Exercise 5

Tobias Raidl, 11717659

2023-11-28

```r
library(ROCit)
```

```
## Warning: package 'ROCit' was built under R version 4.1.3
```

```r
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
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(mltools)
```

```
## Warning: package 'mltools' was built under R version 4.1.3
```

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.1.3
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
df = one_hot(as.data.table(Loan), c("Home", "EmpLen"))
df = dplyr::select(df, -Term)


set.seed(6669)
sample = sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.7,0.3))
```

```
train = df[sample, ]
test = df[!sample, ]
X_train = dplyr::select(train, -Status)
X_test = dplyr::select(test, -Status)
y_train = train$Status
y_test = test$Status

get_misclassification_rate = function(tp, fp, tn, fn) {
  tpr = tp/(tp+fn)
  tnr = tn/(tn+fp)
  eval = list(misclass_rate=(fp+fn)/(fp+tn+fn+tp), balanced_accuracy=(tpr+tnr)/2)
  return(eval)
}
```

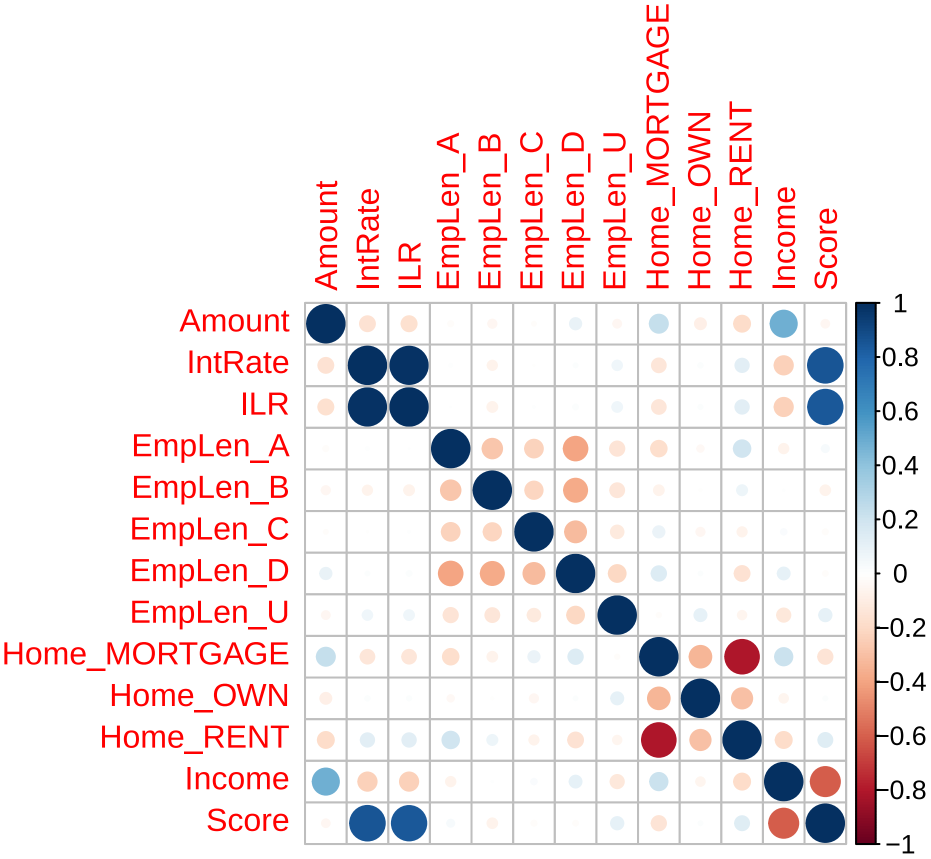
Remove collinearity

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.3
```

```
## corrplot 0.92 loaded
```

```
num_train = train
num_train$Status = as.integer(factor(num_train$Status))
corrplot(cor(X_train))
```

