ADS 503 Final Project

Claire Phibbs, Christopher Richardson, Martin Zagari

2022-06-17

The Data

Our target variable "Risk Performance" along with the first 13 predictor columns

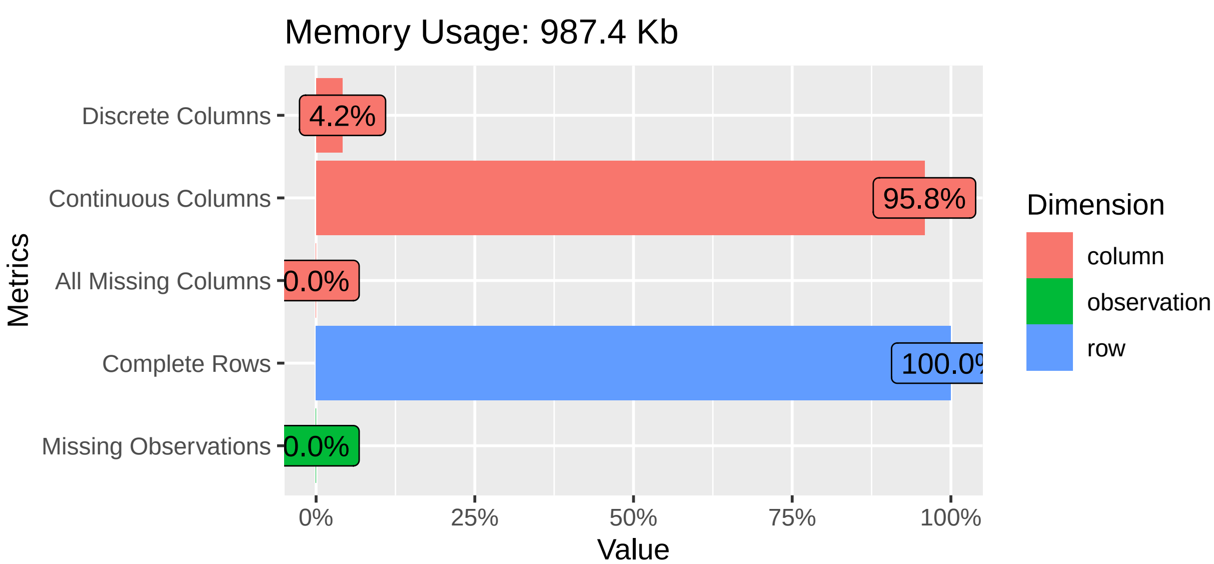
```
file_loc <- '/Volumes/GoogleDrive/My Drive/503/Project 503/Fico Data/heloc_dataset_v1.csv'
heloc <- read.csv(file_loc)

# sort col names for readability purposes
heloc <- heloc[ , order(names(heloc))]
heloc$RiskPerformance <- as.factor(heloc$RiskPerformance)

knitr::kable(heloc[1:4,c(24,1:13)]) %>%
  kableExtra::kable_styling("striped", full_width = F) %>%
  kableExtra::row_spec(0, angle = -90)
```

Predictor columns 14 to 23

```
knitr::kable(heloc[1:4,c(24,14:23)]) %>%
  kableExtra::kable_styling("striped", full_width = F) %>%
  kableExtra::row_spec(0, angle = -90)
```

EDA

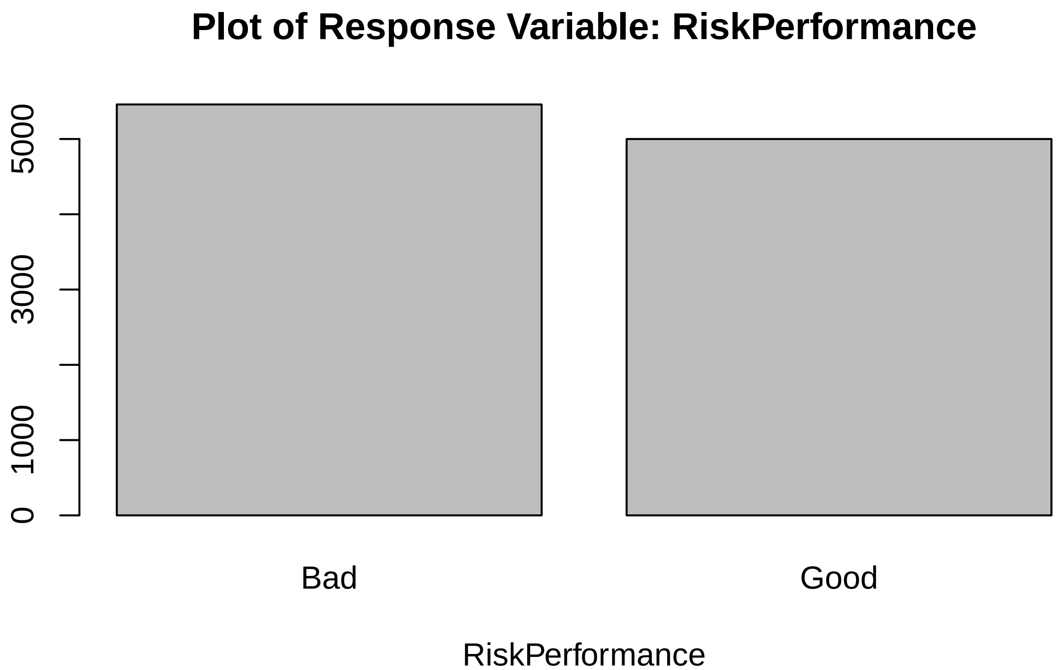
```
DataExplorer::plot_intro(heloc)
```



RiskPerformance	AverageMInFile	ExternalRiskEstimate	MaxDelq2PublicRecLast12M	MaxDelqEver	MSinceMostRecentDelq	MSinceMostRecentInqexcl7days	MSinceMostRecentTradeOpen	MSinceOldestTradeOpen	NetFractionInstallBurden	NetFractionRevolvingBurden	NumBank2NatlTradesWHighUtilization	NumInqLast6M	NumInqLast6Mexcl7days
Bad	84	55	3	5	2	0	4	144	-8	33	1	0	0
Bad	41	61	0	8	-7	0	15	58	-8	0	-8	0	0
Bad	24	67	7	8	-7	0	5	66	66	53	1	4	4
Bad	73	66	6	6	76	0	1	169	83	72	3	5	4

RiskPerformance	NumInstallTradesWBalance	NumRevolvingTradesWBalance	NumSatisfactoryTrades	NumTotalTrades	NumTrades60Ever2DerogPubRec	NumTrades90Ever2DerogPubRec	NumTradesOpeninLast12M	PercentInstallTrades	PercentTradesNeverDelq	PercentTradesWBalance
Bad	1	8	20	23	3	0	1	43	83	69
Bad	-8	0	2	7	4	4	0	67	100	0
Bad	2	4	9	9	0	0	4	44	100	86
Bad	4	6	28	30	1	1	3	57	93	91

```r
# bar plot of response variable; RiskPerformance
barplot(table(heloc$RiskPerformance),
        main="Plot of Response Variable: RiskPerformance",
        xlab="RiskPerformance")
```




```r
table(heloc$RiskPerformance)
```

```
##
##  Bad Good
## 5459 5000
```

A 51:47 split! Nearly a 50:50 balance!

```r
# copy heloc data fore explority purposes
xplore_df <- data.frame(heloc)

noY_bool <- names(xplore_df) != 'RiskPerformance'
xplore_df <- data.frame(xplore_df[, noY_bool])

# No credit history
bool1 <- xplore_df == -9
# No activity in the last year
bool2 <- xplore_df == -8
# No soft hits (those with a 0 assigned have had soft hits)
bool3 <- xplore_df == -7

bool0 <- bool1 | bool2 | bool3

table(xplore_df[bool0])
```

```
##
##    -9    -8    -7
## 13534  6114  6519
```

Interesting to see that 13,534 values are -9 (no history)

```r
xplore_df <- data.frame(heloc)
bool1 <- xplore_df == -9

xplore_df[bool1] <- 0
nines_across_d_board <- xplore_df[rowSums(xplore_df[,noY_bool]) == 0,]
dim(nines_across_d_board)[1]
```

```
## [1] 588
```

We have 588 rows that are complete -9 across all features.

```r
table(nines_across_d_board$RiskPerformance)
```

```
##
##  Bad Good
##  323  265
```

Interesting to see the data setregarding -9s across all features has a 55:45 split. Taking a chance on anything without any background knowledge is a 50/50 split, this was probably intentionally baked into the HELOC data set to skew results in some way.

```r
# the definition of -7 and -8 are similar though not identical
xplore_df <- data.frame(heloc)
bool1 <- xplore_df == -8
bool2 <- xplore_df == -7

xplore_df[bool1] <- NA
xplore_df[bool2] <- NA

inactive_records <- xplore_df[rowSums(is.na(xplore_df)) > 0,]
table(inactive_records$RiskPerformance)
```

```
##
##  Bad Good
## 3574 3792
```

As we see, the dataset has a nearly balanced mixture of those with an inactive credit history; though this lens is through a macroscope and not further dividing the data based on certain features which will be shown via histograms and box plots.

```r
bool1 <- inactive_records == NA
inactive_records[bool1] <- 0
inactive_records <- inactive_records[,noY_bool]
inactive_records <- inactive_records[(rowSums(inactive_records)) == 0,]
# dim(inactive_records)[1]

# unable to figure out how to fix all NA dataframe.
# Code should return a dataframe dim of 0,23.
```

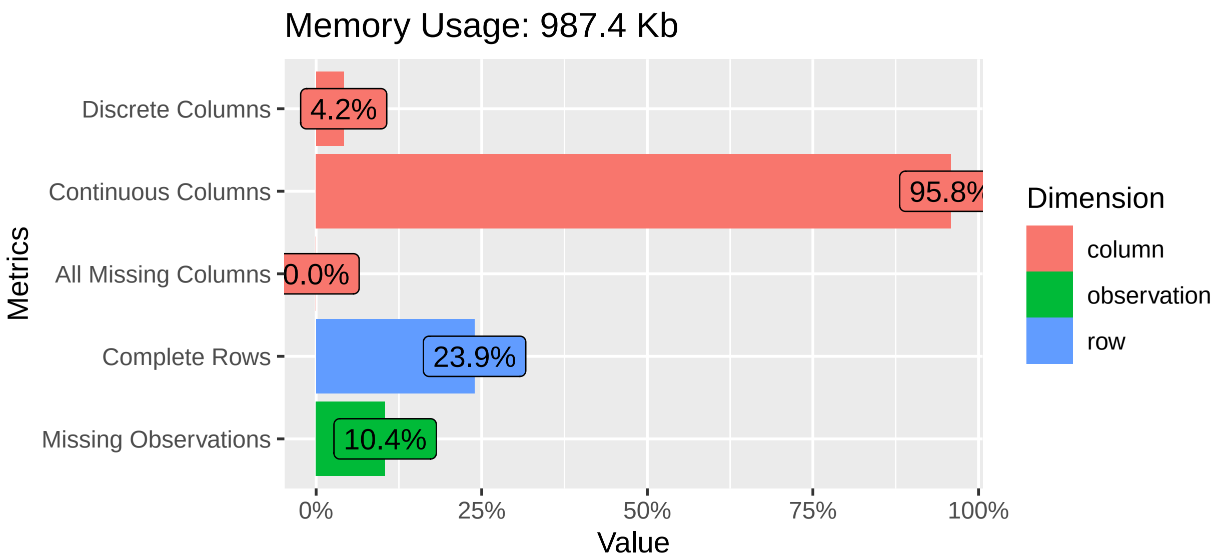
*Note: Cell above changes inactive_records returns a dataframe filled with NAs but with the same structure as HELOC. After further analysis, we concluded there are no records completely filled with -8s or -7s.

This makes sense, for the fact that it is highly likely that people with such records have a mature financial history, while those with a -9 are those starting their financial history; meaning that it is more likely for an individual with a -9 to have -9s accross the board.

Data Pre-Processing

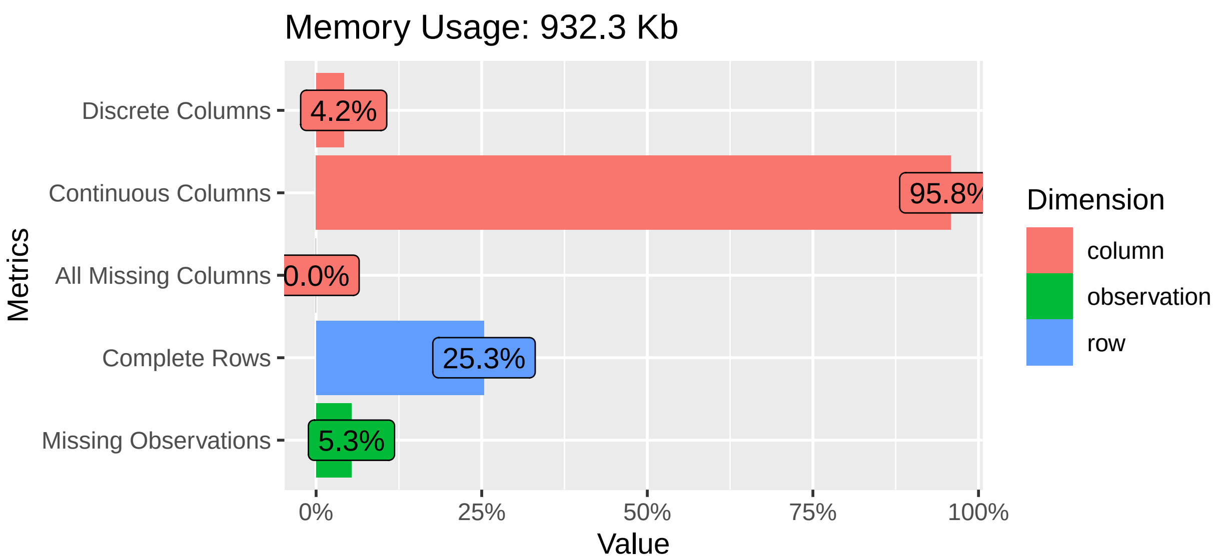
We are going to preprocess the data to get quality data insights.

```r
# -9 = No Credit History
# -8 and -7 = No recent activity
heloc[heloc == -9] <- NA
heloc[heloc == -8] <- NA
heloc[heloc == -7] <- NA
DataExplorer::plot_intro(heloc)
```



It appears that only 76.1% of our data had one or more predictors had a -9,-8,-7. Thus the following code removes all missing values that span across all columns.

```r
# removing missing values from rows that span across all columns (588 values)
heloc_No_NA <- heloc %>% dplyr::filter_at(vars(-RiskPerformance),
                                          any_vars(!is.na(.)))
DataExplorer::plot_intro(heloc_No_NA)
```




```r
# create training indices
set.seed(3)
```

```r
heloc_training <- caret::createDataPartition(heloc_No_NA$RiskPerformance,
                                            p=0.8,
                                            list=FALSE)

# training/set sets
heloc_train <- heloc_No_NA[heloc_training, ]
heloc_test <- heloc_No_NA[-heloc_training, ]

# knn imputation
heloc_impute <- caret::preProcess(heloc_train,
                                  method = 'knnImpute')

heloc_train <- stats::predict(heloc_impute,
                              newdata=heloc_train)

heloc_test <- stats::predict(heloc_impute,
                             newdata=heloc_test)

# remove highly correlated predictors
high_corr <- caret::findCorrelation(stats::cor(heloc_train[, -24]),
                                    0.85)

# removal of high cor predictors
heloc_train <- heloc_train[, -(high_corr)]
heloc_test <- heloc_test[, -(high_corr)]
names(heloc_test[high_corr])
```

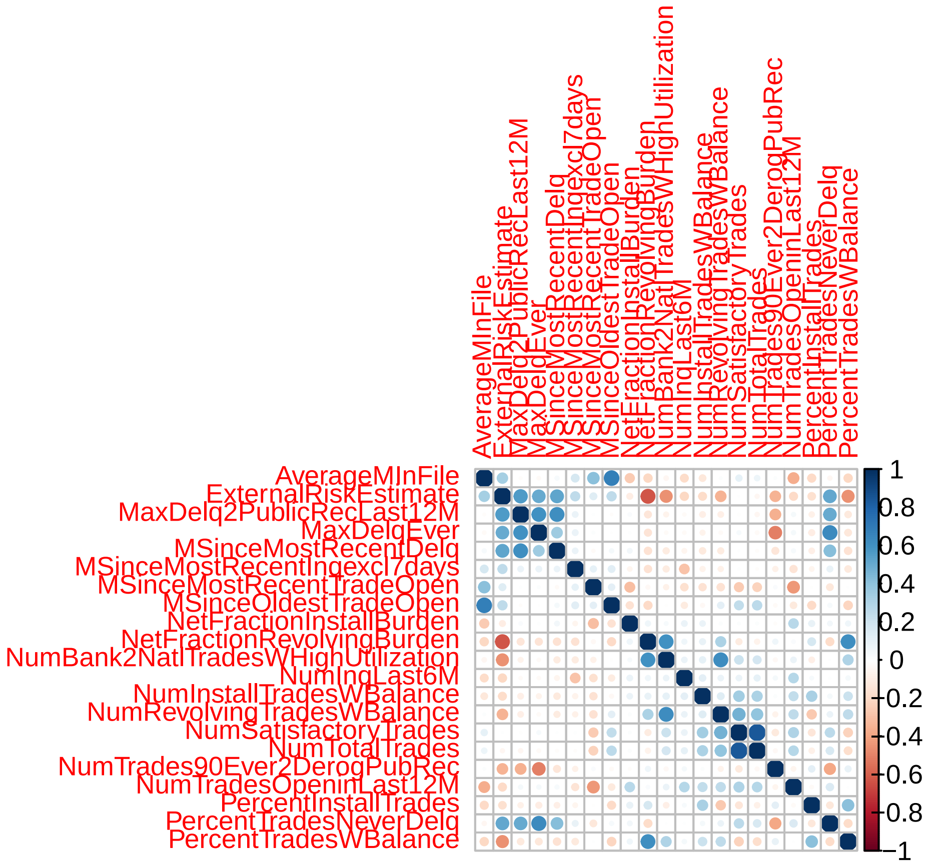
```
## [1] "NumTradesOpeninLast12M"   "NumInstallTradesWBalance"
```

We keep the heloc_train/test dataframes for formula based functions.

```r
no_risk_bool <- names(heloc_train) != 'RiskPerformance'

# x = predictors
heloc_train_x <- heloc_train[,no_risk_bool]
heloc_test_x <- heloc_test[,no_risk_bool]

# y = response/target
heloc_train_y <- heloc_train[,'RiskPerformance']
heloc_test_y <- heloc_test[,'RiskPerformance']
```

EDA

```r
# correlations
corrplot::corrplot(stats::cor(heloc_train_x),
                   number.cex = 0.5,
                   tl.cex = 0.8)
```

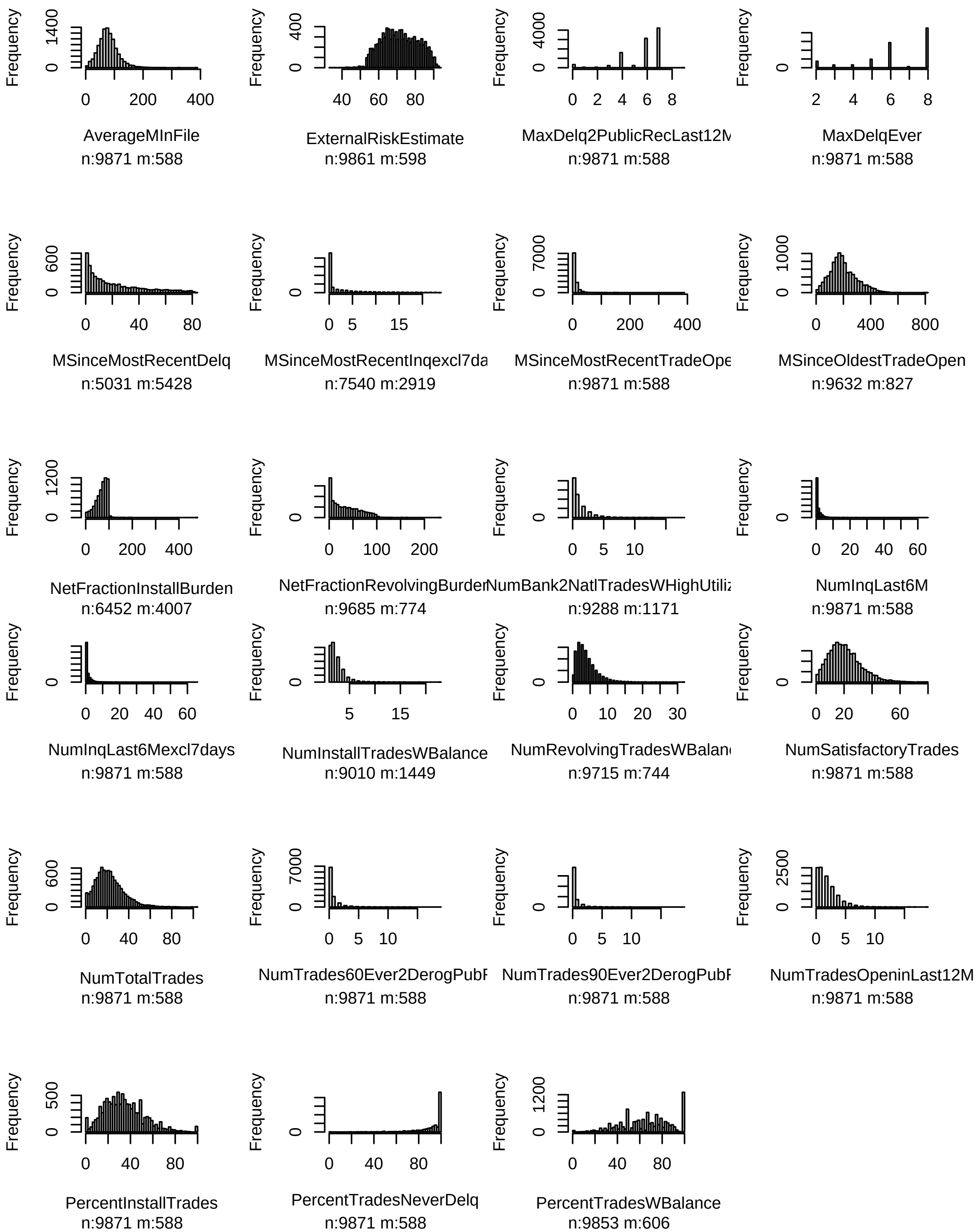


Correlation Plot

Histograms

Overall w/out imputation

```
# histograms to view predictor variable frequencies
par(mfrow=c(3,4))
Hmisc::hist.data.frame(heloc[,1:23])
```




```
skewed <- apply(heloc_train_x, 2, moments::skewness)
skewed[skewed > 0.05]
```

