Application of Data Mining in Underwriting Department

Group Project: MIS-64037- Advanced Data Mining and Analytics

Under Guidance of,

Prof. Dr. Rouzbeh Razavi

Submitted by:

Group 4

Gujja Rithin Rao

Padade Srushti

Deshpande Amruta

Huang Chujun

Table of Contents

Project Goal:

The goal of the project is to design an appropriate asset-management strategy to optimize the economic capital utilization and minimizing the risk to financial investors.

- Objective:

The primary objective of this project is to make on decision on allocation of loan money to the applications received by underwriting department of the bank based on the evaluation of three different scenarios described in the provided information. However, in this report we also elucidate the strategy to allocate the loan money and criteria used to do so by implementing different regression models and choosing the best to reduce the total losses incurred by the bank. The three scenarios are briefly described below:

Scenario 1:

The total capital of $1.4B are supposed to be allocated for giving out loans with fixed term (5-years), and the annual simple interest rate (4.32%) for all approved customers.
With the provided training dataset which contains a list of variables and the "loss" variable which is defined as the percentage of the loan at which the customer was defaulted. The "loss" is expressed in percentage so if loss is 10, then it means that the customer has paid pack 90% of the capital but zero interests. From 25471 customers listed in "test_scenario1_2.csv" file, we approve the loans for the customers to maximize the profit by training the model.

Scenario 2:

In this case, the total budget allocated is $450 million. Keeping all the other conditions same. Again, we try different models to find best one to achieve the maximum profit for the bank and designate 1 or 0 to each customer for approval or rejection of the loan, respectively in the csv file.

Scenario 3:

In this case, each customer has proposed an interest rate (column "Proposed_Interest_Rate"). The total budget for this scenario is $1.4B available to give loans. The requested loan amounts and proposed interest rates are included in the file "test_scenario3.csv"

Data Overview:

The input data consists of information on 80,000 customers in relation to 763 dependent variables to help the bank define their eligibility for financial lending. The first step in data exploration analysis is to locate the empty cells or NA values. Among all the NA values, the rows with all the cells with NA values are completely eliminated, while the rows with few NA values are handled so that NA is replaced with the median of the column to keep the variability of the data constant. The alternative approaches to manage NA values would have been use of mean or KNN regression imputation. However, there are too many variables to calculate k for individual NA values, hence we refrain from using KNN imputation, and use of mean would have changed the variability of the data. So, we settle with the approach to replace the median in some cells with NA values. To avoid losing data, we considered keeping the outliers obtained from the box plot.

In the next step, we perform linear dimensional reduction using lasso with minimizing sum of squares given by below equation which was first published by Tibshirani et al.[1]:

$$\sum_{i=1}^{N}(y_i - \sum_{j} x_{ij\beta_j})^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

Where λ is the tuning parameter that controls the shrinkage of the l1-regression and βj is the constraint parameter to minimize the sum of squares. On normalization of data with caret package we can reduce the variables to 81. The variables with l1 regression coefficients are chosen for building the models.

A new variable called loss binary (0 or 1) was created to be used in the classification model. Those customers with defaulted loans are represented by binary digit 1 and without any defaulted loan are represented by 0. This new variable is then added to the training data set with reduced number of variables. Similar data exploration analysis was run on both test datasets where the NA values were replaced with median of the column.

- Data Cleaning:

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

Here, we have a dataset storing an information of a certain bank customers which contains a list of variables and the target variable that is "loss". The input data consists of information on 80,000 customers in relation to 763 dependent variables to help the bank define their eligibility for financial lending. Here we can see that the dataset has some missing values.

Imputation and removing NA values:

```
Bank_raw_NA <- Bank_raw
for(i in 1:ncol(Bank_raw)){
  Bank_raw_NA[is.na(Bank_raw_NA[,i]), i] <- median(Bank_raw_NA[,i], na.rm = TRUE)
}
```

```
anyNA(Bank_raw)
```

```
## [1] TRUE
```

Here we have located the empty cells or NA values. Among all the NA values, the rows with all the cells with NA values are completely eliminated, while the rows with few NA values are handled so that NA is replaced with the median of the column to keep the variability of the data constant. To avoid losing data, we considered keeping the outliers.

- Data Transformation:

Data transformation is the process of converting data from one format or structure into another format or structure, which includes Normalization and aggregation, reducing data, etc.

