Loan Default Prediction

Group-4

2023-04-19

Loading Packages

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
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(ISLR)
library(dplyr)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
library(plsdepot)
```

```
## Warning: package 'plsdepot' was built under R version 4.2.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.2.3
```

```
## 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

library(pROC)

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

##
## Attaching package: 'pROC'

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

Loading Data

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


na_percent <- colSums(is.na(data)) / nrow(data) * 100

max(na_percent)

## [1] 17.82625
```

Null values for columns ranges from 0 to 17.82% in each column.

As the variables names have been masked and cannot be interpreted properly. Near zero variables are removed and highly correlated variables are removed. Null values are imputed with median impute.

Removing zero variance and high correlated variables and median imputing

```
set.seed(0811)
model_filter <- preProcess(data[ ,-c(763)], method = c("nzv", "corr"))
data_filtered <- predict(model_filter, data)
impute_proc <- preProcess(data[ ,-c(763)], method = c("medianImpute"))
data_filtered <- predict(impute_proc , data_filtered)
na_percent_after <- colSums(is.na(data_filtered)) / nrow(data_filtered) * 100
max(na_percent_after)

## [1] 0
```

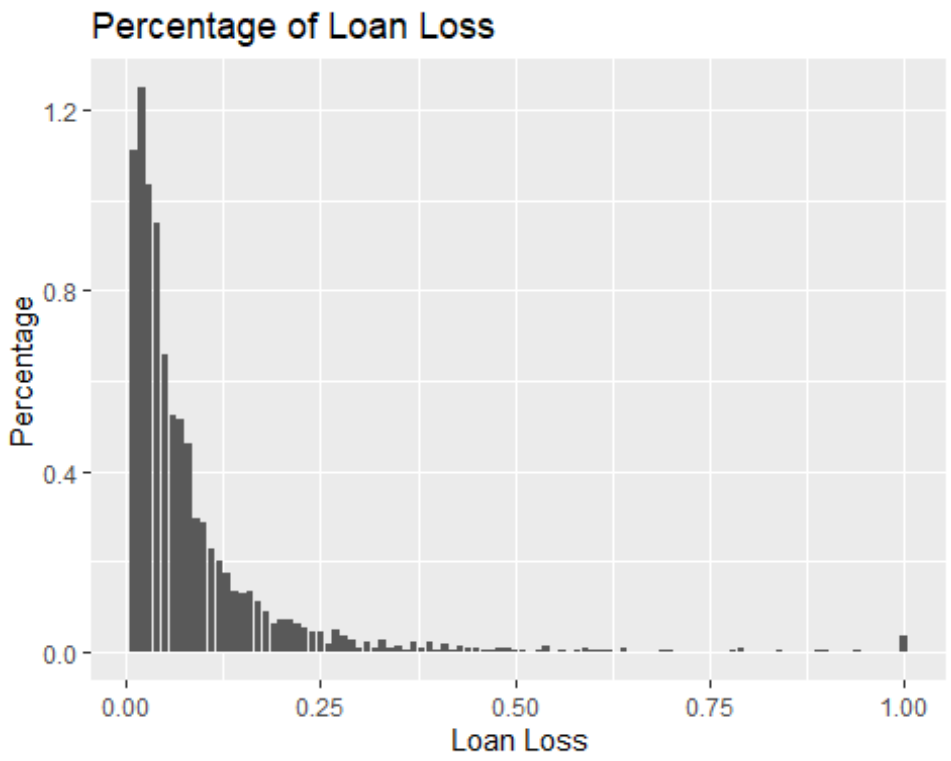
After removing near-zero variance and highly correlated variables, the independent variables were reduced to 247 from 762

Creating a New binary Column for loan default and scaling loss variable

