```
> #Titanic Dataset with  1310 obs. of  14 variables:
> rm(list=ls())
> ls()
character(0)
> data=read.csv("titanic_data.csv",header=TRUE,stringsAsFactors = T) #Loading data
> nrow(data)
[1] 1310
> str(data)
'data.frame':   1310 obs. of  14 variables:
 $ pclass   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ survived : int  1 1 0 0 0 1 1 0 1 0 ...
 $ name     : Factor w/ 1308 levels "","Abbing, Mr. Anthony",..: 23 25 26 27 28 32
47 48 52 56 ...
 $ sex      : Factor w/ 3 levels "","female","male": 2 3 2 3 2 3 2 3 2 3 ...
 $ age      : num  29 0.917 2 30 25 ...
 $ sibsp    : int  0 1 1 1 1 0 1 0 2 0 ...
 $ parch    : int  0 2 2 2 2 0 0 0 0 0 ...
 $ ticket   : Factor w/ 930 levels "","110152","110413",..: 189 51 51 51 51 126 94
17 78 827 ...
 $ fare     : num  211 152 152 152 152 ...
 $ cabin    : Factor w/ 187 levels "","A10","A11",..: 45 81 81 81 81 151 147 17 63
1 ...
 $ embarked : Factor w/ 4 levels "","C","Q","S": 4 4 4 4 4 4 4 4 4 2 ...
 $ boat     : Factor w/ 28 levels "","1","10","11",..: 13 4 1 1 1 14 3 1 28 1 ...
 $ body     : int  NA NA NA 135 NA NA NA NA NA 22 ...
 $ home.dest: Factor w/ 370 levels "","?Havana, Cuba",..: 310 232 232 232 232 238
163 25 23 230 ...
> sum(is.na(data))                         #total 1459 NA values
[1] 1459
> #I will use CARET package for preprocessing of data:
> library(caret)
> preprocvalues=preProcess(data,method=c("medianImpute","center","scale")) #taking
median for all NA with respective variables & adjusting scale
> library(RANN)
> data_pro=predict(preprocvalues,data)
> sum(is.na(data_pro))                         #total 0 NA values
[1] 0
> dv=dummyVars("~.",data_pro,fullRank = T)      # creating dummy variable to handl
e factors
> data_tran=data.frame(predict(dv,data_pro))
> data_tran$survived=as.factor(data_tran$survived)  # converting response variable
in factor
> set.seed(5)
> index <- createDataPartition(data_tran$survived, p=0.75, list=FALSE) #data parti
tion
> train <- data_tran[ index,]        #Traning data=75%
> test<- data_tran[-index,]          #Test data=25%
```

```
> ##############################################Decision Tree######################
#######
> set.seed(3)
> library(rpart)
> m=rpart(survived~.,data=train,method="class",control=rpart.control(minsplit=20,
+                               minbucket=7,maxdepth=10,usesurrogate = 2,xval=10))#
pre-proned method
> library(rattle)
> library(rpart.plot)
> library(RColorBrewer)
> fancyRpartPlot(m)
> printcp(m)

Classification tree:
rpart(formula = survived ~ ., data = train, method = "class",
    control = rpart.control(minsplit = 20, minbucket = 7, maxdepth = 10,
        usesurrogate = 2, xval = 10))

Variables actually used in tree construction:
 [1] age        boat.13    boat.15    boat.16    boat.3     boat.5     boat.7
 [8] boat.A     boat.C     pclass     sex.female sibsp

Root node error: 375/983 = 0.38149

n= 983

        CP nsplit rel error  xerror     xstd
1 0.458667      0   1.00000 1.00000 0.040612
2 0.045333      1   0.54133 0.54400 0.033906
3 0.032000      2   0.49600 0.49867 0.032815
4 0.024000      3   0.46400 0.44267 0.031323
5 0.021333      5   0.41600 0.34933 0.028415
6 0.020000      7   0.37333 0.31733 0.027272
7 0.014667     12   0.22933 0.29333 0.026357
8 0.013333     14   0.20000 0.26667 0.025274
9 0.010000     15   0.18667 0.26133 0.025048
> bestcp=m$cptable[which.min(m$cptable[,"xerror"]),"CP"]
> bestcp                            #Evaluting best cp
[1] 0.01
> pruned=prune(m,cp=bestcp)
> fancyRpartPlot(pruned)
> t=table(train$survived,predict(pruned,type="class"))
> prop.table(table(train$survived,predict(pruned,type="class")))

                     -0.785859287383634 1.27152032698672
  -0.785859287383634         0.59816887       0.02034588
  1.27152032698672           0.05086470       0.33062055
> rownames(t)=paste("Actual",rownames(t),sep=":")
```

```
> colnames(t)=paste("predicted",colnames(t),sep=":")
> t

                            predicted:-0.785859287383634 predicted:1.2715203269867
2
  Actual:-0.785859287383634                          588                         2
0
  Actual:1.27152032698672                             50                        32
5
> prop.table(t)

                            predicted:-0.785859287383634 predicted:1.2715203269867
2
  Actual:-0.785859287383634                   0.59816887                 0.0203458
8
  Actual:1.27152032698672                     0.05086470                 0.3306205
5
> accuracy=sum(diag(t))/sum(t)
> accuracy                           ###Accuracy on traning data=0.9287894
[1] 0.9287894
> t=predict(m,test,type="class")
> s=prop.table(table(t,test$survived))
> s

t                    -0.785859287383634 1.27152032698672
  -0.785859287383634         0.59633028       0.08868502
  1.27152032698672           0.02140673       0.29357798
> accuracy=sum(diag(s))/sum(s)
> accuracy                           ### Accuracy on test data=0.8899083
[1] 0.8899083
> #####################      ROC         #######################
> for_auc=predict(pruned,test,type="prob")
> library(pROC)
> a=auc(test$survived,for_auc[,2])
> a                                          #Area under the curve: 0.8977
Area under the curve: 0.8977
> #Ex:90-100,Good:80-90,fair:70-80,poor:60-70,Fail:50-60
> plot(roc(test$survived,for_auc[,2]),main="Decsion Tree")
> gini_coeff=2*a-1
> gini_coeff                            # Gini Coeff=0.7954851
[1] 0.7954851
> ######################################## Random Forest #########################
###
> library(randomForest)
> set.seed(7)
> rf=randomForest(survived~.,train,ntree=60)  ## wait for few seconds
> #importance(rf)
> varImpPlot(rf)
```

```
> q=predict(rf,test,type="prob")
> library(pROC)
> x=auc(test$survived,q[,2])
> x                               ##Area under the curve: 0.95
Area under the curve: 0.9539
> plot(roc(test$survived,q[,2]))
> gini_coeff=2*x-1
> gini_coeff                          # # Gini Coeff=0.90
[1] 0.907802
> ################################################################################
###
```