Group project (Advance Data mining and Predictive Analytics) - Group-8

2023-05-04

#Loading necessary poackages for current project:

```
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-7

library(pls)

##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2

## The following object is masked from 'package:stats':
##
##     loadings

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(esquisse)
library(ggplot2)
library(randomForest)

## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

#Loading train dataset:

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

#Creating a default column based on, if loss is 0 then 0 and if loss is more than 0 then default is 1

```
bank_model$default <- factor(ifelse(bank_model$loss > 0, 1, 0))

bank_model$loss <- (bank_model$loss / 100)
```

#Checking the missing values in the data set:

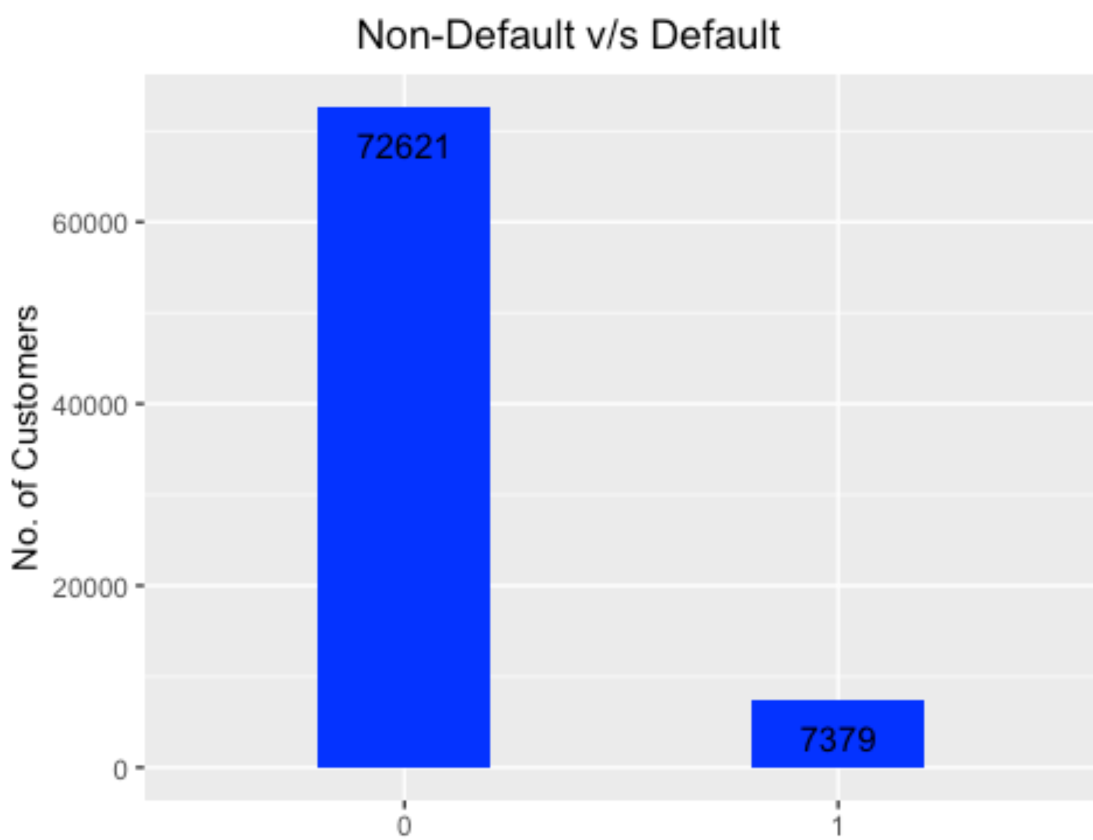
```
row_missing <- rowMeans(is.na(bank_model))

min_missing_values<-min(row_missing)
min_missing_values

## [1] 0

max_missing_values<-max(row_missing)
max_missing_values

## [1] 0.4790576
```

we have maximum missing values in the dataset with percentage of 47%

Visualizations of the dataset:

```
ggplot(bank_model, aes(x=factor(default))) +
  geom_bar(stat="count", width=0.4, fill="blue") +
  labs(title="Non-Default v/s Default") +
  labs(x="", y="No. of Customers") +
  theme(plot.title = element_text(hjust = 0.4)) +
  geom_text(stat='count', aes(label=..count..), vjust=2)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2
3.4.0.
## ℹ Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



#Removing the zero-variances variables and Preprocessing the dataset by removing highly correlated and imputing missing values using "corr" and "medianimpute":

```
zero_var_indices <- nearZeroVar(bank_model[ ,-c(763,764)])

data_cleaned <- bank_model[, -zero_var_indices]

bank_preprocess <- preProcess(data_cleaned[ ,-c(739,740)], method = c("corr",
"medianImpute"))

new_bank_model <- predict(bank_preprocess, data_cleaned)
```

#Removing zero-variance,highly corr variables and imputed missing values : we have new data set "new_bank_model" with 248 attributes.

