Classification Model

Objective of this Project:

This project is to determine whether a loan will default, as well as the loss incurred if it does default. Unlike traditional finance-based approaches to this problem, where one distinguishes between good or bad counterparties in a binary way, we seek to anticipate and incorporate both the default and the severity of the losses that result. In doing so, we are building a bridge between traditional banking, where we are looking at reducing the consumption of economic capital, to an asset-management perspective, where we optimize on the risk to the financial investor

About dataset: *This data contains 762 variables having 80000 records.None of the variables has column headers so it is even more difficult to choose variables based on the relative importance by description. Loss column has two values, either 0 or the percentage of loss.*

Approach:

This file includes the Classification part of the problem where we are trying to check if a customer will default or not.

loading required packages

```
library(caret)

library(dplyr)

library(corrplot)

library(glmnet)

library(tidyverse)
```

Reading the data file

```
data<-read.csv("train_v3.csv")
```

Data Cleaning:

```
#Replacing all NA values with 0

data1 <- data %>% mutate_all(funs(replace_na(.,0)))



# Calculate the percentage of 0 values in each column

null_percent <- apply(data1 == 0, 2, mean)
```

```
# Get the names of the columns with less than or equal to 30% of 0(null) values

cols <- names(null_percent[null_percent <= 0.3])

# Create a new data frame with the selected columns

data2 <- data1[, cols]

#Check if the columns with more than 30% null values are deleted

sums<-(colSums(data2==0)/nrow(data2))*100

#Adding loss column back to data

data3<-cbind(data2,loss=data1$loss)
```

Data Transformation:

```
# Create a new column called 'default' with a value of 1 is loss is above 0 and 0 is loss is 0


data3$default <- ifelse(data3$loss == 0, 0, 1)
data<-data3
```

Data Preparation:

```
#Clearing columns with near zero variance,high correlation and median imputation for missing values for

data<-data[ ,-c(data$f736,data$f764)]

preProcessModel <- preProcess(data, method = c("nzv", "corr", "medianImpute"))

data1 <- predict(preProcessModel, data)
```

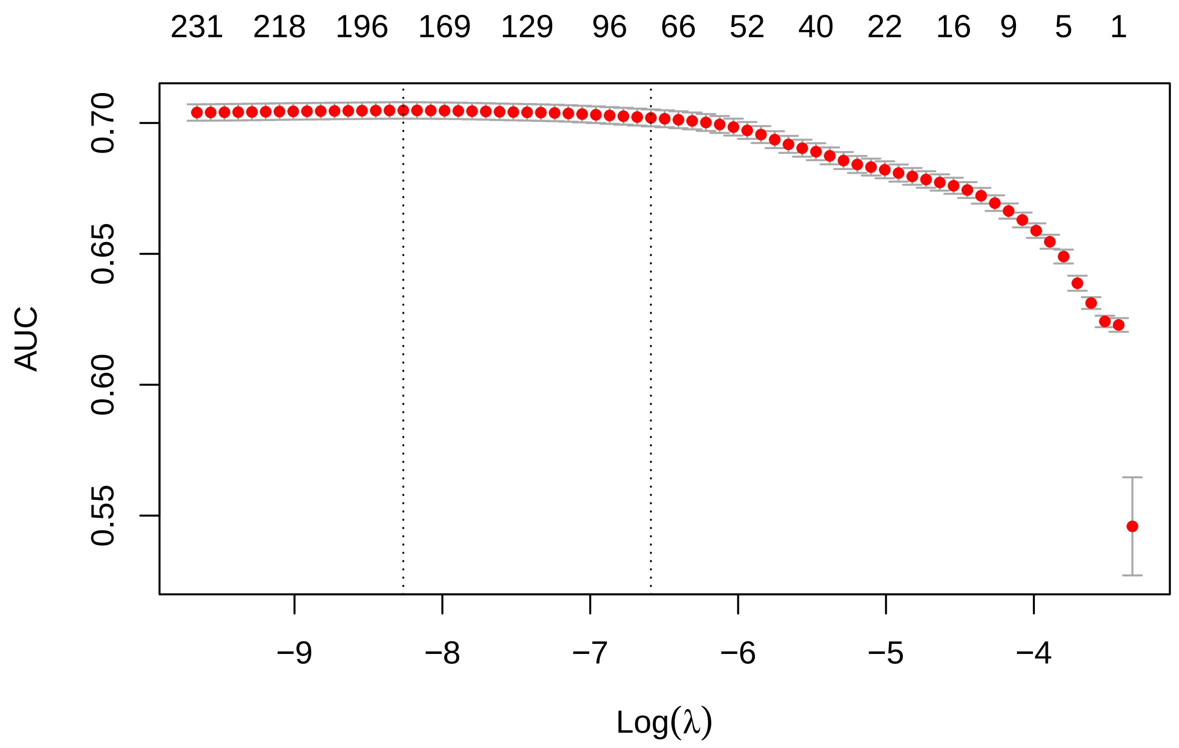
After data cleaning,we are left with 261 columns