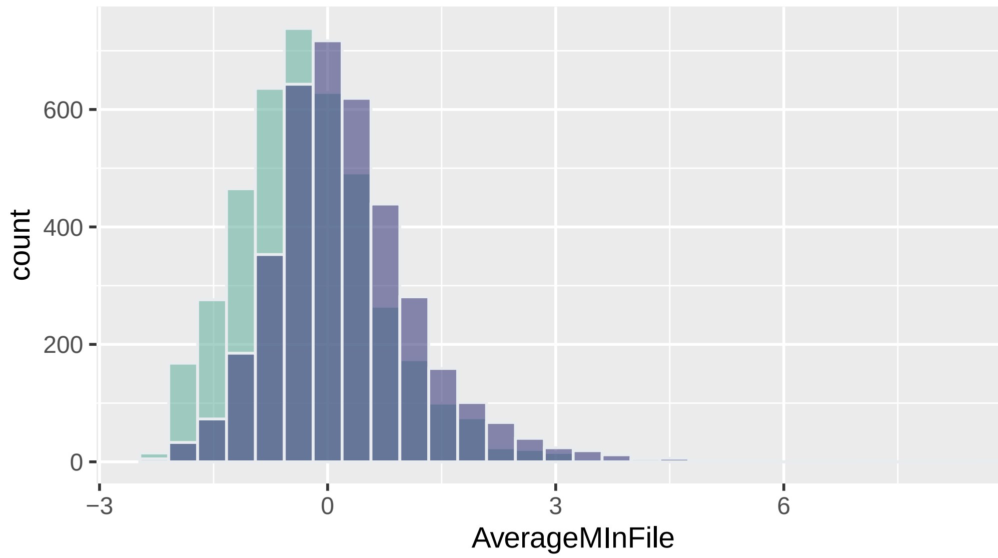
```
##                    AverageMInFile              MSinceMostRecentDelq
```

```
##                           0.9496409                           0.1609974
##        MSinceMostRecentInqexcl7days         MSinceMostRecentTradeOpen
##                           2.3532412                           7.2717628
##               MSinceOldestTradeOpen        NetFractionRevolvingBurden
##                           0.7011420                           0.6041554
## NumBank2NatlTradesWHighUtilization                        NumInqLast6M
##                           2.5735333                           6.8037482
##             NumInstallTradesWBalance        NumRevolvingTradesWBalance
##                           2.6761977                           1.7068060
##               NumSatisfactoryTrades                      NumTotalTrades
##                           0.7914751                           0.9172071
##          NumTrades90Ever2DerogPubRec             NumTradesOpeninLast12M
##                           5.3998292                           1.4957987
##                 PercentInstallTrades
##                           0.6749753
```

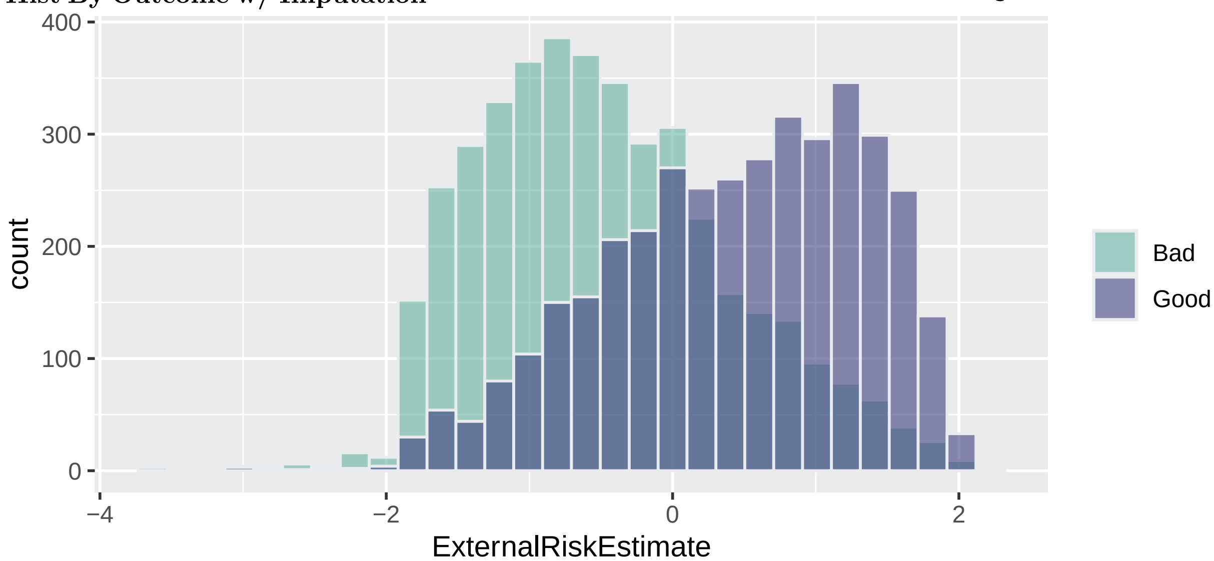
```r
skew_count <- length(skewed[skewed > 0.05])
cat('We have', skew_count,'skewed variables.')
```

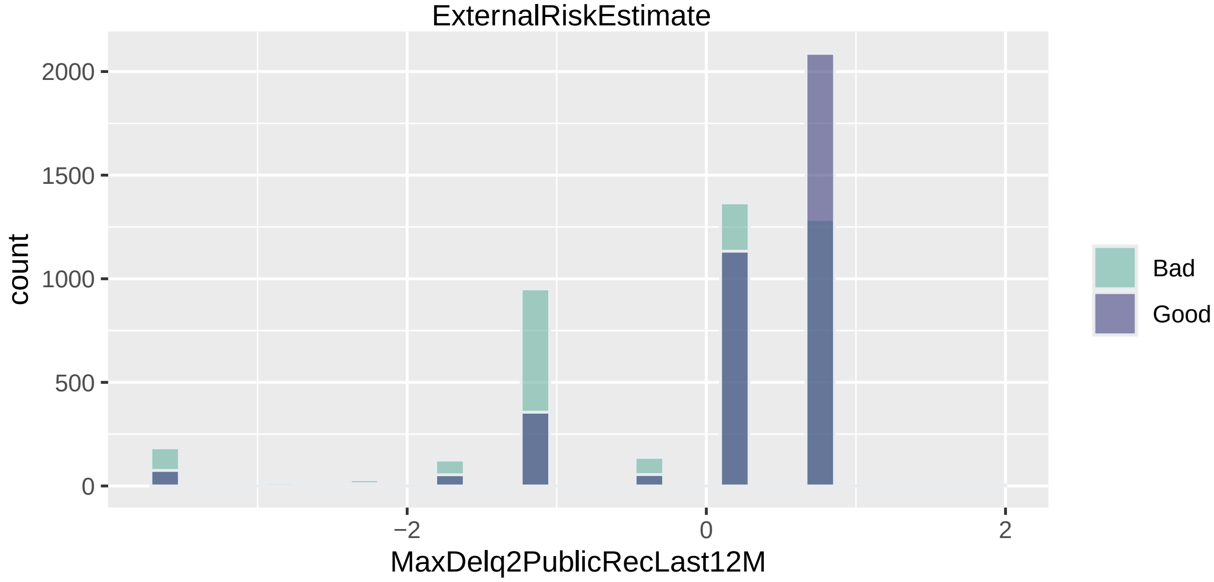
```
## We have 15 skewed variables.
```

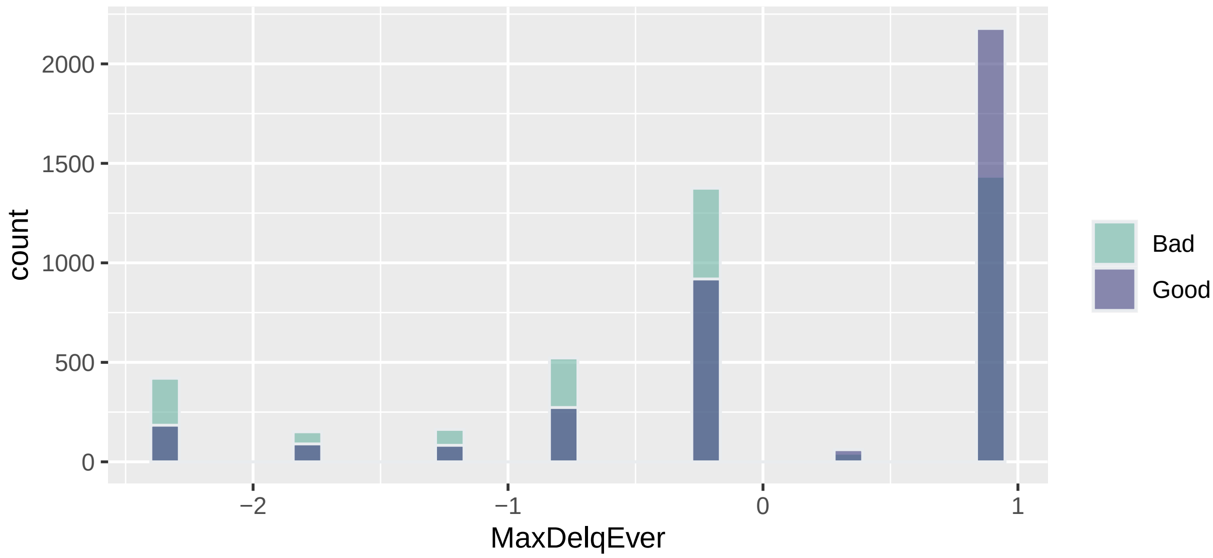
```r
# Deselect Bool Outcome/Response/Target variable
no_risk_bool <- names(heloc_train) != 'RiskPerformance'

# heloc_imputed_full_set <- dplyr::as_tibble(heloc_imputed_full_set)
heloc_ifs_names <- colnames(heloc_train[,no_risk_bool])

# empty list to gather all
plot_list <- list()
for (name in heloc_ifs_names){
  p <- heloc_train %>%
    ggplot( aes(x=heloc_train[,name], fill=RiskPerformance)) +
      geom_histogram( bins=30, color="#e9ecef", alpha=0.6, position = 'identity') +
      scale_fill_manual(values=c("#69b3a2", "#404080")) +
      xlab(name) +
      labs(fill="")
  print(p)
}
```