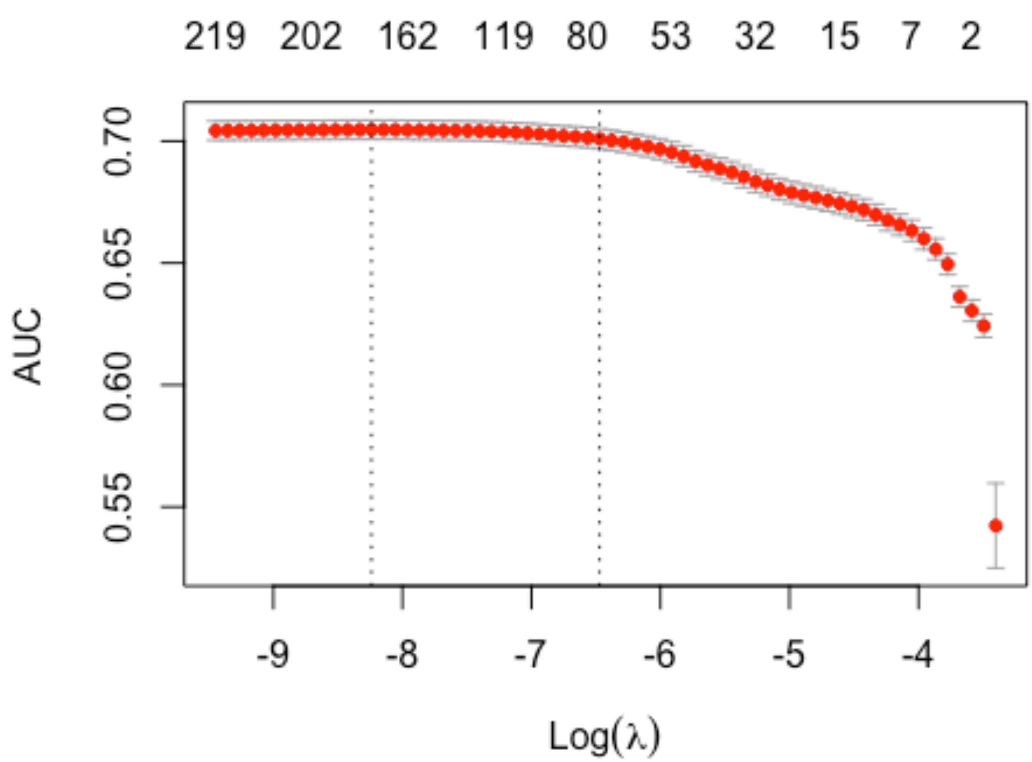
#1.CLASSIFICATION MODEL:

#We first need to run a classification model to classify how many customers are actually defaulting

#We used Lasso and Principle Component Analysis(PCA) for variable selection

#a).Lasso Model: we now run the new_bank_model with 248 attributes for variable selection:

```
set.seed(123)

y <- as.vector(as.factor(new_bank_model$default))

x <- data.matrix(new_bank_model[,-c(247,248)])

lasso_model<- lasso_model<- cv.glmnet(x, y, alpha = 1,  preProcess = c("center", "scale"), family = "binomial", nfolds = 10, type.measure = "auc")

plot(lasso_model)
```




```
lasso_model$lambda.min

## [1] 0.0002640483
```

#Minimum Lambda value returned a total of 180 attributes out of 248 attributes:

#We convert coefficients returned in lasso model into dataframe:

```
# Return the coefficients for the lasso regression at the minimum lambda
value:
coef <- coef(lasso_model, s= "lambda.min")

#Convert the coefficient values into a dataframe:
new_bank_coef<- data.frame(name = coef@Dimnames[[1]][coef@i + 1], coefficient
= coef@x)

#Removing negatives values using "abs" function:
new_bank_coef$coefficient <- abs(new_bank_coef$coefficient)

#Re-arranging the data frame in decreasing order:
new_bank_coef[order(new_bank_coef$coefficient, decreasing = TRUE), ]

##              name  coefficient
## 1   (Intercept) 7.211311e+00
## 28         f129 2.418453e+00
## 59         f268 1.679735e+00
## 116        f471 1.051069e+00
## 178        f768 1.034926e+00
## 13          f57 8.977787e-01
## 136        f604 8.537344e-01
## 25          f99 6.190194e-01
## 71         f298 4.921528e-01
## 74         f306 4.318747e-01
## 55         f240 3.932927e-01
## 128        f546 3.818846e-01
## 177        f765 3.700267e-01
## 127        f536 3.259373e-01
## 179        f774 3.232158e-01
## 37         f153 3.131973e-01
## 140        f615 2.945556e-01
## 176        f756 2.802011e-01
## 18          f70 2.738019e-01
## 129        f556 2.566507e-01
## 56         f243 2.504861e-01
## 145        f630 2.465019e-01
## 15          f65 2.434348e-01
## 14          f61 2.341287e-01
## 45         f198 2.208872e-01
## 70         f297 2.160104e-01
## 69         f292 2.132267e-01
## 27         f104 1.840168e-01
## 77         f313 1.644578e-01
## 118        f479 1.625639e-01
## 141        f616 1.605380e-01
## 19          f71 1.572015e-01
## 22          f80 1.526873e-01
```

```
## 68          f290 1.512196e-01
## 133         f590 1.404800e-01
## 23           f81 1.309131e-01
## 134         f598 1.243174e-01
## 83          f331 1.217854e-01
## 80          f321 1.189038e-01
## 143         f620 1.060489e-01
## 81          f323 1.020564e-01
## 17           f67 9.998148e-02
## 40          f170 9.622995e-02
## 96          f384 9.571386e-02
## 147         f637 8.571398e-02
## 24           f83 8.259875e-02
## 52          f218 7.780791e-02
## 86          f341 7.624593e-02
## 6            f13 7.019709e-02
## 85          f340 6.972299e-02
## 75          f308 6.938225e-02
## 112         f458 6.932195e-02
## 63          f279 6.233926e-02
## 26          f103 6.178347e-02
## 57          f252 6.047895e-02
## 54          f222 5.931565e-02
## 39          f163 5.769257e-02
## 79          f316 5.764058e-02
## 50          f213 5.644315e-02
## 126         f533 5.478935e-02
## 3             f3 5.138892e-02
## 16           f66 4.860839e-02
## 110         f448 4.642512e-02
## 67          f289 4.608358e-02
## 58          f262 4.379036e-02
## 175         f755 4.341701e-02
## 73          f304 3.884648e-02
## 53          f220 3.778266e-02
## 114         f468 3.749926e-02
## 111         f450 3.623018e-02
## 82          f330 3.499625e-02
## 109         f444 3.409257e-02
## 135         f601 3.397650e-02
## 97          f385 3.396528e-02
## 121         f518 3.283955e-02
## 148         f638 3.229011e-02
## 138         f612 3.127051e-02
## 38          f162 2.911762e-02
## 93          f374 2.614121e-02
## 132         f589 2.433474e-02
## 181         f776 2.369116e-02
## 29          f130 2.339993e-02
## 131         f588 2.293142e-02
```

```
## 180        f775 2.213799e-02
## 108        f442 2.174456e-02
## 137        f609 2.153892e-02
## 20          f73 1.965623e-02
## 89         f350 1.874173e-02
## 163        f677 1.812658e-02
## 44         f189 1.682807e-02
## 32         f143 1.606743e-02
## 60         f269 1.570063e-02
## 66         f288 1.565904e-02
## 95         f383 1.564887e-02
## 159        f669 1.551487e-02
## 151        f647 1.542845e-02
## 8           f19 1.542647e-02
## 36         f150 1.526814e-02
## 104        f422 1.457757e-02
## 11          f44 1.445819e-02
## 168        f725 1.436713e-02
## 122        f522 1.407689e-02
## 162        f674 1.309686e-02
## 92         f367 1.303513e-02
## 160        f672 1.297878e-02
## 142        f619 1.293453e-02
## 88         f349 1.228726e-02
## 169        f733 1.196646e-02
## 21          f76 1.192408e-02
## 158        f664 1.171386e-02
## 31         f140 1.165022e-02
## 12          f54 1.096574e-02
## 94         f378 1.082223e-02
## 150        f646 1.040117e-02
## 124        f525 1.014318e-02
## 72         f299 9.879784e-03
## 78         f315 8.661046e-03
## 120        f514 8.568343e-03
## 46         f199 7.107456e-03
## 41         f173 6.782461e-03
## 10          f32 5.922245e-03
## 170        f734 5.686606e-03
## 90         f358 5.336970e-03
## 101        f411 5.190485e-03
## 164        f679 5.088550e-03
## 161        f673 4.987415e-03
## 172        f739 4.861803e-03
## 100        f403 4.795423e-03
## 107        f436 4.068424e-03
## 5            f5 3.986515e-03
## 35         f149 3.900669e-03
## 105        f425 3.866593e-03
## 42         f182 3.805911e-03
```

```
## 43          f188 3.176033e-03
## 76          f312 2.384219e-03
## 33          f144 2.138026e-03
## 153         f650 2.102708e-03
## 49          f212 2.035054e-03
## 154         f651 1.666227e-03
## 149         f645 1.516828e-03
## 156         f654 1.369220e-03
## 9            f29 6.459194e-04
## 155         f652 4.109182e-04
## 165         f680 4.050047e-04
## 119         f499 3.748039e-04
## 91          f361 3.399257e-04
## 152         f649 3.317337e-04
## 139         f614 2.300847e-04
## 64          f285 5.024071e-05
## 4             f4 2.852757e-05
## 106         f428 2.392520e-05
## 173         f742 2.115615e-05
## 130         f587 1.530323e-05
## 99          f394 6.394796e-06
## 84          f333 2.550679e-06
## 62          f278 2.135087e-06
## 30          f139 1.732321e-06
## 48          f203 1.200118e-06
## 2             id 1.050045e-06
## 34          f146 2.250796e-07
## 103         f421 2.246499e-07
## 146         f636 1.233782e-07
## 47          f202 7.852488e-08
## 174         f743 3.973979e-08
## 166         f699 1.719785e-08
## 115         f470 1.164576e-08
## 123         f523 1.027928e-08
## 65          f287 3.415609e-09
## 7            f16 1.468087e-09
## 171         f735 1.305841e-09
## 167         f715 6.683286e-10
## 125         f526 5.990584e-10
## 51          f217 5.529375e-10
## 144         f628 1.143328e-10
## 87          f347 1.380599e-11
## 157         f659 5.688570e-14
## 113         f461 1.279305e-17
## 102         f420 8.661696e-26
## 61          f277 2.839730e-27
## 117         f472 9.415087e-37
## 98          f391 4.903228e-43
```

```r
#Removing intercept columns returned from lasso model:
new_bank_coef<- new_bank_coef[-1, ]

#Converting the data frame to a vector:
new_bank_coef<- as.vector(new_bank_coef$name)

#Adding "default" column to the data frame:
new_bank_coef<- c(new_bank_coef,"default")

#Selecting attributes from original data set "new_bank_model" using
coefficients returned from lasso model i.e., "new_bank_coef"
bank_lasso<-select(new_bank_model, new_bank_coef)
```

```
## Warning: Using an external vector in selections was deprecated in
tidyselect 1.1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(new_bank_coef)
##
##   # Now:
##   data %>% select(all_of(new_bank_coef))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

#b). Principle Component Analysis (PCA):

#We have 180 variables returned from Lasso model that are stored in “bank_lasso”. Now, we further process the variables using PCA.

