Exercise 5

Question 1: We will continue to use the heart attack diagnosis dataset (Patient_data.xlsx) we used for Exercise 3.

Again, the file contains 7998 records. The following screenshot shows you what the dataset actually contains.

	A	B	C	D	E	F	G	H	I
1	age	gender	diabetes	smoker	active	obesity	heartattac	bp	cholesteral
2	54	Female	No	No	Yes	No	No	Hypertension	Normal
3	64	Female	No	No	No	No	Yes	Normal	Normal
4	63	Female	No	No	No	No	Yes	Normal	Highl
5	67	Male	No	Yes	No	No	No	Hypotension	Highl
6	76	Male	No	No	No	No	No	Hypotension	Normal
7	69	Male	No	No	No	No	No	Normal	Normal
8	67	Male	Yes	Yes	Yes	No	Yes	Hypertension	Normal
9	74	Male	No	No	No	Yes	Yes	Normal	Normal
10	69	Male	No	Yes	Yes	No	No	Normal	Highl
11	54	Female	No	Yes	No	No	Yes	Normal	Highl
12	57	Male	No	No	Yes	No	No	Normal	Normal
13	49	Female	No	No	Yes	No	No	Normal	Normal
14	66	Female	No	Yes	No	Yes	Yes	Normal	Normal
15	51	Female	No	No	No	Yes	No	Hypertension	Highl
16	63	Male	No	Yes	Yes	No	Yes	Normal	Highl
17	71	Female	No	Yes	Yes	No	Yes	Normal	Normal
18	70	Female	No	No	Yes	No	No	Normal	Normal
19	76	Male	No	No	Yes	No	No	Normal	Highl
20	50	Male	No	Yes	No	No	Yes	Normal	Normal

A) (10 points) Explore the data set, then use a neural network to model this classification problem (no partition at this step).

```
# Installing and loading all the libraries
#Using the pre-processing and EDA steps similar to previous assignemnt
library(polycor)
library(readxl)
library("readxl")
library('C50')
library(rpart)
library("nnet")
library(devtools)
library(reshape)
library(caret)
library(rattle)
library(psych)#categorical correlation
#install.packages('NeuralNetTools')
library(NeuralNetTools)
#install.packages('neuralnet')
library(neuralnet)
```

```
df <- read_excel("R://downloads//Patient_Data.xlsx")
#now we inspect data to see each variable
str(df)
#since variable type is char, we change it to factor for all the applicable vairables
df[sapply(df, is.character)] <- lapply(df[sapply(df, is.character)],as.factor)
#lets check if they are converted to factors
str(df)
```

From the initial inspection we can see that all the variables except age was given the type as chr, hence I converted all of them to factors for visualization and analysis purposes.

```
> library(psych)#categorical correlation
> df <- read_excel("R://downloads//Patient_Data.xlsx")
> #now we inspect data to see each variable
> str(df)
tibble [7,998 x 9] (S3: tbl_df/tbl/data.frame)
 $ age          : num [1:7998] 54 64 63 67 76 69 67 74 69 54 ...
 $ gender       : chr [1:7998] "Female" "Female" "Female" "Male" ...
 $ diabetes     : chr [1:7998] "No" "No" "No" "No" ...
 $ smoker       : chr [1:7998] "No" "No" "No" "Yes" ...
 $ active       : chr [1:7998] "Yes" "No" "No" "No" ...
 $ obesity      : chr [1:7998] "No" "No" "No" "No" ...
 $ heartattack_s: chr [1:7998] "No" "Yes" "Yes" "No" ...
 $ bp           : chr [1:7998] "Hypertension" "Normal" "Normal" "Hypotension" ...
 $ cholesteral  : chr [1:7998] "Normal" "Normal" "Highl" "Highl" ...
> #since variable type is char, we change it to factor for all the applicable vairables
> df[sapply(df, is.character)] <- lapply(df[sapply(df, is.character)],as.factor)
> #lets check if they are converted to factors
> str(df)
tibble [7,998 x 9] (S3: tbl_df/tbl/data.frame)
 $ age          : num [1:7998] 54 64 63 67 76 69 67 74 69 54 ...
 $ gender       : Factor w/ 2 levels "Female","Male": 1 1 1 2 2 2 2 2 2 1 ...
 $ diabetes     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
 $ smoker       : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 2 1 2 2 ...
 $ active       : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 2 1 2 1 ...
 $ obesity      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 2 1 1 ...
 $ heartattack_s: Factor w/ 2 levels "No","Yes": 1 2 2 1 1 1 2 2 1 2 ...
 $ bp           : Factor w/ 3 levels "Hypertension",..: 1 3 3 2 2 3 1 3 3 3 ...
 $ cholesteral  : Factor w/ 2 levels "Highl","Normal": 2 2 1 1 2 2 2 2 1 1 ...
> 
```

```
#Lets summarize our data
summary(df)
```

From the summary we can say that the dataset contains total 9 variables with only one numeric variable, 7 binary categories and 1 category variable with 3 categories.
From summary we can also observe that the gender almost evenly split same even distribution applies for attribute active as well. Diabetes has most in the no category. Smoker most have answered no. Obesity has the majority categorized as a 'no'. Bp variable has normal, but hypertension has the second highest, followed by hypotension. The cholesterol variable is close to even too.

```
> summary(df)
      age            gender      diabetes    smoker      active      obesity     heartattack_s            bp          cholesteral
 Min.   :45.00   Female:3959   No :7456   No :6346   No :3858   No :6230   No :4491     Hypertension:2011   Highl :3064
 1st Qu.:55.00   Male  :4039   Yes: 542   Yes:1652   Yes:4140   Yes:1768   Yes:3507     Hypotension : 985   Normal:4934
 Median :61.00                                                                          Normal      :5002
 Mean   :61.85
 3rd Qu.:68.00
 Max.   :89.00
> 
```

Now we check the correlation between the binary categorical variables with respect to our dependent variable heartattach_s.
The correlation between binary category variables is calculated using tetrachoric correlation. This test is only performed between variables that have just two potential values.

A tetrachoric correlation can have a value ranging from -1 to 1, where:

- A high negative correlation between the two variables is indicated by a value of -1.
- There is no association between the two variables if the value is 0.
- A significant positive correlation between the two variables is indicated by a value of 1.

```
# to check correlation between each binary categorical data
cc=table(df$diabetes,df$heartattack_s)
tetrachoric(cc)
cc=table(df$gender,df$heartattack_s)
tetrachoric(cc)
cc=table(df$smoker,df$heartattack_s)
tetrachoric(cc)
cc=table(df$active,df$heartattack_s)
tetrachoric(cc)
cc=table(df$obesity,df$heartattack_s)
tetrachoric(cc)
cc=table(df$cholesteral,df$heartattack_s)
tetrachoric(cc)
cc=table(df$bp,df$heartattack_s)
tetrachoric(cc)
```

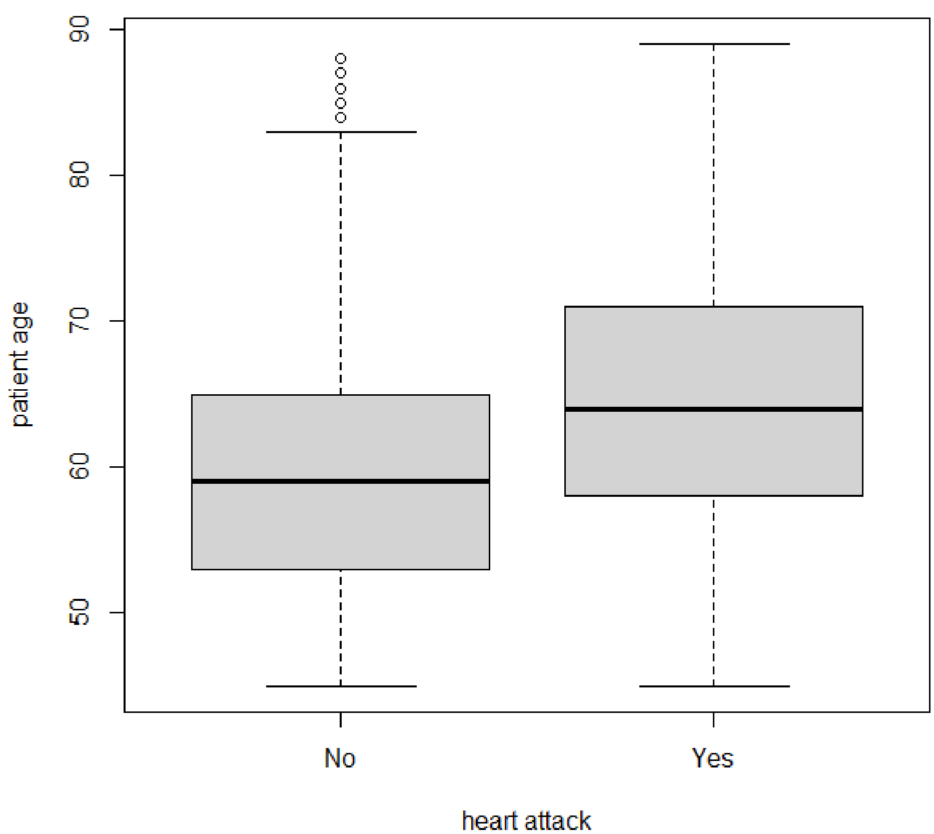
```
library(psych)
cc=table(df$diabetes,df$heartattack_s)
tetrachoric(cc)

cc=table(df$gender,df$heartattack_s)
tetrachoric(cc)

cc=table(df$smoker,df$heartattack_s)
tetrachoric(cc)

cc=table(df$active,df$heartattack_s)
tetrachoric(cc)

cc=table(df$obesity,df$heartattack_s)
tetrachoric(cc)

cc=table(df$cholesteral,df$heartattack_s)
tetrachoric(cc)
```

```
> cc=table(df$diabetes,df$heartattack_s)
> cc

        No  Yes
  No  4394 3062
  Yes   97  445
> tetrachoric(cc)
Call: tetrachoric(x = cc)
tetrachoric correlation
[1] 0.5


>
> cc=table(df$gender,df$heartattack_s)
> tetrachoric(cc)
Call: tetrachoric(x = cc)
tetrachoric correlation
[1] -0.018


>
> cc=table(df$smoker,df$heartattack_s)
> tetrachoric(cc)
Call: tetrachoric(x = cc)
tetrachoric correlation
[1] 0.39

 with tau of
  No   No
0.82 0.15
>
> cc=table(df$active,df$heartattack_s)
> tetrachoric(cc)
Call: tetrachoric(x = cc)
tetrachoric correlation
[1] -0.34

 with tau of
    No     No
-0.044  0.155
>
> cc=table(df$obesity,df$heartattack_s)
> tetrachoric(cc)
Call: tetrachoric(x = cc)
tetrachoric correlation
[1] 0.34
```