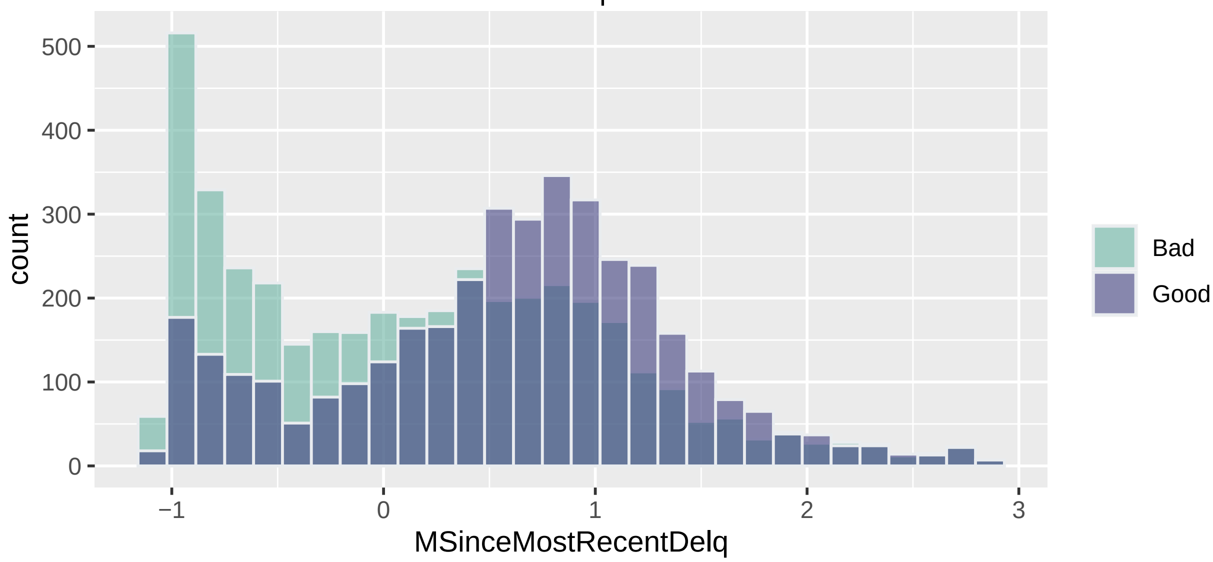


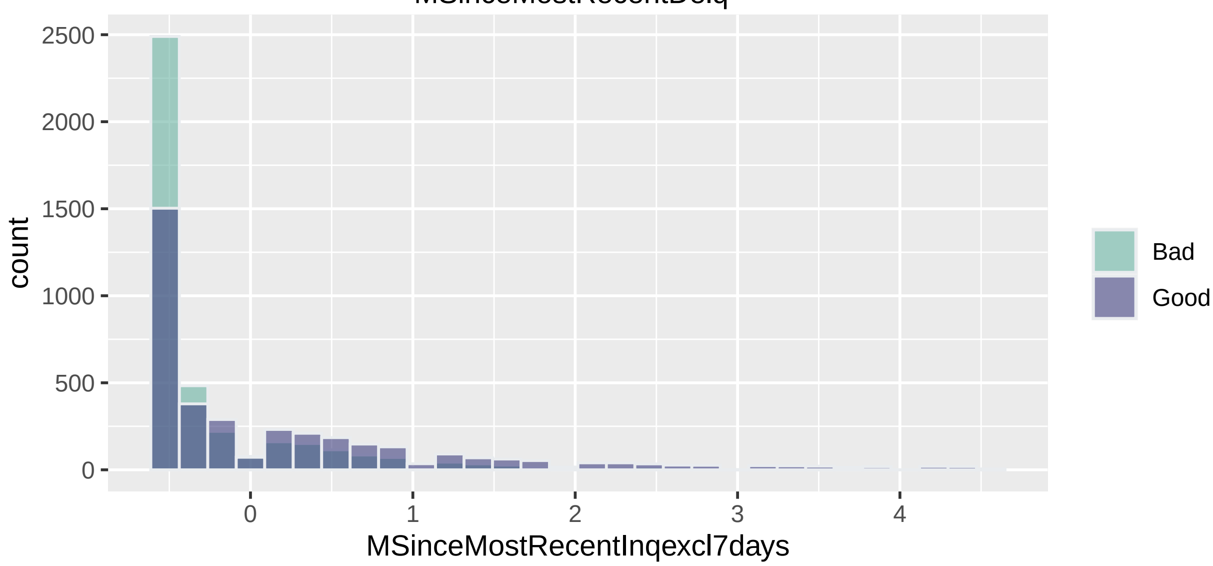
Hist By Outcome w/ Imputation

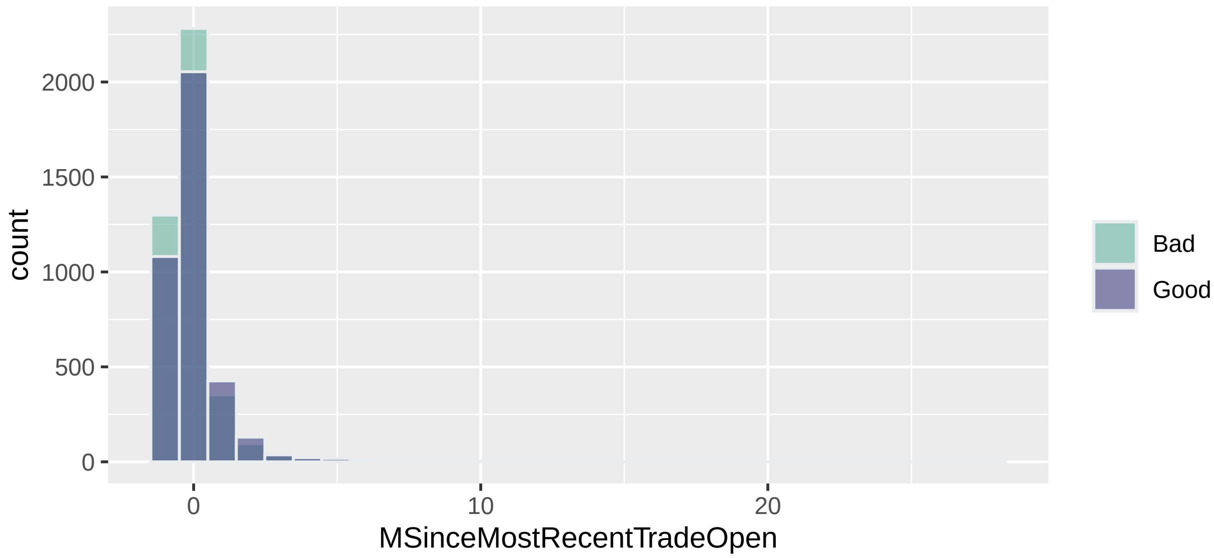


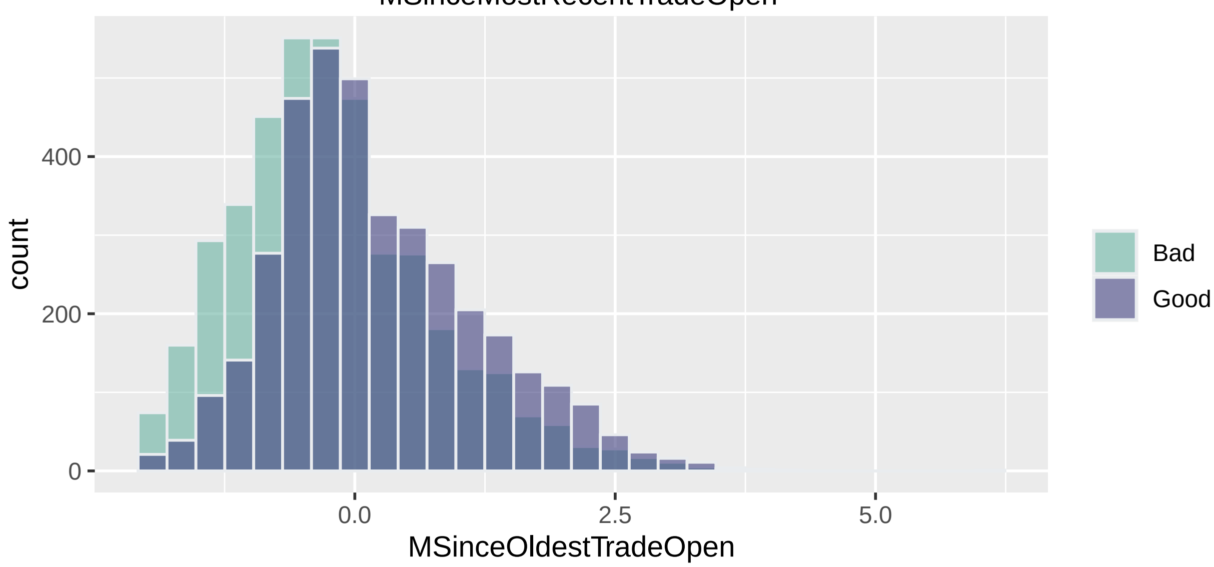


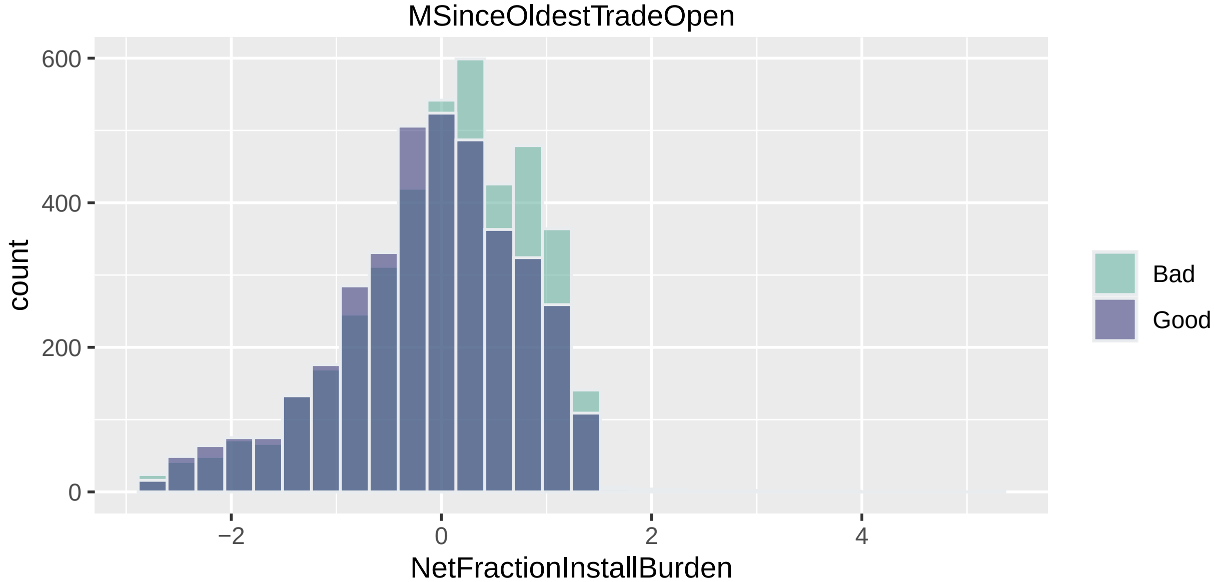

count
2000
1500
1000
500
0
−2
−1
0
1
MaxDelqEver
Bad
Good

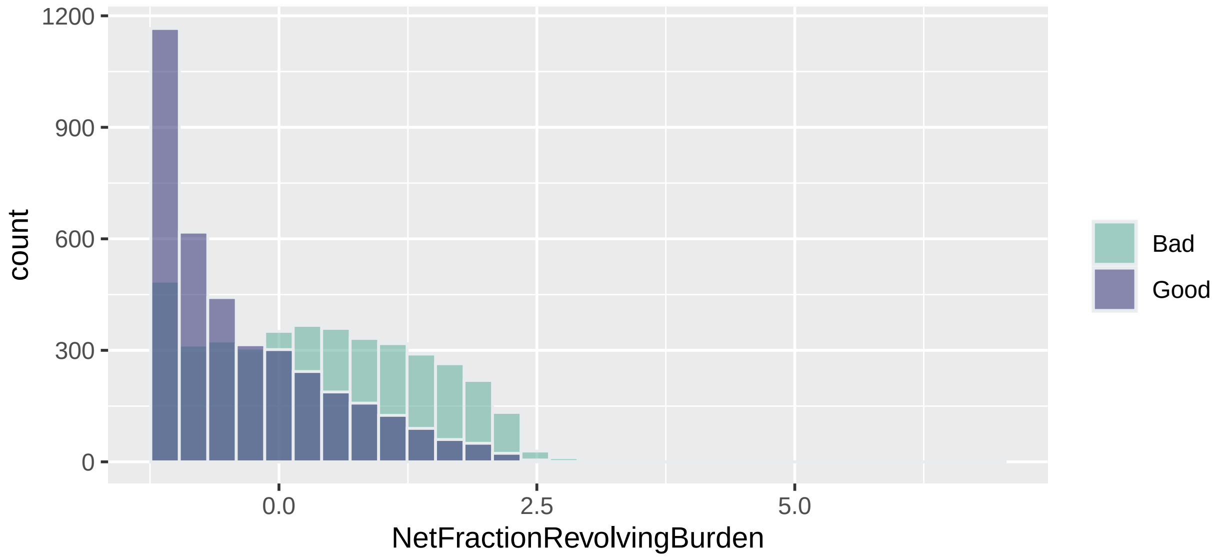

count
500
400
300
200
100
0
−1
0
1
2
3
MSinceMostRecentDelq
Bad
Good


count
2500
2000
1500
1000
500
0
0
1
2
3
4
MSinceMostRecentInqexcl7days
Bad
Good

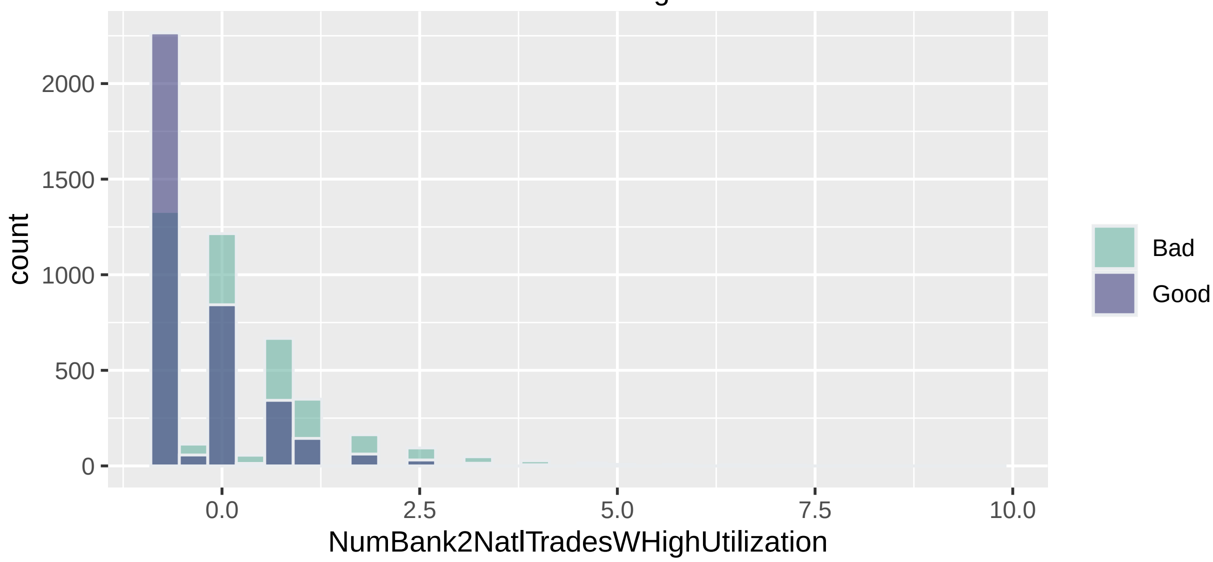

2000
1500
1000
500
0
count
0
10
20
MSinceMostRecentTradeOpen
Bad
Good


400
200
0
count
0.0
2.5
5.0
MSinceOldestTradeOpen
Bad
Good

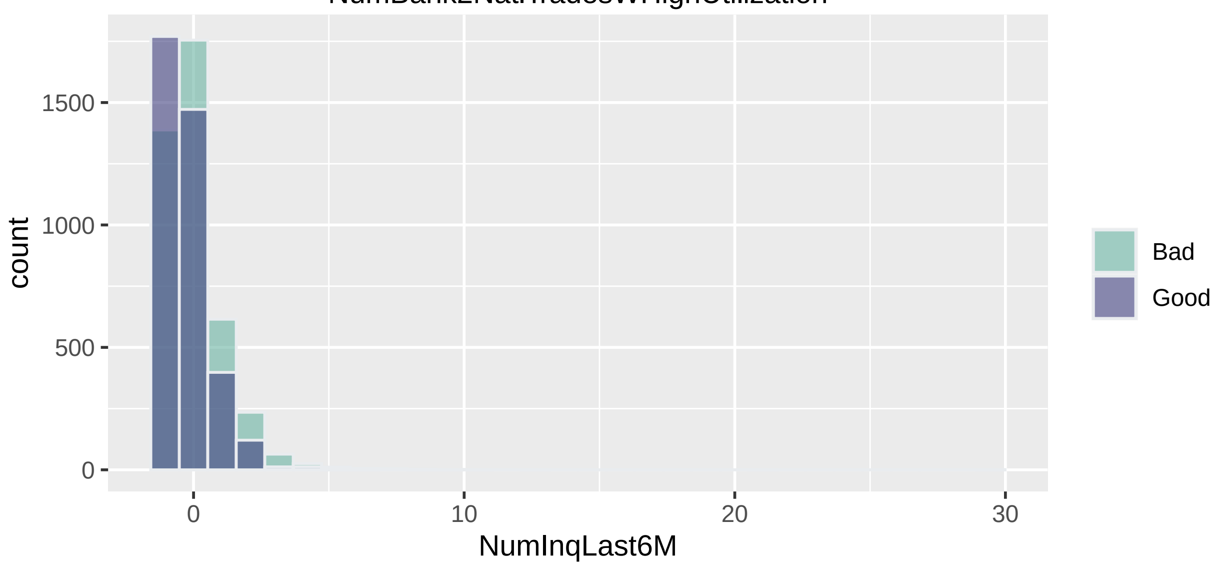

600
400
200
0
count
−2
0
2
4
NetFractionInstallBurden
Bad
Good



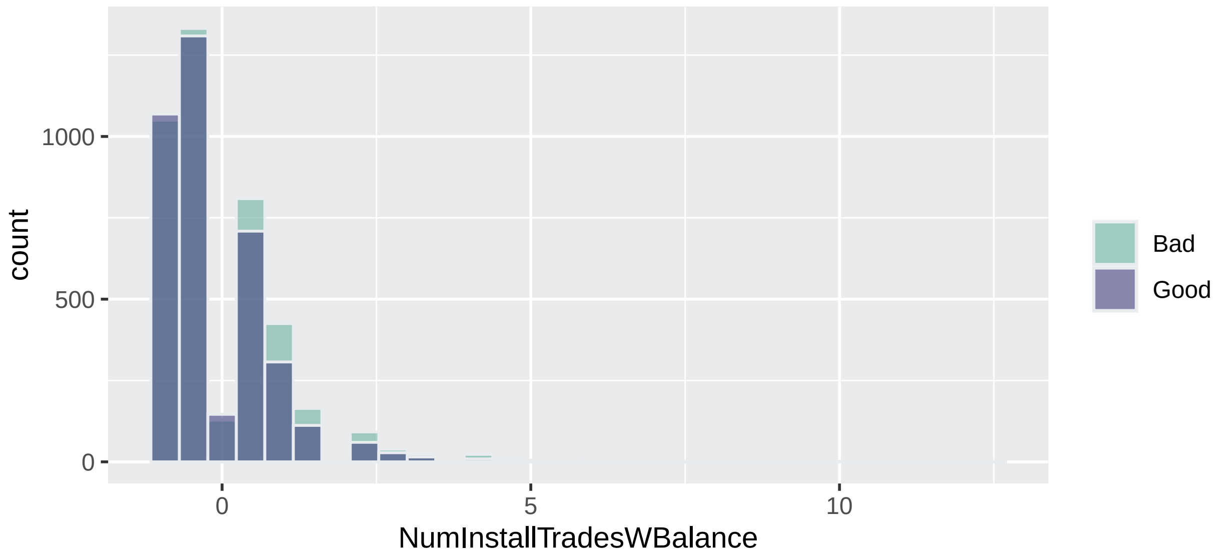
count
1200
900
600
300
0
0.0
2.5
5.0
NetFractionRevolvingBurden
Bad
Good

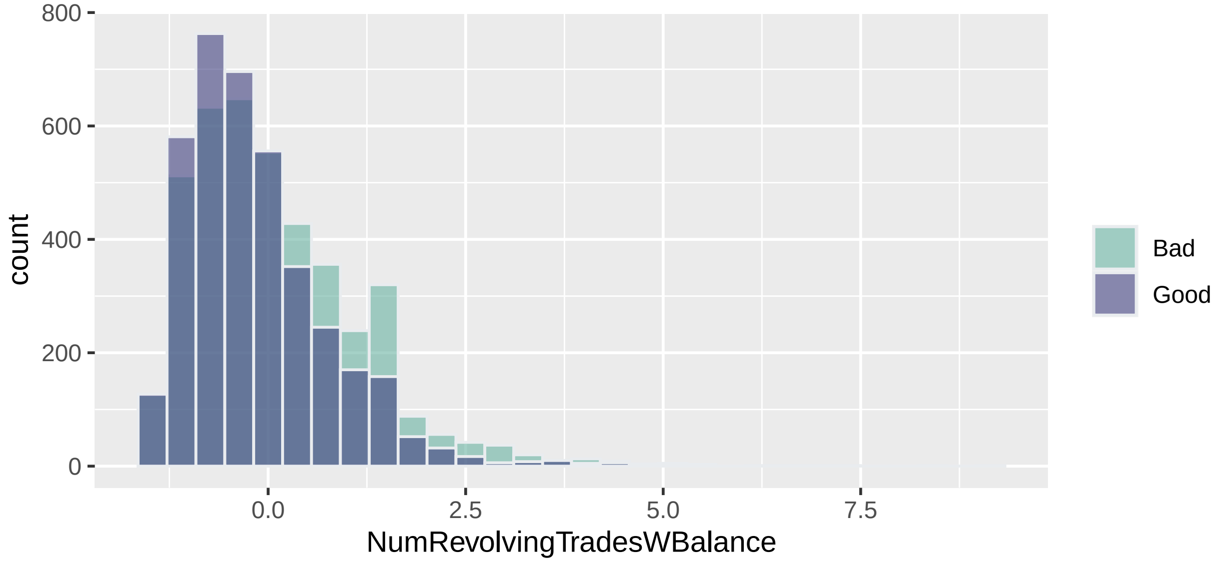


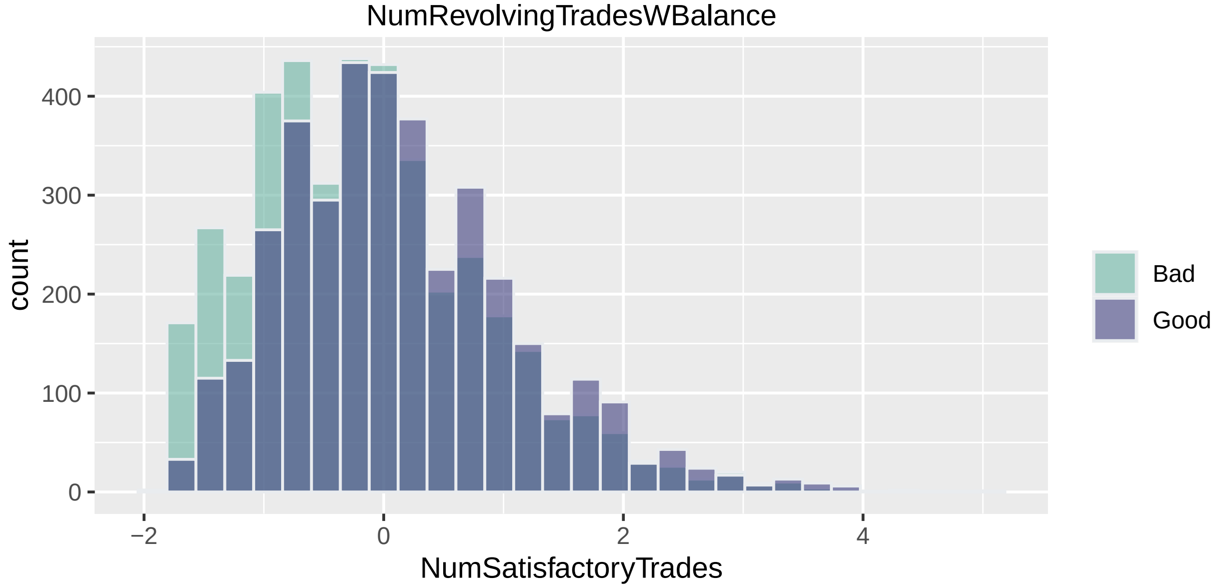
count
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
NumBank2NatlTradesWHighUtilization
Bad
Good

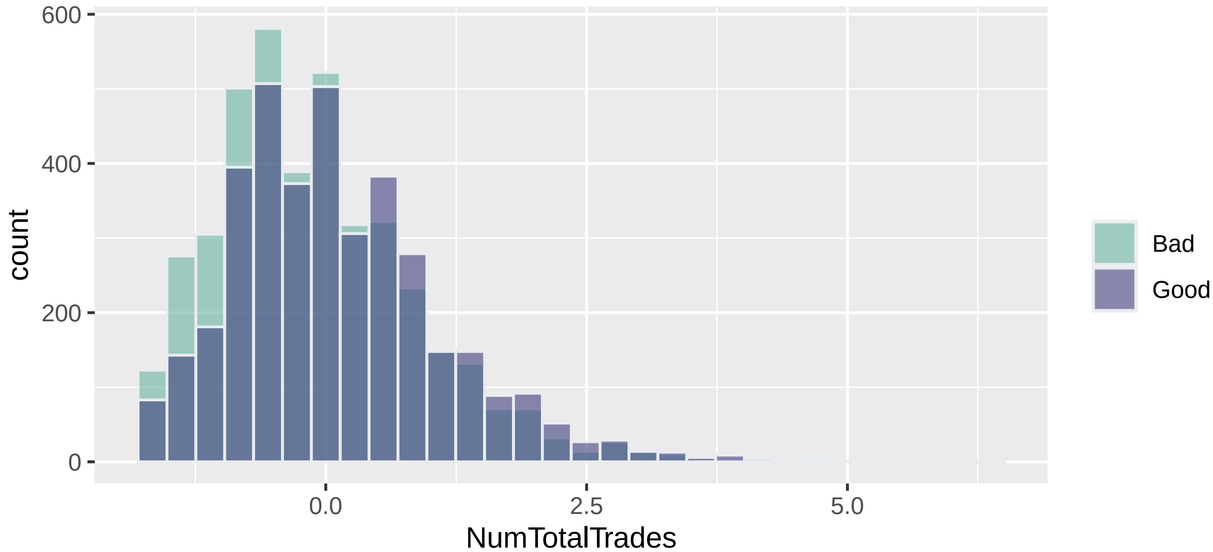


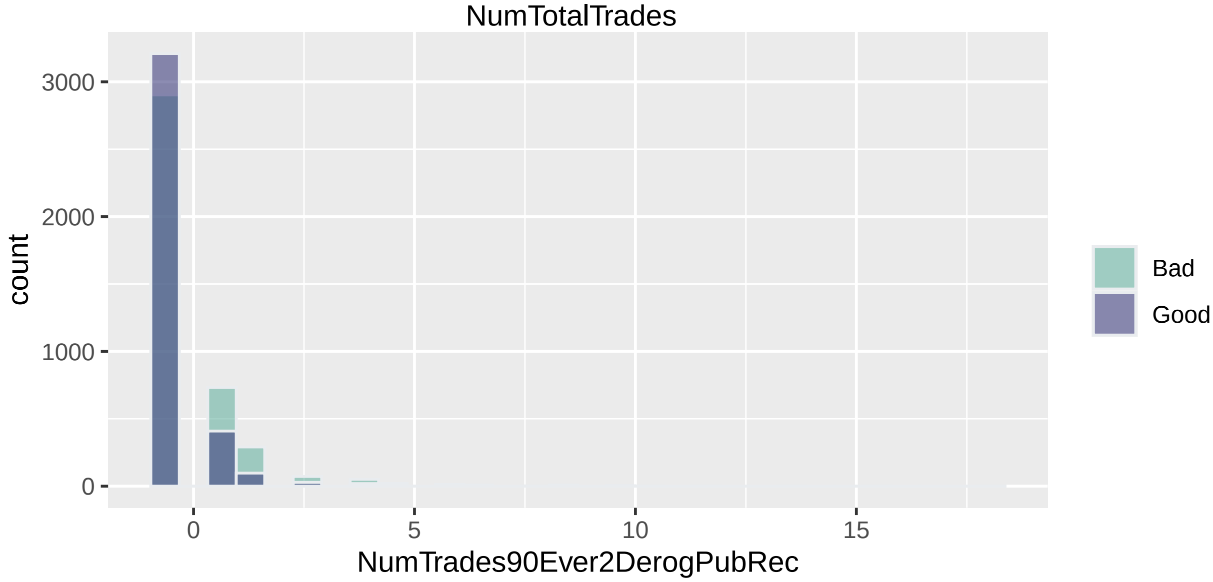
count
1500
1000
500
0
0
10
20
30
NumInqLast6M
Bad
Good