```r
pca_model <- preProcess(bank_lasso[,-c(181)], method = c("center", "scale",
"pca"), thresh = 0.80)

pca_model_1<- predict(pca_model, bank_lasso)

pca_model
```

```
## Created from 80000 samples and 180 variables
##
## Pre-processing:
##   - centered (180)
##   - ignored (0)
##   - principal component signal extraction (180)
##   - scaled (180)
##
## PCA needed 69 components to capture 80 percent of the variance
```

#We have a threshold limit of 0.80 for PCA to ensure 80% of variance is captured. PCA captured 80% in 69 components.

#We are adding default column from previous model to the PCA model:

```
pca_model_1$default <- bank_lasso$default
```

#Creating a train and validation sets from the values returned in PCA model:

```
set.seed(123)

pca_index <- createDataPartition(pca_model_1$default, p = 0.80, list = FALSE)

pca_train <- pca_model_1[pca_index, ]
pca_validate <- pca_model_1[-pca_index, ]
```

#Coverting the "default" column into factor in both train and validation sets:

```
pca_train$default <- as.factor(pca_train$default)
pca_validate$default <- as.factor(pca_validate$default)
```

#Now run the values returned from PCA in random forest model:

```
set.seed(123)

model_rf_pca <- randomForest(default ~ ., data = pca_train, mtry = 5)

print(model_rf_pca)