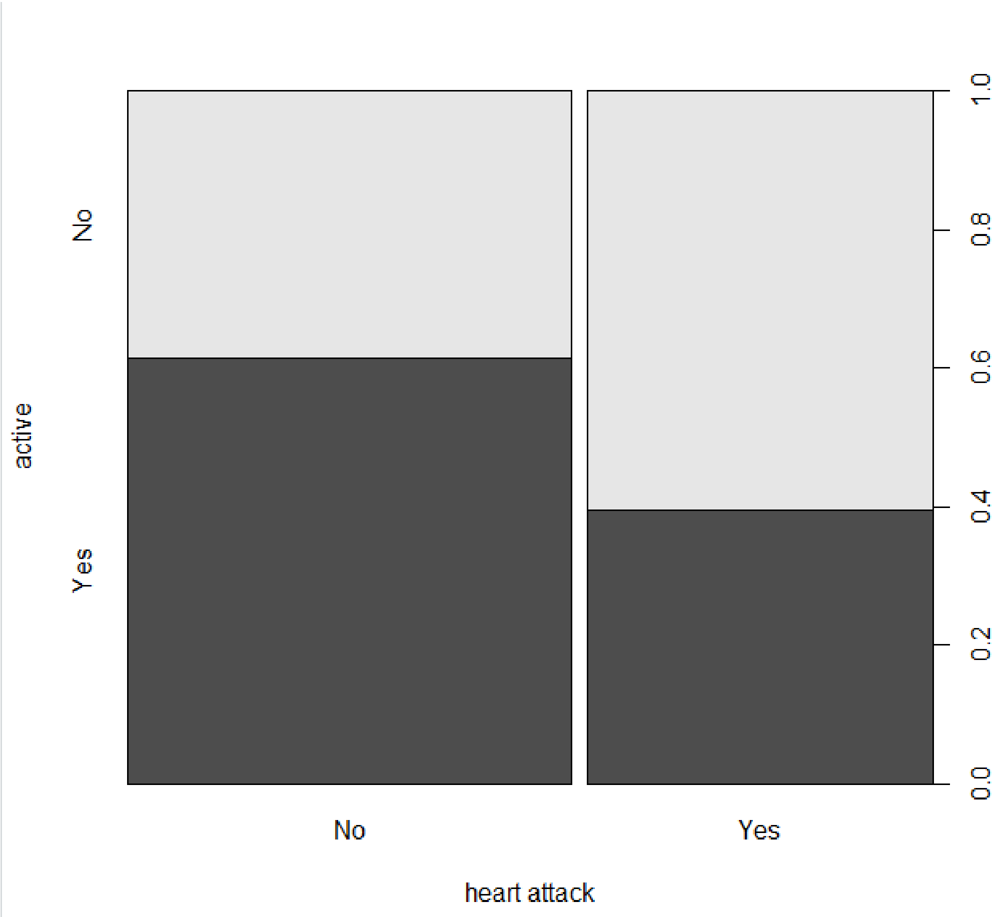
```
> cc=table(df$cholesteral,df$heartattack_s)
> tetrachoric(cc)
Call: tetrachoric(x = cc)
tetrachoric correlation
[1] -0.25

 with tau of
Highl     No
-0.30   0.15
```

From the above correlations we can see that some categorical variables have medium negative correlation with respect to dependent variable and some have medium positive correlation, with diabetes having highest positive correlation of 0.5 , active having lowest negative correlation of -0.34 and gender having almost no correlation (close to 0).

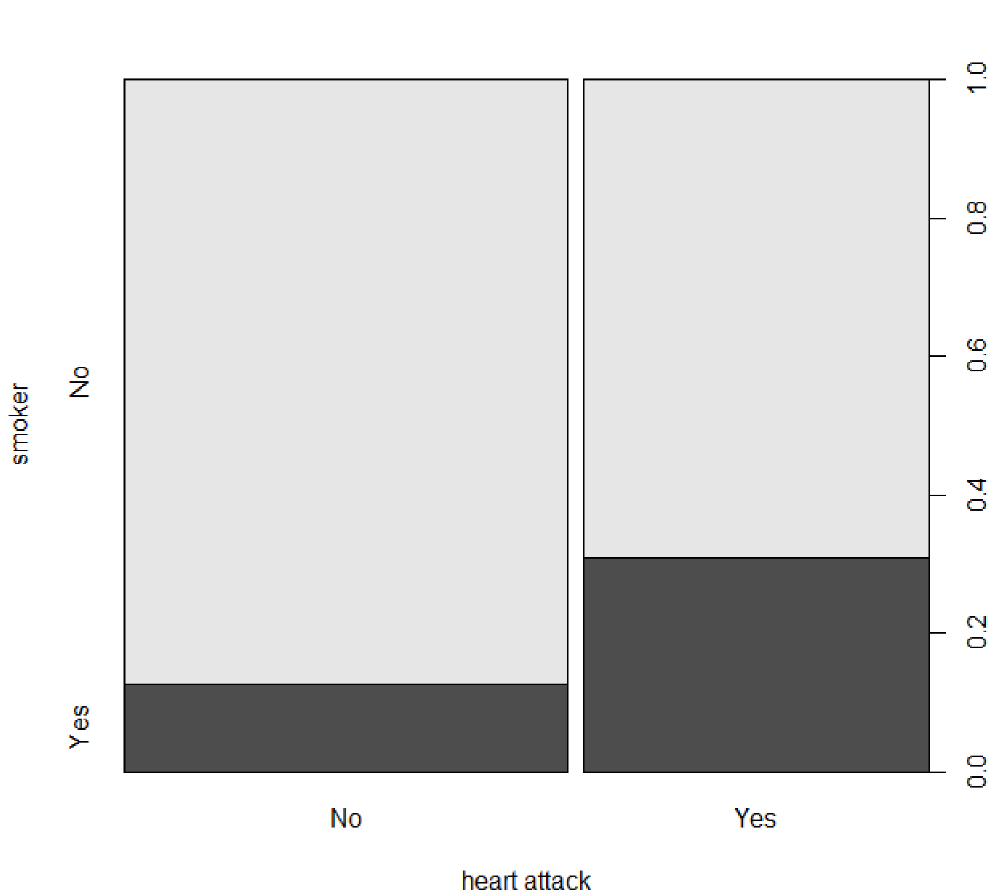
```
#EDA

plot(df$heartattack_s,df$age, xlab="heart attack", ylab="patient age")
plot(df$heartattack_s,df$active, xlab="heart attack", ylab="active")
plot(df$heartattack_s,df$smoker, xlab="heart attack", ylab="smoker")
plot(df$heartattack_s,df$gender, xlab="heart attack", ylab="gender")

#outliers in age variable
```



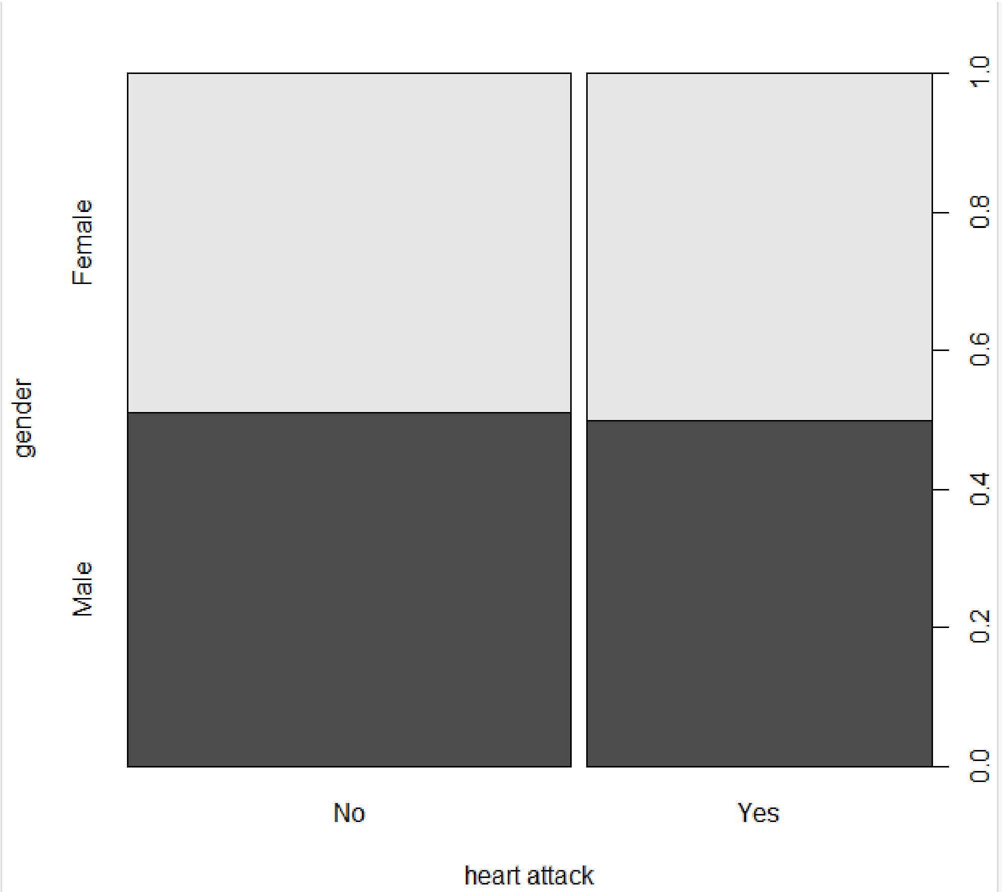
First I have plotted heart attacks wrt to age. For patients with no heart attack, the median age is 58 or 59, whereas for patients who got heart attack the median age is higher, around 62. There also appears to be a few outliers in age with no heart attack. From this we can see that as the age goes up, so does the probability of having a heart attack, hence there is small positive correlation between the two.