```
data_filtered$loan_default <-
as.factor(ifelse(data_filtered$loss>0,"Yes","No"))
data_filtered$loss <- (data_filtered$loss / 100)
```

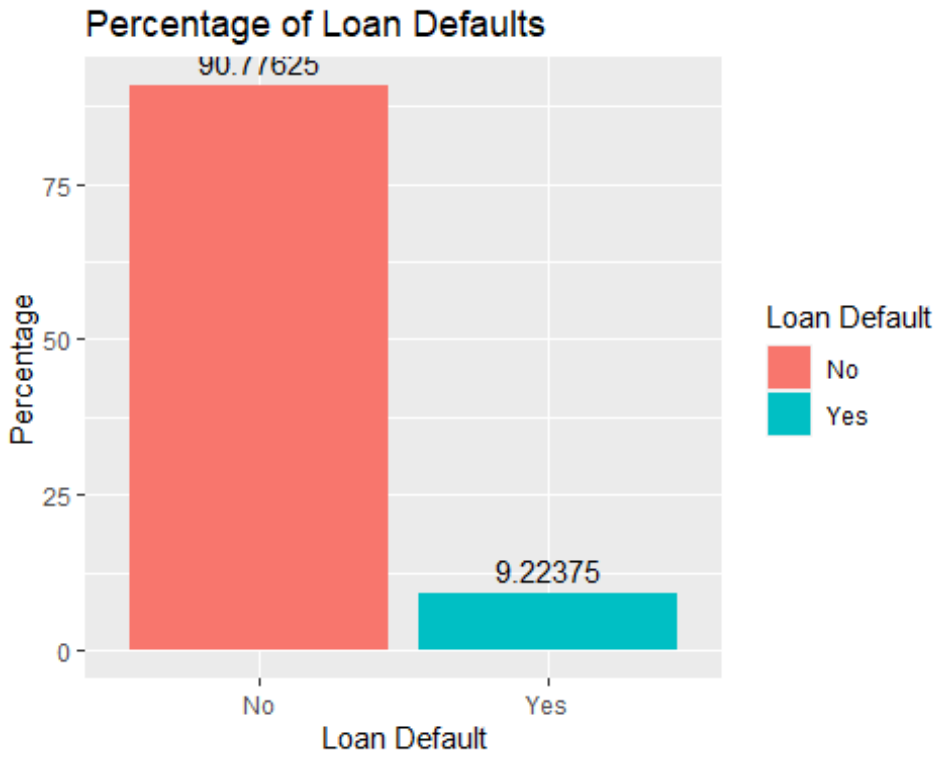
Exploring Target Variables

```
percentage <- data_filtered%>%
  group_by(loan_default)  %>%
  summarise(percentage=(n()/nrow(data_filtered))*100, Total=n())
d2<-data_filtered%>%filter(loan_default=='Yes')

p2<-d2%>%group_by(loss)  %>%
  summarise(percentage=(n()/nrow(data_filtered))*100)



ggplot(p2, aes(x=loss, y=percentage)) +
  geom_bar(stat="identity", position="dodge")  +
  labs(title="Percentage of Loan Loss",
       x="Loan Loss", y="Percentage", fill="Loss")
```



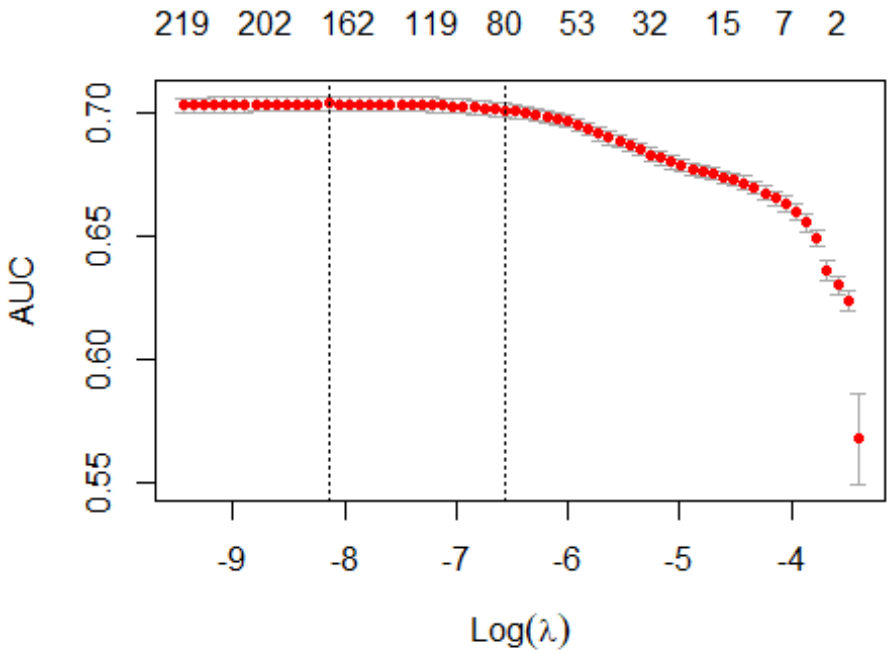