##
## Call:
##  randomForest(formula = default ~ ., data = pca_train, mtry = 5)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 9.22%
## Confusion matrix:
##       0  1  class.error
## 0 58089  8 0.0001377007
## 1  5890 14 0.9976287263

pca_final <- data.frame(actual = pca_validate$default,predict(model_rf_pca,
newdata = pca_validate, type = "prob"))

pca_final$predict <- ifelse(pca_final$X0 > 0.60, 0, 1)

pca <- confusionMatrix(as.factor(pca_final$predict),
as.factor(pca_final$actual),positive='1')

pca
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 14510  1462
##          1    14    13
##
##                Accuracy : 0.9077
##                  95% CI : (0.9032, 0.9122)
##     No Information Rate : 0.9078
##     P-Value [Acc > NIR] : 0.5178
##
##                   Kappa : 0.014
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0088136
##             Specificity : 0.9990361
##          Pos Pred Value : 0.4814815
##          Neg Pred Value : 0.9084648
##              Prevalence : 0.0921933
##          Detection Rate : 0.0008126
##    Detection Prevalence : 0.0016876
##       Balanced Accuracy : 0.5039248
##
##        'Positive' Class : 1
##
```

#Laoding Test data set for predicting the defaulting customers:

```
pca_test <- read.csv("test__no_lossv3.csv")
```

#We are imputing missing values same as we did for train data set using medianimpute method:

```
test_pca_1 <- preProcess(pca_test, method = c("medianImpute"))

test_pca_process<- predict(test_pca_1, pca_test)
```

#Selecting attributes from test data set "test_pca_process" using coefficients returned from lasso model i.e., "new_bank_coef":

```
test_pca_lasso<-select(test_pca_process,
new_bank_coef[new_bank_coef!="default"])
```

#We are processing test model also in PCA to match our train model:

```
set.seed(123)
test_pca_model <- preProcess(test_pca_lasso, method = c("center", "scale",
"pca"), thresh = 0.80)
```

```
test_pca_model_1<- predict(pca_model, test_pca_lasso)
```

#Predicting the test_pca_model_1 using the random forest model "model_rf_pca":

```
set.seed(123)
predictions_pca <-data.frame(id=pca_test$id,predict(model_rf_pca,
test_pca_model_1, type = "prob"))

threshold <- 0.60
predictions_pca$predicted_default <- ifelse(predictions_pca$X0 > threshold,
0, 1)
```

#Filtering the number defaulting customers that was predicting by our random forest model:

```
predictions_pca_filtered<-predictions_pca %>% filter(predicted_default == 1)
predictions_pca_filtered

##          id    X0    X1 predicted_default
## 1086  34664 0.580 0.420                 1
## 1342  20578 0.382 0.618                 1
## 3621  90934 0.564 0.436                 1
## 5105  98853 0.554 0.446                 1
## 6837   4702 0.568 0.432                 1
## 7645   1180 0.586 0.414                 1
## 7715  13944 0.598 0.402                 1
## 8647  57274 0.412 0.588                 1
## 8882  99336 0.388 0.612                 1
## 9128  74118 0.530 0.470                 1
## 9992  21206 0.594 0.406                 1
## 10263 79925 0.574 0.426                 1
## 11564 40386 0.588 0.412                 1
## 11616 23780 0.538 0.462                 1
## 12080  2556 0.598 0.402                 1
## 12892 79483 0.582 0.418                 1
## 13017 71222 0.590 0.410                 1
## 13140 82291 0.324 0.676                 1
## 13289 64413 0.400 0.600                 1
## 13352 85505 0.440 0.560                 1
## 14831 86315 0.312 0.688                 1
## 16841 89098 0.414 0.586                 1
## 17168  2370 0.592 0.408                 1
## 17987 71006 0.288 0.712                 1
## 18197 89775 0.598 0.402                 1
## 20030 50050 0.532 0.468                 1
## 20307 85692 0.572 0.428                 1
## 20333 62466 0.502 0.498                 1
## 21834  1248 0.452 0.548                 1
## 22427 61399 0.566 0.434                 1
## 23437 56201 0.446 0.554                 1
```

```
## 23471   835 0.474 0.526                 1
## 24976 98018 0.520 0.480                 1
```

#Based on the results our random forest model predicted 33 customer will default in the test data set.

#We are binding our results from our classification model to our orginal test dataset:

```
test_2<-pca_test
test_2$predictions <- predictions_pca$predicted_default
test_3<- test_2 %>% filter(predictions==1)
```

#2.REGRESSION MODEL: #We have classified our defaulting customers using classification above. #Now, we create a regression model, to predict loss by the defaulting customers from classification.

#Laoding the train data set:

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

#filtering all the non-defaulting customers from the train data set:

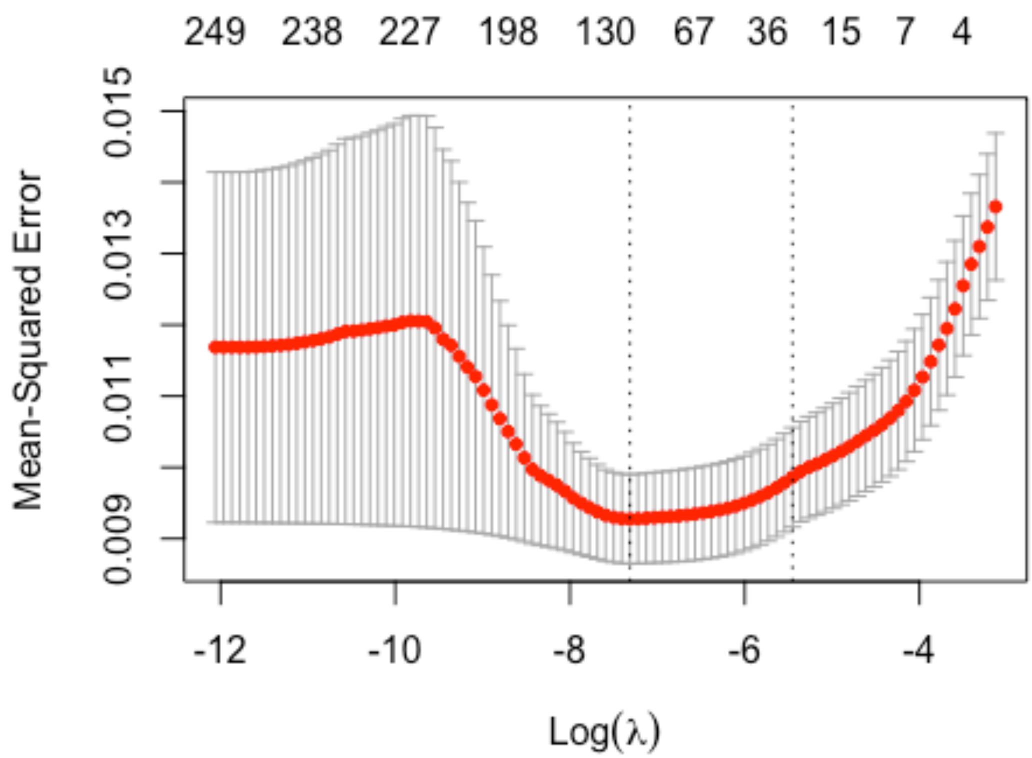
```
new_train_1 <- new_train %>% filter(loss!=0)
new_train_1$loss<- (new_train_1$loss / 100)
```

##Removing the zero-variances variables and Preprocessing the dataset by removing highly correlated and imputing missing values using “corr” and “medianimpute”:

```
zero_var_indices_1 <- nearZeroVar(new_train_1[ ,-c(763)])

train_model <- new_train_1[, -zero_var_indices_1]

new_train_3 <- preProcess(train_model[ ,-c(748)], method = c("medianImpute",
"corr"))

new_train_4 <- predict(new_train_3, train_model)
```

#Lasso model: we are using lasso model for variable selection for the dataset “new_train_4” consisting of 252 attributes:

```
set.seed(123)
x_1 <- as.matrix(new_train_4[ ,-c(252)])
y_2 <- as.vector(new_train_4$loss)

model_lasso <- cv.glmnet(x_1, y_2, alpha = 1, family = "gaussian", nfolds =
10, type.measure = "mse")

plot(model_lasso)
```




```
model_lasso$lambda.min

## [1] 0.0006662975
```

#We convert coefficients returned in lasso model into dataframe:

```
# Return the coefficients for the lasso regression at the minimum lambda
value:
coef_test <- coef(model_lasso, s= "lambda.min")

#Convert the coefficient values into a data frame:
coef_test<- data.frame(name = coef_test@Dimnames[[1]][coef_test@i + 1],
coefficient = coef_test@x)

#Removing negatives values using "abs" function:
coef_test$coefficient <- abs(coef_test$coefficient)

#Re-arranging the data frame in decreasing order:
coef_test[order(coef_test$coefficient, decreasing = TRUE), ]

##              name  coefficient
## 1     (Intercept) 1.723274e-01
## 21           f129 1.376713e-01
## 121          f774 1.273948e-01
```

