Evaluation of Classification Methods

```r
library(ISLR)
```

Model to evaluate

```
## Warning: package 'ISLR' was built under R version 4.0.2
```

```r
data("Carseats")
attach(Carseats)
#create new categorical variables "High"
High <- ifelse(Sales >= 8, "YES", "NO" )
High <- as.factor(High)

#Attach new variable to df & remove 1st column (Sales) of df
Carseats <- data.frame(Carseats, High)
Carseats <- Carseats[-1]

#Divide data into train and test
set.seed(3)
indx <- sample(2, nrow(Carseats), replace=T, prob= c(0.7, 0.3))
train <- Carseats[indx == 1, ]
test <- Carseats[indx ==2, ]

library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.2
```

```r
#TRAIN
tree_model <- rpart(High ~ . , data=train)
#TEST
pred_class <- predict(tree_model, test, type = "class")
actual <- test$High
```

```r
table(pred_class, actual , dnn = c("Predictions", "Actual"))
```

Confusion Matrix

```
##            Actual
## Predictions NO YES
##         NO  62  19
##         YES 13  29
```

```r
accuracy<- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  accuracy <- (y["TN"] + y["TP"])/ sum(y)
  return(as.numeric(accuracy))
}
 accuracy(actual, pred_class)
```

Accurary Function

```
## [1] 0.7398374
```

```
#percent of misclassified records out of the total number of records in the data.
error_rate <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  error_rate <- (y["FP"] + y["FN"])/ sum(y)
  return(as.numeric(error_rate))
}

 error_rate(actual, pred_class)
```

Error Rate Function

```
## [1] 0.2601626
```

```
#out of the instances that are actually in + class, how many of them are predicted in +
recall <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  recall <- (y["TP"] /  (y["TP"]+ y["FN"]))
  return(as.numeric(recall))
}

 recall(actual, pred_class)
```

Recall Function /TR rate/Hit Rate/Sensitivity

```
## [1] 0.6041667
```

```
#% of positive predictions that are correct
precision <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  precision <- (y["TP"] / (y["TP"]+ y["FP"]))
  return(as.numeric(precision))
}

 precision(actual, pred_class)
```

Precision Function

```
## [1] 0.6904762
```

```
# (2*Recall*Precision) / (Recall+Precision)
#combination of recall and precision, make a balance between these 2 measures.
#we want a HIGH F-Score

f_score <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  f_score <- ((2* (y["TP"]/(y["TP"]+ y["FN"])) * (y["TP"]/(y["TP"]+ y["FP"]))) / ((y["TP"]/(y["TP"]+ y[
  return(as.numeric(f_score))
}

 f_score(actual, pred_class)
```

F-Score Function

```
## [1] 0.6444444
```

```
# out of the instances that are actually in - class, how many of them are predicted in +

false_alarm <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  false_alarm <- (y["FP"]/(y["TN"]+ y["FP"]))
  return(as.numeric(false_alarm))
}

 false_alarm(actual, pred_class)
```

False Alarm Function / FP Rate

```
## [1] 0.1733333
```

```
# % of negative predictions that are correct--- (P[predicted=neg | actual = neg]

false_alarm <- function(actual,predictions)
{
  y <- as.vector(table(predictions,actual))
  names(y) <- c("TN","FP","FN","TP")
  false_alarm <- (y["TN"]/(y["TN"]+ y["FP"]))
  return(as.numeric(false_alarm))
}

 false_alarm(actual, pred_class)
```

Specificity Rate

```
## [1] 0.8266667
```