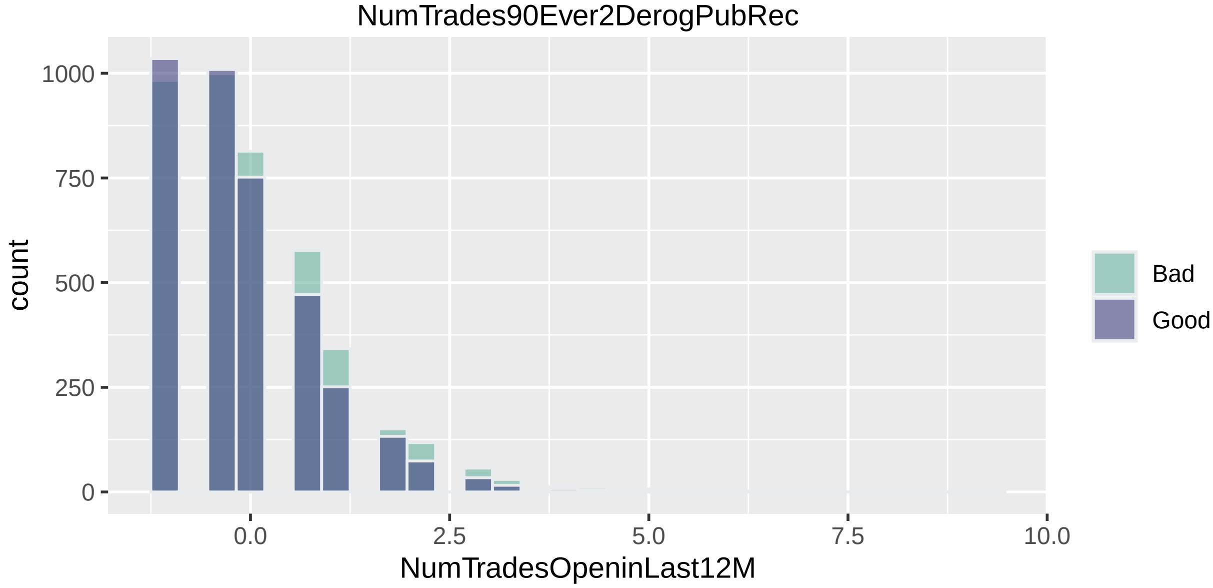

1000
500
0
count
0
5
10
NumInstallTradesWBalance
Bad
Good

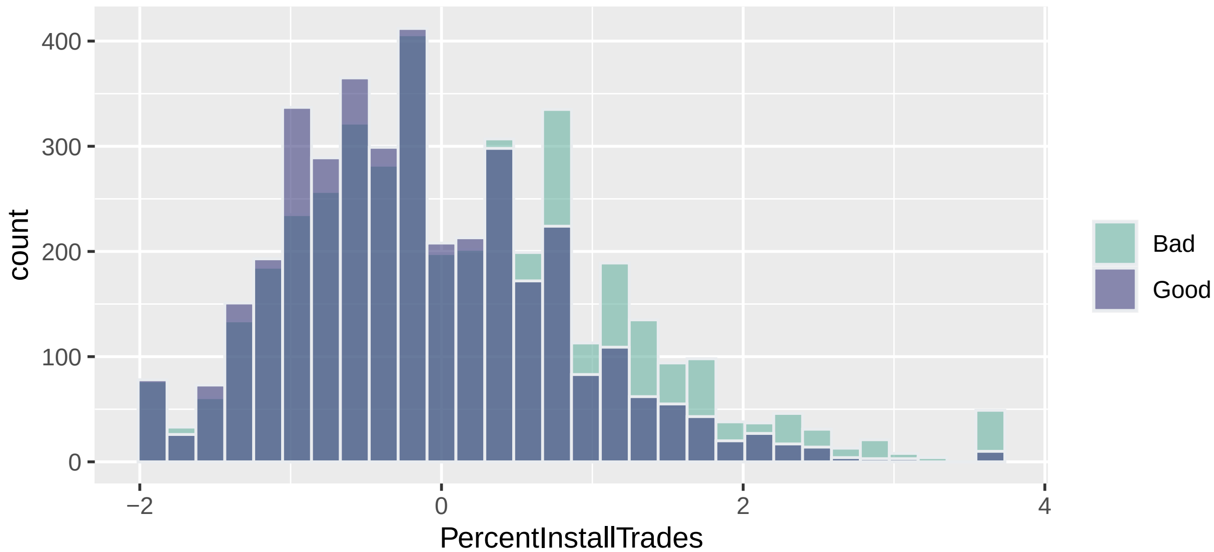

800
600
400
200
0
count
0.0
2.5
5.0
7.5
NumRevolvingTradesWBalance
Bad
Good

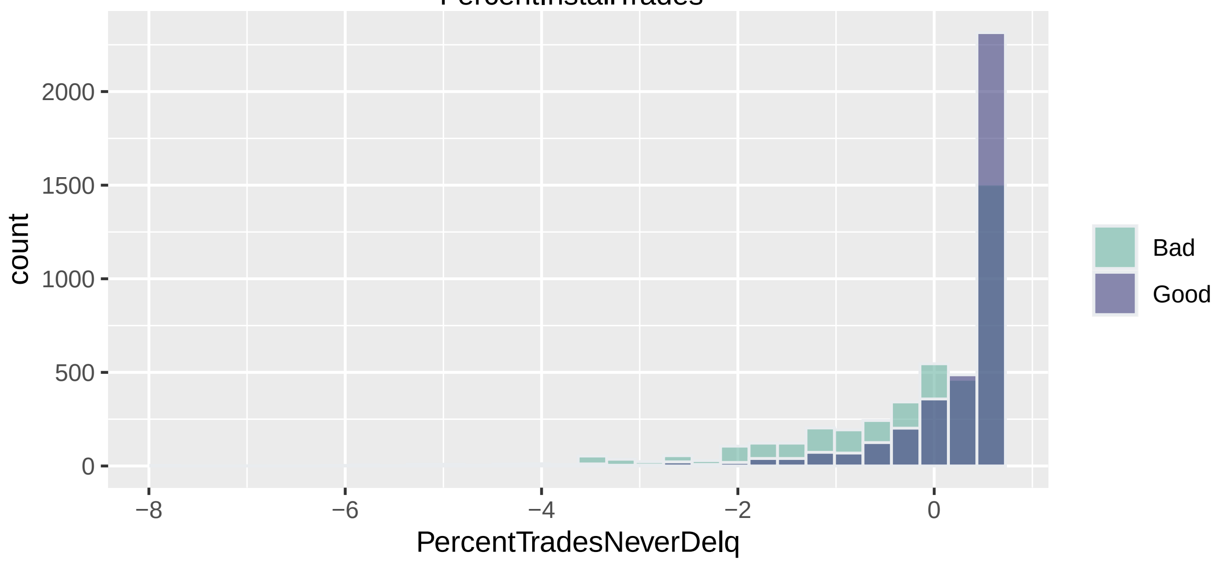

400
300
200
100
0
count
−2
0
2
4
NumSatisfactoryTrades
Bad
Good

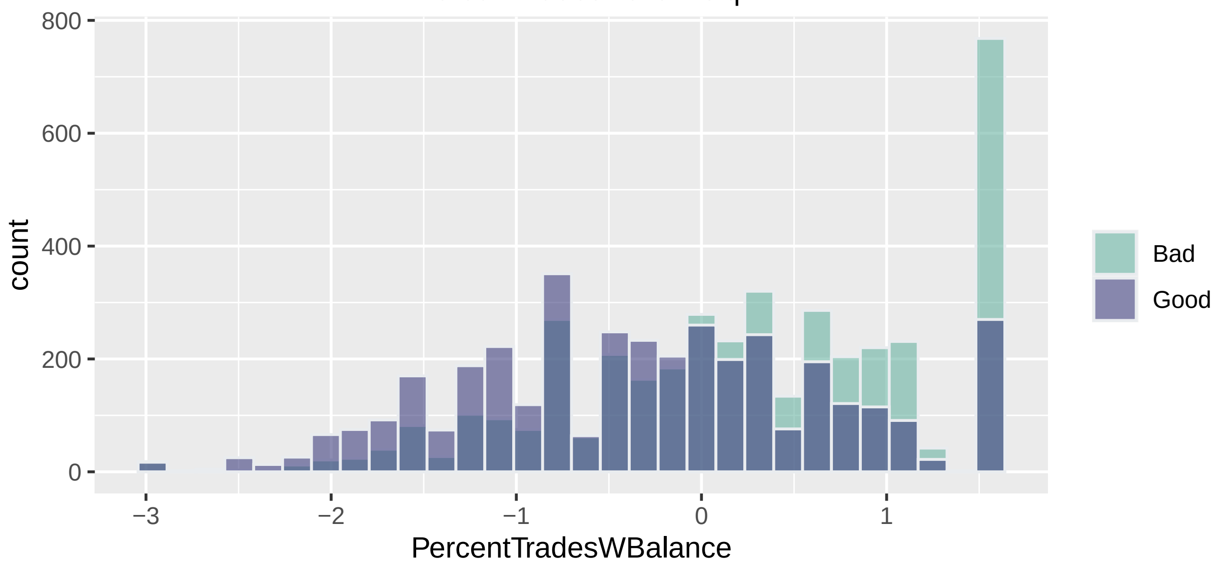

600
400
200
0
count
0.0
2.5
5.0
NumTotalTrades
Bad
Good


3000
2000
1000
0
count
0
5
10
15
NumTrades90Ever2DerogPubRec
Bad
Good


1000
750
500
250
0
count
0.0
2.5
5.0
7.5
10.0
NumTradesOpeninLast12M
Bad
Good

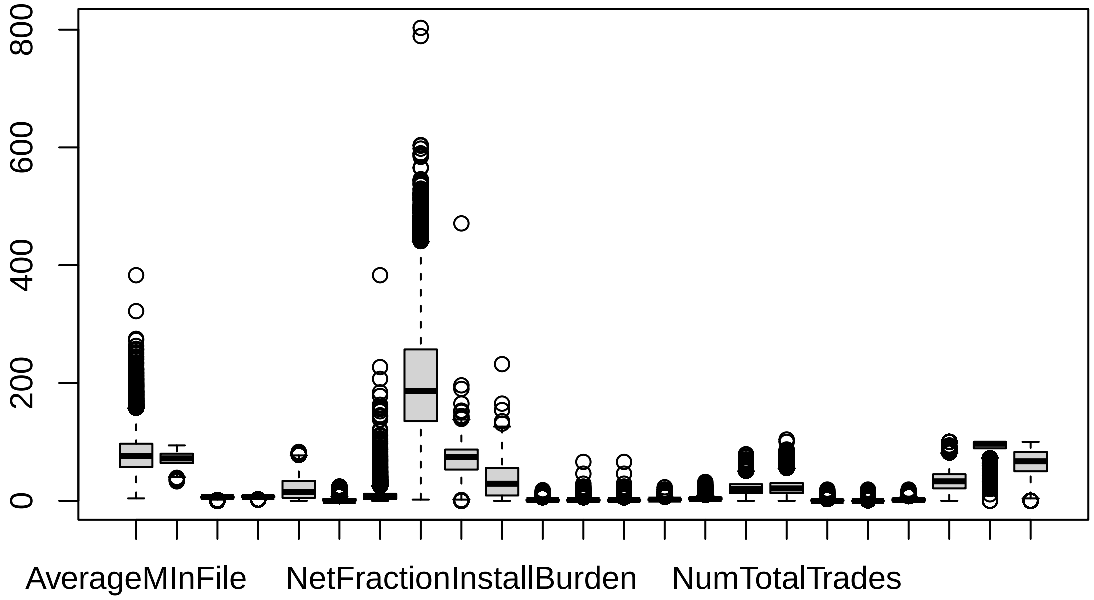

count
400
300
200
100
0
−2
0
2
4
PercentInstallTrades
Bad
Good


count
2000
1500
1000
500
0
−8
−6
−4
−2
0
PercentTradesNeverDelq
Bad
Good


count
800
600
400
200
0
−3
−2
−1
0
1
PercentTradesWBalance
Bad
Good

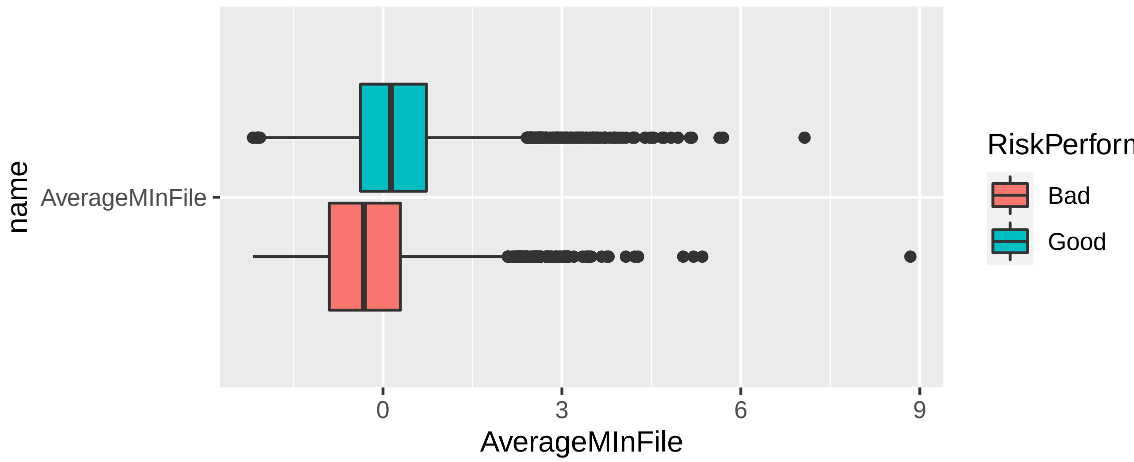
BoxPlots

```
# boxplot to view outliers
boxplot(heloc[, 1:23])
```

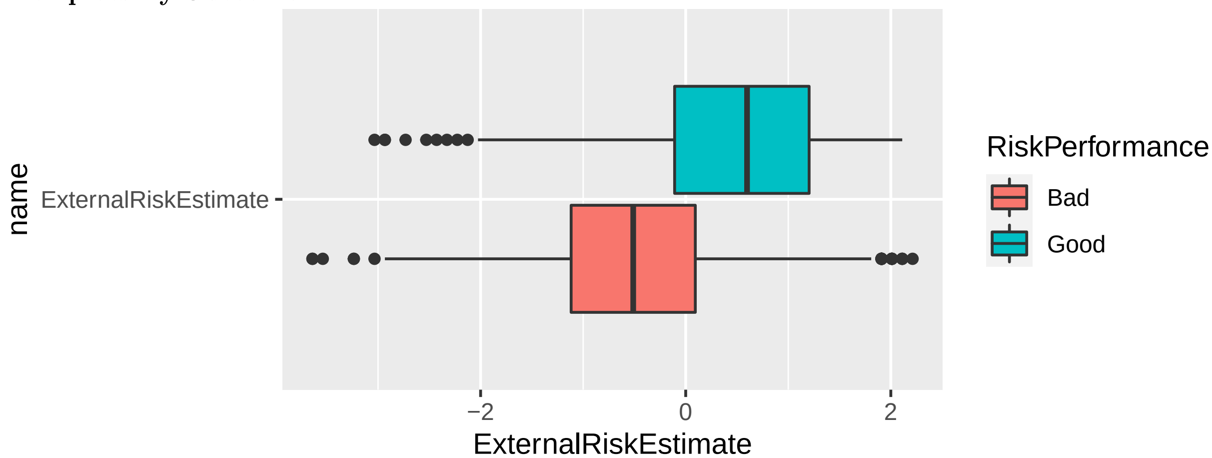


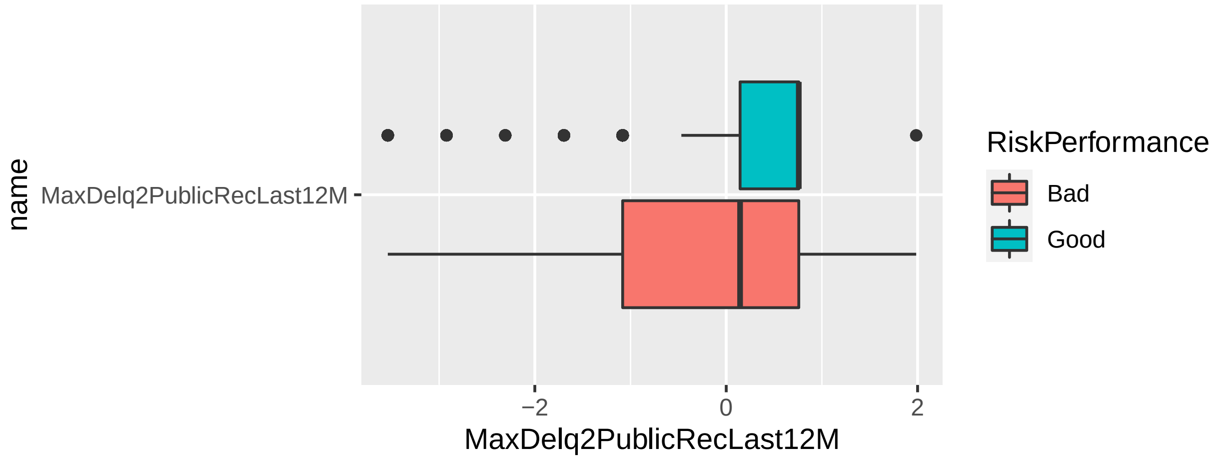
Overall

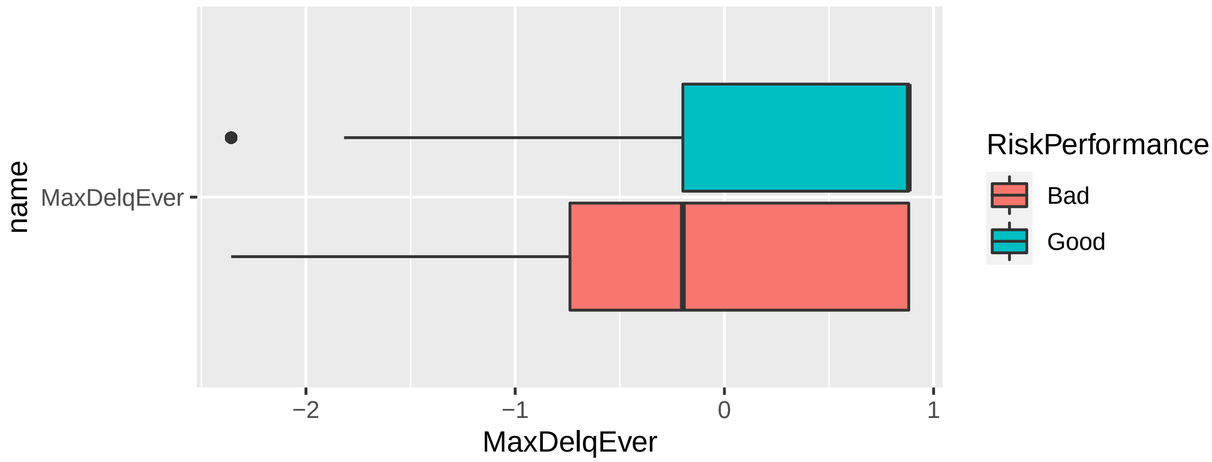
```
for (name in heloc_ifs_names){
  # boxplot to view outliers
  p <- heloc_train %>%
          ggplot( aes(x=heloc_train[,name], y=name, fill=RiskPerformance)) +
          geom_boxplot() +
          xlab(name)
  print(p)
}
```