```
## 119         f766 1.269790e-01
## 120         f768 1.177613e-01
## 38          f268 1.054545e-01
## 33          f229 9.832930e-02
## 78          f556 9.815358e-02
## 77          f546 5.953691e-02
## 36          f243 5.437601e-02
## 34          f238 4.915198e-02
## 15           f70 4.494179e-02
## 87          f615 4.039415e-02
## 43          f297 3.829277e-02
## 22          f130 2.363449e-02
## 29          f198 2.120761e-02
## 12           f57 2.105920e-02
## 42          f291 1.712850e-02
## 55          f402 1.638670e-02
## 85          f604 1.632679e-02
## 118         f765 1.628697e-02
## 95          f637 1.553414e-02
## 45          f316 1.462647e-02
## 92          f631 1.298546e-02
## 83          f598 1.292320e-02
## 8            f32 1.212681e-02
## 82          f589 1.154419e-02
## 91          f629 1.119956e-02
## 47          f324 1.100458e-02
## 14           f67 1.084942e-02
## 16           f71 1.020813e-02
## 13           f65 9.622098e-03
## 20          f109 7.807988e-03
## 10           f47 7.563056e-03
## 5            f13 7.314697e-03
## 37          f248 7.099032e-03
## 89          f621 6.582336e-03
## 79          f566 6.543583e-03
## 69          f509 6.434001e-03
## 48          f340 6.288034e-03
## 58          f413 6.252129e-03
## 72          f522 5.160324e-03
## 23          f140 4.389607e-03
## 30          f199 3.834597e-03
## 81          f588 3.719625e-03
## 46          f322 3.219921e-03
## 19           f93 3.125932e-03
## 18           f81 3.044201e-03
## 39          f281 3.002582e-03
## 41          f289 2.930093e-03
## 40          f288 2.827576e-03
## 28          f188 2.454033e-03
## 57          f412 2.313789e-03
```

```
## 35         f242 2.273488e-03
## 114        f739 1.966726e-03
## 64         f451 1.784469e-03
## 68         f499 1.488715e-03
## 88         f619 1.455426e-03
## 59         f422 1.419574e-03
## 66         f479 1.357014e-03
## 3            f3 1.337901e-03
## 116        f746 1.155371e-03
## 27         f150 9.941292e-04
## 111        f725 9.874954e-04
## 17          f76 9.015019e-04
## 31         f212 8.689184e-04
## 60         f436 8.067489e-04
## 24         f142 7.858284e-04
## 25         f144 7.356802e-04
## 67         f489 7.353847e-04
## 53         f384 7.256467e-04
## 113        f734 7.172839e-04
## 99         f647 7.091819e-04
## 70         f514 6.992333e-04
## 61         f442 6.894109e-04
## 9           f44 6.115054e-04
## 71         f518 4.844990e-04
## 11          f54 4.802613e-04
## 76         f536 4.538973e-04
## 86         f614 3.779220e-04
## 2            f1 3.677045e-04
## 62         f444 3.580527e-04
## 100        f648 3.432132e-04
## 44         f312 3.093357e-04
## 7           f31 2.931348e-04
## 84         f601 2.633167e-04
## 105        f664 2.230797e-04
## 51         f366 2.177829e-04
## 106        f669 2.054076e-04
## 104        f654 1.665254e-04
## 97         f639 1.635168e-04
## 108        f677 1.504512e-04
## 96         f638 1.415521e-04
## 93         f634 1.181377e-04
## 107        f674 1.180069e-04
## 98         f640 1.057873e-04
## 101        f649 8.784743e-05
## 49         f358 8.494809e-05
## 56         f403 7.949754e-05
## 52         f378 6.988383e-05
## 115        f740 6.702796e-05
## 4            f5 4.301011e-05
## 102        f651 4.213577e-05
```

```
## 63         f450 4.153655e-05
## 50         f361 2.601676e-05
## 117        f755 1.873274e-05
## 103        f652 1.370202e-05
## 6           f29 2.931829e-06
## 94         f636 1.392010e-06
## 80         f587 3.085592e-07
## 109        f682 2.565568e-08
## 26         f146 5.071273e-09
## 73         f523 3.885823e-09
## 74         f526 8.275708e-11
## 110        f715 1.333092e-11
## 90         f628 5.595958e-12
## 54         f401 2.974270e-12
## 32         f217 2.430969e-12
## 112        f726 1.721461e-12
## 75         f530 5.685597e-15
## 65         f472 4.191492e-38
```

```
#Removing intercept columns returned from lasso model:
coef_test<- coef_test[-1, ]

#Converting the data frame to a vector:
coef_test<- as.vector(coef_test$name)

#Adding "loss" column to the data frame:
coef_test<- c(coef_test,"loss")