Normalization:

The data needs to be formatted in a normalized manner, so the data changes the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. The normalized data is used only to compute the dimensionality reduction technique LASSO. Other modelling techniques we are using works the same for unnormalized data. We are using the Z-scored Normalization technique to scale the dataset.

```
Bank_Scale <- preProcess(Bank_raw_NA,method = c("center","scale"))
Norm_Bank <- predict(Bank_Scale,Bank_raw_NA)
```

Building Linear Model:

```
linearMod <- lm(loss~., data = Norm_Bank)
```

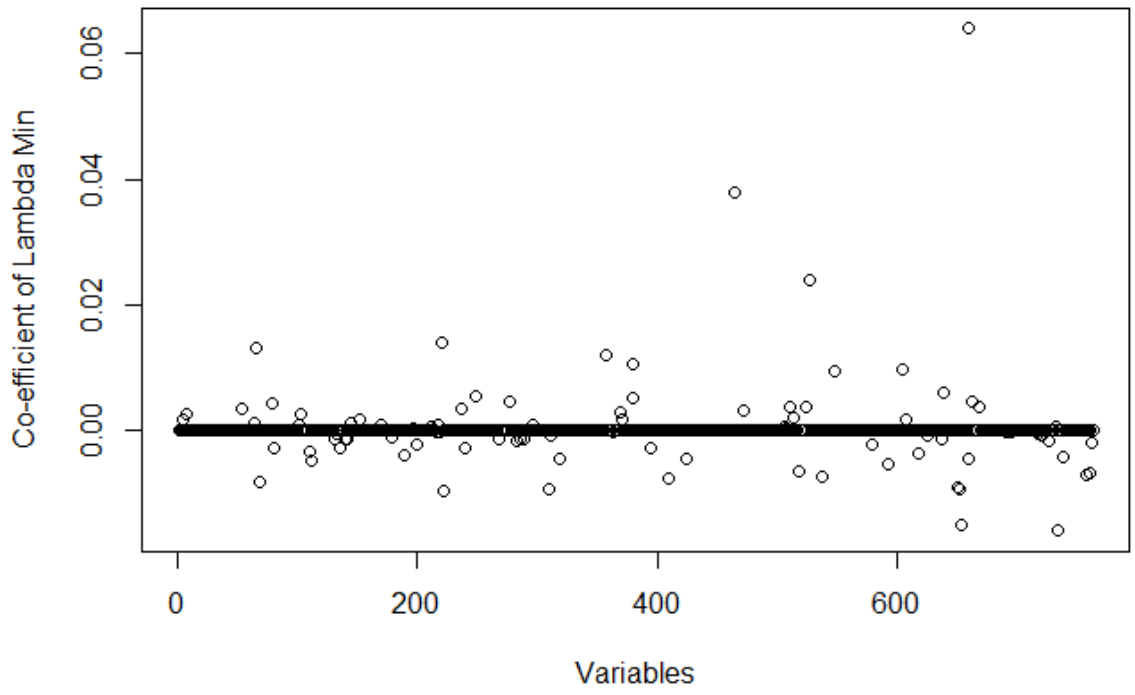
```
glance(linearMod)
```

r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.value <dbl>
0.01766097	0.009626466	0.9951751	2.19814	1.701119e-59

Dimensionality Reduction:

```
Lasso_Model<- cv.glmnet(loss~., data = Norm_Bank, alpha = 1, nlambda = 100)
```

Lambda which gives minimum mean cross validated error is **Lambda.min.**



Variable Selection:

```
```{r, eval = FALSE}
Train <- select(Bank_raw_NA,f3,f5,f55,f67,f70,f80,f82,f102,f103,f111,f112,f132,f136,f145,f153,f180,f190,f198,f200,f218
,f221,f222,f238,f241,f251,f270,f281,f286,f290,f293,f294,f295,f296,f314,f315,f323,f361,f373,f374,f383,f384,f398,f413,f4
28,f471,f479,f514,f518,f522,f526,f533,f536,f546,f556,f588,f604,f617,f620,f631,f650,f652,f655,f663,f665,f666,f673,f674,
f676,f682,f704,f705,f709,f734,f740,f747,f752,f771,f775,f776,loss)
```
```

Factorizing target variable:

From the dataset we have our target variable “loss”. “loss” defines the percentage of the loan at which the customer was defaulted. If “loss” is zero you can imagine that the customer has fully paid back the capital and interest. If the “loss” is greater than zero, it means that the customer has defaulted. “loss” is expressed in percentage so if loss is 10, then it means that the customer has paid pack 90% of the capital but zero interests.
A new variable called loss binary (0 or 1) is created to be used in the classification model. Those customers with defaulted loans are represented by binary digit 1 and without any defaulted loan are represented by 0. This new variable is then added to the training data set with reduced number of variables.

```
lossbin<-rep(0,80000)

for(i in 1:nrow(Train)){
  if(Train[i,80] > 0){
    lossbin[i] <- 1
  }
}

Train$lossbin <- lossbin
Train$lossbin <- as.factor(Train$lossbin)
```

Balancing dataset:

The dataset we have is biased data with ratio of defaulted customers to non-defaulted customers as 1:10. This unequal type of data the model will surely generate a bias. Hence to eliminate model biasing we are randomly gathering equal amount of defaulted to non-defaulted number of customers data.

```
NoLoss <- Train %>% filter(lossbin==0)
Loss <- Train %>% filter(lossbin==1)

Random_NoLoss <- sample(1:nrow(NoLoss),nrow(Loss),replace = FALSE)
NoLoss_Train_Data <- NoLoss[Random_NoLoss,]
Train_Unbaised <- rbind(NoLoss_Train_Data, Loss)
TrainClass_Unbaised <- Train_Unbaised[,-80]
```

- Data Partitioning:

Partitioning is the database process where very large data is divided into smaller parts. Here, we have divided data into Training and Testing data for both classification and regression model.

