# Experiment No. 7

**Title:** To Implement Decision Tree Classifier in R

**Problem**

To build a decision tree, we will proceed with the steps as follow:

1) Import the data and Shuffle the data.
   path <- ("E:/SPIT/R_Practicals/Lab_07/titanic.csv") #Path of data set
   titanic <- read.csv(path) #importing the data set into titanic variable
   shuffle_index <- sample(1:nrow(titanic)) #creating shuffling index to shuffle rows depending on the
   titanic <- titanic[shuffle_index, ] #shuffling the data set based on shuffle index

```
> source('E:/SPIT/R_Practicals/Lab_07/DecisionTree.R', echo=TRUE)

> path <- ("E:/SPIT/R_Practicals/Lab_07/titanic.csv") #Path of data set

> titanic <- read.csv(path) #importing the data set into titanic variable

> head(titanic) #printing head of the data set
  x pclass survived
1 1      1        1
2 2      1        1
3 3      1        0
4 4      1        0
5 5      1        0
6 6      1        1
                                            name    sex
1                    Allen, Miss. Elisabeth Walton female
2                   Allison, Master. Hudson Trevor   male
3                     Allison, Miss. Helen Loraine female
4             Allison, Mr. Hudson Joshua Creighton   male
5 Allison, Mrs. Hudson J C (Bessie Waldo Daniels) female
6                               Anderson, Mr. Harry   male
     age sibsp parch ticket     fare   cabin embarked
1     29     0     0  24160 211.3375      B5        S
2 0.9167     1     2 113781   151.55 C22 C26        S
3      2     1     2 113781   151.55 C22 C26        S
4     30     1     2 113781   151.55 C22 C26        S
5     25     1     2 113781   151.55 C22 C26        S
6     48     0     0  19952    26.55     E12        S
                        home.dest
1                     St Louis, MO
```

```
> shuffle_index <- sample(1:nrow(titanic)) #creating shuffling index to shuffle rows depending on the

> head(shuffle_index)
[1]  757 1242   44  181  959  746

> titanic <- titanic[shuffle_index, ] #shuffling the data set based on shuffle index

> head(titanic)
        x pclass survived
757   757      3        0
1242 1242      3        0
44     44      1        1
181   181      1        1
959   959      3        0
746   746      3        1
                                              name    sex
757                          Davison, Mr. Thomas Henry   male
1242                            Thomas, Mr. Charles P   male
44   Bucknell, Mrs. William Robert (Emma Eliza Ward) female
181                           Kreuchen, Miss. Emilie female
959                     Lefebre, Mrs. Frank (Frances) female
746            Daly, Miss. Margaret Marcella 'Maggie' female
     age sibsp parch ticket     fare cabin embarked
757    ?     1     0 386525     16.1     ?        S
1242   ?     1     0   2621   6.4375     ?        C
44    60     0     0  11813  76.2917   D15        C
181   39     0     0  24160 211.3375     ?        S
959    ?     0     4   4133  25.4667     ?        S
```

2) Clean the dataset

The structure of the data shows some variables have NA's. Data clean up to be done as follows:

Drop variables home. dest, cabin, name, X, and ticket

- Create factor variables for pclass (Upper, Middle, Lower) and survived (Yes, No)
- Drop the NA(na.omit())

```
require(dplyr)    # alternatively, this also loads %>%
# Drop variables
clean_titanic <- titanic %>%
  select(-c(home.dest, cabin, name, x, ticket)) %>%
  #Convert to factor level
  mutate(pclass = factor(pclass, levels = c(1, 2, 3), labels = c('Upper',
'Middle', 'Lower')),
         survived = factor(survived, levels = c(0, 1), labels = c('No',
'Yes'))) %>%
  na.omit()
glimpse(clean_titanic)
```