```
# Create a bar chart of the percentages by category
ggplot(percentage, aes(x=loan_default, y=percentage, fill=loan_default)) +
  geom_bar(stat="identity", position="dodge") +
  geom_text(aes(label=percentage), position=position_dodge(width=0.9),
vjust=-0.5) +
```

```
  labs(title="Percentage of Loan Defaults",
       x="Loan Default", y="Percentage", fill="Loan Default")
```



Applying lasso to reduce variables

```
set.seed(0811)

lasso_cv <- cv.glmnet( as.matrix(data_filtered[ ,-c(247,248)]),
                       data_filtered$loan,
                       preProcess=c("center","scale"),
                       alpha = 1, family = "binomial", type.measure = "auc")

plot(lasso_cv)
```




```
print(paste0("Minimum lambda is ",lasso_cv$lambda.min))
```

```
## [1] "Minimum lambda is 0.00028979267768779"
```

Filtering variables based on Lasso Coefficients

```
set.seed(0811)
lasso_coefs <- coef(lasso_cv, s = "lambda.min")


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

# removing intercept
lasso_coefs_df<- lasso_coefs_df[-1, ]

lasso_names <- as.vector(lasso_coefs_df$name)
lasso_names <- c(lasso_names,"loan_default")
data_filtered_lasso <- select(data_filtered, all_of(lasso_names))
```

After applying lasso model, the variables further reduced to 173

Applying PCA on the data filtered by lasso

```
preproc_pca<- preProcess(data_filtered_lasso[-c(1,175)], method = c("center",
"scale", "pca"), thresh = 0.8)

data_filtered_PCA <- predict(preproc_pca, data_filtered_lasso[-c(1,175)])
```

```
data_filtered_PCA$loan_default<-data_filtered_lasso$loan_default

preproc_pca
```

```
## Created from 80000 samples and 173 variables
##
## Pre-processing:
##   - centered (173)
##   - ignored (0)
##   - principal component signal extraction (173)
##   - scaled (173)
##
## PCA needed 67 components to capture 80 percent of the variance
```

Creating train data 80% and validation 20%

```
set.seed(0811)

index <- createDataPartition(data_filtered_PCA$loan_default, p = 0.80, list =
FALSE)

train_df <- data_filtered_PCA[index, ]
validation_df <-data_filtered_PCA[-index, ]
train_df$loan_default <- as.factor(train_df$loan_default)
validation_df$loan_default <- as.factor(validation_df$loan_default)
```

Building Random Forest model

```
set.seed(0811)
model_rf <- randomForest(loan_default ~ ., data = train_df,ntre=100,mtry=10)

print(model_rf)
```

```
##
## Call:
##  randomForest(formula = loan_default ~ ., data = train_df, ntre = 100,
mtry = 10)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 9.24%
## Confusion matrix:
##        No Yes  class.error
## No  58056  41 0.0007057163
## Yes  5871  33 0.9944105691
```

Predicting on Validation Dataset