As seen in correlation active having lowest negative correlation, we can confirm that with this graph as more the patient is active the lesser the chances of them having a heart attack.

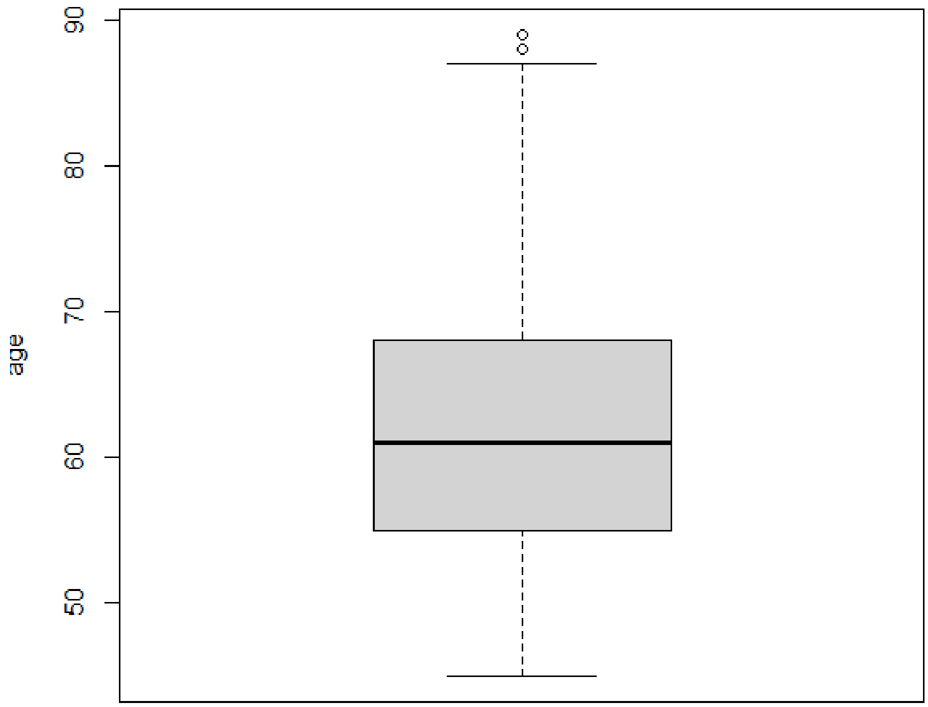


Here the plot proves our previous point that there is slight positive correlation between the the smoker and heart attack as the chances of having a heart attack is higher for patients that smoke.



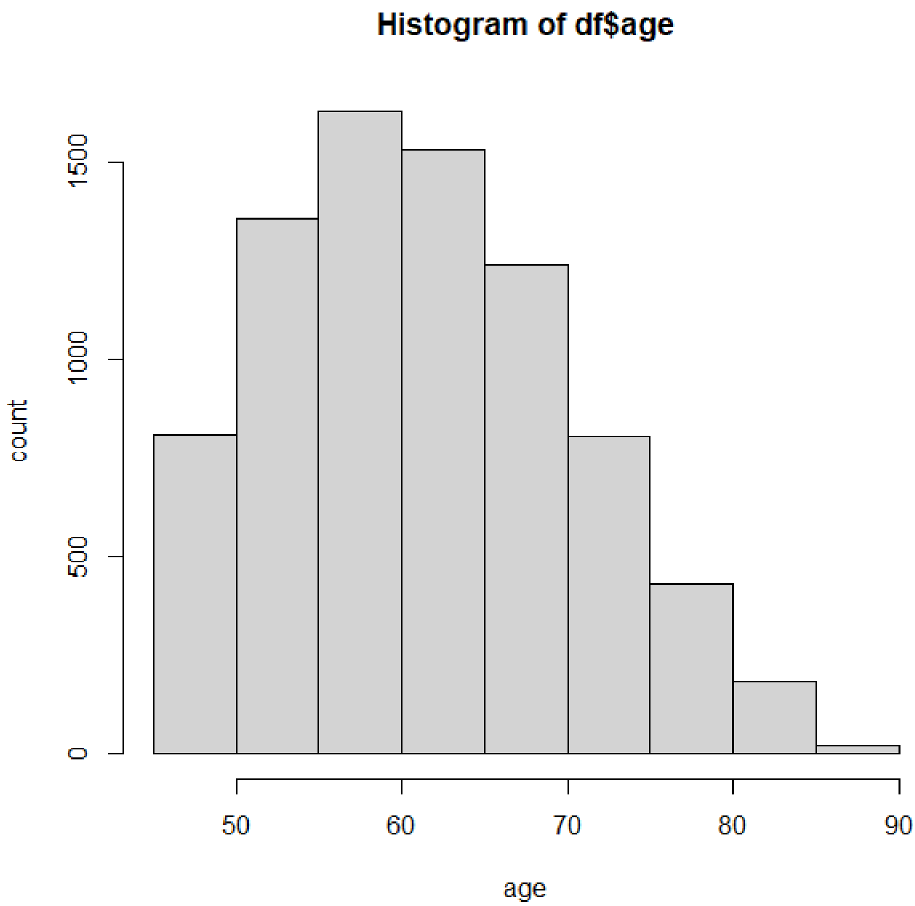
As shown in correlation plot, the gender hardly has any correlation with respect to heart attack. This graph further proves our point.

```
#outliers in age variable
boxplot(df$age)
```



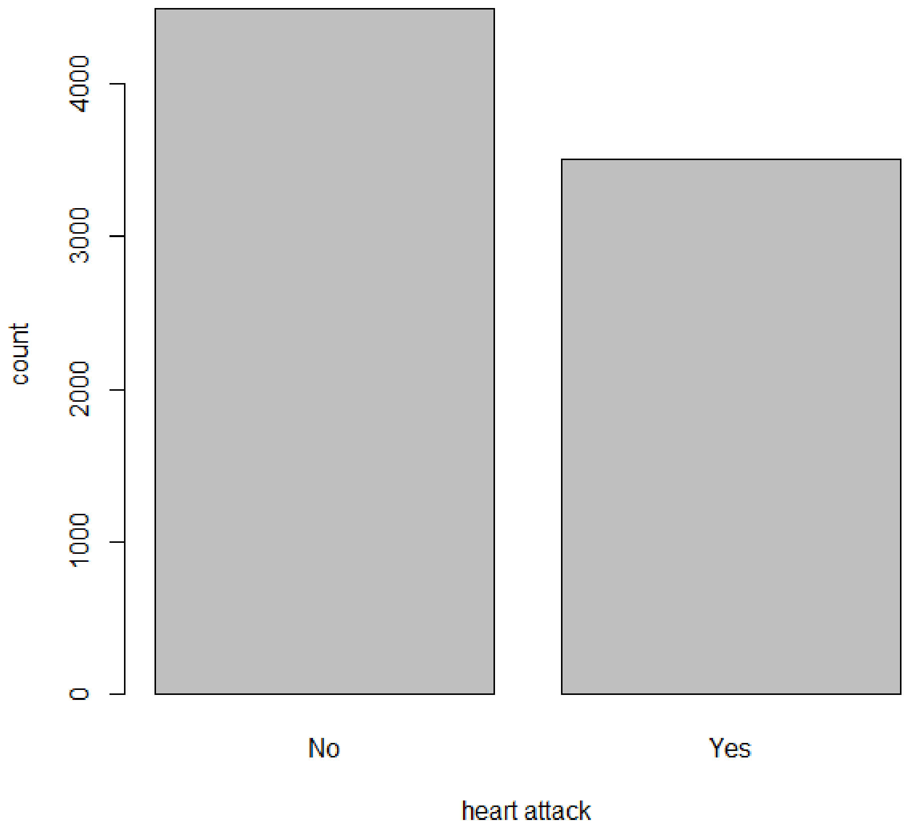
As we had seen some outliers in the age variable, I plotted this boxplot to see there are few outliers (patients with age near to 90), we won't be pruning them as they might hold some valuable information and are not that large in number. Since we are using tree-based methods, they are robust to outliers.

```
#skewness of age variable
hist(df$age,ylab="count",xlab="age")
```



The age variable is slightly right skewed here. Since we are making decision trees, I would not like to normalize and change the age values as the raw values would be better for interpretation.

```
#checking if the data is balanced
barplot(table(df$heartattack_s))
```



From this graph we can observe that the dependent variable (heart attack) is slightly imbalanced here as the number of heart attacks are quite low as compared to patients with no heart attacks. We will be trying to train a model with balanced heart attack attribute as well (please refer the last model in assignment)

```
#to check if there are any duplicate values
#duplicated(df)
sum(duplicated(df))
nrow(df)
df2 =unique(df)
nrow(df2)
```

```
> #to check if there are any duplicate values
> #duplicated(df)
> sum(duplicated(df))
[1] 4492
> nrow(df)
[1] 7998
> df2 =unique(df)
> nrow(df2)
[1] 3506
```

From this we can observe that there are a lot of duplicate values (4492 rows). Once we prune them we will be only left with 3,506 rows from the original data.

```
#check for missing values
sum(is.na(df))
```

```
> #check for missing values
> sum(is.na(df))
[1] 0
```

As seen in the summary there are no missing values in the dataset.