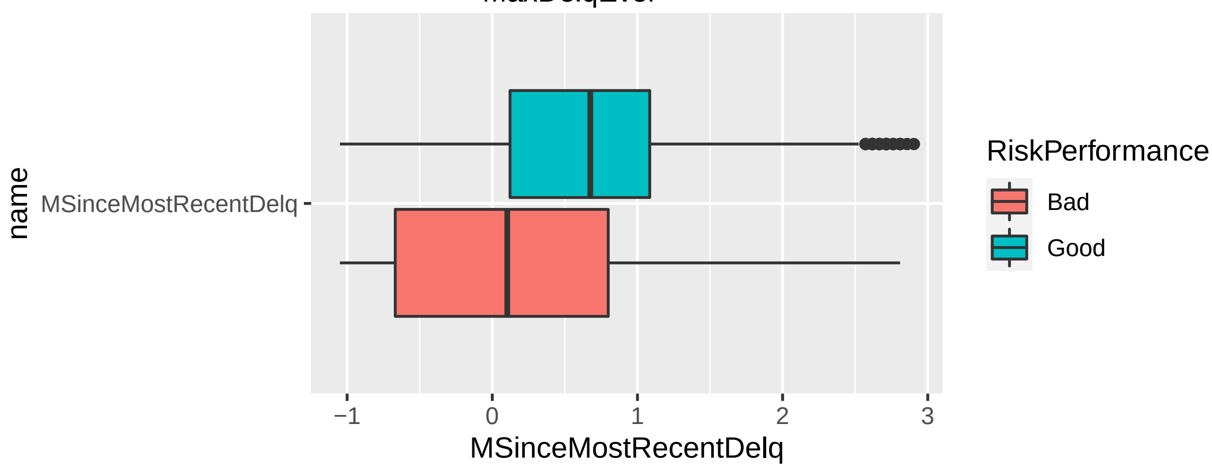


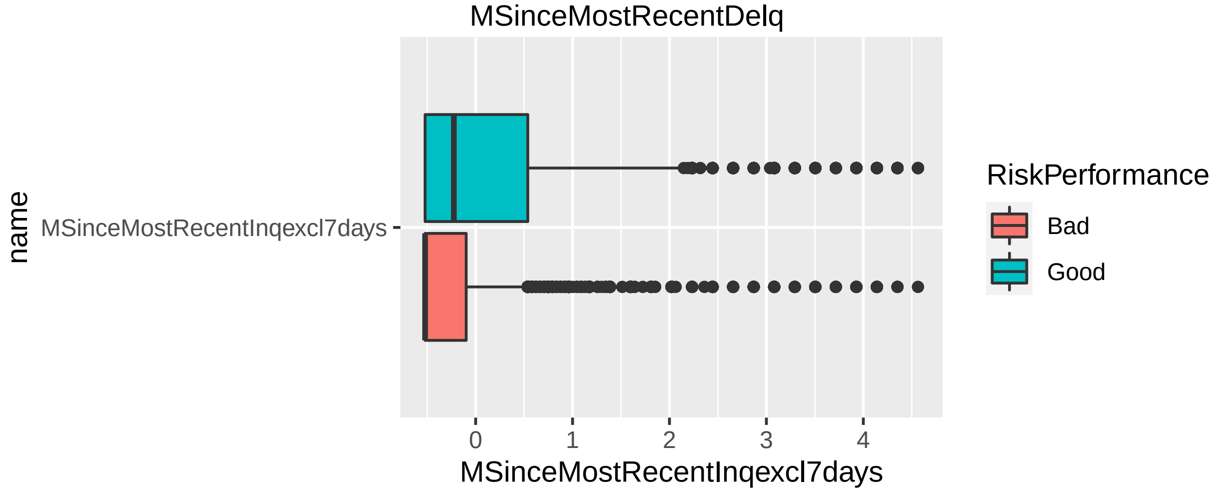
Box plots By Outcome

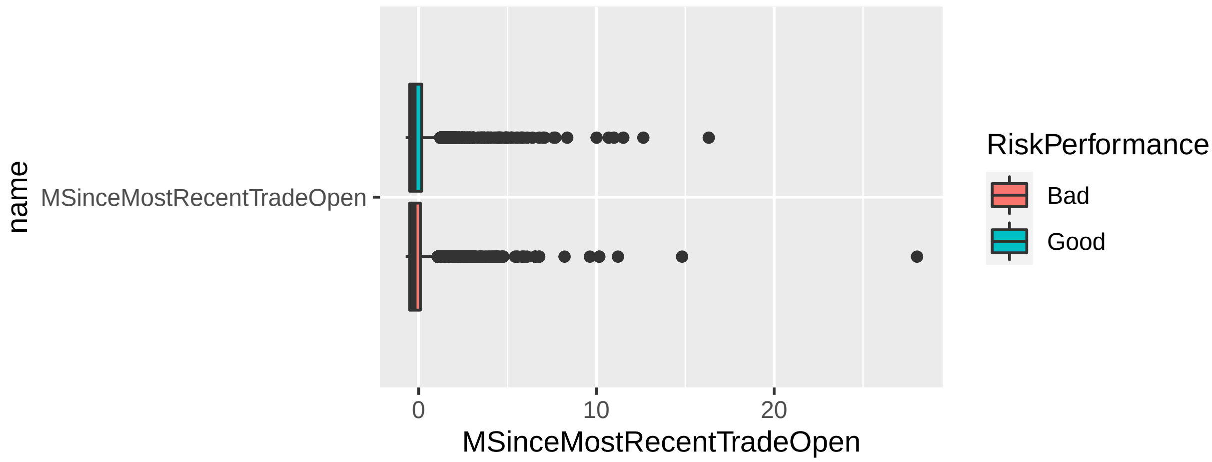


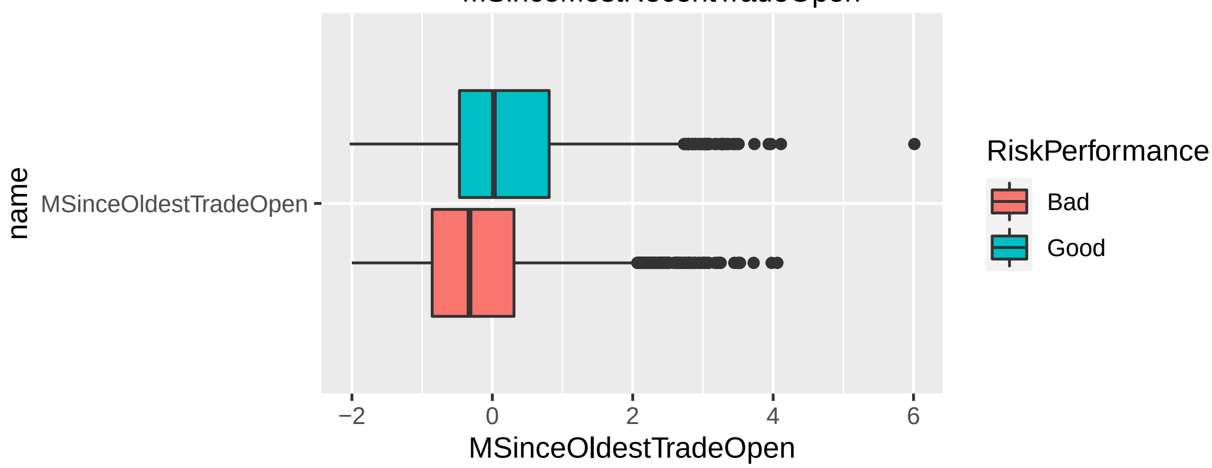


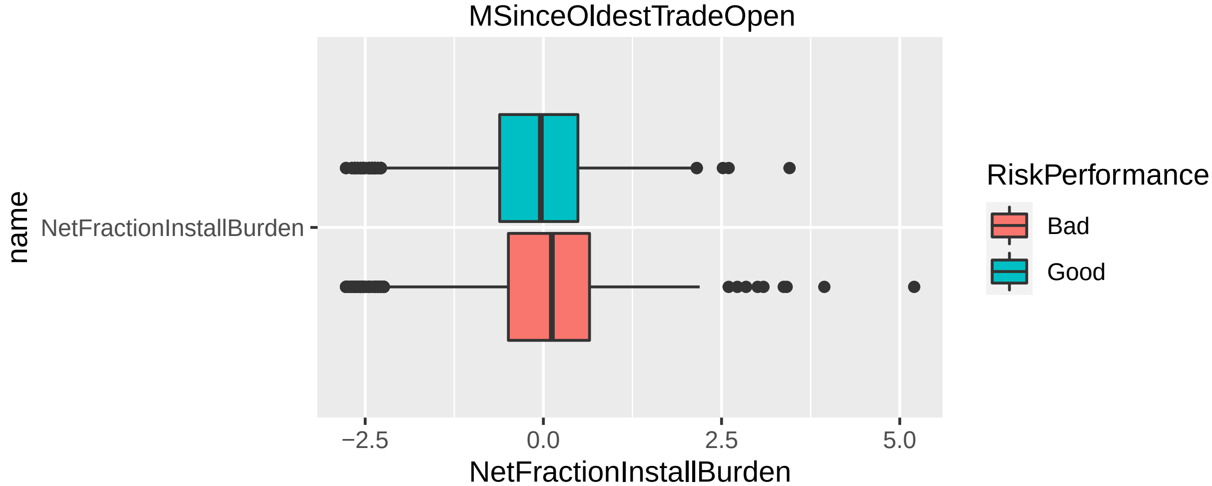

name
MaxDelqEver
−2
−1
0
1
MaxDelqEver
RiskPerformance
Bad
Good

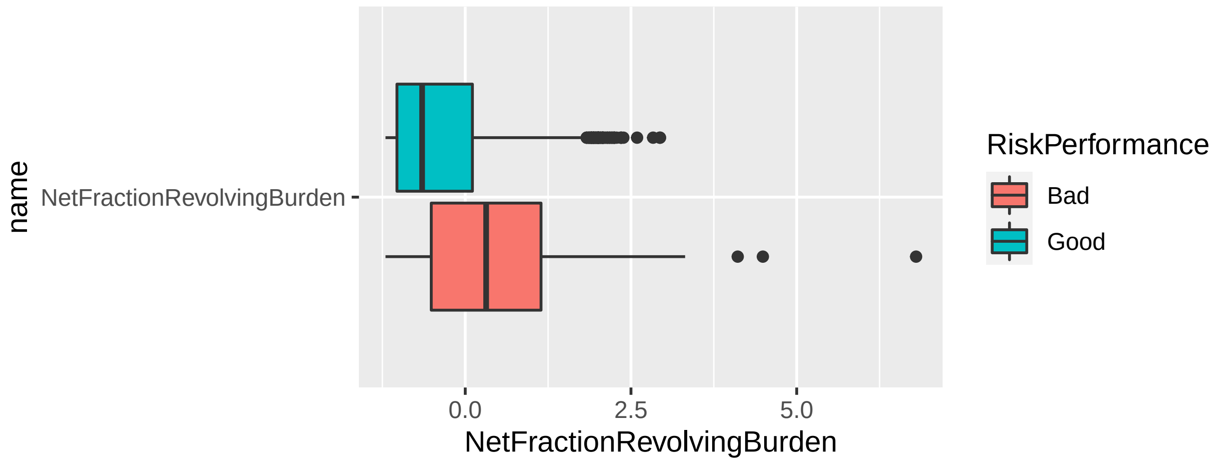

name
MSinceMostRecentDelq
−1
0
1
2
3
MSinceMostRecentDelq
RiskPerformance
Bad
Good


name
MSinceMostRecentInqexcl7days
0
1
2
3
4
MSinceMostRecentInqexcl7days
RiskPerformance
Bad
Good

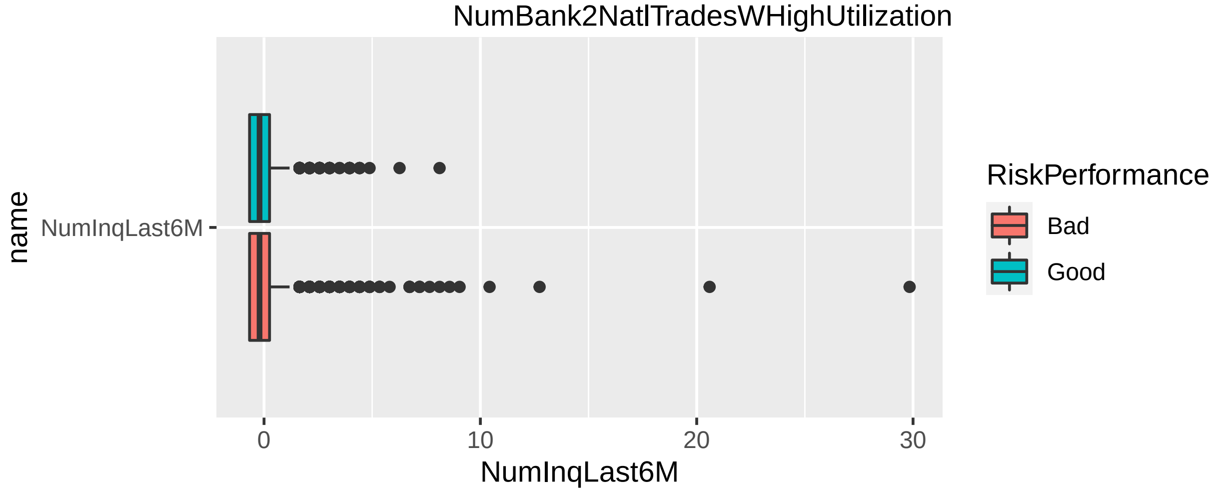

name
MSinceMostRecentTradeOpen
0
10
20
MSinceMostRecentTradeOpen
RiskPerformance
Bad
Good

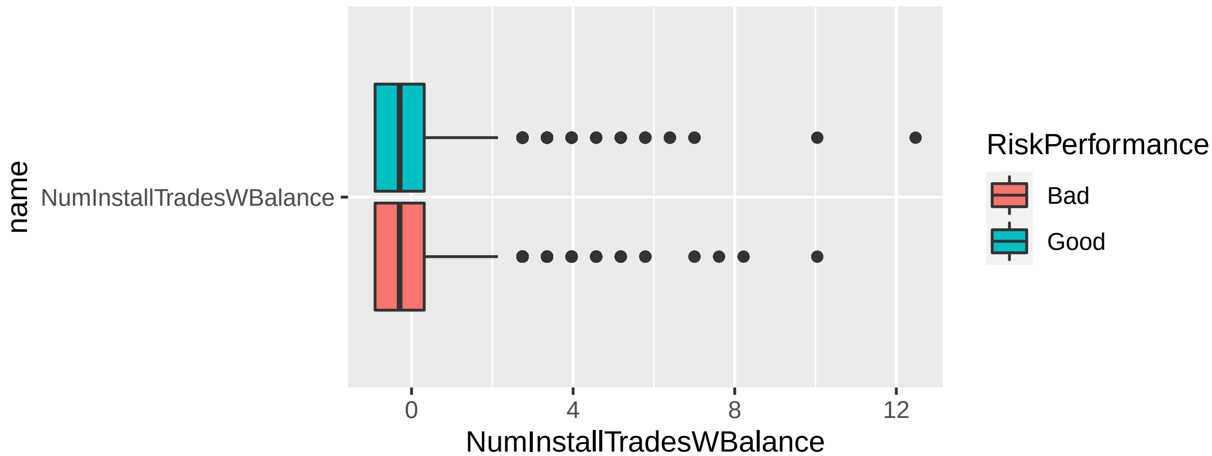

name
MSinceOldestTradeOpen
−2
0
2
4
6
MSinceOldestTradeOpen
RiskPerformance
Bad
Good

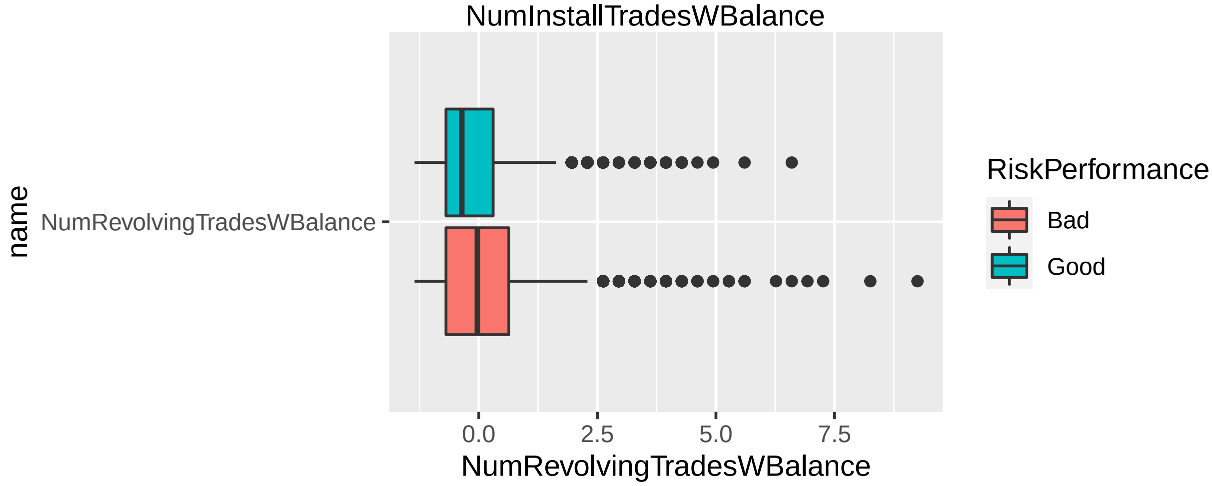

name
NetFractionInstallBurden
−2.5
0.0
2.5
5.0
NetFractionInstallBurden
RiskPerformance
Bad
Good

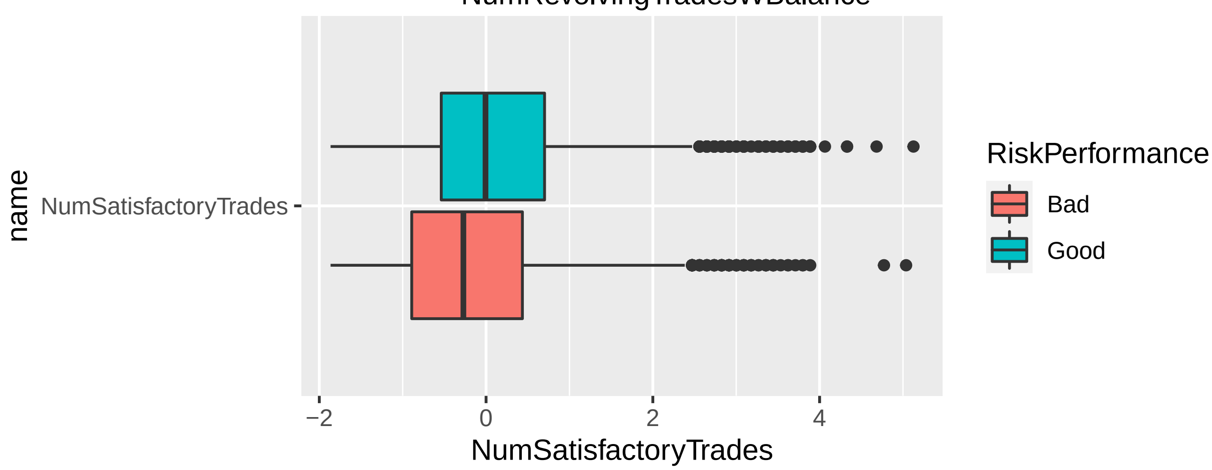

name
NetFractionRevolvingBurden
0.0
2.5
5.0
NetFractionRevolvingBurden
RiskPerformance
Bad
Good

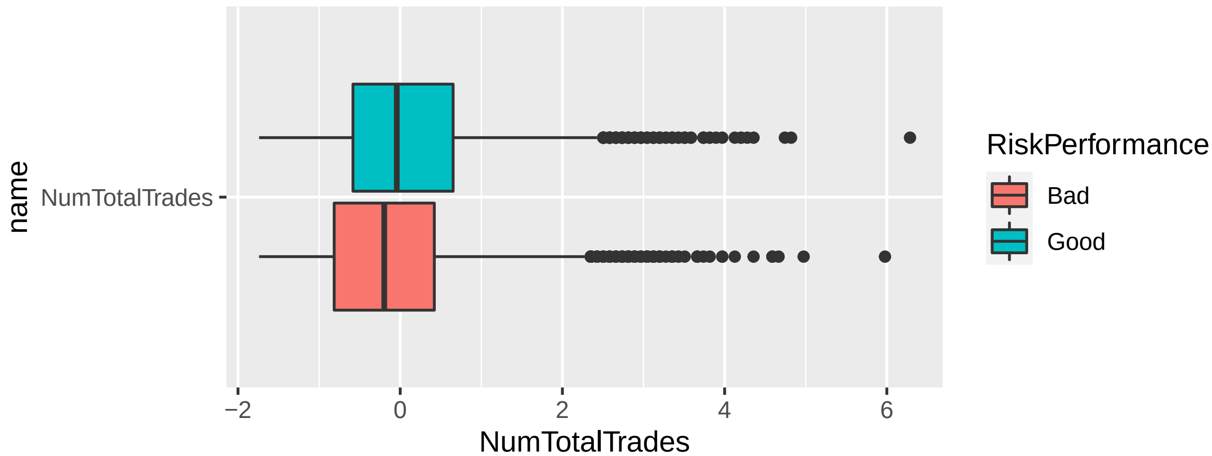

name
NumBank2NatlTradesWHighUtilization
0.0
2.5
5.0
7.5
10.0
NumBank2NatlTradesWHighUtilization
RiskPerformance
Bad
Good