Classification:

To check the performance of the Model we are partitioning the unbaised data. Considering 80% data for TRAINING the model and 20% as VALIDATION data.

```
set.seed(2019)

index_C <- createDataPartition(TrainClass_Unbaised$lossbin, p = 0.8, list = FALSE)
Train_Class <- TrainClass_Unbaised[index_C,] # Train data
Valid_Class <- TrainClass_Unbaised[-index_C,] # Valid data
```

Regression:

To build a regression model we are considering only the data of the customers who are defaulted. Using this data, we can see the percentage by which the customer is defaulted. i.e. from variable “loss”.

```
## Considering variables where loss is only greater than 0
Loss_Data <- filter(Train_Unbaised,lossbin == 1)
Loss_Data <- Loss_Data[,-81]
```

```
set.seed(2019)

index_R <- createDataPartition(Loss_Data$loss, p = 0.8, list = FALSE)
Train_Reg <- Loss_Data[index_R,]
Valid_Reg <- Loss_Data[-index_R,]
```

Modelling Strategy:

Classification and Regression model are used to determine whether a customer loan has been approved or declined.

- Classification:

Here we use Classification to classify defaulters and non-defaulters. In Classification a target function maps each attribute set to one of the predefined class labels. Performance of the regression models is measured with factors like Sensitivity, Accuracy, Specificity, etc. This confusion matrix is derived from the cross table. Classification models that we are using here are: Random Forest, Support Vector Machine, Generalized Linear Model.

Random Forest:

Random Forest is an ensemble learning method for classification, that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Ensemble models in machine learning combine the decisions from multiple models to improve the overall performance. The first algorithm for random decision forests was created by Tin Kam Ho.[2] An extension of the algorithm was developed by Leo Breiman and Adele Cutler who registered "Random Forests" as a trademark.[3] The Random Forest Model built for the classification of defaulters and non-defaulters, gives the below confusion matrix.

Code:

```
RF_Model <- train(lossbin~., data = Train_Class, method = 'rf', ntree = 300)

RF_Valid <- predict(RF_Model, Valid_Class)

RF_ConfusionMatrix <- confusionMatrix(RF_Valid, Valid_Class$lossbin)
```

Cross Table:

	RF_Valid		
Valid_Class$lossbin	0	1	Row Total
0	905	570	1475
	0.614	0.386	0.500
	0.665	0.359	
	0.307	0.193	
1	456	1019	1475
	0.309	0.691	0.500
	0.335	0.641	
	0.155	0.345	
Column Total	1361	1589	2950
	0.461	0.539	

Confusion Matrix:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  905  456
         1  570 1019

               Accuracy : 0.6522
                 95% CI : (0.6347, 0.6694)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3044

 Mcnemar's Test P-Value : 0.000419

            Sensitivity : 0.6136
            Specificity : 0.6908
         Pos Pred Value : 0.6650
         Neg Pred Value : 0.6413
             Prevalence : 0.5000
         Detection Rate : 0.3068
   Detection Prevalence : 0.4614
      Balanced Accuracy : 0.6522

       'Positive' Class : 0
```

With the Negative Prediction Value of 0.6413, Accuracy of 65.22%, while Sensitivity is 61.36% and Specificity of 69.08%.

Support Vector Machine:

Support Vector Machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier. SVMs can efficiently perform a non-linear classification using what is called the 'kernel trick', implicitly mapping their inputs into high-dimensional feature spaces.[4]

The Support Vector Machine Model built for the classification of defaulters and non-defaulters, gives the below,

Code:

```
SVM_Model<-train(lossbin~., data = Train_Class, method="svmLinear")

SVM_Valid <- predict(SVM_Model, Valid_Class)
SVM_ConfusionMatrix <- confusionMatrix(SVM_Valid, Valid_Class$lossbin)
```

Cross Table:

	SVM_Valid		
Valid_Class$lossbin	0	1	Row Total
0	874	601	1475
	0.593	0.407	0.500
	0.673	0.364	
	0.296	0.204	
1	425	1050	1475
	0.288	0.712	0.500
	0.327	0.636	
	0.144	0.356	
Column Total	1299	1651	2950
	0.440	0.560	

Confusion Matrix:

```
          Reference
Prediction    0    1
         0  874  425
         1  601 1050

               Accuracy : 0.6522
                 95% CI : (0.6347, 0.6694)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3044

 Mcnemar's Test P-Value : 4.671e-08

            Sensitivity : 0.5925
            Specificity : 0.7119
         Pos Pred Value : 0.6728
         Neg Pred Value : 0.6360
             Prevalence : 0.5000
         Detection Rate : 0.2963
   Detection Prevalence : 0.4403
      Balanced Accuracy : 0.6522

