Project Name: Predicting the likelihood of a customer defaulting on a loan. .
Submitted By: Hari Oam Chaturvedi
Email ID: Harioam1997@gmail.com
Phone Number: 8839157339

This project has been carried out on R platform Using Various machine learning algorithms.Techniques has used to forecast the loan default of various customers based on their data provided . Datasets contains all the details about customers .

There are around 40 variables in the dataset .

Colums Name	**Description**
UniqueID	Primary Key
disbursed_amount	Amount disbursed for loan
asset_cost	Asset involved for loan (tractor, harvestor)
ltv	Loan To Value
branch_id	Branch which disbursed the loan
supplier_id	Supplier who provided the asset
manufacturer_id	Manufacturer who made the product (mahindra, sonalika etc)
Current_pincode_ID	Pincode of customer
Date.of.Birth	Date of birth of customer
Employment.Type	Employement Type
DisbursalDate	Disbursal date of loan
State_ID	State where transaction happened
Employee_code_ID	Agent involved
MobileNo_Avl_Flag	If mobile is present 1 else 0
Aadhar_flag	if aadhar is present 1 else 0
PAN_flag	if pan is present 1 else 0
VoterID_flag	if voter id is present 1 else 0
Driving_flag	if driving license is present 1 else 0
Passport_flag	if passport is present 1 else 0
PERFORM_CNS.SCORE	Credit bureau score
PERFORM_CNS.SCORE.DESCRIPTION	Credit bureau tagging
PRI.NO.OF.ACCTS	Primary/ Principal account counts of prior loans
PRI.ACTIVE.ACCTS	Primary account counts of active prior loans
PRI.OVERDUE.ACCTS	Primary overdue of accounts (if there were overdue in loans)
PRI.CURRENT.BALANCE	current balance for a loan to be paid

SEC.CURRENT.BALANCE	Current balance for a loan to be paid .
SEC.SANCTIONED.AMOUNT	Secondary sanctioned amount
SEC.DISBURSED.AMOUNT	secondary disbursed amount
PRIMARY.INSTAL.AMT	primary installment amount
SEC.INSTAL.AMT	secondary installment amount
NEW.ACCTS.IN.LAST.SIX.MONTHS	new loans in last 6 months
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	defaulted loans in last 6 months
AVERAGE.ACCT.AGE	average age of a loan for a single customer
CREDIT.HISTORY.LENGTH	history of customer from first loan
NO.OF_INQUIRIES	number of enqiuries for loan done prior to loan
loan_default	Target variable which is to be predicted for this challenge

Data Preprocessiong and Model Selection

1) Data pre-processing was carried out in R , by dplyr package removal of Na`s .
2) Creating dummies for categorical Variable
3) After dividing into 80:20 train and test data , 4 models **Logistic regression , Decision Tree Model , Random Forest Model and XGBoost model** is applied to train the data.
4) Followed by training all the models have been used to forecast the loan default on the test data one at a time.

Model Performances

1) For **Logistic regression** model I have a achieved a balanced F1 Score and KS score at
(0.37) cutoff
4266 TP
5629 FP
22780 FN
9359 TN
27046 P
14988 N
0.4311268 Sn
0.4311268 Precision highest is 0.4750680 for cutoffs 0.46
0.3241424 Accuracy
0.4311268 F1 score highest 0.4750680 at cutoff 0.46
0.218 KS SCore

2) For decision Tree model ,It gave binary answers in 1 or 0 rather than probability which has irregularities since it operated on single tree gave only False Negative and true Negative,failed on prediciting any positive outcome with accuracy of 67%

3) Coming on to **Random Forest** model which performed really well and can be used to deploy in the real world case scenario .
9159 TP
15015 FP
13394 FN
4466 TN
22553 P
19481 N sn
0.3788781 Precision, **0.45 of max precison f1 and acuuracy at 0.45 cutoff**
3.788781e-01 Accuracy
3.241424e-01 **F1 score 0.4750680 0.4750680**
0.3788781 F1 score
0.365 ks score

