ...
trainData           800 obs. of 8 variables
  trainDataX        num [1:800, 1:6] -0.591 -1.217 -1.554 -0.88 -1.651 ...
Values
  accuracy          0.51
  confusion_matrix  'table' int [1:2, 1:2] 46 53 45 56
  i                 50L
  k.optm            num [1:50] 48 46.5 48 46 47.5 45 45 44 47 47 ...
  knn.mod           Factor w/ 2 levels "FALSE","TRUE": 1 1 2 2 2 1 1 2 2 2 ...
  knn.model         Factor w/ 2 levels "FALSE","TRUE": 1 2 2 2 2 1 1 2 2 2 ...
  optimal_k         23L
  sample            logi [1:1000] TRUE TRUE TRUE FALSE TRUE TRUE ...
  testDataY         Factor w/ 2 levels "FALSE","TRUE": 1 2 2 1 2 2 2 1 2 2 ...
  trainDataY        Factor w/ 2 levels "FALSE","TRUE": 1 2 1 2 1 2 2 2 1 1 ...
```

The best accuracy rate for the KNN algorithm was obtained as 51% with k=23 value. The accuracy rate for the Decision Tree algorithm was calculated as 49.5%. According to these results, the KNN algorithm provides higher accuracy compared to the Decision Tree algorithm.

In terms of confusion matrix, the correct prediction rates of positive and negative classes are more balanced in the KNN method.

4. Conclusion

This project aimed to predict whether a patient in a hospital would die by using machine learning algorithms, based on the features given as: ID Age, Race, Sex, Hour, IQ, and Height. To achieve this two different machine learning algorithms: K-Nearest Neighbors (KNN) and

Decision Tree are employed. In both models, the ID parameter was excluded from the analysis as it would not have an impact on predicting the outcome of death.

To create predictive models, both algorithms are employed after preprocessing the data and dividing it into training and testing sets with a k-value of 23, the optimal accuracy rate for the KNN algorithm was 51%. On the other hand, the Decision Tree algorithm provided an accuracy rate of 49.5%.

Based on these results, the KNN algorithm performed with slightly higher accuracy compared to the Decision Tree algorithm. Nevertheless, given that both algorithms' accuracy rates hover around 50%, it can be concluded that the data set requires a better data cleaning procedure. Both models, however, provide a reasonable starting point for predictive analysis in healthcare.