       'Positive' Class : 0
```

With the Negative Prediction Value of 0.6360, Accuracy of 65.22%, while Sensitivity is 59.25% and Specificity of 71.19%.

Generalized Linear Model:

Generalized Linear Model is flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a link function and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.
Generalized linear models were formulated by John Nelder and Robert Wedderburn as a way of unifying various other statistical models.
The Generalized Linear Model built for the classification of defaulters and non-defaulters, gives the below confusion matrix,

Code:

```
```{r, eval = FALSE}
Class_Model <- train(lossbin~.,data = Train_Class, method ='glmnet',
                     tuneGrid = expand.grid(alpha = 0,lambda = c(seq(0.1, 2, by = 0.1), seq(2, 5, 0.5), seq(5, 25,
1))))

Class_Valid <- predict(Class_Model, Valid_Class)
Class_ConfusionMatrix <- confusionMatrix(Class_Valid, Valid_Class$lossbin)
```
```

Cross Table:

Valid_Class$lossbin	Class_Valid 0	1	Row Total
0	889	586	1475
	0.603	0.397	0.500
	0.646	0.373	
	0.301	0.199	
1	488	987	1475
	0.331	0.669	0.500
	0.354	0.627	
	0.165	0.335	
Column Total	1377	1573	2950
	0.467	0.533	

Confusion Matrix:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 889 488
         1 586 987

               Accuracy : 0.6359
                 95% CI : (0.6183, 0.6533)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2719

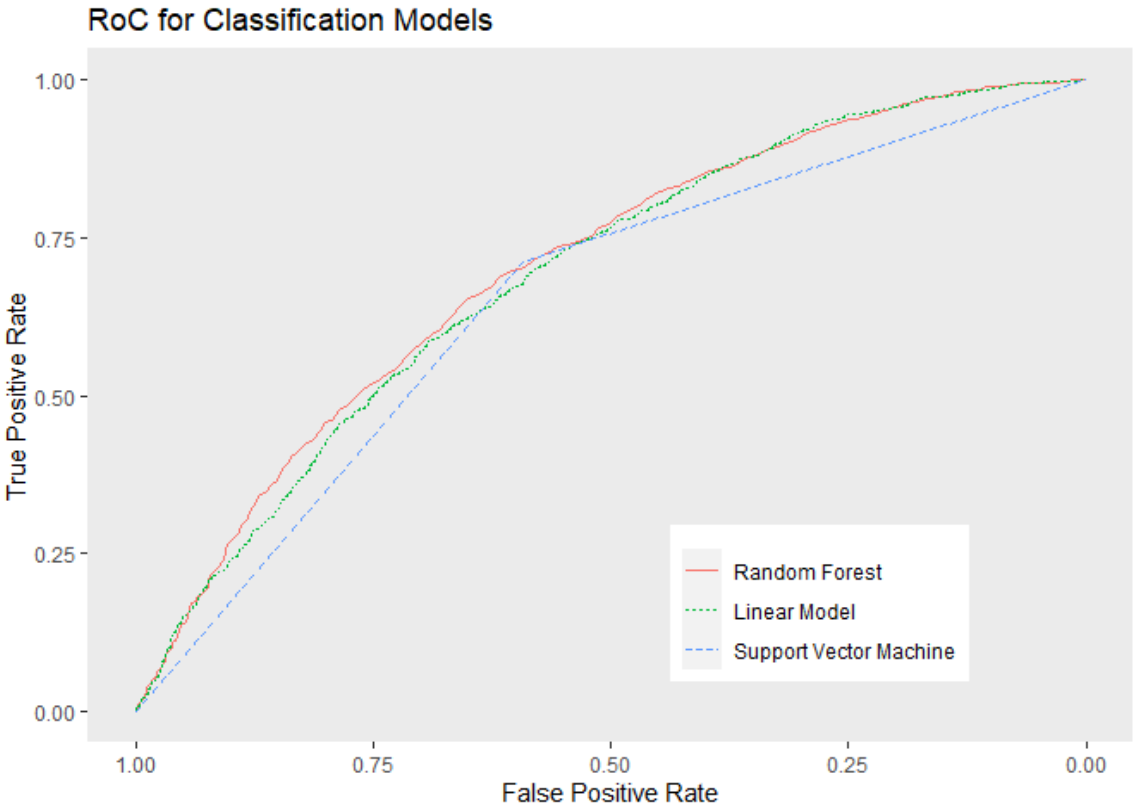
 Mcnemar's Test P-Value : 0.003078

            Sensitivity : 0.6027
            Specificity : 0.6692
         Pos Pred Value : 0.6456
         Neg Pred Value : 0.6275
             Prevalence : 0.5000
         Detection Rate : 0.3014
   Detection Prevalence : 0.4668
      Balanced Accuracy : 0.6359

       'Positive' Class : 0
```

With the Negative Prediction Value of 0.6275, Accuracy of 63.59%, while Sensitivity is 60.27% and Specificity of 66.92%.

Insights:



Receiver operating characteristic (RoC) curve for the three built classification model is plotted above. By looking at the curve we can see that the maximum area is covered by the Random Forest Model. Thus, we will be using the Random Forest model for our test data prediction which is having highest accuracy of 65.22%, sensitivity of 61.36%, precision of 66.50% and specificity of 69.08%.

