MLA-2 CIA-1

TEJASVI B-22121006

2023-10-22

#installation of library packages

Data Loading and Preprocessing

library(readr)  
data <- read\_csv("C:/Users/HP/Desktop/MLA CIA - 4.csv")

## Rows: 30001 Columns: 10  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (4): Cust\_ID, Gender, Occupation, AGE\_BKT  
## dbl (6): Target, Age, Balance, No\_OF\_CR\_TXNS, Holding\_Period, SCR  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

View(data)

sum(is.na(data))

## [1] 105650

data = na.omit(data)  
  
sum(is.na(data))

## [1] 0

library(data.table)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ada)

## Loading required package: rpart

library(xgboost)

# ADA BOOST

#The code splits the data into a training set and a test set. It uses the createDataPartition function to split the data into a 70% training set and a 30% test set based on the 'Target' variable.  
Model Training  
set.seed(123)  
data = data[,-1]  
  
trainIndex =createDataPartition(data$Target, p = 0.7, list = FALSE)  
train\_data = data[trainIndex, ]  
test\_data = data[-trainIndex, ]

Model Evaluation  
  
boost\_model <- ada(Target ~ ., data = train\_data, type = "discrete", control = rpart.control(cp = 0.01))  
  
summary(boost\_model)

## Call:  
## ada(Target ~ ., data = train\_data, type = "discrete", control = rpart.control(cp = 0.01))  
##   
## Loss: exponential Method: discrete Iteration: 50   
##   
## Training Results  
##   
## Accuracy: 0.974 Kappa: 0.558

# XGBoost Model

##

The above line predicts the class labels using the AdaBoost model on the test data. The "type" parameter is set to "class" to get binary class predictions.

## Warning in predict.ada(boost\_model, newdata = test\_data, type = "class"): type=  
## class is undefined: default is 'vector'..

ada\_cm <- confusionMatrix(ada\_pred, as.factor(test\_data$Target))  
  
  
xgb\_pred = predict(xgb\_model, test\_matrix)  
  
xgb\_pred = ifelse(xgb\_pred > 0.5, 1, 0)  
  
xgb\_cm = confusionMatrix(as.factor(xgb\_pred), as.factor(test\_data$Target))  
  
  
  
print(ada\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2186 91  
## 1 10 17  
##   
## Accuracy : 0.9562   
## 95% CI : (0.947, 0.9642)  
## No Information Rate : 0.9531   
## P-Value [Acc > NIR] : 0.2637   
##   
## Kappa : 0.2376   
##   
## Mcnemar's Test P-Value : 1.716e-15   
##   
## Sensitivity : 0.9954   
## Specificity : 0.1574   
## Pos Pred Value : 0.9600   
## Neg Pred Value : 0.6296   
## Prevalence : 0.9531   
## Detection Rate : 0.9488   
## Detection Prevalence : 0.9883   
## Balanced Accuracy : 0.5764   
##   
## 'Positive' Class : 0   
##

The model shows high accuracy (0.9562), but the Kappa and specificity indicate some room for improvement.

print(xgb\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2192 106  
## 1 4 2  
##   
## Accuracy : 0.9523   
## 95% CI : (0.9427, 0.9606)  
## No Information Rate : 0.9531   
## P-Value [Acc > NIR] : 0.6027   
##   
## Kappa : 0.0303   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.99818   
## Specificity : 0.01852   
## Pos Pred Value : 0.95387   
## Neg Pred Value : 0.33333   
## Prevalence : 0.95312   
## Detection Rate : 0.95139   
## Detection Prevalence : 0.99740   
## Balanced Accuracy : 0.50835   
##   
## 'Positive' Class : 0   
##

#The results include the confusion matrix and performance metrics. The XGBoost model shows a slightly lower accuracy (0.9523) compared to AdaBoost.

The Kappa and sensitivity are also not very high.

# Load necessary libraries

ROC Curve Analysis  
library(pROC)

Creating ROC curve for the two algorithms to check the accuracy

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

# Setting up the plotting layout  
par(mfrow=c(1, 2), mar=c(5, 4, 2, 2))  
  
# For AdaBoost Model  
# Getting predicted probabilities for the positive class  
ada\_probs <- predict(boost\_model, newdata = test\_data, type = "prob")[,2]  
  
# Creating a ROC curve  
ada\_roc <- roc(test\_data$Target, ada\_probs)

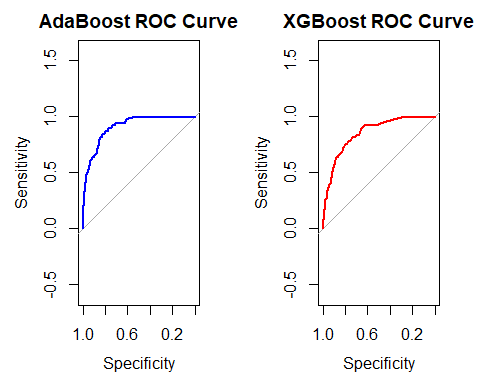
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

plot(ada\_roc, main="AdaBoost ROC Curve", col="blue", lwd=2)  
  
# For XGBoost Model  
# Getting predicted probabilities for the positive class  
xgb\_probs <- predict(xgb\_model, test\_matrix)  
  
# Creating a ROC curve  
xgb\_roc <- roc(test\_data$Target, xgb\_probs)

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

plot(xgb\_roc, main="XGBoost ROC Curve", col="red", lwd=2)



# Resetting the plotting layout  
par(mfrow=c(1, 1), mar=c(5, 4, 2, 2))

ROC curves are commonly used to visualize and compare the performance of binary classification models. They plot the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds.

#Thus from the curves we can identify that ada-boost will have higher ROC curve than compared to XG Boost. Area under the curve is more for ada-boost than compared to XG Boost and the confusion matrix is also proves the same that Adada-boost is better than XG-boost curve.