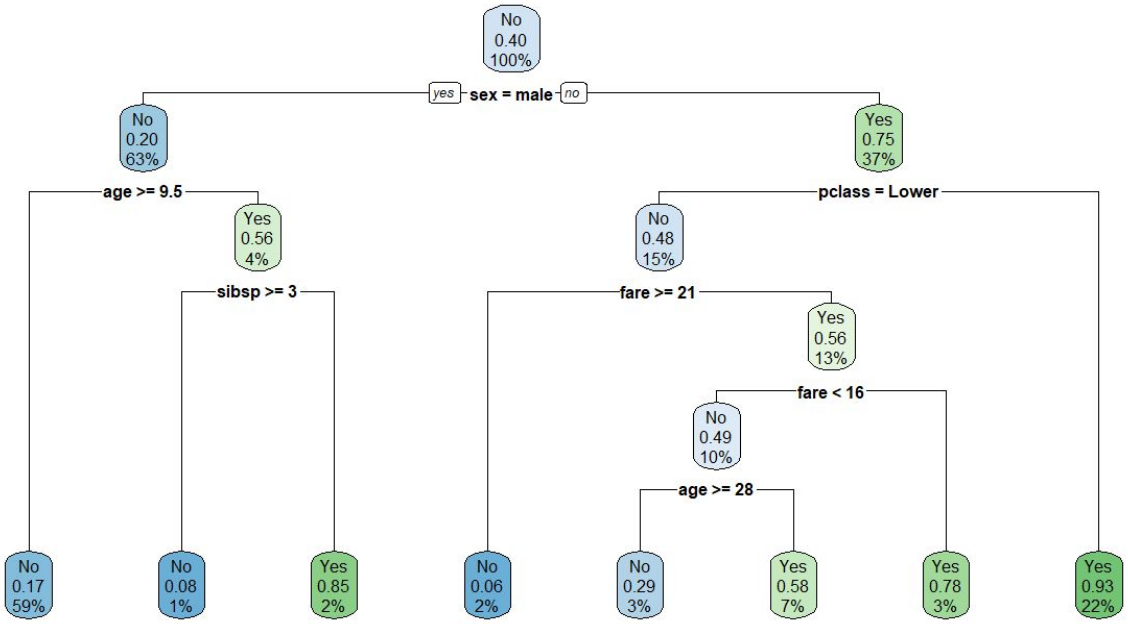
```
> require(dplyr)    # alternatively, this also loads %>%

> # Drop variables
> clean_titanic <- titanic %>%
+   select(-c(home.dest, cabin, name, x, ticket)) %>%
+   #Convert to factor level
+   mutate(pcla .... [TRUNCATED]

> glimpse(clean_titanic)
Rows: 1,045
Columns: 8
$ pclass   <fct> Upper, Upper, Lower, Lower, Lower, Upper...
$ survived <fct> Yes, Yes, Yes, Yes, No, Yes, Yes, Yes, Y...
$ sex      <chr> "female", "female", "female", "female", ...
$ age      <dbl> 60.0, 39.0, 30.0, 31.0, 9.0, 24.0, 40.0,...
$ sibsp    <int> 0, 0, 0, 1, 4, 0, 0, 0, 0, 0, 0, 1, 0, 0...
$ parch    <int> 0, 0, 0, 1, 2, 0, 0, 1, 0, 1, 0, 2, 0, 0...
$ fare     <dbl> 76.2917, 211.3375, 6.9500, 20.5250, 31.2...
$ embarked <chr> "C", "S", "Q", "S", "S", "C", "C", "C", ...
```

3) Create train/test set

```
create_train_test <- function(data, size, train) {
 n_row = nrow(data)
 total_row = size * n_row
 train_sample <- 1: total_row
 if (train == TRUE) {
  return (data[train_sample, ])
 } else {
  return (data[-train_sample, ])
 }
}

data_train <- create_train_test(clean_titanic, 0.8, train = TRUE)
data_test <- create_train_test(clean_titanic, 0.8, train = FALSE)
dim(data_train)
dim(data_test)
```

```
> create_train_test <- function(data, size, train) {
+   n_row = nrow(data)
+   total_row = size * n_row
+   train_sample <- 1: total_row
+   if (trai .... [TRUNCATED]

> data_train <- create_train_test(clean_titanic, 0.8, train = TRUE)

> data_test <- create_train_test(clean_titanic, 0.8, train = FALSE)

> dim(data_train)
[1] 836   8

> dim(data_test)
[1] 209   8

> prop.table(table(data_train$survived))

       No       Yes
0.5849282 0.4150718

> prop.table(table(data_test$survived))

       No       Yes
0.6172249 0.3827751
```

4) Build the model
   require(rpart)
   require(rpart.plot)
   fit <- rpart(survived~., data = data_train, method = 'class')
   rpart.plot(fit, extra = 106)




5) Make prediction

You can predict your test dataset. To make a prediction, you can use the predict() function:

predict_unseen <-predict(fit, data_test, type = 'class')

predict(fit, data_test, type = 'class')# Predict the class (0/1) of the test set

```
> predict_unseen <-predict(fit, data_test, type = 'class')

> predict(fit, data_test, type = 'class')# Predict the class (0/1) of the test set
1046 1047 1048 1049 1050 1051 1052 1056 1057
 Yes   No   No   No   No   No   No  Yes  Yes
1058 1059 1060 1061 1062 1063 1064 1065 1066
 Yes   No   No   No   No  Yes   No   No   No
1067 1068 1069 1070 1071 1073 1074 1075 1076
  No  Yes   No  Yes  Yes   No  Yes   No   No
1077 1078 1080 1081 1082 1083 1085 1089 1090
 Yes  Yes  Yes   No   No   No   No  Yes   No
1091 1092 1093 1094 1095 1097 1098 1099 1100
  No  Yes  Yes   No   No  Yes   No  Yes  Yes
1101 1102 1103 1104 1105 1106 1107 1108 1109
 Yes   No   No  Yes   No  Yes  Yes  Yes   No
1110 1111 1112 1113 1115 1116 1118 1119 1120
 Yes  Yes   No   No  Yes   No   No   No   No
1121 1123 1125 1126 1127 1128 1129 1130 1132
  No  Yes   No   No  Yes   No   No   No   No
1133 1134 1136 1137 1138 1139 1140 1141 1142
  No  Yes  Yes   No   No   No  Yes   No   No
1145 1146 1147 1148 1150 1151 1152 1154 1157
 Yes  Yes   No   No   No   No   No  Yes   No
1158 1160 1161 1162 1164 1165 1166 1167 1168
  No   No   No   No   No  Yes   No   No   No
```

6) Measure performance
You can compute an accuracy measure for classification task with the **confusion matrix**:

The **confusion matrix** is a better choice to evaluate the classification performance. The general idea is to count the number of times True instances are classified are False.

table_mat <- table(data_test$survived, predict_unseen)
table_mat

```
> table_mat <- table(data_test$survived, predict_unseen)

> table_mat
     predict_unseen
       No Yes
  No  109  11
  Yes  29  60
```