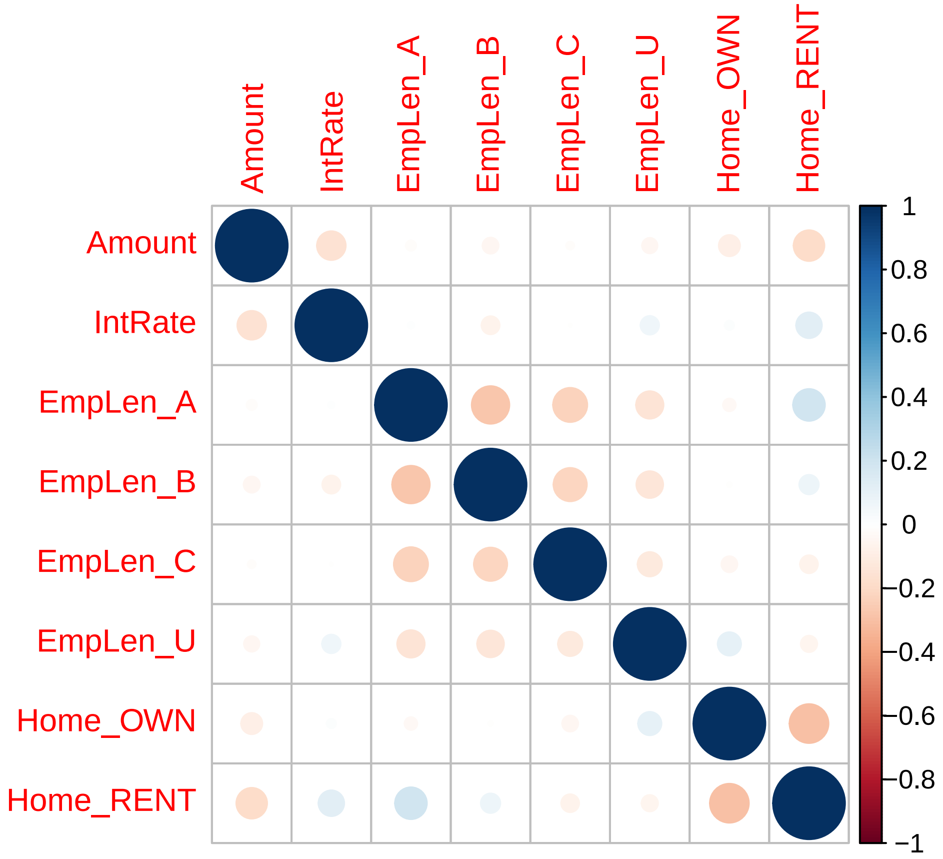


```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.1.3

## Loading required package: lattice
```

```r
indices_to_drop <- findCorrelation(cor(X_train), cutoff = 0.3, names=TRUE)
corrplot(cor(dplyr::select(X_train, -indices_to_drop)))
```

```
## 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(indices_to_drop)
##
##   # Now:
##   data %>% select(all_of(indices_to_drop))
##
## 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.
```



```r
train = dplyr::select(train, -indices_to_drop)
X_train = dplyr::select(X_train, -indices_to_drop)
```

1

(a) Fit model with lda

Yes I removed colinear variables and constant ones in the previous cell as data preprocessing.

```
lda = lda(Status~., train)
summary(lda)
```

```
##         Length Class  Mode     
## prior    2     -none- numeric  
## counts   2     -none- numeric  
## means   16     -none- numeric  
## scaling  8     -none- numeric  
## lev      2     -none- character
## svd      1     -none- numeric  
## N        1     -none- numeric  
## call     3     -none- call     
## terms    3     terms  call     
## xlevels  0     -none- list
```

(b) Evaluation on train set

The lda predicts nearly all observations to be FP. This is probably due to the imbalance between FP to CO ratio in the data set.

```
y_pred = predict(lda, X_train)$class
conf_mat = table(y_train, y_pred)
conf_mat
```

```
##        y_pred
## y_train  CO  FP
##      CO   4  81
##      FP   1 563
```

```
get_misclassification_rate(conf_mat[1,1], conf_mat[2,1], conf_mat[2,2], conf_mat[1,2])
```

```
## $misclass_rate
## [1] 0.1263482
## 
## $balanced_accuracy
## [1] 0.5226429
```