- Regression:

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables. We are using Regression analysis to predict the loss of the bank due to the defaulters. Performance of the regression models is measured with factors- RMSE (root-mean-square error) and Coefficient of determination (R2). Regression models that we are using here are, Lasso Regression, Random Forest, XGBoost.

Lasso Regression:

LASSO (least absolute shrinkage and selection operator) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.
It was originally introduced in geophysics literature in 1986, and later independently rediscovered and popularized in 1996 by Robert Tibshirani[1], who coined the term and provided further insights into the observed performance.

Code:

```
```{r, eval = FALSE}
RegLasso_Model <- train(loss~., data = Train_Reg, method ='glmnet', trControl = trainControl(method = "cv",number = 10),
                        tuneGrid = expand.grid(alpha = 1, lambda = c(seq(0.001, 0.5, by = 0.01))), metric = "Rsquared")
```
```

Performance Measure:

Train_R-Square <dbl>	Valid_R-Square <dbl>	Train_RMSE <dbl>	Valid_RMSE <dbl>
0.3012664	0.2972472	9.783261	9.777089

The Lasso Regression Model built to calculate the loss of the bank due to defaulters has the below performance measuring factors,

R2 = 0.29

RMSE= 9.77

Random Forest:

Random Forest is method, that operates by constructing a multitude of at training time and outputting the class that is the of the classes (classification) or mean prediction (regression) of the individual trees.
The first algorithm for random decision forests was created by. An extension of the algorithm was developed by and who registered "Random Forest".

Code:

```
```{r, eval = FALSE}
RegRF_Model<-train(loss~.,data = Train_Reg, method ='rf', ntree = 500, tuneGrid = expand.grid(.mtry = c(seq(15, 40, by = 5))))
```
```

Performance Measure:

Train_R-Square <dbl>	Valid_R-Square <dbl>	Train_RMSE <dbl>	Valid_RMSE <dbl>
0.9362194	0.3316741	4.138721	9.59937

R2 = 0.331
RMSE= 9.59

XGBoost:

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

Code:

```
```{r, eval = FALSE}
RegXGBoost_Model <- train(loss~., data = Train_Reg, method = "blackboost")
```
```

Performance Measure:

Train_R-Square <dbl>	Valid_R-Square <dbl>	Train_RMSE <dbl>	Valid_RMSE <dbl>
0.4154343	0.2959364	9.067834	9.822344

R2 = 0.29
RMSE= 9.82

Insights:

	Train_R-Square <dbl>	Valid_R-Square <dbl>	Train_RMSE <dbl>	Valid_RMSE <dbl>
Lasso	0.3012664	0.2972472	9.783261	9.777089
Random_Forest	0.9362194	0.3316741	4.138721	9.599370
XGBoost	0.4154343	0.2959364	9.067834	9.822344

By looking at the table Random Forest regression model works very well on the training dataset whereas the performance on the validation data the model performs poorly, which determines overfitting. Thus, only the Lasso regression model performance is similar for both training and test dataset. So, we will be using Lasso regression model with the R-Square value of 0.2972 and less RMSE value of 9.77.

Estimation of model's performance:

Our final classification and regression models selected are used here to predict whether the customer will default or not, also we find by what percentage will they default.

- Classification - Random Forest:

Predicting the Default and Non-default customers in test dataset based on the Random forest model (best classification model) we built previously.

```
FinalRF_Pred <- predict(RF_Model, TestData, type = 'prob')
FinalRF_PredDF <- as.data.frame(FinalRF_Pred)
```

```
FinalRF_Pred_3 <- predict(RF_Model, Prop_interest, type = 'prob')
FinalRF_PredDF_3 <- as.data.frame(FinalRF_Pred_3)
```

- Regression - Lasso:

Evaluating the regression models based on the performance on the training and testing data. From all three models, performance of LASSO model is similar for both data sets.

```
FinalLassoReg_Pred <- predict(RegLasso_Model, TestData)
FinalLassoReg_PredDF <- as.data.frame(FinalLassoReg_Pred)
```

```
FinalLassoReg_Pred_3 <- predict(RegLasso_Model, Prop_interest)
FinalLassoReg_PredDF_3 <- as.data.frame(FinalLassoReg_Pred_3)
```

- Formulation for Loss and Profit:

We are using the Random forest Classification model to find Probability of Default (PD) and the Lasso Regression model for computing Loan Given Default (LGD).

To speculate whether a customer should be approved for the loan or rejected we are computing below parameters and computed profit and loss accordingly.