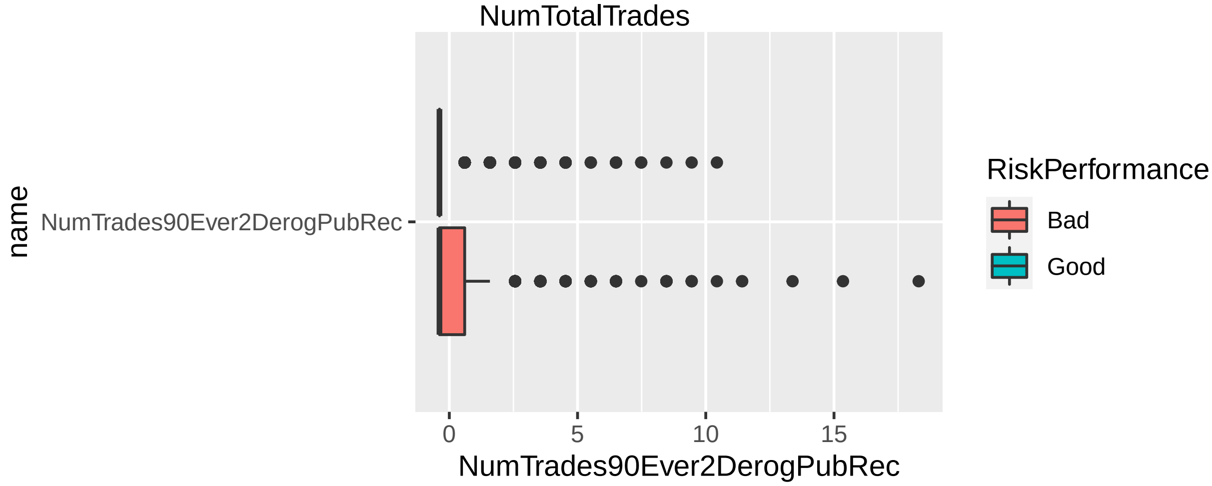

name
NumInqLast6M
0
10
20
30
NumInqLast6M
RiskPerformance
Bad
Good

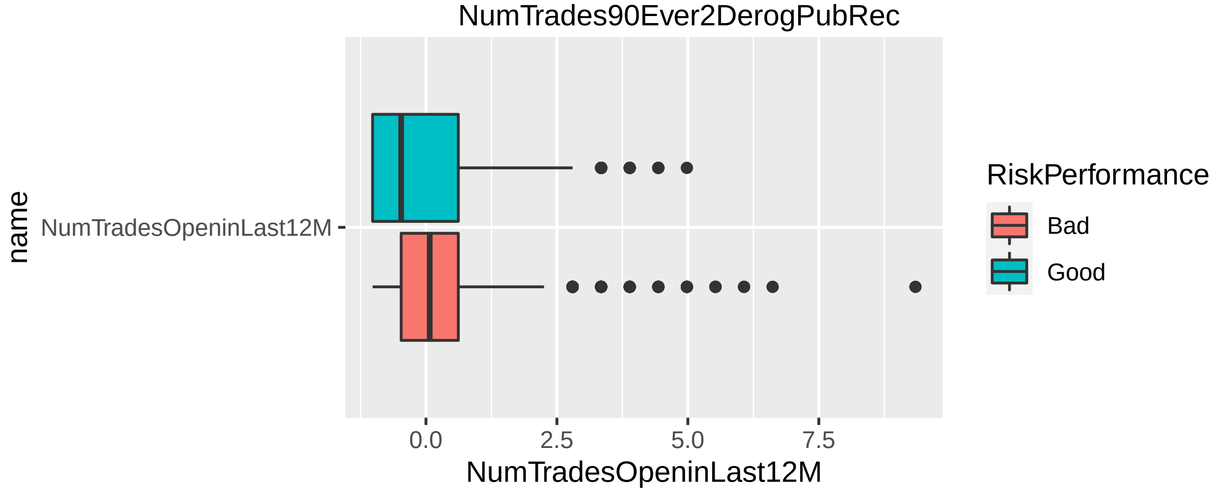

name
NumInstallTradesWBalance
0
4
8
12
NumInstallTradesWBalance
RiskPerformance
Bad
Good

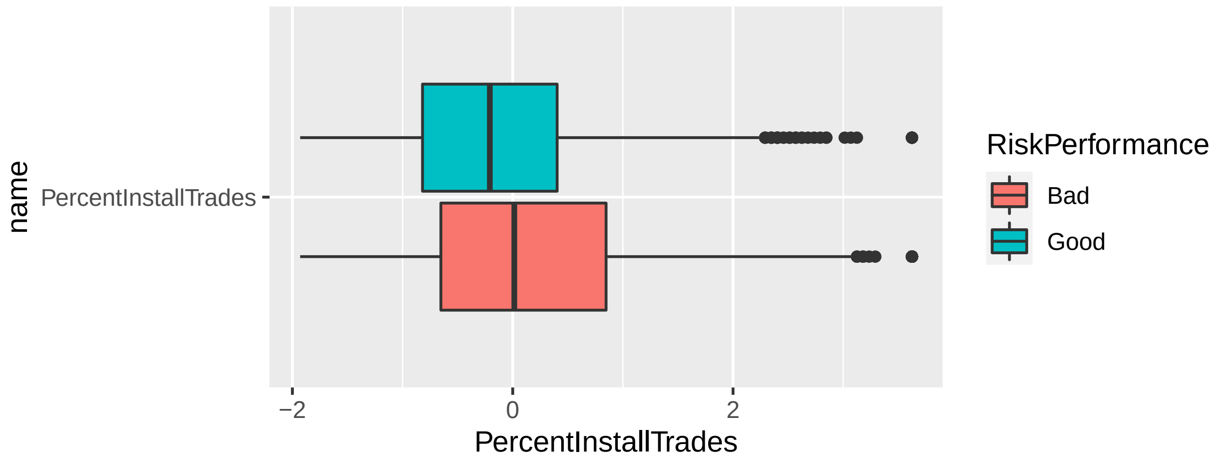

name
NumRevolvingTradesWBalance
0.0
2.5
5.0
7.5
NumRevolvingTradesWBalance
RiskPerformance
Bad
Good

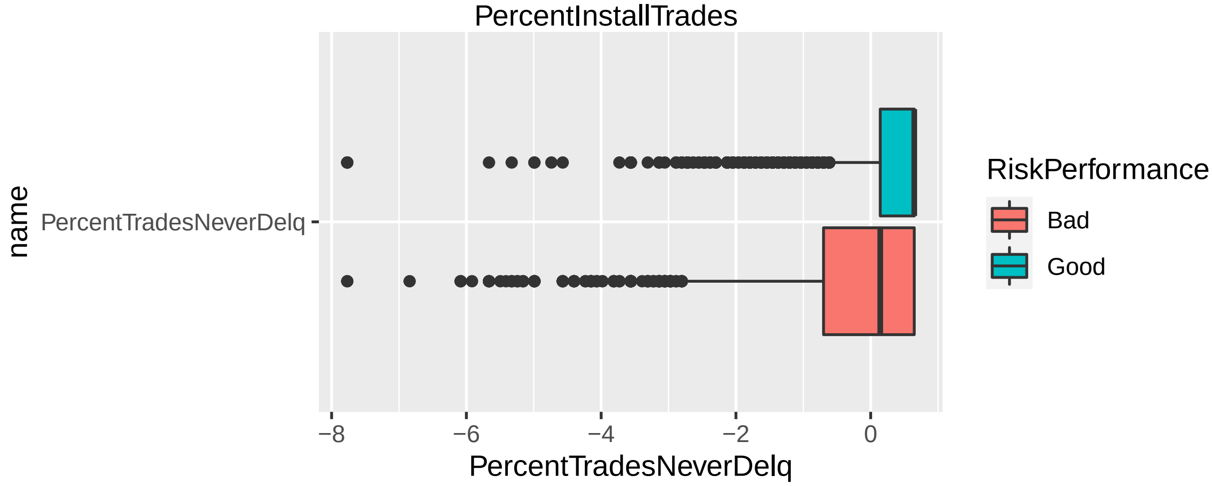

name
NumSatisfactoryTrades
−2
0
2
4
NumSatisfactoryTrades
RiskPerformance
Bad
Good

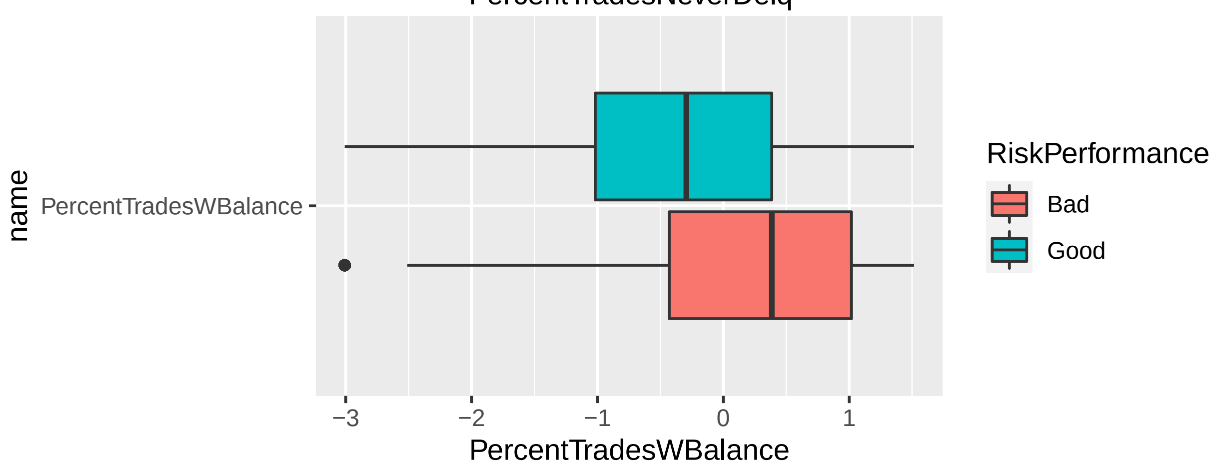

name
NumTotalTrades
−2
0
2
4
6
NumTotalTrades
RiskPerformance
Bad
Good


name
NumTrades90Ever2DerogPubRec
0
5
10
15
NumTrades90Ever2DerogPubRec
RiskPerformance
Bad
Good


name
NumTradesOpeninLast12M
0.0
2.5
5.0
7.5
NumTradesOpeninLast12M
RiskPerformance
Bad
Good







Models

With the following models, we are trying to maximize the SPECIFICITY of the models to prevent any non-qualified loanees being presented with a HELOC loan.

Specificity is defined as: "The specificity is defined as the rate that nonevent samples are predicted as nonevents" (Kuhn & Johnson, 2013)

```
control <- caret::trainControl(method="cv",
                               classProbs=TRUE,
                               savePredictions=TRUE,
```

```
                           summaryFunction=twoClassSummary)
```

```
bestIndex <- function(model){
  # returns top ROC value and surrounding indices from model
  highest_score <- max(model$results$ROC)

  # get row index and convert to type Int
  best_index <- rownames(model$results[model$results$ROC == highest_score,])
  best_index <- as.integer(best_index)

  return(best_index)
}

confusionMatrix <- function(testResults.model){
  caret::confusionMatrix(testResults.model,
                         as.factor(testResults$obs),
                         positive="Good")
}

importanceRanker <- function(model_varImp.importance, name){
  # coerce ranking into dataframe structure for manipulation
  df <- data.frame(model_varImp.importance)

  df_column_count <- dim(df)[2]
  # if dataframe has 2 columns, reduce to 1
  # NOTE: Values in both columns are the same
  if (df_column_count == 2){
    # grab only 1 column
    df <- df[,1, drop=FALSE]
  }

  # overwrite name
  names(df) <- name
  # reverse order ranking (highest num ranked = 1)
  df[,name] <- rank(-df[,name])
  df <- t(df)
  return(df)
}

modelScoreBoard <- function(testResults){
  # feed in testResults dataframe and out comes a model scoreboard!
  scoreboard <- data.frame()

  bool <- names(testResults) != 'obs'
  col_names <- colnames(testResults[,bool])

  for (colname in col_names){

    testResults.model <- testResults[,colname]
    cf <- caret::confusionMatrix(testResults.model,
                                 as.factor(testResults$obs),
                                 positive="Good")

    # gather testResults
```

```r
    acc <- data.frame(metric=cf$overall[1])
    # gather Precision, Sensitivity, Specificity, & F1
    metrics <- list(cf$byClass[c(5,1,2,7)])
    metrics <- data.frame(Metrics=metrics)
    names(metrics) <- 'metric'
    # gather all metrics in 1 df
    metrics <- rbind(acc,metrics)
    names(metrics) <- colname

    metrics <- t(metrics)
    scoreboard <- rbind(scoreboard, metrics)
  }
  return(scoreboard)
}

# helper function for roc
roc_build <- function(model) {
  THE_ROC <- roc(response = model$pred$obs,
                 predictor = model$pred$Bad,
                 levels = rev(levels(model$pred$obs)))
  return(THE_ROC)
}
```

Discriminant Classification Models

LDA

```r
set.seed(100)
lda_model <- caret::train(x=heloc_train_x,
                          y=heloc_train_y,
                          method="lda",
                          metric="ROC",
                          trControl=control)

lda_modelRoc <- roc_build(lda_model)
```

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

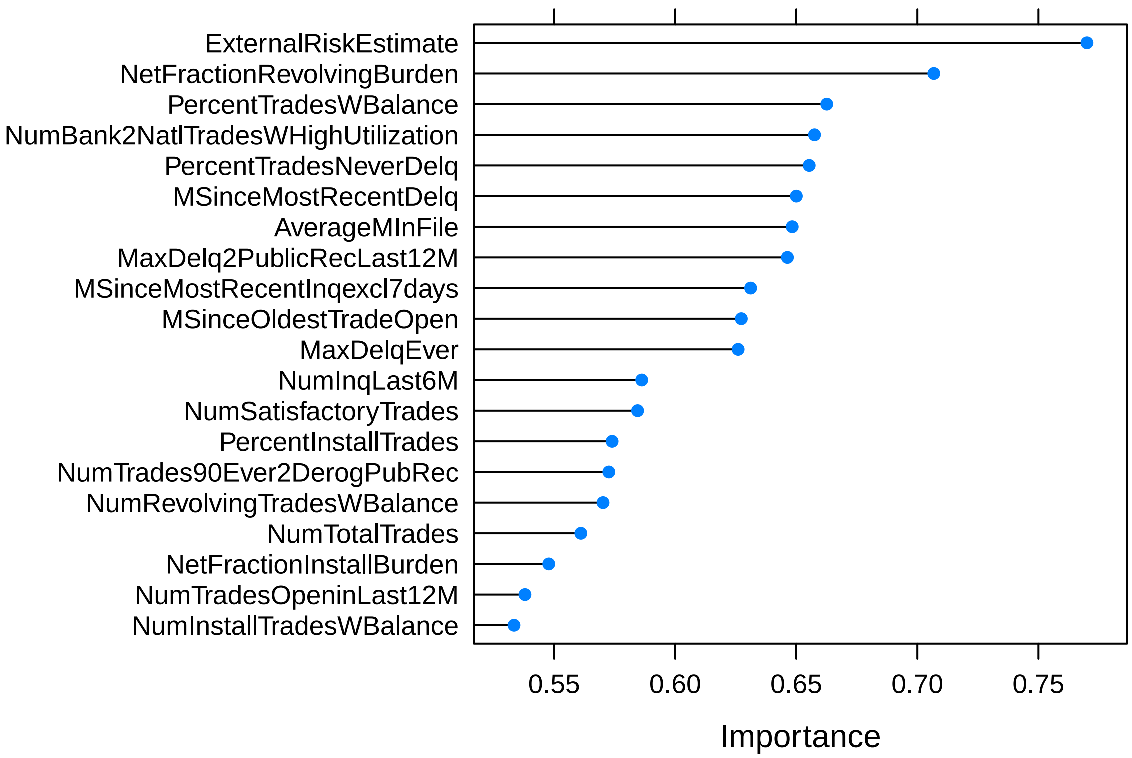
```r
lda_model
```

```
## Linear Discriminant Analysis
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7998689  0.7527435  0.7114538
```

```r
lda_predictions <- stats::predict(lda_model, heloc_test_x)

# create dataframe to store
testResults <- data.frame(obs=heloc_test_y,
                          lda_model=lda_predictions)

# confusion matrix
confusionMatrix(testResults$lda_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  768  288
##       Good 259  659
##
##                Accuracy : 0.7229
##                  95% CI : (0.7026, 0.7426)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4442
##
##  Mcnemar's Test P-Value : 0.2312
##
##             Sensitivity : 0.6959
##             Specificity : 0.7478
##          Pos Pred Value : 0.7179
##          Neg Pred Value : 0.7273
##              Prevalence : 0.4797
##          Detection Rate : 0.3338
##    Detection Prevalence : 0.4650
##       Balanced Accuracy : 0.7218
##
##        'Positive' Class : Good
##
```

```r
lda_varImp <- caret::varImp(lda_model, scale=FALSE)
lda_varImpRanks <- importanceRanker(lda_varImp$importance, 'lda')
plot(lda_varImp, top=20)
```



Logistic Regression

```
set.seed(100)
logreg_model <- caret::train(x=heloc_train_x,
                             y=heloc_train_y,
                             method="glm",
                             metric="ROC",
                             trControl=control)

testResults$log_reg_model <- stats::predict(logreg_model, heloc_test_x)
logreg_modelRoc <- roc_build(logreg_model)
```

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

```
logreg_model
```

```
## Generalized Linear Model
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8000842  0.7554216  0.7130383
```

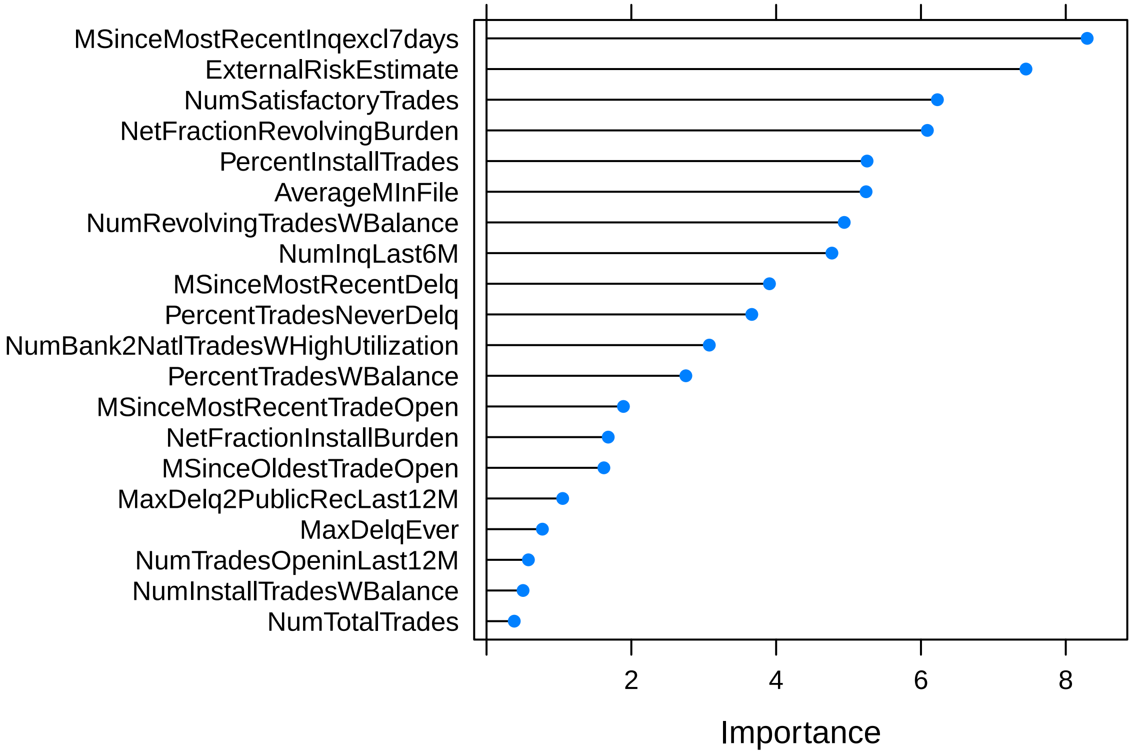
```r
logreg_model$finalModel$coefficients
```

```
##                       (Intercept)                     AverageMInFile
##                     -0.1803777116                       0.2470129097
##              ExternalRiskEstimate           MaxDelq2PublicRecLast12M
##                      0.4763324385                       0.0454362796
##                       MaxDelqEver              MSinceMostRecentDelq
##                      0.0320452303                       0.1630765084
##      MSinceMostRecentInqexcl7days          MSinceMostRecentTradeOpen
##                      0.2726958674                      -0.0650335190
##              MSinceOldestTradeOpen          NetFractionInstallBurden
##                      0.0649987498                      -0.0543543815
##        NetFractionRevolvingBurden NumBank2NatlTradesWHighUtilization
##                     -0.2929249411                      -0.1376246387
##                       NumInqLast6M           NumInstallTradesWBalance
##                     -0.1638154810                       0.0167061783
##        NumRevolvingTradesWBalance             NumSatisfactoryTrades
##                     -0.2278121804                       0.3689635005
##                     NumTotalTrades         NumTrades90Ever2DerogPubRec
##                      0.0194119014                       0.0001783062
##             NumTradesOpeninLast12M              PercentInstallTrades
##                     -0.0196622137                      -0.1878716750
##             PercentTradesNeverDelq              PercentTradesWBalance
##                      0.1716749452                       0.1173618900
```

```r
confusionMatrix(testResults$log_reg_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  766  288
##       Good 261  659
##
##                Accuracy : 0.7219
##                  95% CI : (0.7015, 0.7416)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4422
##
##  Mcnemar's Test P-Value : 0.2671
##
##             Sensitivity : 0.6959
##             Specificity : 0.7459
##          Pos Pred Value : 0.7163
##          Neg Pred Value : 0.7268
##              Prevalence : 0.4797
##          Detection Rate : 0.3338
##    Detection Prevalence : 0.4661
##       Balanced Accuracy : 0.7209
##
##        'Positive' Class : Good
##
```

```r
lr_varImp <- caret::varImp(logreg_model, scale=FALSE)
lr_varImpRanks <- importanceRanker(lr_varImp$importance, 'lr')
plot(lr_varImp, top=20)
```




```r
#get raw probs from model
predictions <- predict(logreg_model, heloc_test_x, type = 'prob')
predictions$OBS <- as.factor(heloc_test$RiskPerformance)
predictions <- predictions %>%
  mutate(lr10 = as.factor(if_else(Bad > 0.1, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr20 = as.factor(if_else(Bad > 0.2, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr30 = as.factor(if_else(Bad > 0.3, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr40 = as.factor(if_else(Bad > 0.4, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr50 = as.factor(if_else(Bad > 0.5, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr60 = as.factor(if_else(Bad > 0.6, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr70 = as.factor(if_else(Bad > 0.7, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr80 = as.factor(if_else(Bad > 0.8, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr90 = as.factor(if_else(Bad > 0.9, 'Bad', 'Good')))

# cf function
cost_confusionMatrix <- function(prediction.rate){
```

```
  cm <- caret::confusionMatrix(prediction.rate,
                               predictions$OBS,
                               positive = "Bad")
  return(cm)
}

CF10 <- cost_confusionMatrix(predictions$lr10)
CF20 <- cost_confusionMatrix(predictions$lr20)
CF30 <- cost_confusionMatrix(predictions$lr30)
CF40 <- cost_confusionMatrix(predictions$lr40)
CF50 <- cost_confusionMatrix(predictions$lr50)
CF60 <- cost_confusionMatrix(predictions$lr60)
CF70 <- cost_confusionMatrix(predictions$lr70)
CF80 <- cost_confusionMatrix(predictions$lr80)
CF90 <- cost_confusionMatrix(predictions$lr90)

Costs = matrix(c(0,-1000*.85,-60,60), ncol=2, nrow=2)

Prev = matrix(c(9.6/50,9.6/50,2,2), ncol=2, nrow=2)

CF10$table
```