Now we build neural networks without data splitting:

```
h <- preProcess(df[1], method = c("range"))
df3 <- cbind(predict(h, df[,-7]),heartattack_s = df$heartattack_s)
df3
#building NN with iterations as 1000 and 10 units in hidden layer
model1 <- nnet(heartattack_s ~ ., data = df3, size = 10, maxit = 1000)
plotnet(model1)
View(model1)
#now we calculate the accuracy
pred <- predict(model1, df3[,1:8], type = "class")
mean(pred == df3$heartattack_s)
pred_table = table(predicted = pred, actual = df$heartattack_s)

##########################################################################################
```

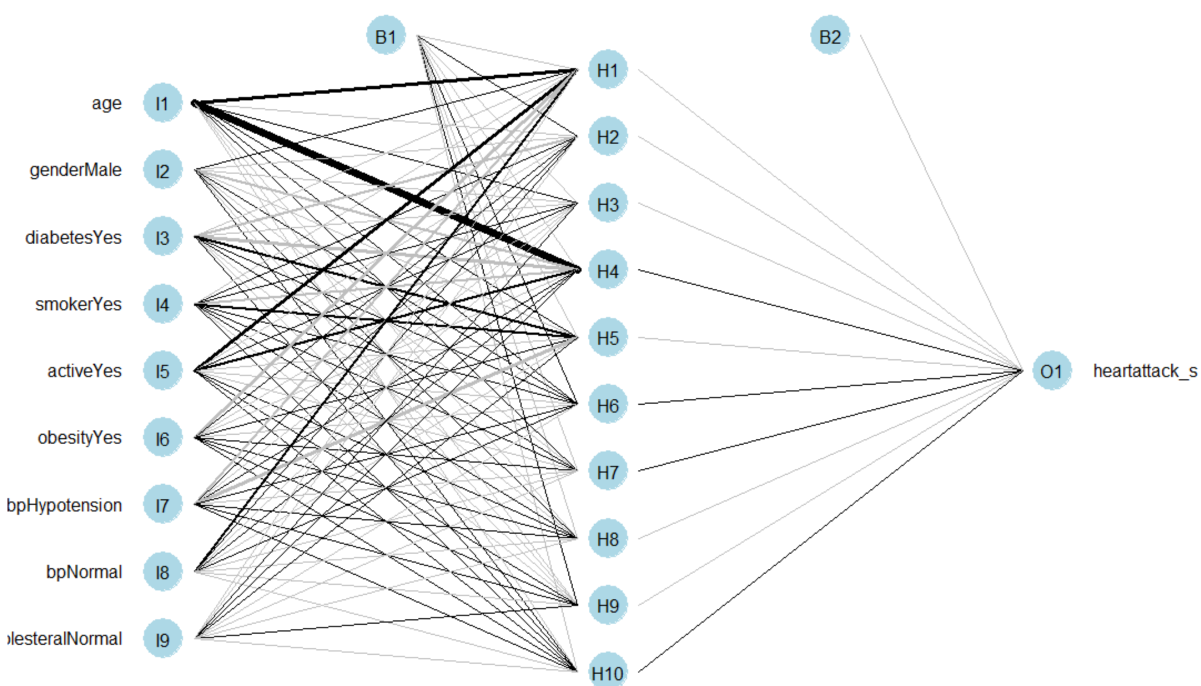
```
> h <- preProcess(df[1], method = c("range"))
> df3 <- cbind(predict(h, df[,-7]),heartattack_s = df$heartattack_s)
> #df3
> #building NN with iterations as 1000 and 10 units in hidden layer
> model1 <- nnet(heartattack_s ~ ., data = df3, size = 10, maxit = 1000)
# weights:  111
initial  value 5592.461517
iter  10 value 4341.479284
iter  20 value 4305.380644
iter  30 value 4278.574817
iter  40 value 4257.970301
iter  50 value 4248.030212

iter 620 value 4168.661540
iter 630 value 4168.502538
iter 640 value 4168.282342
final  value 4168.191769
converged
> plotnet(model1)
> View(model1)
> #now we calculate the accuracy
> pred <- predict(model1, df3[,1:8], type = "class")
> mean(pred == df3$heartattack_s)
[1] 0.7423106
> pred_table = table(predicted = pred, actual = df$heartattack_s)
>
```

As age had outliers as well as low and high value, I preprocessed it to scale it between 0 and 1. Then we build our neural network on this data for 1000 iterations and 10 hidden units in the first and only hidden layer. From the output we can see that model converged at 640th iteration and did not run till 1000.

The accuracy for this non- partitioned model turned out to be 74.23%, which isn't that great. Below is the confusion matrix.

```
> pred_table
         actual
predicted   No  Yes
      No  3727 1297
      Yes  764 2210
>

> rec <- pred_tab[2,2]/sum(pred_table[,2])
> rec
[1] 0.5075563
>
```

From the confusion matrix we can see that the false negative classification is 1297 which isn't a good number in heart attack diagnosis as it indicates that the model predicted patients wouldn't have attacks and did have them. Recall accounts for the false negative cases, here the recall isn't good its 50.7%



Above is the diagram of our neural network. We can see that each categorical variable has selected category taken, also here the dark lines between the input and hidden layer represent the positive weights associated.

B) (20 points) Comparing with a C5.0 model you had in Exercise 6, is there any difference?

```
#building the c5.0 model
c50tree1 <- C5.0(df[,-7], as.factor(df$heartattack_s))
#summary
summary(c50tree1)
```

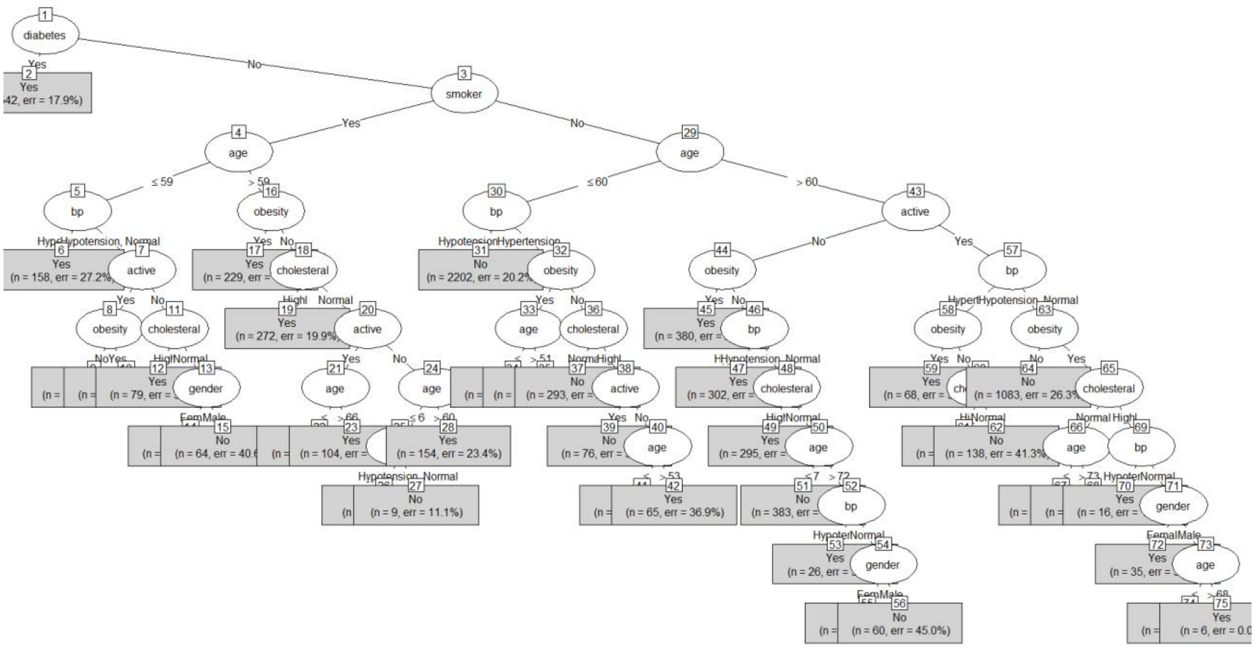
```
> summary(c50tree1)
```

```
Call:
C5.0.default(x = df[, -7], y = as.factor(df$heartattack_s))


C5.0 [Release 2.07 GPL Edition]         Sat Apr 09 17:42:20 2022
-------------------------------

Class specified by attribute `outcome'

Read 7998 cases (9 attributes) from undefined.data

Decision tree:

diabetes = Yes: Yes (542/97)
diabetes = No:
:...smoker = Yes:
    :...age <= 59:
    :   :...bp = Hypertension: Yes (158/43)
    :   :   bp in {Hypotension,Normal}:
    :   :   :...active = Yes:
    :   :       :...obesity = No: No (262/75)
    :   :       :   obesity = Yes: Yes (42/17)
    :   :       active = No:
    :   :       :...cholesteral = High1: Yes (79/27)
    :   :           cholesteral = Normal:
    :   :           :...gender = Female: Yes (60/26)
    :   :               gender = Male: No (64/26)
    :   age > 59:
    :   :...obesity = Yes: Yes (229/26)
    :       obesity = No:
    :       :...cholesteral = High1: Yes (272/54)
    :           cholesteral = Normal:
    :           :...active = Yes:
    :               :...age <= 66: No (90/34)
    :               :   age > 66: Yes (104/39)
    :               active = No:
    :               :...age > 60: Yes (154/36)
    :                   age <= 60:
    :                   :...bp = Hypertension: Yes (3)
    :                       bp in {Hypotension,Normal}: No (9/1)
    smoker = No:
    :...age <= 60:
        :...bp in {Hypotension,Normal}: No (2202/445)
        :   bp = Hypertension:
        :   :...obesity = Yes:
        :       :...age <= 51: No (45/16)
        :       :   age > 51: Yes (121/44)
        :       obesity = No:
        :       :...cholesteral = Normal: No (293/64)
        :           cholesteral = High1:
        :           :...active = Yes: No (76/29)
        :               active = No:
        :               :...age <= 53: No (51/19)
        :                   age > 53: Yes (65/24)
        age > 60:
        :...active = No:
```

```
age > 60:
:...active = No:
    :...obesity = Yes: Yes (380/93)
    :   obesity = No:
    :   :...bp = Hypertension: Yes (302/100)
    :       bp in {Hypotension,Normal}:
    :       :...cholesteral = High: Yes (295/124)
    :           cholesteral = Normal:
    :           :...age <= 72: No (383/136)
    :               age > 72:
    :               :...bp = Hypotension: Yes (26/8)
    :                   bp = Normal:
    :                   :...gender = Female: Yes (62/28)
    :                       gender = Male: No (60/27)
    active = Yes:
    :...bp = Hypertension:
        :...obesity = Yes: Yes (68/21)
        :   obesity = No:
        :   :...cholesteral = High: Yes (78/32)
        :       cholesteral = Normal: No (138/57)
        bp in {Hypotension,Normal}:
        :...obesity = No: No (1083/285)
            obesity = Yes:
            :...cholesteral = Normal:
                :...age <= 73: No (104/31)
                :   age > 73: Yes (26/10)
                cholesteral = High:
                :...bp = Hypotension: Yes (16/2)
                    bp = Normal:
                    :...gender = Female: Yes (35/12)
                        gender = Male:
                        :...age <= 68: No (15/2)
                            age > 68: Yes (6)


Evaluation on training data (7998 cases):

	    Decision Tree   
	  ----------------  
	  Size      Errors  

	    38 2110(26.4%)   <<


	   (a)   (b)    <-classified as
	  ----  ----
	  3628   863    (a): class No
	  1247  2260    (b): class Yes


	Attribute usage:

	100.00% diabetes
	 93.22% age
	 93.22% smoker
	 77.86% bp
	 61.18% obesity
	 51.71% active
	 32.06% cholesteral
	  3.78% gender
```

```
#Now let us compute the accuracy of our model
predictions <- predict(c50tree1, df)
mean(predictions==df$heartattack_s)
```

```
> plot(c50tree1,type="simple",cex=.7)
> #Now let us compute the accuracy of our model
> predictions <- predict(c50tree1, df)
> mean(predictions==df$heartattack_s)
[1] 0.736184
>
```




```
> rec <- pred_table2[2,2]/sum(pred_table2[,2])
> rec
[1] 0.6444254
>
```