```
validation_predicted <- data.frame(actual = validation_df$loan_default,
                    predict=predict(model_rf, newdata = validation_df[-68],
type = "response"))


confusion_matrix <- confusionMatrix(as.factor(validation_predicted$predict),
as.factor(validation_predicted$actual),positive='Yes')

confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  14521  1473
##        Yes     3     2
##
##                Accuracy : 0.9077
##                  95% CI : (0.9032, 0.9122)
##     No Information Rate : 0.9078
##     P-Value [Acc > NIR] : 0.5178
##
##                   Kappa : 0.0021
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0013559
##             Specificity : 0.9997934
##          Pos Pred Value : 0.4000000
##          Neg Pred Value : 0.9079030
##              Prevalence : 0.0921933
##          Detection Rate : 0.0001250
##    Detection Prevalence : 0.0003125
##       Balanced Accuracy : 0.5005747
##
##        'Positive' Class : Yes
##
```

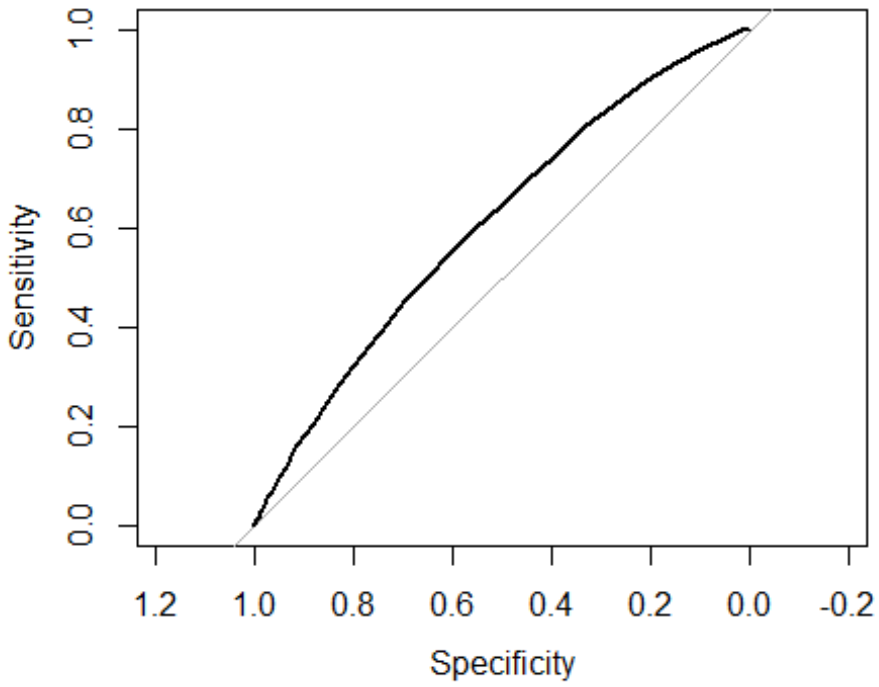
Determining threshold cutoff based on AUC

```
library(ROCR)
pred_probs <- predict(model_rf, newdata = validation_df[-68],type='prob')[,
"Yes"]

rf.roc <- roc(validation_df$loan_default,pred_probs)
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plot(rf.roc)
```




```
auc(rf.roc)

## Area under the curve: 0.6097
```

Loading Test Data

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

preprocessing test data

```
set.seed(0811)
## filtering lasso coefficients
lasso_names_2 <- lasso_names[lasso_names != "loan_default"]
test_data_filtered<-select(test_data, lasso_names_2)

## Note: Using an external vector in selections is ambiguous.
## ℹ Use `all_of(lasso_names_2)` instead of `lasso_names_2` to silence this
message.
## ℹ See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## imputing null values with median impute
final_test_1<- predict(impute_proc, test_data_filtered)

## pca preprocess
final_test_pca<-predict(preproc_pca,final_test_1)
```

Predicting the Random Forest on test data

```
final_predict<- data.frame(id=test_data$id,
                    predict(model_rf, newdata =final_test_pca[-1], type =
"prob"))

final_predict$predict <- ifelse(final_predict$No > 0.59, 0, 1)
test_data$loss<-final_predict$predict
final_test_2<-test_data%>%filter(loss==1)
```

Building Model for Defaulted Loss Customers and performing Regression

Creating subset of Defaulters