Computing Profit and Loss:

Expected Loss = (PD * Loan Amount * LGD)
Expected Profit = ((1-PD) * Loan Amount * Year * Percentage of Interest)
Profit gained = Expected profit – Expected loss

If the expected profit is greater than the expected loss, then the bank would see profits while minimizing the default of the customer.

```{r, eval=FALSE}
# Calculating the Expected loss if the loan is approved
Computed_Loss <- NULL
for(i in 1:nrow(Bank_Test)){
  Computed_Loss[i] <- (FinalRF_PredDF[i,2]*Bank_Test$requested_loan[i]*(FinalLassoReg_PredDF[i,1]/100))
}

#Expected Profit for the span of 5 year at the constatnt annual interest rate of 4.32%
Computed_Profit <- NULL
for(i in 1:nrow(Bank_Test)){
  Computed_Profit[i] <- (FinalRF_PredDF[i,1]*Bank_Test$requested_loan[i]*5*(4.32/100))
}
```

```{r, eval=FALSE}
#Similarly for the Test data 2 where we are calculating the expected profit and expected loss for the requested loan amount with prosed intrest rate.

Prop_Loss <- NULL
for(i in 1:nrow(Prop_interest_Test)){
  Prop_Loss[i] <- (FinalRF_PredDF_3[i,2]*Prop_interest_Test$requested_loan[i]*(FinalLassoReg_PredDF_3[i,1]/100))
}

Prop_Profit <- NULL
for(i in 1:nrow(Prop_interest_Test)){
  Prop_Profit[i] <- (FinalRF_PredDF_3[i,1]*Prop_interest_Test$requested_loan[i]*5*(Prop_interest_Test$Proposed_Intrest_Rate[i]/100))
}
```

- Scenarios:

Deliverables for three different scenarios:

Scenario 1:

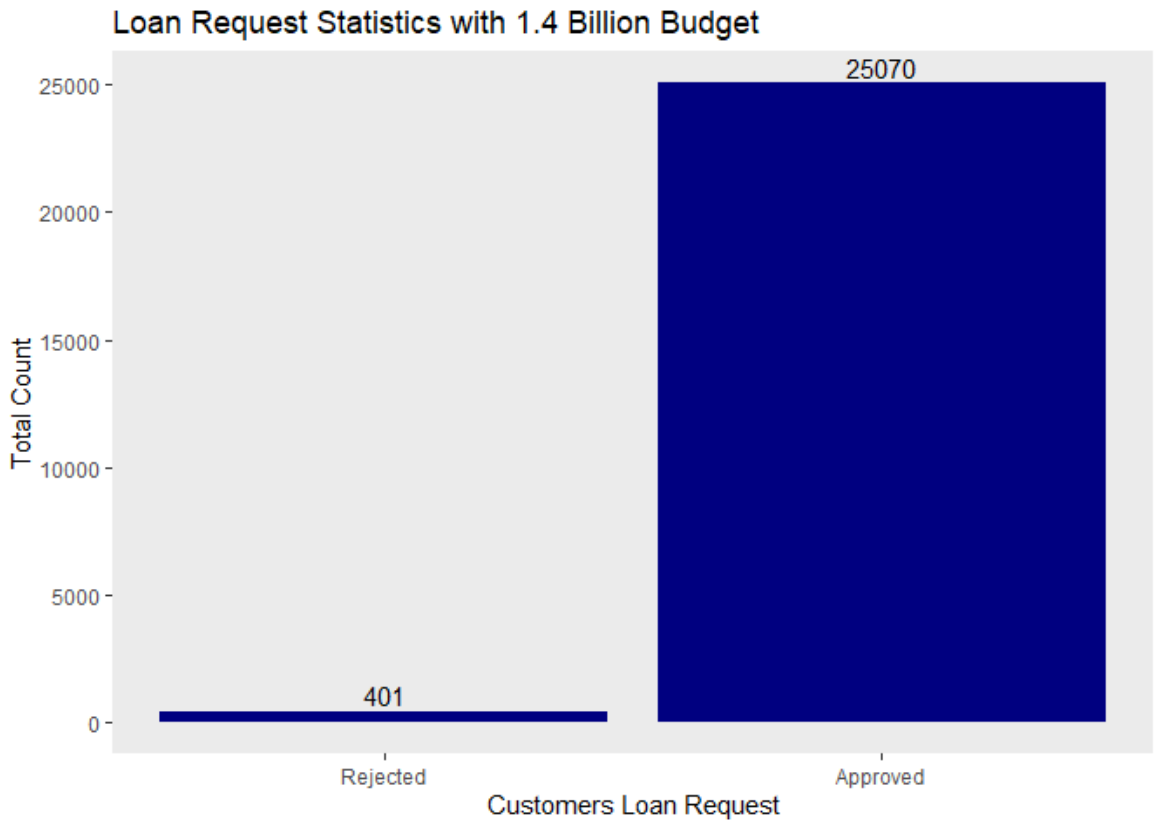
We have underwriting for loan request for the customers with the overall Bank capital of $1.4 Billion, annual interest rate is 4.32% for a fixed term of 5 Years. We are arranging the customers sorted order of maximum profit gained. Also, we are formulating cumulative sum with respect to requested loan of each customers.

```{r}
sum(Bank_Test$requested_loan)
```