As we can observe diabetes is the first segregation attribute in the tree. This result is quite obvious as in our EDA we got to know diabetes had the highest correlation. It shows if someone has diabetes, they will be categorized as having a heart attack. Whereas if they don't have diabetes, they continue getting segregated down the tree. After Diabetes we can see that age and smoker were the next two top variable used for segregation, again this was quite evident from EDA. The error rate for this model is 26.4%. From the confusion matrix we can see that the false negative classification is 1247 which isn't a good number in heart attack diagnosis. Lastly, we can also see the attribute usage, as discussed diabetes, age, smoker is on top of the list. Recall accounts for the false negatives' cases, here the recall isn't great either, its 64.4%

We can see that the accuracy of our c5 model is 73.6% while that of neural network is 74.23%, hence there is not much difference in the accuracies between the two models. Though we can say that since false negative cases are more important for this data, neural networks give worse performance for non-partitioned data as the false negative number are slightly higher by a difference of 50 cases (with much lower percentage of recall 50%). Also, in the case of c5 model we can get the representation of attribute usage, which is not the case in neural networks. Hence for some cases like this c5 model might be preferred as the user will know which attributes are more important in determining the heart attacks.

C) (30 points) Using partition ratio 80:20 to rerun your neural network model. Do you have the similar accuracy for the test data? If not, how will you improve your model?

```
#Buliding NN with partition
set.seed(100)
sampl <- createDataPartition(df$heartattack_s, p = 0.80, list = FALSE)
df_train <- df[sampl,]
View(df_train)
df_test <- df[-sampl,]
View(df_test)
heart_partition <- preProcess(df_train[1], method = c("range"))
df_train <- cbind(predict(heart_partition, df_train[-7]),heartattack_s = df_train$heartattack_s)
df_test  <- cbind(predict(heart_partition, df_test[-7]), heartattack_s = df_test$heartattack_s)

model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 10)
model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 10, maxit = 1000)
plotnet(model1_partition)
#calculating the accuracy and confusion matrix of the train data
pred <- predict(model1_partition, df_train[,1:8], type = "class")
mean(pred == df_train$heartattack_s)
pred_tab = table(predicted = pred, actual = df_train$heartattack_s)
pred_tab
#calculating the accuracy and confusion matrix of the test data
test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")
mean(test_predictions == df_test$heartattack_s)
pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)
pred_tab_test
```

```
> #Buliding NN with partition
> set.seed(100)
> sampl <- createDataPartition(df$heartattack_s, p = 0.80, list = FALSE)
> df_train <- df[sampl,]
> View(df_train)
> df_test <- df[-sampl,]
> View(df_test)
> heart_partition <- preProcess(df_train[1], method = c("range"))
> df_train <- cbind(predict(heart_partition, df_train[-7]),heartattack_s = df_train$heartattack_s)
> df_test  <- cbind(predict(heart_partition, df_test[-7]), heartattack_s = df_test$heartattack_s)
>
> model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 10)
# weights:  111
initial  value 7466.019818
iter  10 value 3463.828592
iter  20 value 3420.304708
iter  30 value 3405.698223
iter  40 value 3396.881248
iter  50 value 3389.419729
iter  60 value 3380.071955
iter  70 value 3373.941292
iter  80 value 3364.751829
iter  90 value 3355.646808
iter 100 value 3347.560673
final  value 3347.560673
stopped after 100 iterations
> model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 10, maxit = 1000)
# weights:  111
initial  value 5188.656642
iter  10 value 3479.939007
iter  20 value 3432.034588
iter  30 value 3411.532612
```

```
iter 490 value 3305.304395
final  value 3305.292698
converged
> plotnet(model1_partition)
> #calculating the accuracy and confusion matrix of the train data
> pred <- predict(model1_partition, df_train[,1:8], type = "class")
> mean(pred == df_train$heartattack_s)
[1] 0.7446476
> pred_tab = table(predicted = pred, actual = df_train$heartattack_s)
> pred_tab
         actual
predicted   No  Yes
      No  2985 1026
      Yes  608 1780
> #calculating the accuracy and confusion matrix of the test data
> test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")
> mean(test_predictions == df_test$heartattack_s)
[1] 0.7185741
> pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)
> pred_tab_test
         actual
predicted  No Yes
      No  732 284
      Yes 166 417
> 
```

```
> rec <- pred_tab[2,2]/sum(pred_tab[,2])
> rec
[1] 0.634355
> 
```

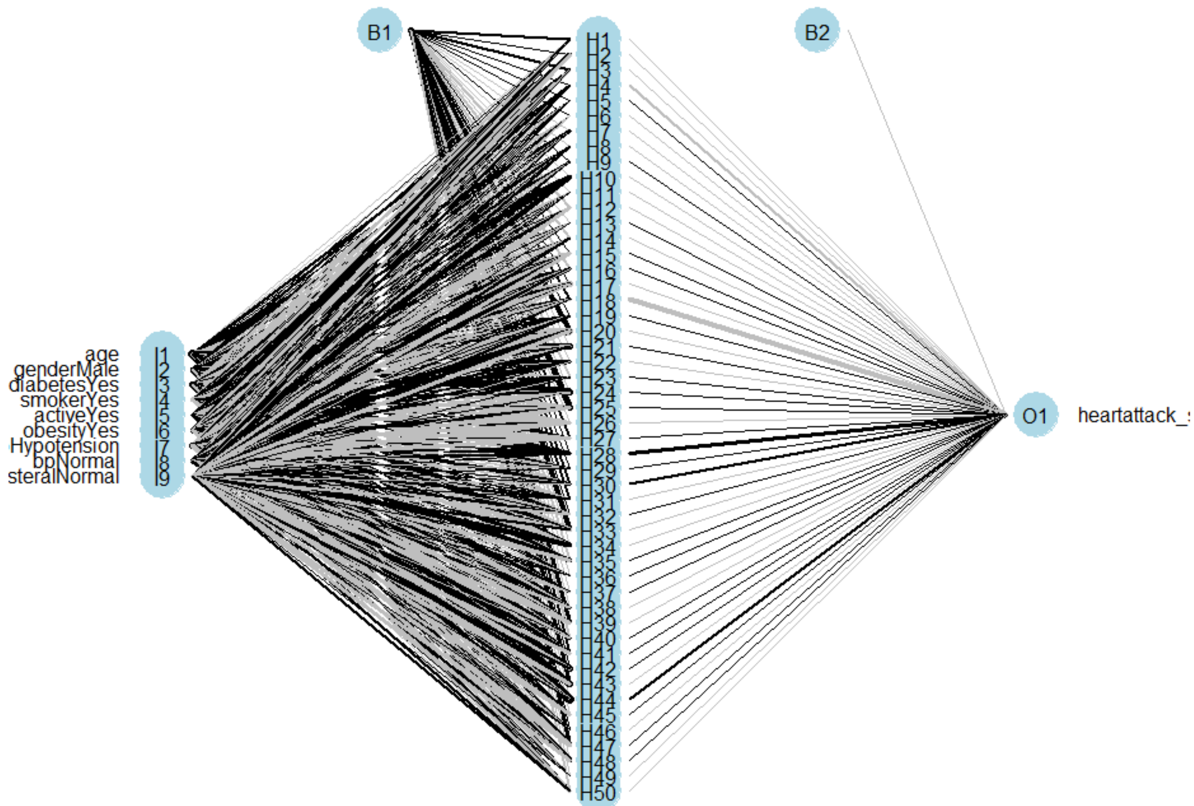
```
> pred_tab_test
         actual
predicted  No Yes
      No  732 284
      Yes 166 417
> rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])
> rec
[1] 0.5948645
```

As we can see that the train set gave an accuracy of 74.4 percent with recall as 63.44% which is much higher than the results observed in non-partitioned data. But let's check for the test set as they show the true results on unseen data.

From the above screenshots we can see the accuracy for test is 71.8% and the recall is about 59.4% which is still a significant increase from 50% recall score in non-partitioned data. Hence, we can say that partitioning does lead to a better recall score and almost similar accuracy.