```
default_data<- data
default_data$loan_default <-
as.factor(ifelse(data_filtered$loss>0,"Yes","No"))
default_data$loss <- (default_data$loss / 100)
default_data <- subset(default_data, default_data$loan_default == 'Yes')
```

Preprocessing Data

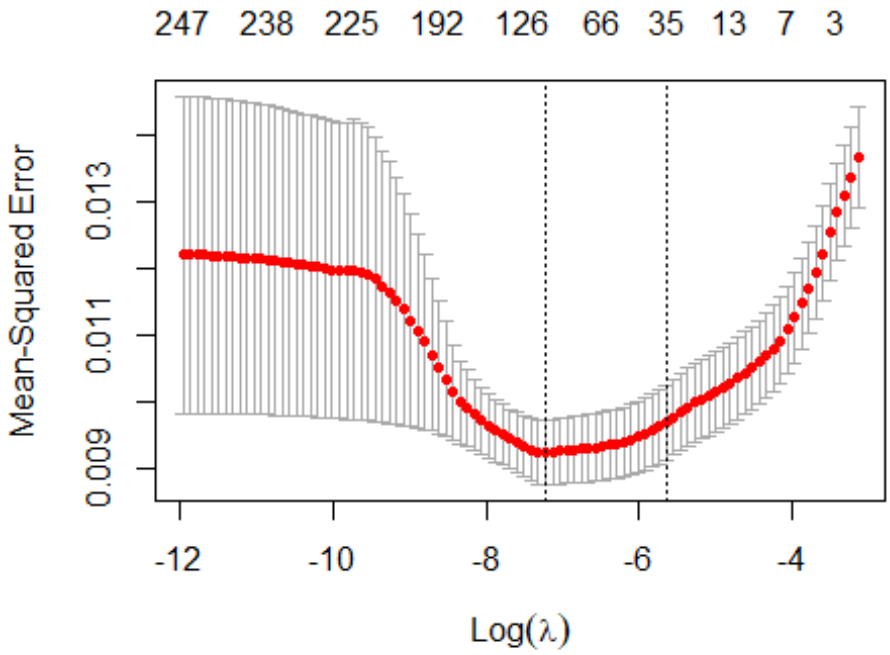
```
set.seed(0811)
model_1 <- preProcess(default_data[ ,-c(763,764)], method = c("nzv", "corr"
))

filtered_default <- predict(model_1, default_data)
impute_proc_2 <- preProcess(filtered_default[ ,-c(763,764)], method =
c("medianImpute" ))

filtered_default <- predict(impute_proc_2, filtered_default)
```

This preprocess model took variables from 762 variables down to 253 variables by removing near zero and highly correlated variables, and imputed the median values into missing values.

These values will be fed into our lasso regression model.

Applying lasso to reduce variables.

```
set.seed(0811)

lasso_cv_default <- cv.glmnet(as.matrix(filtered_default[ ,-c(1,252,253)]),
filtered_default$loss, alpha = 1, family = "gaussian", type.measure = "mse")
plot(lasso_cv_default)
```




```
print(paste0("Optimal Lambda for Lasso regression is
",lasso_cv_default$lambda.min))

## [1] "Optimal Lambda for Lasso regression is 0.000731260689041878"
```

Filtering Coefficients based on lasso

```
coefs_default <- coef(lasso_cv_default, s = "lambda.min")

coefs_default_2 <- data.frame(name =
coefs_default@Dimnames[[1]][coefs_default@i + 1], coefficient =
coefs_default@x)

coefs_default_2 <- coefs_default_2[-1, ]

# Turn the names into a vector

coefs_default_2<- as.vector(coefs_default_2$name)

# Add "loss" variable back into the vector

coefs_default_2 <- c(coefs_default_2,"loss")

