### Installing packages and loading library

library(readr)

### Setting up the working directory and loading data file

setwd("D:/great learning/5. Machin Learning/Project-5")

cars\_mydata = read.csv("Cars-1.csv", header = TRUE)

View(cars\_mydata)

str(cars\_mydata)

### Exploratory data analysis

summary(cars\_mydata)

cars\_mydata1 = na.omit(cars\_mydata)

summary(cars\_mydata1)

### Check for multi co-linearity

library(caTools)

library(corrplot)

cars\_mydata2 = cars\_mydata1[-c(2,9)]

cars\_mydata2\_corr = cor(cars\_mydata2)

corrplot(cars\_mydata2\_corr, method = "number")

### Treating multi co-linearity

library(nFactors)

cars\_mydata3 = cars\_mydata1[c(1,5,6)] # Considering only correlated variables Age, Work Exp, Salary

names(cars\_mydata3)

View(cars\_mydata3)

cars\_mydata3\_corr = cor(cars\_mydata3)

cars\_mydata3\_corr

ev = eigen(cor(cars\_mydata3))

ev

EigenValue = ev$values

EigenValue

solution = fa(r = cars\_mydata3, nfactors = 1, rotate = "none", fm = "pa")

solution

solution1 = fa(r = cars\_mydata3, nfactors = 1, rotate = "varimax", fm = "pa")

solution1

solution1$scores

### Binding factor with other attributes

cars\_mydata\_bind = cbind(cars\_mydata1[c(2,3,4,7,8,9)],solution1$scores)

names(cars\_mydata\_bind) = c("Gender","Engineer", "MBA", "Distance", "License", "Transport", "Employee")

names(cars\_mydata\_bind)

View(cars\_mydata\_bind)

### Preparing data for SMOTE analysis

set.seed(1000)

cars.split = sample.split(cars\_mydata\_bind,SplitRatio = 0.70)

cars.train = subset(cars\_mydata\_bind, cars.split == TRUE)

cars.test = subset(cars\_mydata\_bind, cars.split == FALSE)

View(cars.train)

View(cars.test)

### Logistic Regression

cars.logistic = glm(Transport~., data = cars.train, family = binomial(link = "logit"))

cars.test$log.pred = predict(cars.logistic, cars.test, type = "response")

table(cars.test$Transport, cars.test$log.pred>0.5)

### KNN Fit

library(class)

knn.fit = knn(train = cars.train, test = cars.test, cl = cars.train[c(6)], k = 2, prob = TRUE)

table(cars.test[,6], knn.fit)

### Naive Bayes

library(e1071)

cars.naive = naiveBayes(x = cars.train[,1:5], y = as.factor(cars.train[,6]))

pred.nb = predict(cars.naive, newdata = cars.test[-c(6)])

table(cars.test[,6], pred.nb)

### Bagging

install.packages('ipred')

library(ipred)

library(rpart)

cars.bagging = bagging(Transport~., data = cars.train, control = rpart.control(maxdepth = 5, minsplit = 4))

cars.test$pred.Transport = predict(cars.bagging, cars.test)

table(cars.test$Transport, cars.test$pred.Transport)

### Boosting (Gradient Boosting Method)

install.packages('gbm')

library(gbm)

gbm.fit = gbm(formula = Transport~., distribution = "bernoulli", data = cars.train, n.trees = 1000,

interaction.depth = 1, shrinkage = 0.001, cv.folds = 7, n.cores = NULL, verbose = FALSE)

cars.test$pred.Transport = predict(gbm.fit, cars.test, type = "response")

table(cars.test$Transport, cars.test$pred.Transport>0.5)

### XG Boosting

install.packages('xgboost')

library(xgboost)

cars\_features\_train = as.matrix(cars.train[-c(6)])

cars\_label\_train = as.matrix(cars.train[-c(1,2,3,4,5,7)])

cars\_features\_test = as.matrix(cars.test[-c(6)])

xgb.fit = xgboost(data = cars\_features\_train, label = cars\_label\_train, eta = 0.001, max\_depth = 3, min\_child\_weight = 3,

nrounds = 10000, nfold = 5, objective = "binary:logistic", verbose = 0, early\_stopping\_rounds = 10)

cars.test$xgb.pred.Transport = predict(xgb.fit, cars\_features\_test)

table(cars.test$Transport, cars.test$xgb.pred.Transport>0.5)

### Trying new combination of data

tp\_xgb = vector()

lr = c(0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1)

md = c(1,3,5,7,9,15)

nr = c(2, 50, 100, 1000, 10000)

for(i in md) {

xgb.fit = xgboost(data = cars\_features\_train, label = cars\_label\_train, eta = 0.7, max\_depth = 5, nrounds = 50,

nfold = 5, objective = "binary:logistic", verbose = 1, early\_stopping\_rounds = 10)

}

cars.test$xgb.pred.Transport = predict(xgb.fit, cars\_features\_test)

tp\_xgb = cbind(tp\_xgb, sum(cars.test$Transport ==1 & cars.test$xgb.pred.Transport>=0.5))

### Now we put them all into our best fit

xgb.fit = xgboost(data = cars\_features\_train, label = cars\_label\_train, eta = 0.7, max\_depth = 5, nrounds = 50,

nfold = 5, objective = "binary:logistic", verbose = 1, early\_stopping\_rounds = 10)

cars.test$xgb.pred.Transport = predict(xgb.fit, cars\_features\_test)

sum(cars.test$class ==1 & cars.tet$xgb.pred.Transport>=0.5)

table(cars.test$class, cars.test$xgb.pred.Transport>=0.5)

### Working with SMOTE

install.packages('DMwR')

library(DMwR)

table(cars\_mydata1$Transport)

split = sample.split(cars\_mydata1,SplitRatio = 0.70)

smote.train = subset(cars\_mydata1, split == TRUE)

smote.test = subset(cars\_mydata1, split == FALSE)

smote.train$Transport = as.factor(smote.train$Transport)

balanced.cars = SMOTE(Transport~., smote.train, perc.over = 4800, k = 5, perc.under = 1000)

table(balanced.cars$Transport)

View(balanced.cars)

### Now put SMOTE data into best XGBoost

smote\_features\_train = as.matrix(balanced.cars[-c(9)])

smote\_label\_train = as.matrix(balanced.cars$Transport)

smote.xgb.fit = xgboost(data = smote\_features\_train, label = smote\_label\_train, eta = 0.3, max\_depth = 5,

nrounds = 30, nfold = 5, objective = "binary:logistic", verbose = 0, early\_stopping\_rounds = 10)

smote\_features\_test = as.matrix(smote.test[-c(6)])

smote.test$smote.pred.Transport = predict(smote.xgb.fit, smote\_features\_test)

table(smote.test$Transport, smote.test$smote.pred.Transport>=0.5)