Cost Matrix Threshold Analysis

```
##             Reference
## Prediction   Bad Good
##        Bad  1023  895
##        Good    4   52
```

```
sum(CF10$table*Costs*Prev)
```

```
## [1] -101812.8
```

```
CF20$table
```

```
##             Reference
## Prediction Bad Good
##        Bad  993  730
##        Good  34  217
```

```
sum(CF20$table*Costs*Prev)
```

```
## [1] -67108.8
```

```
CF30$table
```

```
##             Reference
## Prediction Bad Good
##        Bad  934  531
##        Good  93  416
```

```
sum(CF30$table*Costs*Prev)
```

```
## [1] -28977.6
```

```
CF40$table
```

```
##             Reference
## Prediction Bad Good
```

```
##        Bad  859  378
##        Good 168  569
```

```
sum(CF40$table*Costs*Prev)
```

```
## [1] -4497.6
```

```
CF50$table
```

```
##           Reference
## Prediction Bad Good
##        Bad  766  288
##        Good 261  659
```

```
sum(CF50$table*Costs*Prev)
```

```
## [1] 1924.8
```

```
CF60$table
```

```
##           Reference
## Prediction Bad Good
##        Bad  652  180
##        Good 375  767
```

```
sum(CF60$table*Costs*Prev)
```

```
## [1] 9240
```

```
CF70$table
```

```
##           Reference
## Prediction Bad Good
##        Bad  510  103
##        Good 517  844
```

```
sum(CF70$table*Costs*Prev)
```

```
## [1] 4545.6
```

```
CF80$table
```

```
##           Reference
## Prediction Bad Good
##        Bad  318   36
##        Good 709  911
```

```
sum(CF80$table*Costs*Prev)
```

```
## [1] -10708.8
```

```
CF90$table
```

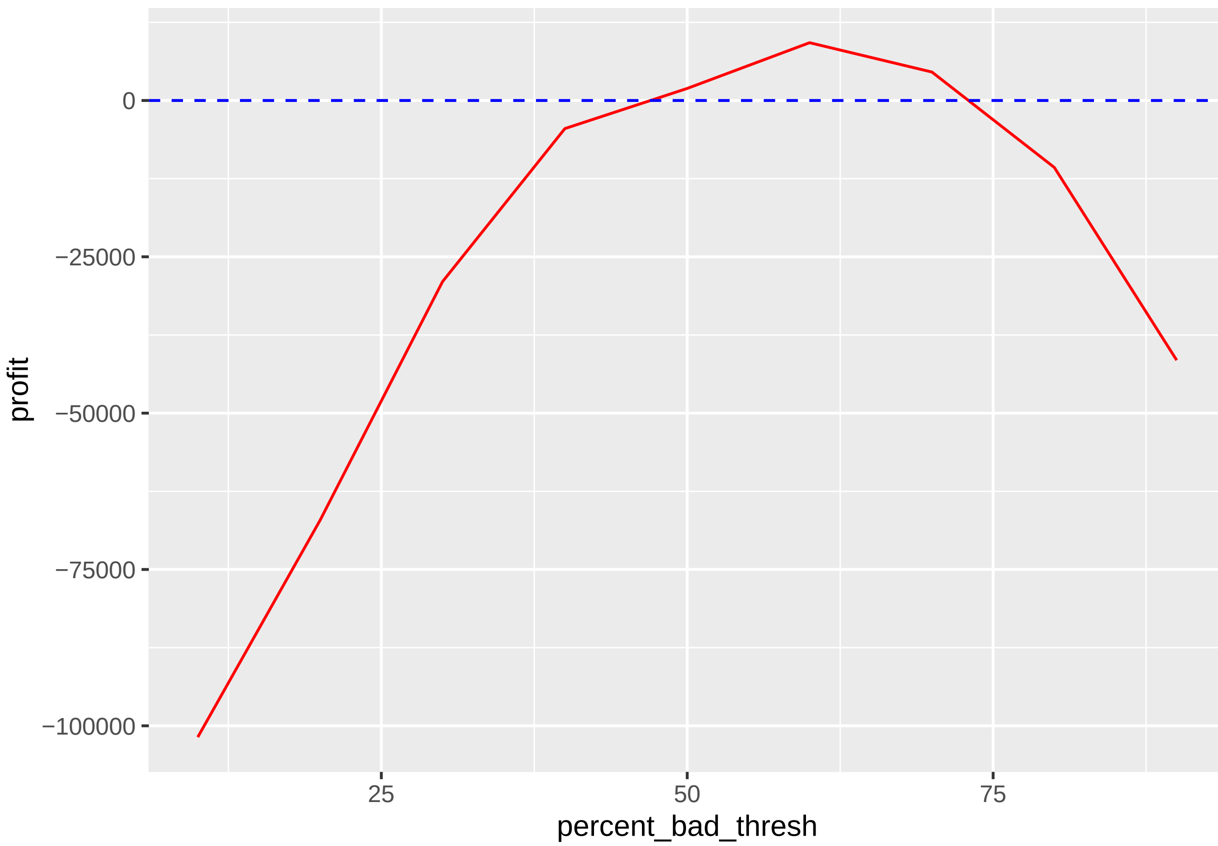
```
##           Reference
## Prediction Bad Good
##        Bad   85    6
##        Good 942  941
```

```
sum(CF90$table*Costs*Prev)
```

```
## [1] -41534.4
```

```
PlotProf<-data.frame(percent_bad_thresh = c(10,20,30,40,50,60,70,80,90),
                     profit = c(sum(CF10$table*Costs*Prev),
```

```
                                 sum(CF20$table*Costs*Prev),
                                 sum(CF30$table*Costs*Prev),
                                 sum(CF40$table*Costs*Prev),
                                 sum(CF50$table*Costs*Prev),
                                 sum(CF60$table*Costs*Prev),
                                 sum(CF70$table*Costs*Prev),
                                 sum(CF80$table*Costs*Prev),
                                 sum(CF90$table*Costs*Prev)))

ggplot(PlotProf, aes(y=profit, x=percent_bad_thresh)) +
  geom_line(colour = 'red') +
  geom_hline(yintercept=0, linetype='dashed', color='blue')
```



Penalized Logistic Regression

```
set.seed(100)
glmnGrid <- expand.grid(alpha=c(0, 0.1, 0.2, 0.4, 0.6, 0.8, 1),
                        lambda=seq(0.01, 0.2, length=5))

logreg_penalized_model <- caret::train(x=heloc_train_x,
                                       y=heloc_train_y,
                                       method="glmnet",
                                       metric="ROC",
                                       tuneGrid=glmnGrid,
                                       trControl=control)
```

```r
# bestIndex(logreg_penalized_model)
logreg_penalized_model$results[3:7,1:5]
```

```
##   alpha lambda       ROC      Sens      Spec
## 3   0.0 0.1050 0.7987157 0.7583402 0.7043291
## 4   0.0 0.1525 0.7981792 0.7607732 0.7024836
## 5   0.0 0.2000 0.7976588 0.7619904 0.7011636
## 6   0.1 0.0100 0.8001371 0.7551771 0.7114524
## 7   0.1 0.0575 0.7992612 0.7551777 0.7043277
```

```r
logreg_penalized_modelRoc <- roc_build(logreg_penalized_model)
```

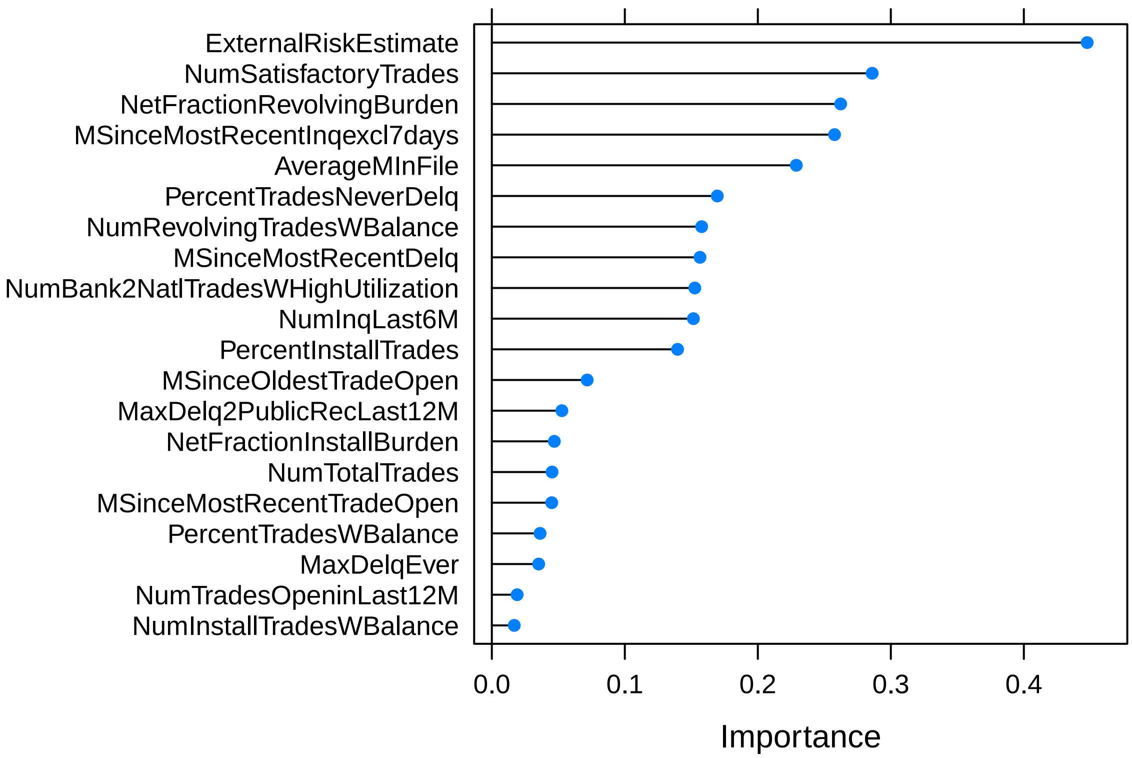
```
## Setting direction: controls < cases
```

Based on the best ROC, has the best specificity, our main metric. This enables us to insure that we only accept best qualified candidates, thus reducing the risk of a loanee defaulting on a $100,000 loan.

```r
# Utilizing Best Model
testResults$logreg_penalized_model <- stats::predict(logreg_penalized_model,
                                                     heloc_test_x)
# confusion matrix
confusionMatrix(testResults$logreg_penalized_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  773  282
##       Good 254  665
##
##                Accuracy : 0.7285
##                  95% CI : (0.7083, 0.748)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4554
##
##  Mcnemar's Test P-Value : 0.2435
##
##             Sensitivity : 0.7022
##             Specificity : 0.7527
##          Pos Pred Value : 0.7236
##          Neg Pred Value : 0.7327
##              Prevalence : 0.4797
##          Detection Rate : 0.3369
##    Detection Prevalence : 0.4656
##       Balanced Accuracy : 0.7274
##
##        'Positive' Class : Good
##
```

```r
lr_penalized_varImp <- caret::varImp(logreg_penalized_model, scale=FALSE)
lr_pen_varImpRanks <- importanceRanker(lr_penalized_varImp$importance, 'lr_penalized')
plot(lr_penalized_varImp, top=20)
```



Nonlinear Classification Models

Flexibble Discriminant Analysis

```
# set.seed(100)
# fdaGrid <- expand.grid(degree=c(1,2),
#                        nprune=seq(14, 20, 1))
#
# set.seed(100)
# fdaModel <- caret::train(x = heloc_train_x,
#                          y = heloc_train_y,
#                          method = "fda",
#                          metric = "ROC",
#                          trControl=control,
#                          tuneGrid=fdaGrid)
# # bestIndex(fdaModel)
# fdaModel$results[1:18,1:5]
```

As we see, the best is that of nprune 16 with a specificity of 72.15% (index = 15).

```
fdaGrid <- expand.grid(degree=1,
                       nprune=16)

set.seed(100)
fdaModel <- caret::train(x = heloc_train_x,
                         y = heloc_train_y,
                         method = "fda",
                         metric = "ROC",
                         trControl=control,
                         tuneGrid=fdaGrid)
```

```
## Loading required package: earth
## Loading required package: Formula
## Loading required package: plotmo
## Loading required package: plotrix
## Loading required package: TeachingDemos
```

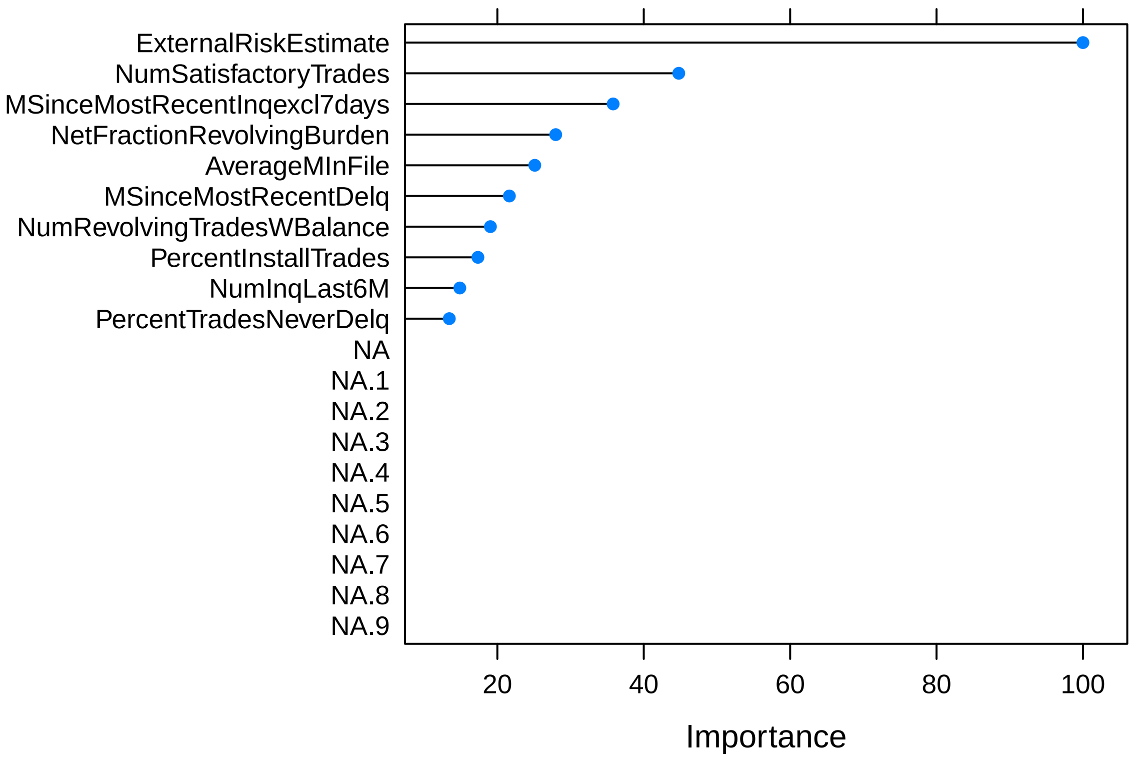
```
fda_modelRoc <- roc_build(fdaModel)
```

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

```
testResults$fda_model <- predict(fdaModel,
                                 heloc_test_x)
# confusion matrix
confusionMatrix(testResults$fda_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  762  265
##       Good 265  682
##
##                Accuracy : 0.7315
##                  95% CI : (0.7114, 0.751)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4621
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7202
##             Specificity : 0.7420
##          Pos Pred Value : 0.7202
##          Neg Pred Value : 0.7420
##              Prevalence : 0.4797
##          Detection Rate : 0.3455
##    Detection Prevalence : 0.4797
##       Balanced Accuracy : 0.7311
##
##        'Positive' Class : Good
##
```

```
fda_varImp <- caret::varImp(fdaModel, scale=FALSE)
fda_varImpRanks <- importanceRanker(fda_varImp$importance, 'fda_penalized')
plot(fda_varImp, top=20)
```



Neural Network

```
# set.seed(100)
# nnetGrid <- expand.grid(size = 1:2,
#                         decay = c(0, 0.1, 0.25, 0.5, 0.75, 1))
#
# nnetModel <- caret::train(x = heloc_train_x,
#                           y = heloc_train_y,
#                           method = "nnet",
#                           tuneGrid = nnetGrid,
#                           metric = "ROC",
#                           trace = FALSE,
#                           maxit = 2000,
#                           trControl = control)
# nnetModel$bestTune$results[1:6,1:5]
```

It appears that the model starts to over fit once the decay goes to 0.1. The nnet model chose size=2 with decay of 2, with nearly identical ROC, Sensitivity, and Specificity scores and thus size 1 with decay 0 is our choice.

```
set.seed(100)
nnetGrid <- expand.grid(size = 1,
                        decay = 0)

nnetModel <- caret::train(x = heloc_train_x,
                   y = heloc_train_y,
                   method = "nnet",
                   tuneGrid = nnetGrid,
                   metric = "ROC",
                   trace = FALSE,
```

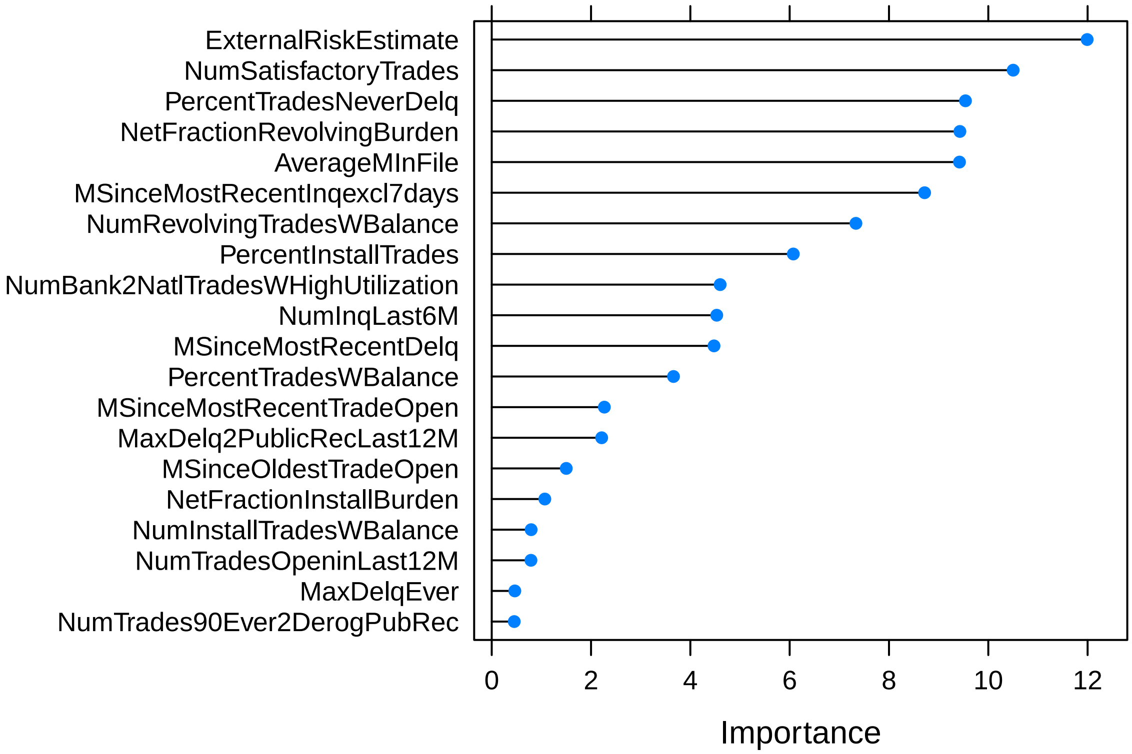
```r
                   maxit = 2000,
                   trControl = control)

nnet_modelRoc <- roc_build(nnetModel)
```