#Selecting attributes from original data set "new_train_4" using coefficients returned from lasso model i.e., "coef_test"
final_model<-select(new_train_4, coef_test)
```

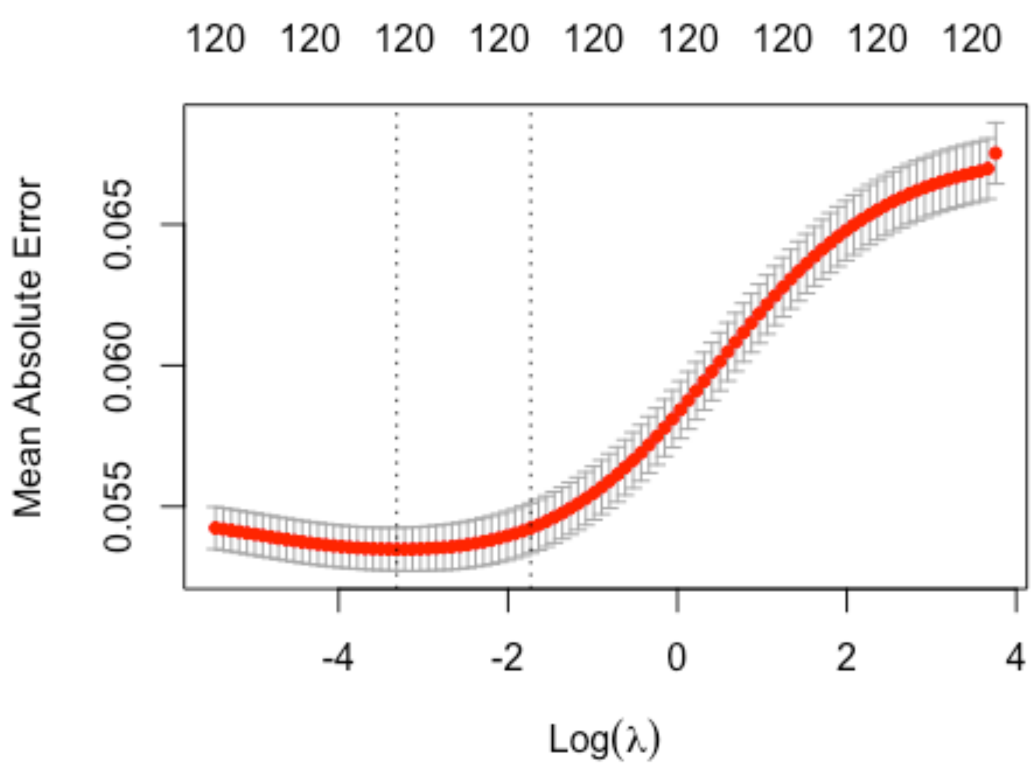
```
## Warning: Using an external vector in selections was deprecated in
tidyselect 1.1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(coef_test)
##
##   # Now:
##   data %>% select(all_of(coef_test))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

##Creating a train and validation sets from lasso model dataset “final_model”:

```
set.seed(123)

bank_index_1 <- createDataPartition(final_model$loss, p = 0.80, list = FALSE)

bank_train_1 <- final_model[bank_index_1, ]
bank_validate_1 <- final_model[-bank_index_1, ]
```

#Creating a ridge model on the train data set from above:

```
x_3 <- as.matrix(bank_train_1[ ,-c(121)])
y_3 <- as.vector(bank_train_1$loss)

ridge_model<- cv.glmnet(x_3, y_3, alpha = 0, family = "gaussian", nfolds =
10, type.measure = "mae")

plot(ridge_model)
```




```
ridge_model$lambda.min

## [1] 0.03637917

coef_final <- coef(ridge_model, s = "lambda.min")
```

validating the Ridge model using “bank_validate_1” using “MAE” metrics:

```
x_4 <- as.matrix(bank_validate_1[ ,-c(121)])
y_4 <- as.vector(bank_validate_1$loss)

predicted_loss <- predict(ridge_model, s = ridge_model$lambda.min, newx =
x_4)

## Evaluating Performance.

mae <- mean(abs((predicted_loss - y_4)))
mae_final <- cbind(y_4,predicted_loss)

print(mae)

## [1] 0.0575126
```

#Selecting attributes from original data set “test_3” using coefficients returned from lasso model i.e., “coef_test”

```
predict_9595<-select(test_3, coef_test[coef_test!="loss"])
```

#Imputing missing values in updated dataset “predict_9595”:

```
set.seed(123)
final_preprocess <- preProcess(predict_9595, method = c("medianImpute"))

final_preprocess_1 <- predict(final_preprocess, predict_9595)
```

#Predciting loss using ridge model by defaulting customers:

```
default_loss<-as.data.frame(round(abs(predict(ridge_model, s =
ridge_model$lambda.min, newx = as.matrix(final_preprocess_1)))*100))
```

#Storing loss given default values into a csv file:

```
loss_given_default <- cbind.data.frame(predictions_pca_filtered,
default_loss)

s<-left_join(predictions_pca,loss_given_default,by='id')

s$loss <- ifelse(s$predicted_default.x==0,0,s$s1)

final_predicted_file<-data.frame(id=s$id,loss=s$loss)

write.csv(final_predicted_file, "final_predicted_file.csv")
```