4) Deployment of **XGBoost** model is quite effective for this problem of forecasting of loan default ,Performed really well with the test data .
0.697 cutoff for highest in **precision and f1 score and sn 0.764705882, 0.764705882 0.764705882** as well that is and accuracy
3.21 constant for ks and f1 equilibrium score at cutoff at 0.33 cutoff

The codes for All the model codes is enclosed below for your references .
I would go for **XGBoost** model as it gave an highest precision , accuracy ,F1 score of 76% for the model , outperforming any other model .where Target Variable is Loan Default and
IV`s which are of no significance to the model and are excluded from the model.

UniqueID ,branch_id ,manufacturer_id ,Current,pincode_ID ,DisbursalDate -State_ID -Employee_code_ID ,

MobileNo_Avl_Flag Aadhar_flag , Aadhar_flag ,SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT

,PERFORM_CNS.SCORE ,PAN_flag SEC.SANCTIONED.AMOUNT ,empl_selfemployed ,PRI.SANCTIONED.AMOUNT

,VoterID_flag, New_CREDIT.HISTORY.LENGTH_months , PRI.ACTIVE.ACCTS ,Driving_flag ,Passport_flag

All the variables which are **very significant** to the predicitions ;

disbursed_amount , SEC.NO.OF.ACCTS , SEC.ACTIVE.ACCTS , SEC.OVERDUE.ACCTS , SEC.CURRENT.BALANCE , asset_cost , ltv