default_lasso_filtered <- select(filtered_default, all_of(coefs_default_2))
```

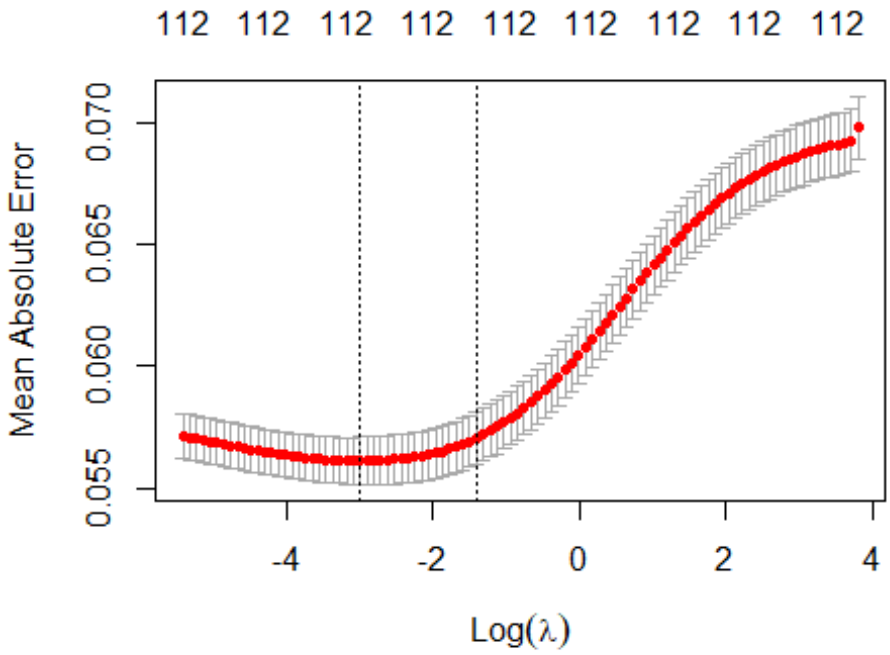
Lasso penalized regression further reduced the data set from 253 variables to 115 variables. These will be taken and entered into Ridge regression model to calculate loss given default.

Partitioning Train 80% and Validation 20%

```
set.seed(0811)

index_2 <- createDataPartition(default_lasso_filtered$loss, p = 0.80, list =
FALSE)

train_df_2 <- default_lasso_filtered[index_2, ]
validation_df_2 <- default_lasso_filtered[-index_2, ]

ridge_preproc<-preProcess(train_df_2[-113],method = c("center","scale"))

train_norm<-predict(ridge_preproc,train_df_2)

valid_norm<-predict(ridge_preproc,validation_df_2)
```

Ridge Regression for loss given default rate prediction

```
ridge_model <- cv.glmnet(as.matrix(train_norm[ ,-c(113)]),
                         train_norm$loss,
                         alpha= 0,
                         family = "gaussian",
                         type.measure = "mae")

plot(ridge_model)
```




```
print(paste0("optimal lambda for ridge is ",ridge_model$lambda.min))
```

```
## [1] "optimal lambda for ridge is 0.0502885651239036"
```

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

Validating Ridge model and calculating MAE

```
predicted_loss <- predict(ridge_model, s = ridge_model$lambda.min, newx =
as.matrix(valid_norm[ ,-c(113)]))


MAE_default = mean(abs((predicted_loss - as.vector(valid_norm$loss))))
comparison <- cbind(as.vector(valid_norm$loss),predicted_loss)

print(paste0("MAE for ridge model is ",MAE_default))
```

```
## [1] "MAE for ridge model is 0.0513700269511958"
```

Preprocessing test data

```
coefs_default_2 <- coefs_default_2[coefs_default_2 != "loss"]


final_test_filtered_2<-select(final_test_2, coefs_default_2)
```

```
## Note: Using an external vector in selections is ambiguous.
## ℹ Use `all_of(coefs_default_2)` instead of `coefs_default_2` to silence
this message.
```

```
## ℹ See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

final_test_filtered_3 <- predict(impute_proc_2, final_test_filtered_2)


final_test_filtered_3 <- predict(ridge_preproc, final_test_filtered_3)
```

Testing on unknown test data

```
final_test_filtered_4<- as.matrix(final_test_filtered_3)
predicted_loss_test <-  data.frame(id=
final_test_2$id,predict=(predict(ridge_model, s = ridge_model$lambda.min,
newx = final_test_filtered_4) )*100)
```

Writing to csv file

```
predicted_loss_test$predicted_loss<-
if_else(predicted_loss_test$s1<0,0,round(predicted_loss_test$s1))

pred<-left_join(final_predict,predicted_loss_test,by='id')

pred$loss <- ifelse(pred$predict==0,0,pred$predicted_loss)

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

write.csv(final_predicted_file, "final_prediction_file_Group_6.csv",
row.names = FALSE)
```