Feature Selection using LASSO Model

```
#Creating a matrix

X <- as.matrix(data1[ ,-c(261)])

#Setting default column to factor levels

Y <- as.vector(as.factor(data1$default))

#Building the model with 10 fold cross-validation and performance measure as AUC curve

model<- cv.glmnet(X, Y, alpha = 1, family = "binomial", nfolds = 10, type.measure = "auc")

summary(model)
```

```
##              Length Class  Mode
## lambda     69     -none- numeric
## cvm        69     -none- numeric
## cvsd       69     -none- numeric
## cvup       69     -none- numeric
## cvlo       69     -none- numeric
## nzero      69     -none- numeric
## call        7     -none- call
## name        1     -none- character
## glmnet.fit 13     lognet list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
## index       2     -none- numeric
```

```
plot(model)
```



```
#Finding minimum value for lambda

model$lambda.min
```

```
## [1] 0.0002575271
```

```
#Saving the coefficients for lambda minimum

coefs <- coef(model, s = "lambda.min")
```

```r
# Taking the coefficient values into a data frame for processing

coefs <- data.frame(name = coefs@Dimnames[[1]][coefs@i + 1], coefficient = coefs@x)

# Rounding to the absolute value of all the coefficients in the model

coefs$coefficient <- abs(coefs$coefficient)

# Finding the largest coefficient in the model

coefs <- coefs[order(coefs$coefficient, decreasing = TRUE), ]


# Taking out the intercept from the data frame

coefs <- coefs[-1, ]

#Converting the coefficients to a  vector type

coefs <- as.vector(coefs$name)

#Combining the default column back to the coefficients

coefs <- c(coefs,"default")

data_new<-select(data1,coefs)
```

Using Lasso model, out of 261 variables, nearly 80 variables were eliminated and only 180 variables are selected for further analysis.

Partitioning data into 85% training and 15% validation

```r
#Splitting data into Training and Validation

set.seed(6782)

Split_data <- createDataPartition(data_new$default,p=.85,list=FALSE,times=1)

Training <- data_new[Split_data,]

Validation <- data_new[-Split_data,]
```

Modeling Strategy:

Gradient Boosting Machines-

We used GBMs as they can handle large datasets with ease, as they can be parallelized. Moreover, GBM can reduce overfitting, which is a common problem in machine learning, by using regularization techniques such as early stopping and shrinkage.

```r
#Building a Gradient Boosting Machine model
library(gbm)

#Cross-vlidation with 10 folds and number of trees are set to 50
```

```r
gbm_model <- gbm(default ~ ., data = Training, distribution = "bernoulli",
                 n.trees = 100, interaction.depth = 9, shrinkage = 0.02,
                 bag.fraction = 0.8, train.fraction = 0.85,
                 n.minobsinnode = 15,cv.folds = 10)
```

Deploying the model on the Validation Data:

```r
predictions <- predict(gbm_model, newdata = Validation, n.trees = 100, type = "response")

#Threshold value is set to 50 %

predictions<-ifelse(predictions>0.5,1,0)

# Building Confusion Matrix

Final_data<-cbind(Validation,predictions)


Final_data$predictions<-as.factor(Final_data$predictions)

Final_data$default<-as.factor(Final_data$default)

confusionMatrix<-confusionMatrix(Final_data$default,Final_data$predictions)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10920     0
##          1  1079     1
##
##                Accuracy : 0.9101
##                  95% CI : (0.9048, 0.9151)
##     No Information Rate : 0.9999
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0017
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9100758
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.0009259
##              Prevalence : 0.9999167
##          Detection Rate : 0.9100000
##    Detection Prevalence : 0.9100000
##       Balanced Accuracy : 0.9550379
##
##        'Positive' Class : 0
##
```

```
#Precision<- TP/TP+FP = 0.005%
```

We are considering Precision as our performance metrics as we need to consider the effect of False Positives and False Negatives as well. For a bank, approving a loan for a customer who would be defaulting would be more expensive. Hence, Precision is being used instead of Accuracy.

Here Precision was found to be very low .This was expected because from our EDA we have seen that the data has imbalanced classes of default to non-default with almost 1:10 ratio.Therefore we will use stratified sampling for splitting and run the model again

Stratified Sampling:

```
data_sample<-data_new

# Calculate the smallest class size

min_length <- min(table(data_sample$default))

# Sample the minority class to the size of the smallest class

balanced_spam <- data_sample %>%group_by(default) %>%sample_n(min_length) %>%ungroup()

# Check the class distribution after balancing

table(balanced_spam$default)
```