, PRI.OVERDUE.ACCTS , SEC.NO.OF.ACCTS , PRI.CURRENT.BALANCE , NO.OF_INQUIRIES , PRIMARY.INSTAL.AMT ,

SEC.INSTAL.AMT , DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS , CREDIT.HISTORY.LENGTH and NEW.ACCTS.IN.LAST.SIX.MONTHS

```
setwd("C:/jrt 4")
getwd()
library(readxl)
data = read.csv("C:/jrt 4/default_data.csv")View(data)
summary(data)
head(data,10)
library(dplyr)
glimpse(data)
data[3,]

apply(data,2,function(data)sum(is.na(data)))
is.data.frame(data)
sum(is.na(data))
####no na valuues or missing values present
###now   we create dummy variables for the categorical variablesglimpse(data)
##working with date of birth column to calculate age
data$age <- as.numeric(format(Sys.Date(), "%Y")) - as.numeric(substring(data$Date.of.Birth, 7, 10))##so I can see some
age are negetive which has to be ommited
data=data[data$age>=20,]
####choosen age greater than 20 only
##now moving on to employment type
sort(table(data$Employment.Type)) data<-
mutate(data,
            empl_selfemployed = as.numeric(Employment.Type %in% ("Self employed")),
            empl_salaried = as.numeric(Employment.Type %in% ("Salaried"))
)
data=data %>% select(-Employment.Type)
glimpse(data)

glimpse(data)
nrow(data)
###now coming on to next varaible char
table(data$branch_id) table(data$supplier_id)
###I will remove all this variables to containing id because it will create lot of dimensionality curselibrary(dplyr)
glimpse(data)
table(data$PERFORM_CNS.SCORE.DESCRIPTION)
data <- mutate(data,
               cns_high_risk = as.numeric(PERFORM_CNS.SCORE.DESCRIPTION == "High Risk"), cns_low_risk =
               as.numeric(PERFORM_CNS.SCORE.DESCRIPTION == "Low Risk"), cns_medium_risk =
               as.numeric(PERFORM_CNS.SCORE.DESCRIPTION == "Medium Risk"), cns_very_high_risk =
               as.numeric(PERFORM_CNS.SCORE.DESCRIPTION == "Very High Risk"),cns_very_low_risk =
               as.numeric(PERFORM_CNS.SCORE.DESCRIPTION == "Very Low Risk"), cns_not_scored =
               as.numeric(PERFORM_CNS.SCORE.DESCRIPTION == "Not Scored")
)
glimpse(data)
###now moving on to next variable
data=data %>%
  select(-PERFORM_CNS.SCORE.DESCRIPTION )
glimpse(data)
```

```
install.packages("stringr") library(stringr)

library(dplyr)
glimpse(data)

data$AVERAGE.ACCT.AGE_months <- with(data, {
  years <- as.numeric(substring(AVERAGE.ACCT.AGE, 1, 1))
  months <- as.numeric(substring(AVERAGE.ACCT.AGE, 5, 6))
  total_months <- years * 12 + months return(total_months)
})
data$New_CREDIT.HISTORY.LENGTH_months <- with(data, {
  years <- as.numeric(substring(CREDIT.HISTORY.LENGTH, 1, 1))
  months <- as.numeric(substring(CREDIT.HISTORY.LENGTH, 5, 6))
  total_months <- years * 12 + months
  return(total_months)
})

data=data %>%
  select(-CREDIT.HISTORY.LENGTH,-AVERAGE.ACCT.AGE,)


glimpse(data)
apply(data,2,function(data) sum(is.na(data)))
data=na.omit(data)
'
#######data is preprocessed for modelset.seed(77)
s=sample(1:nrow(data),0.80*nrow(data))
data_train=data[s,]
data_test=data[-s,]
###train and test data are madeglimpse(data_train)
#########
model_1=lm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
           -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_Fla
           -Aadhar_flag
           -PAN_flag
           -VoterID_flag
           -Driving_flag
           -Passport_flag
           ,data = data_train)
sort(vif(model_1),decreasing = TRUE)
install.packages("car")
library(car)
as.factor(data_train)
str(data_train)
glimpse(data_train) ##using
for glm
model_2=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
            -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_Fl
            -Aadhar_flag
            -PAN_flag
```

```
                  -VoterID_flag
                  -Driving_flag
                  -Passport_flag
                  , family = "binomial"
                 ,data = data_train)
sort(vif(model_2),decreasing = T)
summary(model_2)
model_2_1=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
                    -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_
                    -Aadhar_flag - SEC.DISBURSED.AMOUNT
                    -PAN_flag
                    -VoterID_flag