The total amount of all the loan request is approximately $1.27 Billion.

```{r, eval=FALSE}
Test_temp <- Bank_Test %>% mutate(profit = Computed_Profit, loss = Computed_Loss) %>%
  mutate(totalgain = profit - loss) %>% arrange(desc(totalgain)) %>% mutate(cumsum = cumsum(requested_loan))

Predict_1 <- NULL
for(i in 1:nrow(Bank_Test)){
  if(Test_temp$totalgain[i] > 0){
    Predict_1[i] <- 1
  }
  else{
    Predict_1[i] <- 0
  }
}
```



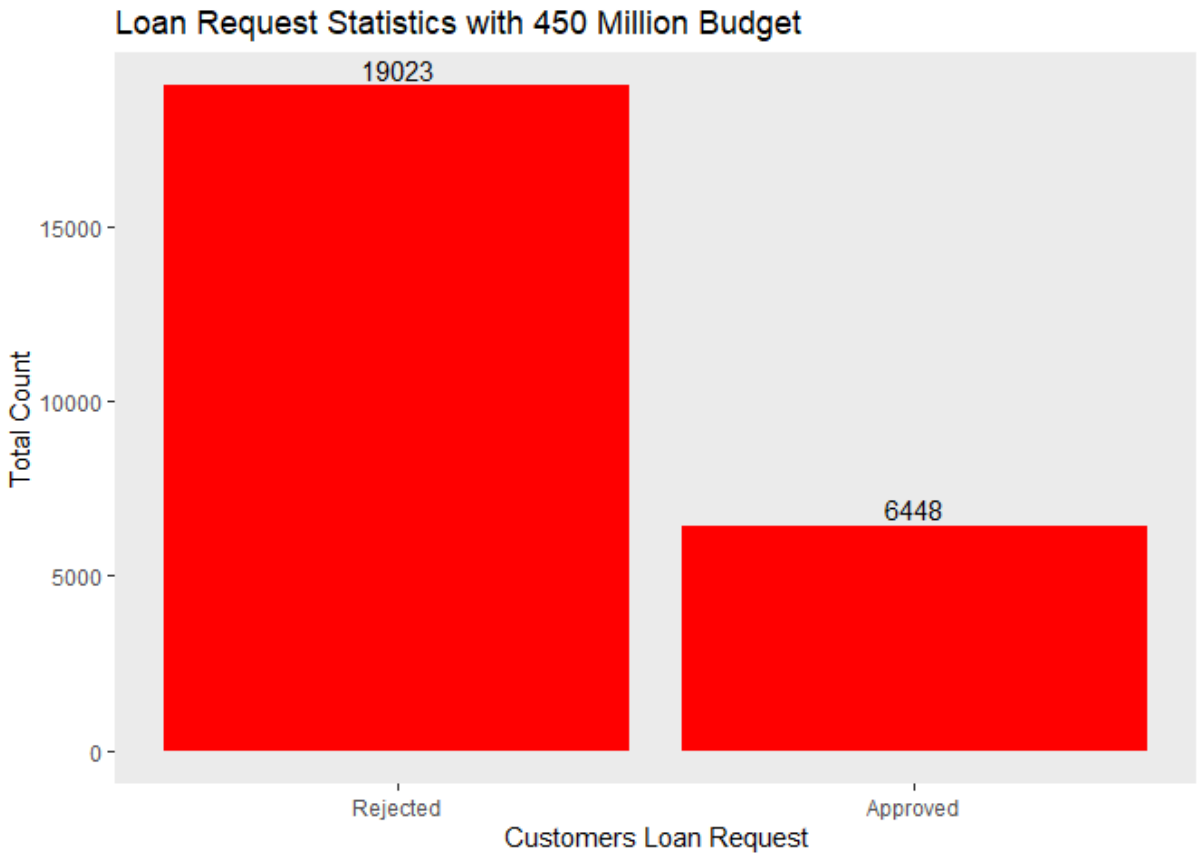
Total Loan applications approved under scenario 1 are 25070 with the budget spend of approximately $1.25 Billion.

Scenario 2:

We have underwriting for loan request for the customers with the overall bank capital of $450 million, annual interest rate is 4.32% for a fixed term of 5 years. We are arranging the customers sorted order of maximum profit gained as the requested loan amount is higher than the budget. Hence the computed cumulative sum with respect to requested loan of each customers will be used to filter the customers with maximum gain and under budget.

```{r, eval=FALSE}
Predict_2 <- NULL

for(i in 1:nrow(Bank_Test)){
  if(Test_temp$cumsum[i] < 450000000){
    Predict_2[i] <- 1
  }
  else{
    Predict_2[i] <- 0
  }
}
```