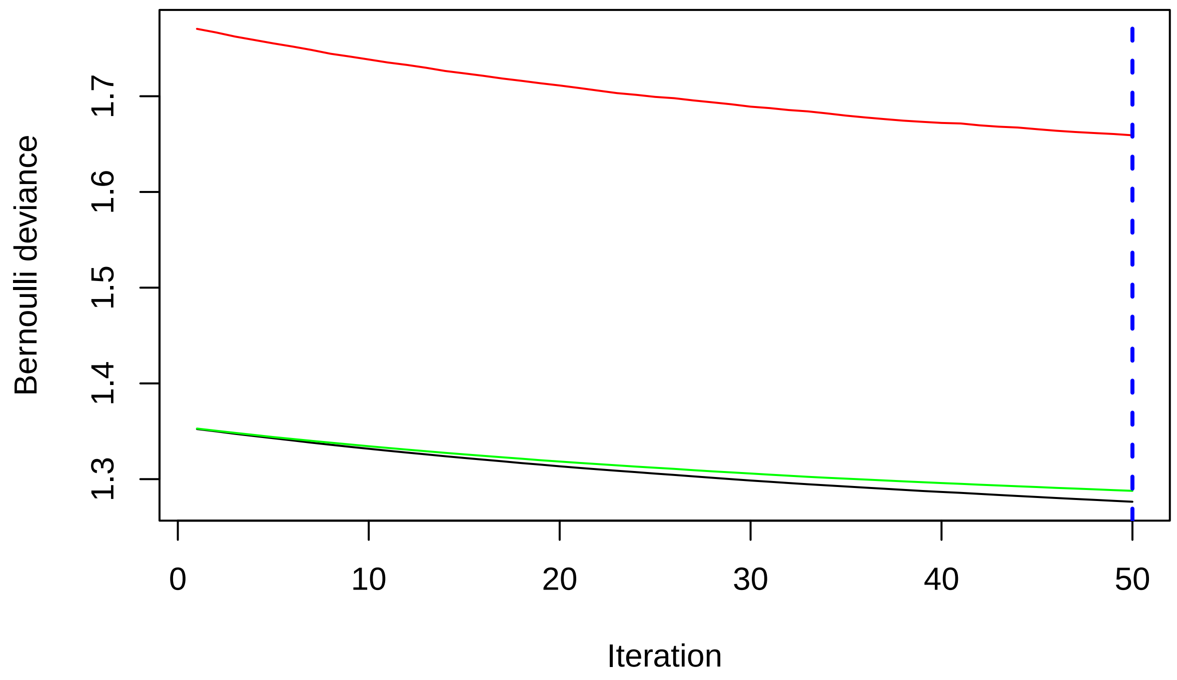
```
##
##    0    1
## 7379 7379
```

Splitting data into 85% train and 15% test within our balanced data

```
set.seed(6782)


Split_data_sample <- createDataPartition(balanced_spam$default,p=.85,list=FALSE,times=1)

Training_sample <- balanced_spam[Split_data_sample,]

Validation_sample <- balanced_spam[-Split_data_sample,]
```

Building the model again using the new representative dataset

```
library(gbm)
gbm_model <- gbm(default ~ ., data = Training_sample, distribution = "bernoulli",
                 n.trees = 50, interaction.depth = 5, shrinkage = 0.02,
                 bag.fraction = 0.80, train.fraction = 0.85,
                 n.minobsinnode = 25,cv.folds = 10)

cv_error<-gbm.perf(gbm_model,method="cv")
```



```
cv_error
```

```
## [1] 50
```

Predicition on Validation data

```
predictions_sample <- predict(gbm_model, newdata = Validation_sample, n.trees = 50, type = "response")


# Here the threshold value is set to be 35% ,that we are becoming more pickier while selecting customer

predictions_sample<-ifelse(predictions_sample>0.35,1,0)


Final_data_sample<-cbind(Validation_sample,predictions_sample)


Final_data_sample$predictions_sample<-as.factor(Final_data_sample$predictions_sample)

Final_data_sample$default<-as.factor(Final_data_sample$default)

#Building Confusion Matrix

confusionMatrix_sample<-confusionMatrix(Final_data_sample$default,Final_data_sample$predictions_sample)

confusionMatrix_sample
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 319 787
##          1 120 986
##
##                Accuracy : 0.59
##                  95% CI : (0.5691, 0.6106)
##     No Information Rate : 0.8015
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1799
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7267
##             Specificity : 0.5561
##          Pos Pred Value : 0.2884
##          Neg Pred Value : 0.8915
##              Prevalence : 0.1985
##          Detection Rate : 0.1442
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.6414
##
##        'Positive' Class : 0
##
```

Calculating Precision for the new model

Precision<-TP/(TP+FP)= 1004/(1004+102)=90.77%

Running the model on test data

```
Test_data<-read.csv("testv3.csv")

Test_Prediction<-predict(gbm_model, newdata = Test_data, n.trees = 50, type = "response")


Test_Prediction<-ifelse(Test_Prediction>0.35,1,0)


Test_datafile<- cbind(Test_data,Test_Prediction)


Default_test<- subset(Test_datafile, Test_datafile$Test_Prediction == 1)


#Writing data frame to a new csv file containing the list of customers who defaults the loan

#write.csv(Default_test,file="defaulted_test_customers.csv")
```

From the above predictions, only customers who would default will be considered and the amount of loss will be predicted using Regression models.