                    -Driving_flag
                    -Passport_flag
                    , family = "binomial"
                    ,data = data_train)
sort(vif(model_2_1),decreasing = T)
model_2_2=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
                    -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_
                    -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT
                    -PAN_flag
                    -VoterID_flag
                    -Driving_flag
                    -Passport_flag
                    , family = "binomial"
                    ,data = data_train)
sort(vif(model_2_2),decreasing = T)
model_2_3=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
                    -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_
                    -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCORE
                    -PAN_flag
                    -VoterID_flag
                    -Driving_flag
                    -Passport_flag
                    , family = "binomial"
                    ,data = data_train)
sort(vif(model_2_3),decreasing = T)
model_2_4=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
                    -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_
                    -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCORE -asset_cost
                    -PAN_flag
                    -VoterID_flag
                    -Driving_flag
                    -Passport_flag
                    , family = "binomial"
                    ,data = data_train)
sort(vif(model_2_4),decreasing = T)
model_2_5=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
                    -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_
                    -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCORE -asset_cost
                    -PAN_flag - SEC.SANCTIONED.AMOUNT
                    -VoterID_flag
                    -Driving_flag
                    -Passport_flag
```

```
                    , family = "binomial"
                    ,data = data_train)
sort(vif(model_2_5),decreasing = TRUE)
model1_2_6=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
                    -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl
                    -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCORE -asset_cos
                    -PAN_flag - SEC.SANCTIONED.AMOUNT - self_employed
                    -VoterID_flag
                    -Driving_flag
                    -Passport_flag
                    , family = "binomial"
                    ,data = data_train)
 sort(vif(model1_2_6),decreasing = TRUE)
 model1_2_7=glm(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID
                      -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_A
                      -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCORE -asset_c
                      -PAN_flag - SEC.SANCTIONED.AMOUNT - empl_selfemployed -PRI.SANCTIONED.AMOUNT
                      -VoterID_flag-   New_CREDIT.HISTORY.LENGTH_months-PRI.ACTIVE.ACCTS
                      -Driving_flag
                      -Passport_flag
                      , family = "binomial"
                      ,data = data_train)
 glimpse(data_train)
 vif_values <- vif(model1_2_7)
 sort(vif(model1_2_7),decreasing = TRUE)
 summary(model1_2_7)

 View(cor(data_train)) model1_2_7=step(model1_2_7)

####
data_train=data_train %>% select(-Date.of.Birth)
data_test=data_test %>% select(-Date.of.Birth)
# Load the required library
library(rpart)
# Load the required library
library(rpart)

# Create a decision tree model
tree_model <- rpart(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID
                         -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileN
                         -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCORE -asse
                         -PAN_flag - SEC.SANCTIONED.AMOUNT - empl_selfemployed -PRI.SANCTIONED.AMOUNT
                         -VoterID_flag-   New_CREDIT.HISTORY.LENGTH_months-PRI.ACTIVE.ACCTS
                         -Driving_flag
                         -Passport_flag