To increase the efficiency of train and test set, and to try to increase the test recall we try to increase the number of hidden units in the hidden layer to 50.

```
##trying to increase recall and accuracy
model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 50, maxit = 1000)
plotnet(model1_partition)
#calculating the accuracy and confusion matrix of the train data
pred <- predict(model1_partition, df_train[,1:8], type = "class")
mean(pred == df_train$heartattack_s)

#calculating the accuracy and confusion matrix of the test data
test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")
mean(test_predictions == df_test$heartattack_s)
pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)
pred_tab_test
rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])
rec
```




```
iter 790 value 2892.153312
iter 800 value 2892.149553
final  value 2892.147770
converged
> plotnet(model1_partition)
> #calculating the accuracy and confusion matrix of the train data
> pred <- predict(model1_partition, df_train[,1:8], type = "class")
> mean(pred == df_train$heartattack_s)
[1] 0.7755899
>
> #calculating the accuracy and confusion matrix of the test data
> test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")
> mean(test_predictions == df_test$heartattack_s)
[1] 0.6979362
> pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)
> pred_tab_test
         actual
predicted  No Yes
      No  719 304
      Yes 179 397
> rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])
> rec
[1] 0.5663338
```

We can see that the model now converged at 800th iteration and the accuracy for both train and test increased by 2-3%. Whereas the recall for test dropped to 56.6%

Now we try building the same model with decay addition of 0.01

```
model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 50, maxit = 10000, decay = 0.01)
plotnet(model1_partition)
#calculating the accuracy and confusion matrix of the train data
pred <- predict(model1_partition, df_train[,1:8], type = "class")
mean(pred == df_train$heartattack_s)

#calculating the accuracy and confusion matrix of the test data
test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")
mean(test_predictions == df_test$heartattack_s)
pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)
pred_tab_test
rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])
rec
```

```
> model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 50, maxit = 10000, decay = 0.01)
# weights:  551
initial  value 4827.417164
iter  10 value 3457.510434
iter  20 value 3425.986405
iter  30 value 3408.423788
iter  40 value 3385.167187
iter  50 value 3366.617929
```

```
iter3810 value 3147.412085
iter3820 value 3147.403362
final  value 3147.399353
converged
> plotnet(model1_partition)
> #calculating the accuracy and confusion matrix of the train data
> pred <- predict(model1_partition, df_train[,1:8], type = "class")
> mean(pred == df_train$heartattack_s)
[1] 0.7657446
>
> #calculating the accuracy and confusion matrix of the test data
> test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")
> mean(test_predictions == df_test$heartattack_s)
[1] 0.7073171
> pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)
> pred_tab_test
         actual
predicted  No Yes
      No  726 296
      Yes 172 405
> rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])
> rec
[1] 0.5777461
```

This model took 3820 iterations to converge but showed a slightly better results in terms of test and train accuracy, also the recall increased by around 1%

Question 2) We will use the credit default dataset (default_of_credit_card_clients.xlsx) for this exercise. Here comes the data set information again:

Source: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Attribute Information:

This dataset has a binary variable, default payment (Yes = 1, No = 0), as the response variable. The following 23 variables as explanatory variables:

- X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- X2: Gender (1 = male; 2 = female).
- X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
- X4: Marital status (1 = married; 2 = single; 3 = others).
- X5: Age (year).
- X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.
- X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.

A) (40) Explore the data set, then use a Bayesian Network to model this classification problem (using partition 80:20).

```
###################### PART-2  ##################################################
library(e1071)
library("klaR")
library("caret")
library(magrittr)
library(dplyr)
library(readxl)

df_c <- read_excel("R:/downloads/default_of_credit_card_clients.xlsx", skip = 1)
view(df_c)
str(df_c)
summary(df_c)
#Removing id
df_c <- df_c[-1]
#replacing 4/4+ with 4 (others category) in education variable
df_c$EDUCATION[df_c$EDUCATION != c("1","2","3","4")] <- 4
#replacing 3/3+ with 3 (others category) in marriage variable
df_c$MARRIAGE[df_c$MARRIAGE != c("1","2","3")] <- 3
str(df_c)
```

First after checking the attribute information we can see that for attributes education and marriage, values 4 and above and, 3 and above are marked as other categories. Hence we change all the values above those values to 4 and 3 for each attribute respectively. Since we don't need ID, I removed it.

```
>
> df_c <- read_excel("R:/downloads/default_of_credit_card_clients.xlsx", skip = 1)
> View(df_c)
> str(df_c)
tibble [30,000 x 25] (S3: tbl_df/tbl/data.frame)
 $ ID                        : num [1:30000] 1 2 3 4 5 6 7 8 9 10 ...
 $ LIMIT_BAL                 : num [1:30000] 20000 120000 90000 50000 50000 50000 500000 100000
 $ SEX                       : num [1:30000] 2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION                 : num [1:30000] 2 2 2 2 2 1 1 2 3 3 ...
 $ MARRIAGE                  : num [1:30000] 1 2 2 1 1 2 2 2 1 2 ...
 $ AGE                       : num [1:30000] 24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0                     : num [1:30000] 2 -1 0 0 -1 0 0 0 0 -2 ...
 $ PAY_2                     : num [1:30000] 2 2 0 0 0 0 0 -1 0 -2 ...
 $ PAY_3                     : num [1:30000] -1 0 0 0 -1 0 0 -1 2 -2 ...
 $ PAY_4                     : num [1:30000] -1 0 0 0 0 0 0 0 0 -2 ...
 $ PAY_5                     : num [1:30000] -2 0 0 0 0 0 0 0 0 -1 ...
 $ PAY_6                     : num [1:30000] -2 2 0 0 0 0 0 -1 0 -1 ...
 $ BILL_AMT1                 : num [1:30000] 3913 2682 29239 46990 8617 ...
 $ BILL_AMT2                 : num [1:30000] 3102 1725 14027 48233 5670 ...
 $ BILL_AMT3                 : num [1:30000] 689 2682 13559 49291 35835 ...
 $ BILL_AMT4                 : num [1:30000] 0 3272 14331 28314 20940 ...
 $ BILL_AMT5                 : num [1:30000] 0 3455 14948 28959 19146 ...
 $ BILL_AMT6                 : num [1:30000] 0 3261 15549 29547 19131 ...
 $ PAY_AMT1                  : num [1:30000] 0 0 1518 2000 2000 ...
 $ PAY_AMT2                  : num [1:30000] 689 1000 1500 2019 36681 ...
 $ PAY_AMT3                  : num [1:30000] 0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4                  : num [1:30000] 0 1000 1000 1100 9000 ...
 $ PAY_AMT5                  : num [1:30000] 0 0 1000 1069 689 ...
 $ PAY_AMT6                  : num [1:30000] 0 2000 5000 1000 679 ...
 $ default payment next month: num [1:30000] 1 1 0 0 0 0 0 0 0 0 ...
```

$ default payment next month: num [1:30000] 1 1 0 0 0 0 0 0 0 0 ...

```
> summary(df_c)
       ID          LIMIT_BAL            SEX          EDUCATION        MARRIAGE          AGE       
 Min.   :    1   Min.   :  10000   Min.   :1.000   Min.   :0.000   Min.   :0.000   Min.   :21.00  
 1st Qu.: 7501   1st Qu.:  50000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:28.00  
 Median :15000   Median : 140000   Median :2.000   Median :2.000   Median :2.000   Median :34.00  
 Mean   :15000   Mean   : 167484   Mean   :1.604   Mean   :1.853   Mean   :1.552   Mean   :35.49  
 3rd Qu.:22500   3rd Qu.: 240000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:41.00  
 Max.   :30000   Max.   :1000000   Max.   :2.000   Max.   :6.000   Max.   :3.000   Max.   :79.00  
     PAY_0             PAY_2             PAY_3             PAY_4             PAY_5             PAY_6        
 Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000  
 1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000  
 Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000  
 Mean   :-0.0167   Mean   :-0.1338   Mean   :-0.1662   Mean   :-0.2207   Mean   :-0.2662   Mean   :-0.2911  
 3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000  
 Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000  
   BILL_AMT1         BILL_AMT2        BILL_AMT3         BILL_AMT4         BILL_AMT5        BILL_AMT6      
 Min.   :-165580   Min.   :-69777   Min.   :-157264   Min.   :-170000   Min.   :-81334   Min.   :-339603  
 1st Qu.:   3559   1st Qu.:  2985   1st Qu.:   2666   1st Qu.:   2327   1st Qu.:  1763   1st Qu.:   1256  
 Median :  22382   Median : 21200   Median :  20089   Median :  19052   Median : 18105   Median :  17071  
 Mean   :  51223   Mean   : 49179   Mean   :  47013   Mean   :  43263   Mean   : 40311   Mean   :  38872  
 3rd Qu.:  67091   3rd Qu.: 64006   3rd Qu.:  60165   3rd Qu.:  54506   3rd Qu.: 50191   3rd Qu.:  49198  
 Max.   : 964511   Max.   :983931   Max.   :1664089   Max.   : 891586   Max.   :927171   Max.   : 961664  
    PAY_AMT1         PAY_AMT2          PAY_AMT3         PAY_AMT4         PAY_AMT5           PAY_AMT6       
 Min.   :     0   Min.   :      0   Min.   :     0   Min.   :     0   Min.   :     0.0   Min.   :     0.0  
 1st Qu.:  1000   1st Qu.:    833   1st Qu.:   390   1st Qu.:   296   1st Qu.:   252.5   1st Qu.:   117.8  
 Median :  2100   Median :   2009   Median :  1800   Median :  1500   Median :  1500.0   Median :  1500.0  
 Mean   :  5664   Mean   :   5921   Mean   :  5226   Mean   :  4826   Mean   :  4799.4   Mean   :  5215.5  
 3rd Qu.:  5006   3rd Qu.:   5000   3rd Qu.:  4505   3rd Qu.:  4013   3rd Qu.:  4031.5   3rd Qu.:  4000.0  
 Max.   :873552   Max.   :1684259   Max.   :896040   Max.   :621000   Max.   :426529.0   Max.   :528666.0  
 default payment next month
 Min.   :0.0000            
 1st Qu.:0.0000            
 Median :0.0000            
 Mean   :0.2212            
 3rd Qu.:0.0000            
 Max.   :1.0000            
> #Removing id
```

```
> #Removing id
> df_c <- df_c[-1]
> #replacing 4/4+ with 4 (others category) in education variable
> df_c$EDUCATION[df_c$EDUCATION != c("1","2","3","4")] <- 4
> #replacing 3/3+ with 3 (others category) in marriage variable
> df_c$MARRIAGE[df_c$MARRIAGE != c("1","2","3")] <- 3
> str(df_c)
tibble [30,000 x 24] (S3: tbl_df/tbl/data.frame)
 $ LIMIT_BAL                 : num [1:30000] 20000 120000 90000 50000 50000 50000 500000 1(
 $ SEX                       : num [1:30000] 2 2 2 2 1 1 1 2 2 1 ...
 $ EDUCATION                 : num [1:30000] 4 2 4 4 4 4 4 4 4 4 ...
 $ MARRIAGE                  : num [1:30000] 1 2 3 1 3 3 3 2 3 3 ...
 $ AGE                       : num [1:30000] 24 26 34 37 57 37 29 23 28 35 ...
 $ PAY_0                     : num [1:30000] 2 -1 0 0 -1 0 0 0 0 -2 ...
 $ PAY_2                     : num [1:30000] 2 2 0 0 0 0 0 -1 0 -2 ...
 $ PAY_3                     : num [1:30000] -1 0 0 0 -1 0 0 -1 2 -2 ...
 $ PAY_4                     : num [1:30000] -1 0 0 0 0 0 0 0 0 -2 ...
 $ PAY_5                     : num [1:30000] -2 0 0 0 0 0 0 0 0 -1 ...
 $ PAY_6                     : num [1:30000] -2 2 0 0 0 0 0 -1 0 -1 ...
 $ BILL_AMT1                 : num [1:30000] 3913 2682 29239 46990 8617 ...
 $ BILL_AMT2                 : num [1:30000] 3102 1725 14027 48233 5670 ...
 $ BILL_AMT3                 : num [1:30000] 689 2682 13559 49291 35835 ...
 $ BILL_AMT4                 : num [1:30000] 0 3272 14331 28314 20940 ...
 $ BILL_AMT5                 : num [1:30000] 0 3455 14948 28959 19146 ...
 $ BILL_AMT6                 : num [1:30000] 0 3261 15549 29547 19131 ...
 $ PAY_AMT1                  : num [1:30000] 0 0 1518 2000 2000 ...
 $ PAY_AMT2                  : num [1:30000] 689 1000 1500 2019 36681 ...
 $ PAY_AMT3                  : num [1:30000] 0 1000 1000 1200 10000 657 38000 0 432 0 ...
 $ PAY_AMT4                  : num [1:30000] 0 1000 1000 1100 9000 ...
 $ PAY_AMT5                  : num [1:30000] 0 0 1000 1069 689 ...
 $ PAY_AMT6                  : num [1:30000] 0 2000 5000 1000 679 ...
 $ default payment next month: num [1:30000] 1 1 0 0 0 0 0 0 0 0 ...
```

