```
# load packages into R
library(caret)
library(ggplot2)
library(ROCR)
library(rpart)
library(rpart.plot)
library(splitstackshape)
library(yardstick)
# loading data into R
cs <- read.csv("C:/Users/Sneha R/OneDrive/Desktop/required_data.csv")
# get summary of all variables
summary(cs)
# plot graphs to understand different variables
ggplot(data = cs, aes(x = `ClientSatisfaction`, y = `ComplainPriorityID`, color =
`ClientSatisfaction`))+
  geom_jitter() +
  theme_classic()
# convert into numeric function
cs$ID <- as.numeric(cs$ID)
cs$ClientSatisfaction <- as.numeric(cs$ClientSatisfaction)
# split data into train and test model
train_index <- sample(1:nrow(cs), size = 0.8*nrow(cs))
# create train data set using train index
train_data <- cs[train_index, ]
# create test data set using train index
test_data <- cs[-train_index, ]
# build logistic regression using glm function
cs_logit <- glm(ClientSatisfaction ~ StateID+ ResolveDays+
                  ComplainPriorityID+ ComplainTypeID+
                  ComplainCategoryID + ComplainStatusID +
                  ComplainSourceID +ExpectedReimbursement,
```

```
            data = train_data,
            family = "binomial")
summary(cs_logit)
# run logistic regression with variables that are statistically significant only
newcs_logit <-  glm(ClientSatisfaction ~ResolveDays + ComplainPriorityID +
                  ComplainTypeID + ComplainSourceID +
                ComplainCategoryID ,
                  data = train_data,
                  family = "binomial")
summary(newcs_logit)
# predict the success of the test data
predict_testdata <- predict(newcs_logit, test_data, type = "response")
# create confusion matrix for test data
cm<- confusionMatrix(data = as.factor(as.numeric(predict_testdata > 0.5)),
            reference = as.factor(as.numeric(test_data$ClientSatisfaction)))
# plot confusion matrix
TClass <- factor(c(0, 0, 1, 1))
PClass <- factor(c(0, 1, 0, 1))
Y      <- c(10,160,14,1352)
df <- data.frame(TClass, PClass, Y)
ggplot(data =  df, mapping = aes(x = TClass, y = PClass)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "light green", high = "red") +
  theme_bw() + theme(legend.position = "none")
# creating prediction of training data to check for its accuracy
predict_traindata <- predict(newcs_logit, train_data, type = "response")
# create confusion matrix for train data
confusion_matrix <- confusionMatrix(data = as.factor(as.numeric(predict_traindata > 0.5)),
            reference = as.factor(as.numeric(train_data$ClientSatisfaction)))
```

```
# all the data used are in original units, and cannot be compared alongside other variables
# in order to compare all the variables influencing client satisfaction we normalize the variables
# normalize all significant variables
cs$ResolveDays_norm  <- (cs$ResolveDays - min(cs$ResolveDays))/
            (max(cs$ResolveDays) - min(cs$ResolveDays))

cs$ComplainCategoryID_norm  <- (cs$ComplainCategoryID - min(cs$ComplainCategoryID))/
  (max(cs$ComplainCategoryID) - min(cs$ComplainCategoryID))

cs$ComplainPriorityID_norm  <- (cs$ComplainPriorityID - min(cs$ComplainPriorityID))/
  (max(cs$ComplainPriorityID) - min(cs$ComplainPriorityID))

cs$ComplainTypeID_norm  <- (cs$ComplainTypeID - min(cs$ComplainTypeID))/
  (max(cs$ComplainTypeID) - min(cs$ComplainTypeID))

cs$ComplainSourceID_norm  <- (cs$ComplainSourceID - min(cs$ComplainSourceID))/
  (max(cs$ComplainSourceID) - min(cs$ComplainSourceID))
#summary
summary(cs)
# split data into train and test model
train_index_n <- sample(1:nrow(cs), size = 0.8*nrow(cs))
# create train data set using train index
train_data_n <- cs[train_index_n, ]
# create test data set using train index
test_data_n <- cs[-train_index_n, ]
# run logistic regression with normalized variables
csn_logit <-  glm(ClientSatisfaction ~ResolveDays_norm + ComplainPriorityID_norm +
                  ComplainTypeID_norm + ComplainSourceID_norm +
```

```
                  ComplainCategoryID_norm ,
                data = train_data_n,
                family = "binomial")
summary(csn_logit)
# predict the success of the test data
predict_testdata_n <- predict(csn_logit, test_data_n, type = "response")
# create confusion matrix for test data
CF_n <- confusionMatrix(data = as.factor(as.numeric(predict_testdata_n > 0.5)),
            reference = as.factor(as.numeric(test_data_n$ClientSatisfaction)))
# plot confusion matrix
TClass <- factor(c(0, 0, 1, 1))
PClass <- factor(c(0, 1, 0, 1))
Y      <- c(11,128,19,1378)
df <- data.frame(TClass, PClass, Y)
ggplot(data =  df, mapping = aes(x = TClass, y = PClass)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "light green", high = "red") +
  theme_bw() + theme(legend.position = "none")
# creating prediction of training data to check for its accuracy
predict_traindata_n <- predict(csn_logit, train_data_n, type = "response")
# create confusion matrix for train data
CFT_n <- confusionMatrix(data = as.factor(as.numeric(predict_traindata_n > 0.5)),
            reference = as.factor(as.numeric(train_data_n$ClientSatisfaction)))
# plot confusion matrix for train data
TClass <- factor(c(0, 0, 1, 1))
PClass <- factor(c(0, 1, 0, 1))
Y      <- c(57,624,62,5397)
df <- data.frame(TClass, PClass, Y)
ggplot(data =  df, mapping = aes(x = TClass, y = PClass)) +
```

```
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "light green", high = "red") +
  theme_bw() + theme(legend.position = "none")
# testing confusion matrix
CF_n
# training Confusion Matrix
CFT_n
# create ROC and AUC
# create new object (predicted values for my object)
pred_val <- prediction(predict_testdata_n, test_data_n$ClientSatisfaction)
perf_lr <- performance(pred_val, "tpr", "fpr")
# plot AUC
plot(perf_lr)
# run regression for resolve days
resolve_source<-  glm(ResolveDays ~ ComplainSourceID + ComplainTypeID_norm+
                      ComplainPriorityID_norm + ComplainPriorityID,
                    data = train_data_n,
                    family = "binomial")
resolve_source
# build GINI decision tree
class_tree <- rpart(cs$ClientSatisfaction ~ cs$ResolveDays + cs$ComplainPriorityID +
                      cs$ComplainTypeID + cs$ComplainSourceID +
                      cs$ComplainCategoryID,
                    method = "class",
                    control = rpart.control(minsplit = 150,
                                            minbucket = 150,
                                            cp = 0.007
                                            ))
```

```
# plot tree using rpart.plot function
rpart.plot(class_tree, type = 2, extra = 2)
# Calculate the probability of business success
# resolve days
round(exp(-0.017569)-1,4)
# complain priority
round(exp(1.569340)-1,4)
# complain type
round(exp(0.430110)-1,4)
# complain source
round(exp(-0.203120)-1,4)
# complain category
round(exp(0.023085)-1,4)
```