                         , data = data_train, method = "class")

# Visualize the decision tree
install.packages("rpart.plot")
library(rpart.plot) prp(tree_model)
summary(tree_model)
```

```
resid_1=resid(model1_2_7) ####visualizing
through randon forest# Load the required
library install.packages("randomForest")
library(randomForest)

# Create a random forest model
rf_model_1 <- randomForest(loan_default ~ . -UniqueID -branch_id -manufacturer_id -Current_pincode_ID
                           -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag
                           -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCOR
                           -PAN_flag - SEC.SANCTIONED.AMOUNT   -PRI.SANCTIONED.AMOUNT
                           -VoterID_flag-   New_CREDIT.HISTORY.LENGTH_months-PRI.ACTIVE.ACCTS
                           -Driving_flag
                           -Passport_flag
                          , data = data_train, ntree = 100)

# Print an overview of the random forest model
print(rf_model_1)
summary(rf_model_1)
rf_model_2 <- randomForest(loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID
                           -DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag
                           -Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCOR
                           -PAN_flag - SEC.SANCTIONED.AMOUNT - empl_selfemployed -PRI.SANCTIONED.AMOUNT
                           -VoterID_flag-   New_CREDIT.HISTORY.LENGTH_months-PRI.ACTIVE.ACCTS
                           -Driving_flag
                           -Passport_flag

                           , data = data_train, ntree = 200)
summary(rf_model_2)

####moving on to xboost
# Install the xgboost package
install.packages("xgboost")

# Load the xgboost library
library(xgboost)
glimpse(data_train)


# Define the formula, excluding the unwanted variables
formula= (loan_default~.-UniqueID -branch_id -manufacturer_id -Current_pincode_ID -Date.of.Birth
-DisbursalDate -State_ID -Employee_code_ID -MobileNo_Avl_Flag -Aadhar_flag -MobileNo_Avl_Flag
-Aadhar_flag - SEC.DISBURSED.AMOUNT - PRI.DISBURSED.AMOUNT -PERFORM_CNS.SCORE -asset_cost
-PAN_flag - SEC.SANCTIONED.AMOUNT - empl_selfemployed -PRI.SANCTIONED.AMOUNT
-VoterID_flag-   New_CREDIT.HISTORY.LENGTH_months-PRI.ACTIVE.ACCTS
-Driving_flag
-Passport_flag)


# Create a DMatrix for the training data, excluding specified variables
data_matrix <- xgb.DMatrix(data = as.matrix(data_train[, setdiff(names(data_train), all.vars(formula))]##giving paramteres
# Define hyperparameters
params <- list(
```

```
    objective = "binary:logistic",eta =
    0.1,
    max_depth = 6,
    subsample = 0.8,
    colsample_bytree = 0.8,
    min_child_weight = 1, eval_metric =
    "logloss"
)
View(data_matrix)
##train the model
xgb_model <- xgboost(data = data_matrix, params = params, nrounds = 100, verbose = 1)
summary(xgb_model)
####so done with building model now we will start with predictions

nrow(data_test)



model1_2_7logpred=predict(model1_2_7,data_test,type = "response")
View(model1_2_7logpred) data_test$logistic_score=model1_2_7logpred
View(data_test[,c("loan_default","logistic_score")])
log_pred_table=data_test[,c("loan_default","logistic_score")] ###storing it in different table###converting score to
0 and 1s on ,,assuming cutoff 0.5 is best suited
cutoff=0.5 log_pred_table$logistic_score_01=as.numeric(log_pred_table$logistic_score>0.5)
View(log_pred_table)

# Calculate True Positives (TP)
TP <- sum(log_pred_table$loan_default == 1 & log_pred_table$logistic_score_01 ==1)

# Calculate False Positives (FP)
FP <- sum(log_pred_table$loan_default == 0 & log_pred_table$logistic_score_01 == 1)

# Calculate True Negatives (TN)
TN <- sum(log_pred_table$loan_default == 0 & log_pred_table$logistic_score_01 == 0 )

# Calculate False Negatives (FN)
FN <- sum(log_pred_table$loan_default == 1 & log_pred_table$logistic_score_01 == 0)

# Print the results
cat("True Positives (TP):", TP, "\n")
cat("False Positives (FP):", FP, "\n")
cat("True Negatives (TN):", TN, "\n")
cat("False Negatives (FN):", FN, "\n")P=TP+FN
N=TN+FP
Total=P+N