Now we can see that the “default payment next month” attribute has spaces between the words of the attribute title, this will be tough to call with the data frame, hence I renamed it.

After which I changed all the categorical variables type from char to factors and got rid of the duplicate rows.

```
names(df_c)[names(df_c) == "default payment next month"] <- "defaultpayment"
#factor type convertitions
df_c$SEX <- as.factor(df_c$SEX)
df_c$EDUCATION <- as.factor(df_c$EDUCATION)
df_c$MARRIAGE <- as.factor(df_c$MARRIAGE)
df_c$PAY_0 <- as.factor(df_c$PAY_0)
df_c$PAY_2 <- as.factor(df_c$PAY_2)
df_c$PAY_3 <- as.factor(df_c$PAY_3)
df_c$PAY_4 <- as.factor(df_c$PAY_4)
df_c$PAY_5 <- as.factor(df_c$PAY_5)
df_c$PAY_6 <- as.factor(df_c$PAY_6)
df_c$default_payment_next_month <- as.factor(df_c$defaultpayment)
str(df_c)
#Checking and getting rid of duplicates
df_c <- df_c[!duplicated(df_c), ]
head(df_c)
```

```
#performing train-test split
set.seed(100)
trainIndex <- createDataPartition(df_c$defaultpayment, p=0.8, list=FALSE)
data_train <- df_c[ trainIndex,]
data_test <- df_c[-trainIndex,]
#upsampling the defaultpayments
data_train<-upSample(x names(x) rain[,-ncol(data_train)],y=data_train$defaultpayment)
table(data_train$Class)
names(data_train)[names(data_train) == "Class"] <- "defaultpayment"
```

```
> #performing train-test split
> set.seed(100)
> trainIndex <- createDataPartition(df_c$defaultpayment, p=0.8, list=FALSE)
> data_train <- df_c[ trainIndex,]
> data_test <- df_c[-trainIndex,]
> #upsampling the defaultpayments
> data_train<-upSample(x=data_train[,-ncol(data_train)],y=data_train$defaultpayment)
> table(data_train$Class)

    0     1 
18671 18671 
> names(data_train)[names(data_train) == "Class"] <- "defaultpayment"
> #performing train-test split
> set.seed(100)
> trainIndex <- createDataPartition(df_c$defaultpayment, p=0.8, list=FALSE)
> data_train <- df_c[ trainIndex,]
> data_test <- df_c[-trainIndex,]
> table(data_train$defaultpayment)

    0     1 
18671  5306 
> #upsampling the defaultpayments
> data_train<-upSample(x=data_train[,-ncol(data_train)],y=data_train$defaultpayment)
> names(data_train)[names(data_train) == "Class"] <- "defaultpayment"
> table(data_train$defaultpayment)

    0     1 
18671 18671 
> 
```

After splitting the data to 80% train and 20% test we up-sample the dependent variable (default payment) to deal with the data imbalance. In the screenshots attached we can see that the attribute is now balanced, and we can build the naïve bayes model on it.

```
# build naive bayes model
Naive_Bayes_Model=naiveBayes(defaultpayment ~., data=data_train)
Naive_Bayes_Model

#calculating the accuracy and confusion matrix of the train data
credit_predictions_train <- predict(Naive_Bayes_Model, data_train)
cred_table=table(credit_predictions_train,data_train$defaultpayment)
mean(credit_predictions_train == data_train$defaultpayment)
rec <- cred_table[2,2]/sum(cred_table[,2])

rec
#calculating the accuracy and confusion matrix of the test data
credit_predictions_test <- predict(Naive_Bayes_Model, data_test)
cred_table=table(credit_predictions_test,data_test$defaultpayment)
mean(credit_predictions_test == data_test$defaultpayment)
rec <- cred_table[2,2]/sum(cred_table[,2])
rec
```

```
> # build naive bayes model
> Naive_Bayes_Model=naiveBayes(defaultpayment ~., data=data_train)
> Naive_Bayes_Model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  0   1 
0.5 0.5 

Conditional probabilities:
   LIMIT_BAL
Y       [,1]     [,2]
  0 178851.5 131614.6
  1 131322.5 115399.3

   SEX
Y           1         2
  0 0.3863210 0.6136790
  1 0.4325424 0.5674576

   EDUCATION
Y            1          2          3          4
  0 0.09169300 0.11541963 0.03899095 0.75389642
  1 0.07680360 0.11857962 0.04697124 0.75764555

   MARRIAGE
Y           1         2         3
  0 0.1505543 0.1801189 0.6693268
  1 0.1664078 0.1633549 0.6702373

   AGE
Y       [,1]     [,2]
  0 35.44202 9.084620
  1 35.82052 9.718418

   PAY_0
Y             -2           -1            0            1            2            3            4            5
  0 0.1016549730 0.2030957099 0.5508542660 0.1023512399 0.0364201168 0.0032670987 0.0010711799 0.0005891489
  1 0.0565047400 0.1456269080 0.2799528681 0.1894381661 0.2784532162 0.0360987628 0.0086765572 0.0019281238
   PAY_0
Y              6            7            8
  0 0.0002677950 0.0001071180 0.0003213540
  1 0.0007498259 0.0008569439 0.0017138878

   PAY_2
Y             -2           -1            0            1            2            3            4            5
  0 0.1305768304 0.2184671416 0.5670290825 0.0010176209 0.0746612394 0.0051952225 0.0021423598 0.0004284720
  1 0.1060468106 0.1432703122 0.3697177441 0.0007498259 0.3377430239 0.0297788013 0.0073375823 0.0023030368
   PAY_2
Y              6            7            8
  0 0.0001071180 0.0003213540 0.0000535590
  1 0.0013389749 0.0017138878 0.0000000000

   PAY_3
Y            -2          -1           0           1           2           3           4           5
  0 0.140592362 0.216003428 0.556959991 0.000160677 0.079695785 0.004284720 0.001231857 0.000374913
  1 0.116383697 0.139414065 0.407691072 0.000107118 0.300091050 0.022226983 0.007176905 0.001606770
   PAY_3
Y             6           7           8
```

```
   PAY_4
Y             6            7            8
  0 0.0001606770 0.0003749130 0.0000000000
  1 0.0004820310 0.0070697874 0.0001606770

   PAY_5
Y           -2          -1           0           2           3           4           5           6
  0 0.154464142 0.198864549 0.589738097 0.052005784 0.002838627 0.001338975 0.000374913 0.000000000
  1 0.137753736 0.135772053 0.477960473 0.213593273 0.017460232 0.008194526 0.001338975 0.000535590
   PAY_5
Y            7           8
  0 0.000374913 0.000000000
  1 0.007230464 0.000160677

   PAY_6
Y            -2           -1            0            2            3            4            5            6
  0 0.1662471212 0.2042204488 0.5663863746 0.0588077768 0.0026243908 0.0009105029 0.0002677950 0.0002142360
  1 0.1500723046 0.1451984361 0.4575009373 0.2153071608 0.0183707354 0.0043918376 0.0010176209 0.0021959188
   PAY_6
Y             7            8
  0 0.0003213540 0.0000000000
  1 0.0055701355 0.0003749130

   BILL_AMT1
Y       [,1]     [,2]
  0 52037.97 73511.90
  1 49288.44 74660.62

   BILL_AMT2
Y       [,1]    [,2]
  0 49805.73 71127.5
  1 47862.78 72341.0

   BILL_AMT3
Y       [,1]     [,2]
  0 47647.97 69091.96
  1 45894.89 69132.21

   BILL_AMT4
Y       [,1]     [,2]
  0 43935.88 64795.22
  1 42994.71 65283.48

   BILL_AMT5
Y       [,1]     [,2]
  0 40812.86 60970.00
  1 40572.46 62700.01

   BILL_AMT6
Y       [,1]     [,2]
  0 39351.96 60068.40
  1 39391.83 61170.92

   PAY_AMT1
Y       [,1]      [,2]
  0 6357.333 18671.654
  1 3457.572  9679.364
```

```
    PAY_AMT1
Y         [,1]      [,2]
  0 6357.333 18671.654
  1 3457.572  9679.364

    PAY_AMT2
Y         [,1]     [,2]
  0 6686.972 24103.68
  1 3460.547 11817.13

    PAY_AMT3
Y         [,1]     [,2]
  0 5882.108 19823.22
  1 3456.582 14494.61

    PAY_AMT4
Y         [,1]     [,2]
  0 5325.597 16515.04
  1 3188.234 11793.51

    PAY_AMT5
Y         [,1]     [,2]
  0 5241.644 16026.58
  1 3188.100 12313.99

    PAY_AMT6
Y         [,1]     [,2]
  0 5722.014 18691.36
  1 3385.171 13086.82

>
> #calculating the accuracy and confusion matrix of the train data
> credit_predictions_train <- predict(Naive_Bayes_Model, data_train)
> cred_table=table(credit_predictions_train,data_train$defaultpayment)
> mean(credit_predictions_train == data_train$defaultpayment)
[1] 0.5925232
> rec <- cred_table[2,2]/sum(cred_table[,2])
>
> rec
[1] 0.8912752
> #calculating the accuracy and confusion matrix of the test data
> credit_predictions_test <- predict(Naive_Bayes_Model, data_test)
> cred_table=table(credit_predictions_test,data_test$defaultpayment)
> mean(credit_predictions_test == data_test$defaultpayment)
[1] 0.4276656
> rec <- cred_table[2,2]/sum(cred_table[,2])
> rec
[1] 0.907994
>
```