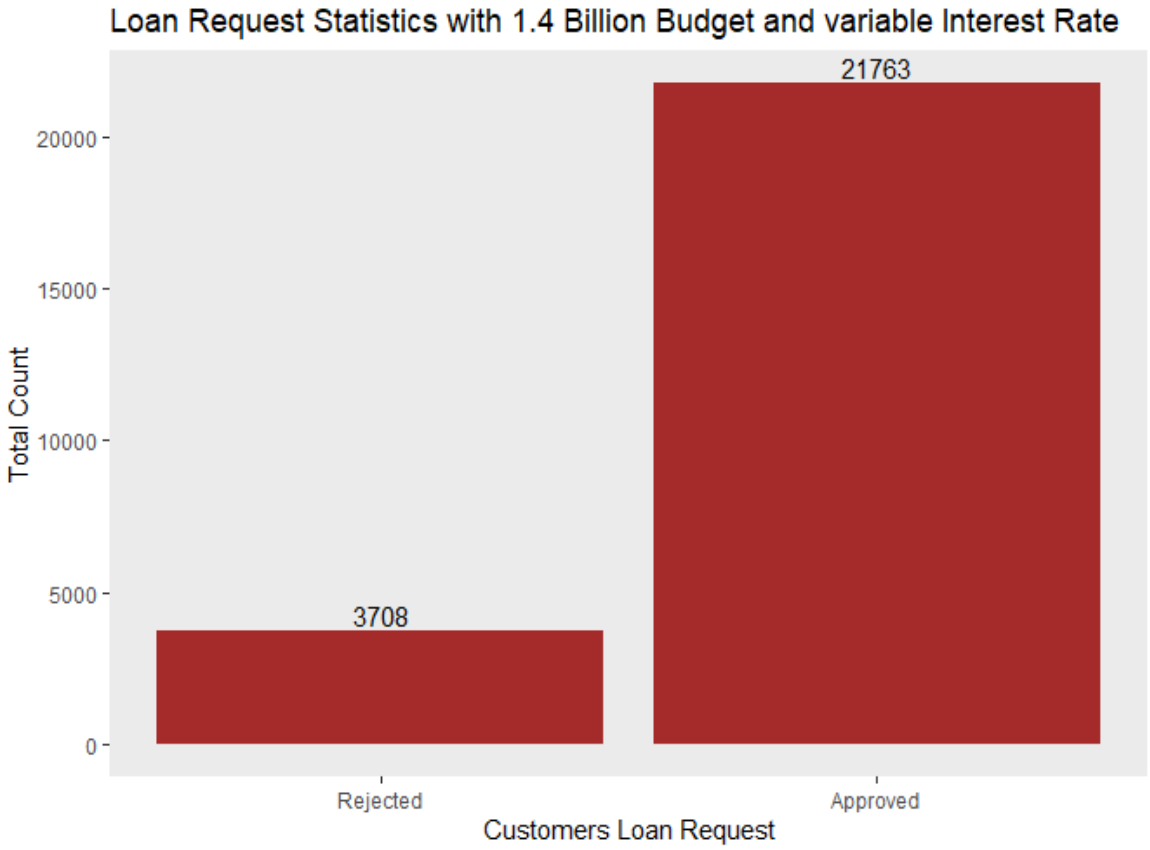
Total Loan applications approved under scenario 2 are **6448** with the budget spend of **$449,943579 Million**.

Scenario 3:

We have underwriting for loan request for the customers with the overall *Bank capital* of **$1.4 Billion**, *annual interest rate* is **proposed by the customer** thus it varies for each customer with the fixed term of **5 Years**.

```
```{r, eval=FALSE}
PropTest_temp <- Prop_interest_Test %>% mutate(profit = Prop_Profit, loss = Prop_Loss) %>%
  mutate(totalgain = profit - loss) %>% arrange(desc(totalgain)) %>% mutate(cumsum = cumsum(requested_loan))

#Decision Making
Decision <- NULL
for(i in 1:nrow(Prop_interest_Test)){
  if(PropTest_temp$totalgain[i] > 0){
    Decision[i] <- 1
  }
  else{
    Decision[i] <- 0
  }
}
```
```



Total Loan applications approved with variable Interest rate under scenario 3 are 21763 whereas, rejected are 3708.

Conclusion:

As an underwriting department employee, the decision of approval and rejection of loan for the customers in three different scenarios is outlooked with regression and classification model. The classification uses Random Forest and Lasso model for regression. For the first scenario where bank has budget of $1.4 billion, the total number of customers were 25471 and out of which the loan is approved for 25070 customers. Second scenario is like the first except for the lower budget of $450 million. In this only 6448 customers get their loan approved. And for the third, each customer is proposing the interest rate and bank has budget of $1.4 billion. The number of customers approved were 21763.

Thus, while trying to cross over the drift with traditional banking method, looking at reduction of the consumption of economic capital, to an asset-management perspective, while optimizing the risk to the financial investor.

References:

1. Tishbirani, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* vol. 58 267–288 (1996).
2. Kam Ho, T. Random Decision Forests. *Proc. 3rd Int. Conf. Doc. Anal. Recognition, Montr. Quebec, Canada* **1**, 278–282 (1995).
3. Breiman, L. ST4_Method_Random_Forest. *Mach. Learn.* **45**, 5–32 (2001).
4. Cortes, C. & Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.* **20**, 273–297 (1995).

Author Contribution:

NAME	CODE	REPORT	PRESENTATION
Gujja Rithin	✓		✓
Padade Srushti	✓	✓	
Deshpande Amruta	✓	✓	
Huang Chujun	✓		✓