###this cutoff is not suitable so we will check for all the cutoffs which suits the best
cutoffdataframe=data.frame(cutoff=0,TP=0,FP=0,FN=0,TN=0)
View(cutoffdataframe)
cutoffs = round(seq(0, 1, length = 101), 3)
log_pred_table$logistic_score=round(log_pred_table$logistic_score,2) ####now to iterate
we will mutate other tables in it
```

```
for (cutoff in cutoffs) {
   predicted_1=as.numeric(log_pred_table$logistic_score>cutoff) TP <-
   sum(log_pred_table$loan_default == 1 & predicted_1 == 1)FP <-
   sum(log_pred_table$loan_default == 0 & predicted_1 == 1)
   TN <- sum(log_pred_table$loan_default == 0 & predicted_1 == 0 )FN <-
   sum(log_pred_table$loan_default == 1 & predicted_1 == 0)
   cutoffdataframe=rbind(cutoffdataframe,c(cutoff,TP,FP,TN,FN))
   }
View(cutoffdataframe)
cutoffdataframe=cutoffdataframe[-1,]
####calculate in cutoffdataframe wwhich is better for the what cutoofff##based on
various metric such as precision accuracy and all
cutoffdataframe=mutate(cutoffdataframe,
                                   P=TP+FN,
                                   N=TN+FP,
                                   Sn = TP/(TP+FP), PRECISION
                                   = TP/(TP+FP),
                                   ACCURACY = (TP+TN)/(TP+FP+TN+FN) , F1_SCORE
                                   = 2*PRECISION*Sn/(PRECISION+Sn),
                                   KS_SCORE = round(abs((TP/P)-(FP/N)),3)
                                   )
##for logisitic regrsn the primary objective is to reduce losses due to loan defaults by minimizing ###false negatives (i.e.,
correctly identifying loans that will default), you should focus on maximizin####ADDING BOTH THE DATA FRAME F1_SCORE: F1
Score is the harmonic mean of precision and recall. It prov### and minimizing false positives (precision). By maximizing the F1
Score, you are finding the cutoff####KS_SCORE: KS Score measures the maximum difference between the cumulative
distribution functions of###negative (non-defaults) classes. It is commonly used in credit scoring to find the cutoff that maxim
####A high KS Score corresponds to a higher true positive rate and a lower false negative rate.
log_pred_table$logistic_score_real=as.numeric(log_pred_table$logistic_score>0.37)
View(log_pred_table)##(0.37)
cutoff
#4266 TP
#5629 FP
#22780 fn
#9359 tn
#27046 p
#14988 N
#0.4311268 Sn
#0.4311268 Precisoin highest is 0.4750680 for cutoffs 0.46
#0.3241424 Accuracy
#0.4311268 F1   highest 0.4750680
#0.218 KS SCore
install.packages("writexl") library(writexl)
write_xlsx(log_pred_table,"log_pred_table.xlsx")
write_xlsx(xgcutoffdataframe,"xgcutoffdataframe.xlsx")
####   so my cutoff will be the cutoff value of 0.37 yields the highest KS Score (0.365) and a relativellibrary(ggplot2)
# Create a line graph with readable scales
ggplot(cutoffdataframe, aes(x = cutoff)) +
   geom_line(aes(y = KS_SCORE, color = "KS"), size = 1) + geom_line(aes(y =
   F1_SCORE, color = "F1"), size = 1) + scale_color_manual(values = c("KS" =
   "red", "F1" = "blue")) +
   labs(x = "Cutoff", y = "Value", title = "KS and F1 Scores vs. Cutoff") +
```

```
  theme_minimal() +
  scale_x_continuous(breaks = seq(0.28, 0.69, by = 0.05)) +
  scale_y_continuous(labels = scales::percent_format(scale = 1))
####now will do it for decison treeemodel
data_test$dtree_score = tree_model
dtreepred_table= data_test[,c("loan_default","dtree_score")]
View(dtreepred_table)
## to check accuracy
# Assuming your data is in a data frame called 'df'
correct_dtreepred <- sum(dtreepred_table$loan_default == dtreepred_table$dtree_score)
dtree_accuracy <- correct_dtreepred / nrow(dtreepred_table)
cat("DTREE_Accuracy: ", dtree_accuracy, "\n")
## as there are 67% 0 values in loan_default column and   dtree_perd has
##given all 0s so it shows 67% accuracy   cat("DTREE_Accuracy: ", dtree_accuracy, "\n")
###DTREE_Accuracy:  0.6758576 ,,not a single 1s so we move on to rf data_test=data_test %>% select(-
logistic_score ,- dtree_score )
##predicitng rf model
glimpse(data_test)
View(data_test)
data_test = data_test %>% select(-logistic_score)
rf_predict = predict(rf_model_2 , newdata = data_test, type = "response")

