# XGBoost

#good for large dataset

#herefeature scaling not require but in deep learning its compulsary

#best model in term of good accuracy model and fast execution speed

#high performance, keep all interpration in model, high speed

# Importing the dataset

dataset = read.csv('Churn\_Modelling.csv')

dataset = dataset[4:14]

# Encoding the categorical variables as factors

dataset$Geography = as.numeric(factor(dataset$Geography,

levels = c('France', 'Spain', 'Germany'),

labels = c(1, 2, 3)))

dataset$Gender = as.numeric(factor(dataset$Gender,

levels = c('Female', 'Male'),

labels = c(1, 2)))

# Splitting the dataset into the Training set and Test set

# install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(dataset$Exited, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Fitting XGBoost to the Training set

#install.packages('xgboost')

library(xgboost)

classifier = xgboost(data = as.matrix(training\_set[-11]), label = training\_set$Exited, nrounds = 10)#type F1....do parameter tunning

#whole training set without dependent variable

#data parameter accept matrix

#nrounds--max no of iteration

#RESULT::rmse-ROOT MEAN SQUARE ERROR decrease continously

# Predicting the Test set results

y\_pred = predict(classifier, newdata = as.matrix(test\_set[-11]))

y\_pred = (y\_pred >= 0.5)

# Making the Confusion Matrix

cm = table(test\_set[, 11], y\_pred)

# Applying k-Fold Cross Validation

# install.packages('caret')

library(caret)

folds = createFolds(training\_set$Exited, k = 10)

cv = lapply(folds, function(x) {

training\_fold = training\_set[-x, ]

test\_fold = training\_set[x, ]

classifier = xgboost(data = as.matrix(training\_set[-11]), label = training\_set$Exited, nrounds = 10)#here classifier is XGBoosT(just copy paste)

y\_pred = predict(classifier, newdata = as.matrix(test\_fold[-11]))# here is in no of fold

y\_pred = (y\_pred >= 0.5)# since output is probability so convert to binary 1 or 0

cm = table(test\_fold[, 11], y\_pred)

accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])

return(accuracy)

})

accuracy = mean(as.numeric(cv))

#we find accuracy 88% (prof that best accuracy and best model)