(c) As expected, inference on the test set results in a higher misclassification rate and balanced accuracy. Not far off though.

```
y_pred = predict(lda, X_test)$class
conf_mat = table(y_test, y_pred)
conf_mat
```

```
##       y_pred
## y_test  CO  FP
##     CO   1  45
##     FP   1 204
```

```
get_misclassification_rate(conf_mat[1,1], conf_mat[2,1], conf_mat[2,2], conf_mat[1,2])
```

```
## $misclass_rate
## [1] 0.1832669
##
## $balanced_accuracy
## [1] 0.5084305
```

2

(a) Undersampling

Undersample train set

```
table(train$Status)
```

```
##
##  CO  FP
##  85 564
```

```
class_counts = table(train$Status)
majority_class = names(class_counts)[which.max(class_counts)]
minority_class = names(class_counts)[which.min(class_counts)]
minority_count = class_counts[minority_class]
majority_indices = which(train$Status == majority_class)
sampled_majority_indices = sample(majority_indices, minority_count)
train_under = rbind(train[train$Status == minority_class],
↪  train[sampled_majority_indices])
table(train_under$Status)
```

```
##
## CO FP
## 85 85
```

```
X_train_under = dplyr::select(train_under, -Status)
y_train_under = train_under$Status
```

Evaluate undersampling for train set ~10% increase in balanced accuracy for train set.

```
lda_under = lda(Status~., train_under)
y_pred_under = predict(lda_under, X_train_under)$class
conf_mat = table(y_train_under, y_pred_under)
conf_mat
```

```
##               y_pred_under
## y_train_under CO FP
##            CO 49 36
##            FP 27 58
```

```
get_misclassification_rate(conf_mat[1,1], conf_mat[2,1], conf_mat[2,2], conf_mat[1,2])
```

```
## $misclass_rate
## [1] 0.3705882
##
## $balanced_accuracy
## [1] 0.6294118
```

Evaluate undersampling for test set Performs a little worse than on the train set

```
lda_under = lda(Status~., train_under)
y_pred_under = predict(lda_under, X_test)$class
conf_mat = table(y_test, y_pred_under)
conf_mat
```

```
##       y_pred_under
## y_test  CO  FP
##     CO  24  22
##     FP  91 114
```

```
get_misclassification_rate(conf_mat[1,1], conf_mat[2,1], conf_mat[2,2], conf_mat[1,2])
```

```
## $misclass_rate
## [1] 0.4501992
##
## $balanced_accuracy
## [1] 0.5389183
```

(b) Oversampling

```
class_counts = table(train$Status)
majority_class = names(class_counts)[which.max(class_counts)]
minority_class = names(class_counts)[which.min(class_counts)]
majority_count = class_counts[majority_class]
minority_indices = which(train$Status == minority_class)
sampled_minority_indices = sample(minority_indices, majority_count, replace=TRUE)
train_over = rbind(train[train$Status == majority_class],
↪  train[sampled_minority_indices])
table(train_over$Status)
```

```
##
##  CO  FP
## 564 564
```

```
X_train_over = dplyr::select(train_over, -Status)
y_train_over = train_over$Status
```

Oversample train set

```
lda_over = lda(Status~., train_over)
y_pred_over = predict(lda_over, X_train_over)$class
conf_mat = table(y_train_over, y_pred_over)
conf_mat
```

```
##             y_pred_over
## y_train_over  CO  FP
##           CO 316 248
##           FP 185 379
```

```
get_misclassification_rate(conf_mat[1,1], conf_mat[2,1], conf_mat[2,2], conf_mat[1,2])
```

```
## $misclass_rate
## [1] 0.3838652
```

```
##
## $balanced_accuracy
## [1] 0.6161348
```

Oversample test set

```
lda_over = lda(Status~., train_over)
y_pred_over = predict(lda_over, X_test)$class
conf_mat = table(y_test, y_pred_over)
conf_mat
```

```
##       y_pred_over
## y_test  CO  FP
##     CO  23  23
##     FP  78 127
```

```
get_misclassification_rate(conf_mat[1,1], conf_mat[2,1], conf_mat[2,2], conf_mat[1,2])
```

```
## $misclass_rate
## [1] 0.4023904
##
## $balanced_accuracy
## [1] 0.5597561
```