data_test$rfpred=rf_predict rf_pred_table=data_test[,c("loan_default","rfpred")]
View(rf_pred_table)
###now to find out best cutoff for the rfmodel
rfcutoffdataframe=data.frame(cutoff=0,TP=0,FP=0,FN=0,TN=0)
View(rfcutoffdataframe)
cutoffs = round(seq(0, 1, length = 101), 3)
rf_pred_table$rfpred=round(rf_pred_table$rfpred,2) ####now to
iterate we will mutate other tables in itfor (cutoff in cutoffs) {
  predicted_rf=as.numeric(rf_pred_table$rfpred>cutoff)
  TP <- sum(rf_pred_table$loan_default == 1 & predicted_rf == 1)FP <-
  sum(rf_pred_table$loan_default == 0 & predicted_rf == 1)TN <-
  sum(rf_pred_table$loan_default == 0 & predicted_rf == 0 )FN <-
  sum(rf_pred_table$loan_default == 1 & predicted_rf == 0)
  rfcutoffdataframe=rbind(cutoffdataframe,c(cutoff,TP,FP,TN,FN))
}
View(rfcutoffdataframe)
rfcutoffdataframe=rfcutoffdataframe[-1,]
###0.31cutoff
TP
FP
F
N
T
N
P
N
Sn
PRECISION
ACCURACY
F1_SCORE
KS_SCORE
##9159 TP
```

```
##15015 FP
##13394 FN
##4466 TN
##22553 P
##19481   N sn
#0.3788781 Precision 0.45 of max precison   f1 and acuuracy at 0.45 cutoff#3.788781e-01
Accuracy
#3.241424e-01 F1 score 0.4750680 0.4750680
#0.3788781 F1 score
#0.365 ks score

##drawing f1 and ks for the graph to find best suited cutoff
ggplot(rfcutoffdataframe, aes(x = cutoff)) +
    geom_line(aes(y = KS_SCORE, color = "KS"), size = 1) + geom_line(aes(y =
    F1_SCORE, color = "F1"), size = 1) + scale_color_manual(values = c("KS" =
    "red", "F1" = "blue")) +
    labs(x = "Cutoff", y = "Value", title = "KS and F1 Scores vs. Cutoff") +theme(axis.text.x =
  element_text(angle = 45, hjust = 1)) scale_x_continuous(breaks = seq(0.31, 0.50, by =
  0.02)) + scale_y_continuous(labels = scales::percent_format(scale = 1))
    ###here we can see 0.33 is the best cutoof
  rf_pred_table$predicted_rf_real=as.numeric(rf_pred_table$rfpred>0.33) xgb_pred <-
  predict(xgb_model, data_matrix_test)

  View(xgb_pred)


  ###now move on to next model of xgboost#
  Create DMatrix for the test data
  glimpse(data_test)
##this is done to change it full data_test to dbl or numeric..removing some factor and char and int##then only it can be
converted to xgbmatric data to be predicted
glimpse(data_test)
data_matrix_test <- xgb.DMatrix(data = as.matrix(data_test[, setdiff(names(data_test), all.vars(formuladata_test$xgb_score =
xgb_pred
View(data_test) xgb_predtable=data_test[,c("loan_default","xgb_score")]
View(xgb_predtable)
##now for cutofffdata frame
xgcutoffdataframe=data.frame(cutoff=0,TP=0,FP=0,FN=0,TN=0)
View(xgcutoffdataframe)
cutoffs = round(seq(0, 1, length = 100), 3)
xgb_predtable$xgbscore=round(xgb_predtable$xgbscore,2) ####now
to iterate we will mutate other tables in it for (cutoff in cutoffs) {
    predicted_xg=as.numeric(xgb_predtable$xgbscore>cutoff)
    TP <- sum(xgb_predtable$loan_default == 1 & predicted_xg == 1)FP <-
    sum(xgb_predtable$loan_default == 0 & predicted_xg == 1)TN <-
    sum(xgb_predtable$loan_default == 0 & predicted_xg == 0 )FN <-
    sum(xgb_predtable$loan_default == 1 & predicted_xg == 0)
        xgcutoffdataframe=rbind(xgcutoffdataframe,c(cutoff,TP,FP,TN,FN))
}
View(xgcutoffdataframe)
xgcutoffdataframe=xgcutoffdataframe[-1,]
```

```
###now we will all matrics accuracy,f1score etc
xgcutoffdataframe=mutate(xgcutoffdataframe,
                         P=TP+FN,
                         N=TN+FP,
                         Sn = TP/(TP+FP), PRECISION
                         = TP/(TP+FP),
                         ACCURACY = (TP+TN)/(TP+FP+TN+FN) , F1_SCORE
                         = 2*PRECISION*Sn/(PRECISION+Sn),
                         KS_SCORE = round(abs((TP/P)-(FP/N)),3)
)
####0.697 cutoof for precision and f1 score and sn as well that is and accuracy 3.21 const##for ks and f1
equilibrium score at cutoff at 0.33 cutoff
xgb_predtable$xgb_predicted = as.numeric(xgb_predtable$xgb_score>0.33)
View(xgb_predtable)
glimpse(xgbpre)

####i think xgb is best model with better accuracy and recall,,,fi score and ks score is also better aninstall.packages("writexl")
library(writexl) write_xlsx(xgb_predtable,"xgb_predtable.xlsx")
write_xlsx(xgcutoffdataframe,"xgcutoffdataframe.xlsx")
install.packages("knitr")
```