Finally with naïve Bayes model, we can see that the train accuracy is 59% and 42.7% for test accuracy, which isn't that great. The recall score is good as its around 90% for both (which indicates low false negative cases on both balanced train set and imbalanced test set).

Appendix (The complete version of your solution scripts in R)

```
# Installing and loading all the libraries
#Using the pre-processing and EDA steps similar to previous assignemnt
library(polycor)
library(readxl)
library("readxl")
library('C50')
library(rpart)
library("nnet")
library(devtools)
library(reshape)
library(caret)
library(rattle)
library(psych)#categorical correlation
#install.packages('NeuralNetTools')
library(NeuralNetTools)
#install.packages('neuralnet')
library(neuralnet)

df <- read_excel("R://downloads//Patient_Data.xlsx")
#now we inspect data to see each variable
str(df)
#since variable type is char, we change it to factor for all the applicable vairables
df[sapply(df, is.character)] <- lapply(df[sapply(df, is.character)],as.factor)
```

```
#lets check if they are converted to factors
str(df)
#Lets summarize our data
summary(df)

# to check correlation between each binary categorical data
cc=table(df$diabetes,df$heartattack_s)
tetrachoric(cc)
cc=table(df$gender,df$heartattack_s)
tetrachoric(cc)
cc=table(df$smoker,df$heartattack_s)
tetrachoric(cc)
cc=table(df$active,df$heartattack_s)
tetrachoric(cc)
cc=table(df$obesity,df$heartattack_s)
tetrachoric(cc)
cc=table(df$cholesteral,df$heartattack_s)
tetrachoric(cc)

#EDA
plot(df$heartattack_s,df$age, xlab="heart attack", ylab="patient age")
plot(df$heartattack_s,df$active, xlab="heart attack", ylab="active")
plot(df$heartattack_s,df$smoker, xlab="heart attack", ylab="smoker")
plot(df$heartattack_s,df$gender, xlab="heart attack", ylab="gender")

#outliers in age variable
boxplot(df$age, ylab="age")
```

```
#skewness of age variable
hist(df$age,ylab="count",xlab="age")

#checking if the data is balanced
barplot(table(df$heartattack_s),ylab="count",xlab="heart attack")

#to check if there are any duplicate values
#duplicated(df)
sum(duplicated(df))
nrow(df)
df2 =unique(df)
nrow(df2)

#check for missing values
sum(is.na(df))

##############################################################################
############################
h <- preProcess(df[1], method = c("range"))
df3 <- cbind(predict(h, df[,-7]),heartattack_s = df$heartattack_s)
#df3
#building NN with iterations as 1000 and 10 units in hidden layer
model1 <- nnet(heartattack_s ~ ., data = df3, size = 10, maxit = 1000)
plotnet(model1)
View(model1)
#now we calculate the accuracy
pred <- predict(model1, df3[,1:8], type = "class")
```

```
mean(pred == df3$heartattack_s)

pred_table = table(predicted = pred, actual = df$heartattack_s)

rec <- pred_tab[2,2]/sum(pred_table[,2])

rec


###############################################################################
#############################


#Building the c5 model

model2 <- C5.0(heartattack_s ~ ., data=df)

summary(model2)

plot(model2, type = "simple" , cex =0.7 , main = 'heart attack diagnosis decision tree')

#calculating the accuracy of our c5 moodel

acc <- predict(model2, df, type = "class")

mean(acc == df$heartattack_s)

pred_table2 = table(predicted = acc, actual = df$heartattack_s)

pred_table2

rec <- pred_table2[2,2]/sum(pred_table2[,2])

rec

#######f#######################################################################


#Buliding NN with partition

set.seed(100)

sampl <- createDataPartition(df$heartattack_s, p = 0.80, list = FALSE)

df_train <- df[sampl,]

View(df_train)

df_test <- df[-sampl,]

View(df_test)
```

```
heart_partition <- preProcess(df_train[1], method = c("range"))
df_train <- cbind(predict(heart_partition, df_train[-7]),heartattack_s = df_train$heartattack_s)
df_test  <- cbind(predict(heart_partition, df_test[-7]), heartattack_s = df_test$heartattack_s)

model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 10, maxit = 1000)
plotnet(model1_partition)
#calculating the accuracy and confusion matrix of the train data
pred <- predict(model1_partition, df_train[,1:8], type = "class")
mean(pred == df_train$heartattack_s)

#calculating the accuracy and confusion matrix of the test data
test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")
mean(test_predictions == df_test$heartattack_s)
pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)
pred_tab_test
rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])
rec


##trying to increase recall and accuracy
model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 50, maxit = 1000)
plotnet(model1_partition)
#calculating the accuracy and confusion matrix of the train data
pred <- predict(model1_partition, df_train[,1:8], type = "class")
mean(pred == df_train$heartattack_s)

#calculating the accuracy and confusion matrix of the test data
```

```
test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")

mean(test_predictions == df_test$heartattack_s)

pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)

pred_tab_test

rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])

rec


model1_partition <- nnet(heartattack_s ~ ., data = df_train, size = 50, maxit = 10000, decay =
0.01)

plotnet(model1_partition)

#calculating the accuracy and confusion matrix of the train data

pred <- predict(model1_partition, df_train[,1:8], type = "class")

mean(pred == df_train$heartattack_s)


#calculating the accuracy and confusion matrix of the test data

test_predictions <- predict(model1_partition, df_test[,1:8], type = "class")

mean(test_predictions == df_test$heartattack_s)

pred_tab_test = table(predicted = test_predictions, actual = df_test$heartattack_s)

pred_tab_test

rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])

rec




##################### PART-2
###################################################################

library(e1071)

library("klaR")
```

```
library("caret")

library(magrittr)

library(dplyr)

library(readxl)


df_c <- read_excel("R:/downloads/default_of_credit_card_clients.xlsx", skip = 1)

str(df_c)

summary(df_c)

head(df_c)

#Removing id

df_c <- df_c[-1]

#replacing 4/4+ with 4 (others category) in education variable

df_c$EDUCATION[df_c$EDUCATION != c("1","2","3","4")] <- 4

#replacing 3/3+ with 3 (others category) in marriage variable

df_c$MARRIAGE[df_c$MARRIAGE != c("1","2","3")] <- 3

str(df_c)


names(df_c)[names(df_c) == "default payment next month"] <- "defaultpayment"

#factor type convertitions

df_c$SEX <- as.factor(df_c$SEX)

df_c$EDUCATION <- as.factor(df_c$EDUCATION)

df_c$MARRIAGE <- as.factor(df_c$MARRIAGE)

df_c$PAY_0 <- as.factor(df_c$PAY_0)

df_c$PAY_2 <- as.factor(df_c$PAY_2)

df_c$PAY_3 <- as.factor(df_c$PAY_3)

df_c$PAY_4 <- as.factor(df_c$PAY_4)

df_c$PAY_5 <- as.factor(df_c$PAY_5)
```

```
df_c$PAY_6 <- as.factor(df_c$PAY_6)

df_c$default_payment_next_month <- as.factor(df_c$defaultpayment)

str(df_c)

#Checking and getting rid of duplicates

df_c <- df_c[!duplicated(df_c), ]

head(df_c)



#performing train-test split

set.seed(100)

trainIndex <- createDataPartition(df_c$default_payment_next_month, p=0.8, list=FALSE)

data_train <- df_c[ trainIndex,]

data_test <- df_c[-trainIndex,]

# build naive bayes model

Naive_Bayes_Model=naiveBayes(default_payment_next_month ~., data=data_train)

Naive_Bayes_Model


#calculating the accuracy and confusion matrix of the train data

credit_predictions_train <- predict(Naive_Bayes_Model, data_train)

table(credit_predictions_train,data_train$default_payment_next_month)

mean(credit_predictions_train == data_train$default_payment_next_month)

rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])

#calculating the accuracy and confusion matrix of the test data

credit_predictions_test <- predict(Naive_Bayes_Model, data_test)

table(credit_predictions_test == data_test$default_payment_next_month)

mean(credit_predictions_test == data_test$default_payment_next_month)

rec <- pred_tab_test[2,2]/sum(pred_tab_test[,2])
```