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

```r
testResults$nnet_model <- predict(nnetModel,
                                  heloc_test_x)
# confusion matrix
confusionMatrix(testResults$nnet_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  769  291
##       Good 258  656
##
##                Accuracy : 0.7219
##                  95% CI : (0.7015, 0.7416)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4421
##
##  Mcnemar's Test P-Value : 0.172
##
##             Sensitivity : 0.6927
##             Specificity : 0.7488
##          Pos Pred Value : 0.7177
##          Neg Pred Value : 0.7255
##              Prevalence : 0.4797
##          Detection Rate : 0.3323
##    Detection Prevalence : 0.4630
##       Balanced Accuracy : 0.7207
##
##        'Positive' Class : Good
##
```

```r
nn_varImp <- caret::varImp(nnetModel, scale=FALSE)
nn_varImpRanks <- importanceRanker(nn_varImp$importance, 'nn')
plot(nn_varImp, top=20)
```



Classification Trees

Boosted Tree

```
# gbmGrid <- expand.grid(interaction.depth =  c(2,3),
#                        n.trees = c(1000,2000,3000,4000), #default val = 1000
#                        shrinkage = c(0.01, 0.1),
#                        n.minobsinnode = c(5,10)) # default val = 10
# set.seed(100)
# gbmModel <- caret::train(x = heloc_train_x,
#                          y = heloc_train_y,
#                          method = "gbm",
#                          tuneGrid = gbmGrid,
#                          verbose = FALSE,
#                          metric = "ROC",
#                          trControl= control)
# gbmModel$results
```

```
gbmGrid <- expand.grid(interaction.depth = 2,
                       n.trees = 1000,
                       shrinkage = 0.01,
                       n.minobsinnode = 5)
set.seed(100)
gbmModel <- caret::train(x = heloc_train_x,
                         y = heloc_train_y,
                         method = "gbm",
                         tuneGrid = gbmGrid,
                         verbose = FALSE,
                         metric = "ROC",
                         trControl= control)
```

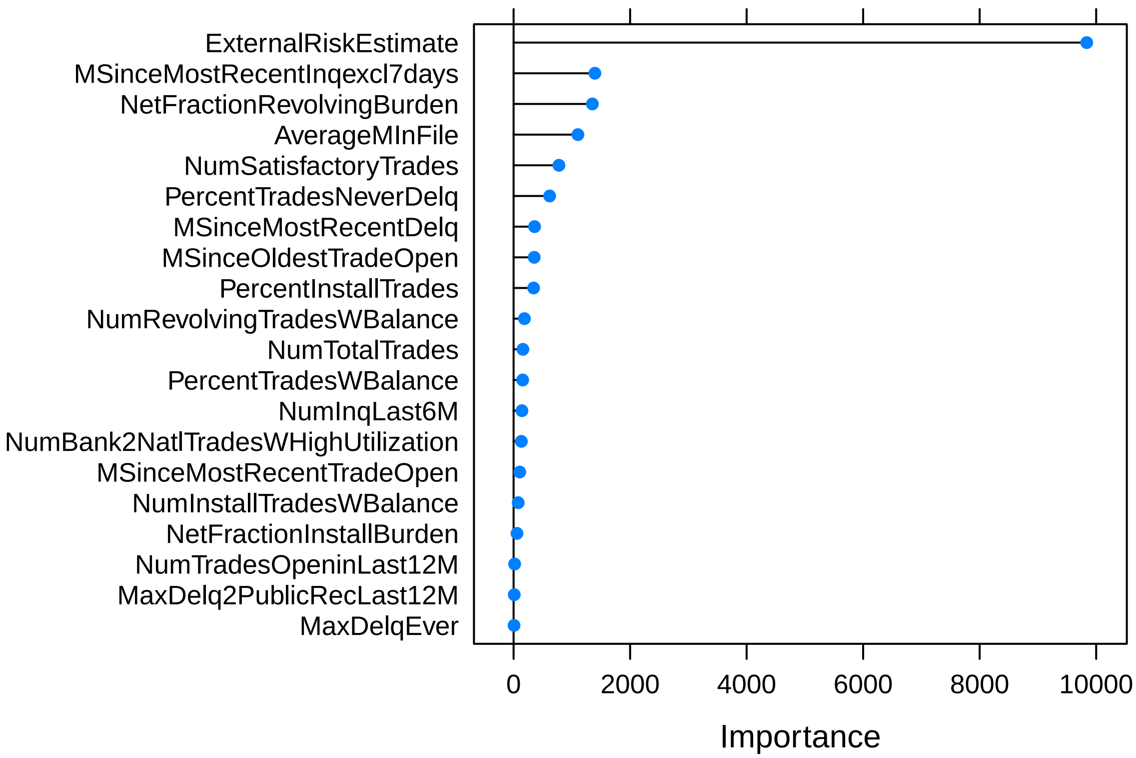
```r
gbm_modelRoc <- roc_build(gbmModel)
```

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

```r
testResults$gbm_model <- predict(gbmModel,
                                 heloc_test_x)
# confusion matrix
confusionMatrix(testResults$gbm_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  775  285
##       Good 252  662
##
##                Accuracy : 0.728
##                  95% CI : (0.7078, 0.7475)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4543
##
##  Mcnemar's Test P-Value : 0.1673
##
##             Sensitivity : 0.6990
##             Specificity : 0.7546
##          Pos Pred Value : 0.7243
##          Neg Pred Value : 0.7311
##              Prevalence : 0.4797
##          Detection Rate : 0.3354
##    Detection Prevalence : 0.4630
##       Balanced Accuracy : 0.7268
##
##        'Positive' Class : Good
##
```

```r
gbm_varImp <- caret::varImp(gbmModel, scale=FALSE)
gbm_varImpRanks <- importanceRanker(gbm_varImp$importance, 'gbm')
plot(gbm_varImp, top=20)
```




```
# gbmGrid <- expand.grid(interaction.depth = c(2,3),
#                          n.trees = c(1000,2000),
#                          shrinkage = c(0.01, 0.1),
#                          n.minobsinnode = c(5,10))
# set.seed(100)
# gbmMono_model <- caret::train(x = heloc_train_x,
#                                y = heloc_train_y,
#                                method = "gbm",
#                                var.monotone = c(-1,-1,-1,-1,-1,-1,-1,-1,
#                                                 1,1,1,1,0,0,-1,0,-1,1,
#                                                 0,-1,1),
#                                tuneGrid = gbmGrid,
#                                verbose = FALSE,
#                                metric = "ROC",
#                                trControl= control)
# gbmMono_model$results
```

```
gbmGrid <- expand.grid(interaction.depth = 2,
                       n.trees = 1000,
                       shrinkage = 0.01,
                       n.minobsinnode = 5)
set.seed(100)
gbmMono_model <- caret::train(x = heloc_train_x,
                              y = heloc_train_y,
                              var.monotone = c(-1,-1,-1,-1,-1,-1,-1,-1,
                                               1,1,1,1,0,0,-1,0,-1,1,
                                               0,-1,1),
```

```
                             method = "gbm",
                             tuneGrid = gbmGrid,
                             verbose = FALSE,
                             metric = "ROC",
                             trControl= control)

gbmMono_modelRoc <- roc_build(gbmMono_model)
```

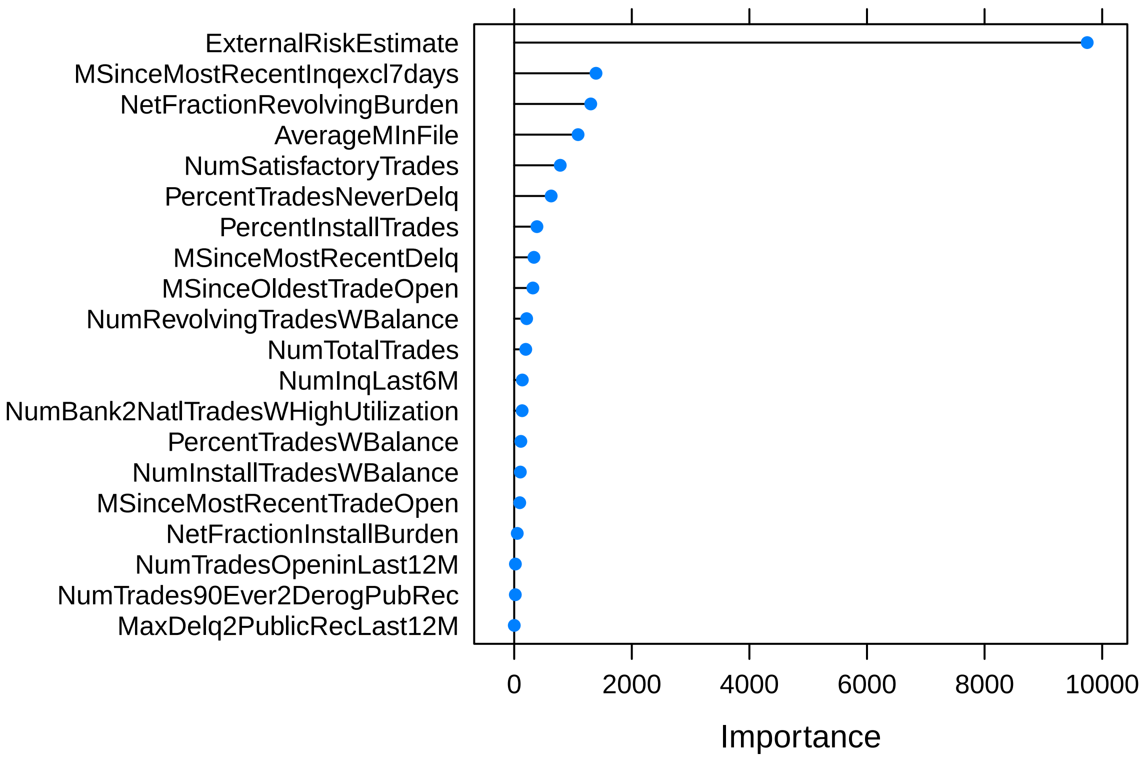
GBM with monotonic constraints

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

```
testResults$gbmMono_model <- predict(gbmMono_model,
                                     heloc_test_x)
confusionMatrix(testResults$gbmMono_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  772  281
##       Good 255  666
##
##                Accuracy : 0.7285
##                  95% CI : (0.7083, 0.748)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4555
##
##  Mcnemar's Test P-Value : 0.2802
##
##             Sensitivity : 0.7033
##             Specificity : 0.7517
##          Pos Pred Value : 0.7231
##          Neg Pred Value : 0.7331
##              Prevalence : 0.4797
##          Detection Rate : 0.3374
##    Detection Prevalence : 0.4666
##       Balanced Accuracy : 0.7275
##
##        'Positive' Class : Good
##
```

```
gmbMono_varImp <- caret::varImp(gbmMono_model, scale=FALSE)
gbmMono_varImpRanks <- importanceRanker(gmbMono_varImp$importance, 'gbmMono')
plot(gmbMono_varImp, top=20)
```



CART

```
# set.seed(100)
# rpart_grid <- expand.grid(cp=c(0.0005, 0.001250, 0.0015, 0.00175, 0.002))
#
# rpart_model <- caret::train(x=heloc_train_x,
#                              y=heloc_train_y,
#                              method="rpart",
#                              metric="ROC",
#                              trControl=control,
#                              tuneGrid = rpart_grid)
#
# testResults$rpart_model <- predict(rpart_model, heloc_test_x)
#
# rpart_model
```

As we see, specificity for this model is one of the worst out of all the models we have.

```
set.seed(100)
rpart_grid <- expand.grid(cp=0.00175)

rpart_model <- caret::train(x=heloc_train_x,
                            y=heloc_train_y,
                            method="rpart",
                            metric="ROC",
                            trControl=control,
                            tuneGrid = rpart_grid)

rpart_modelRoc <- roc_build(rpart_model)
```

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

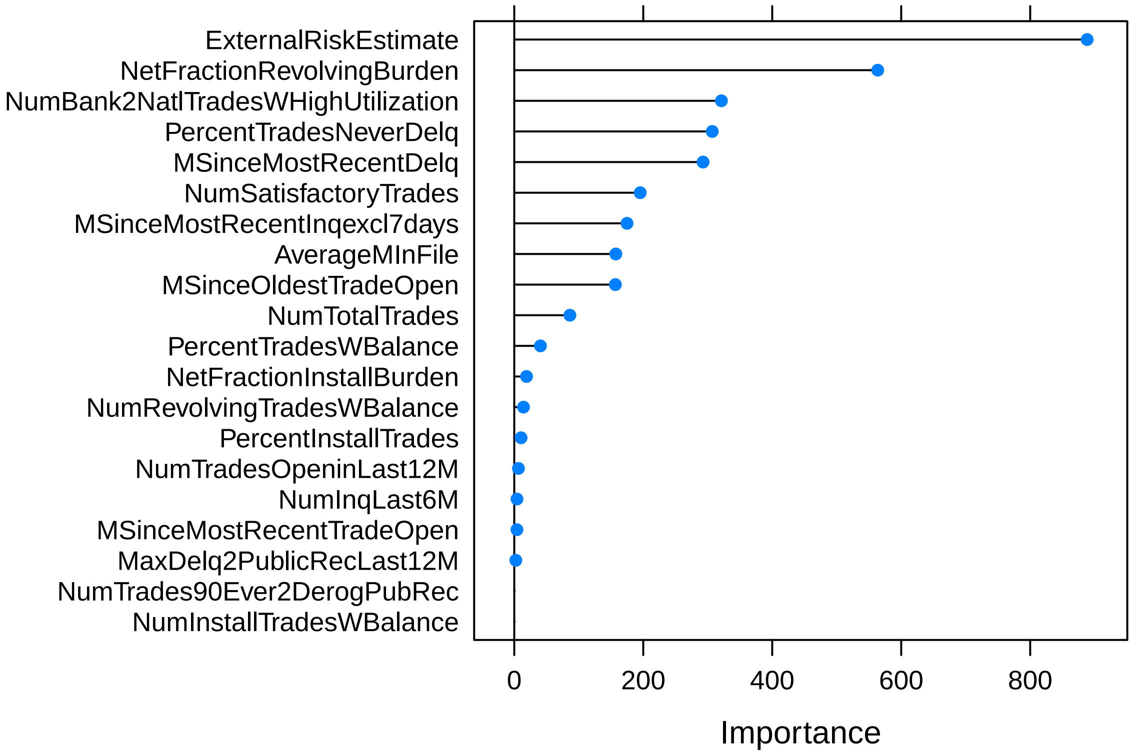
```r
rpart_model
```

```
## CART
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7558931  0.7551742  0.671846
##
## Tuning parameter 'cp' was held constant at a value of 0.00175
```

```r
testResults$rpart_model <- predict(rpart_model, heloc_test_x)

confusionMatrix(testResults$rpart_model)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Bad Good
##       Bad  748  282
##       Good 279  665
##
##                Accuracy : 0.7158
##                  95% CI : (0.6953, 0.7356)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4306
##
##  Mcnemar's Test P-Value : 0.9327
##
##             Sensitivity : 0.7022
##             Specificity : 0.7283
##          Pos Pred Value : 0.7044
##          Neg Pred Value : 0.7262
##              Prevalence : 0.4797
##          Detection Rate : 0.3369
##    Detection Prevalence : 0.4782
##       Balanced Accuracy : 0.7153
##
##        'Positive' Class : Good
##
```

```r
rpart_varImp <- caret::varImp(rpart_model, scale=FALSE)
rpart_varImpRanks <- importanceRanker(rpart_varImp$importance, 'RPart')
plot(rpart_varImp, top=20)
```



Random Forest

```
# set.seed(100)
# rf_grid <- expand.grid(mtry=c(5,10,15))
#
# randomForest_model <- caret::train(x=heloc_train_x,
#                                   y=heloc_train_y,
#                                   method="rf",
#                                   metric="ROC",
#                                   trControl=control,
#                                   tuneGrid = rf_grid)
#
# randomForest_model
```

```
set.seed(100)
rf_grid <- expand.grid(mtry=5)

randomForest_model <- caret::train(x=heloc_train_x,
                                   y=heloc_train_y,
                                   method="rf",
                                   metric="ROC",
                                   trControl=control,
                                   tuneGrid = rf_grid)

randomForest_modelRoc <- roc_build(randomForest_model)
```

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

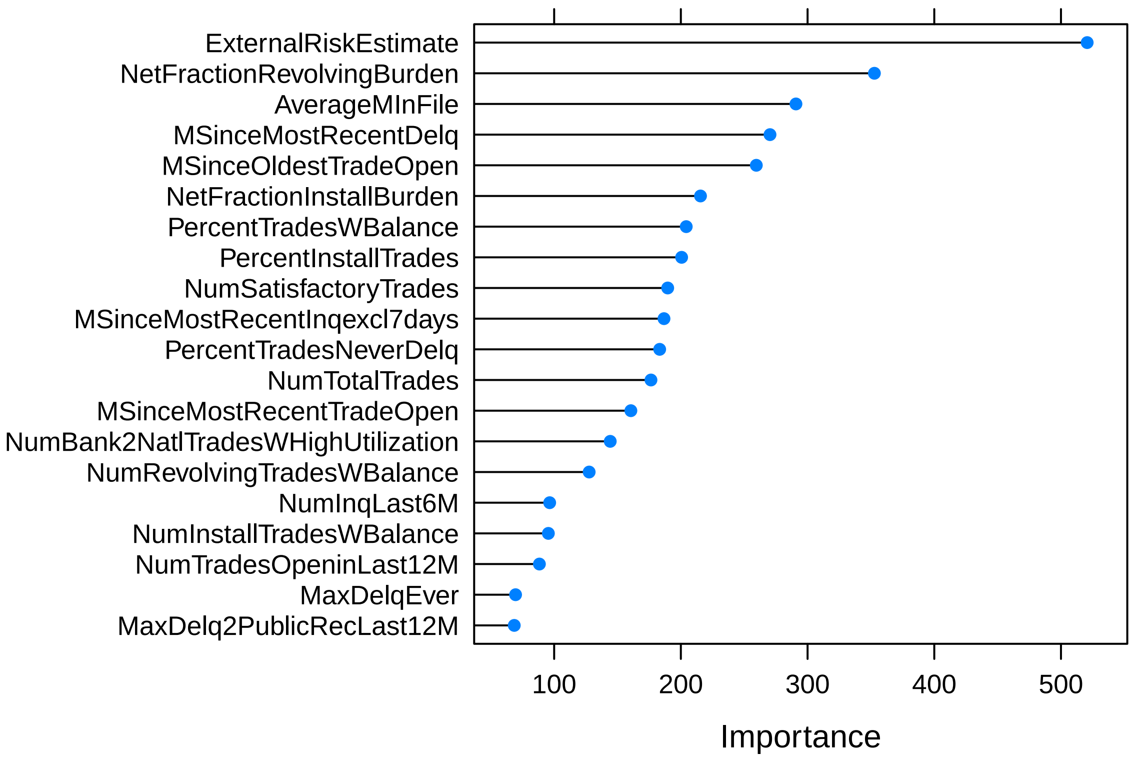
```
randomForest_model
```

```
## Random Forest
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7951729  0.7736746  0.6847978
##
## Tuning parameter 'mtry' was held constant at a value of 5
```

```
testResults$randomForest_model <- predict(randomForest_model, heloc_test_x)

confusionMatrix(testResults$randomForest_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  790  291
##       Good 237  656
##
##                Accuracy : 0.7325
##                  95% CI : (0.7124, 0.7519)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.463
##
##  Mcnemar's Test P-Value : 0.02108
##
##             Sensitivity : 0.6927
##             Specificity : 0.7692
##          Pos Pred Value : 0.7346
##          Neg Pred Value : 0.7308
##              Prevalence : 0.4797
##          Detection Rate : 0.3323
##    Detection Prevalence : 0.4524
##       Balanced Accuracy : 0.7310
##
##        'Positive' Class : Good
##
```

```
rf_varImp <- caret::varImp(randomForest_model, scale=FALSE)
rf_varImpRanks <- importanceRanker(rf_varImp$importance, 'RF')
plot(rf_varImp, top=20)
```



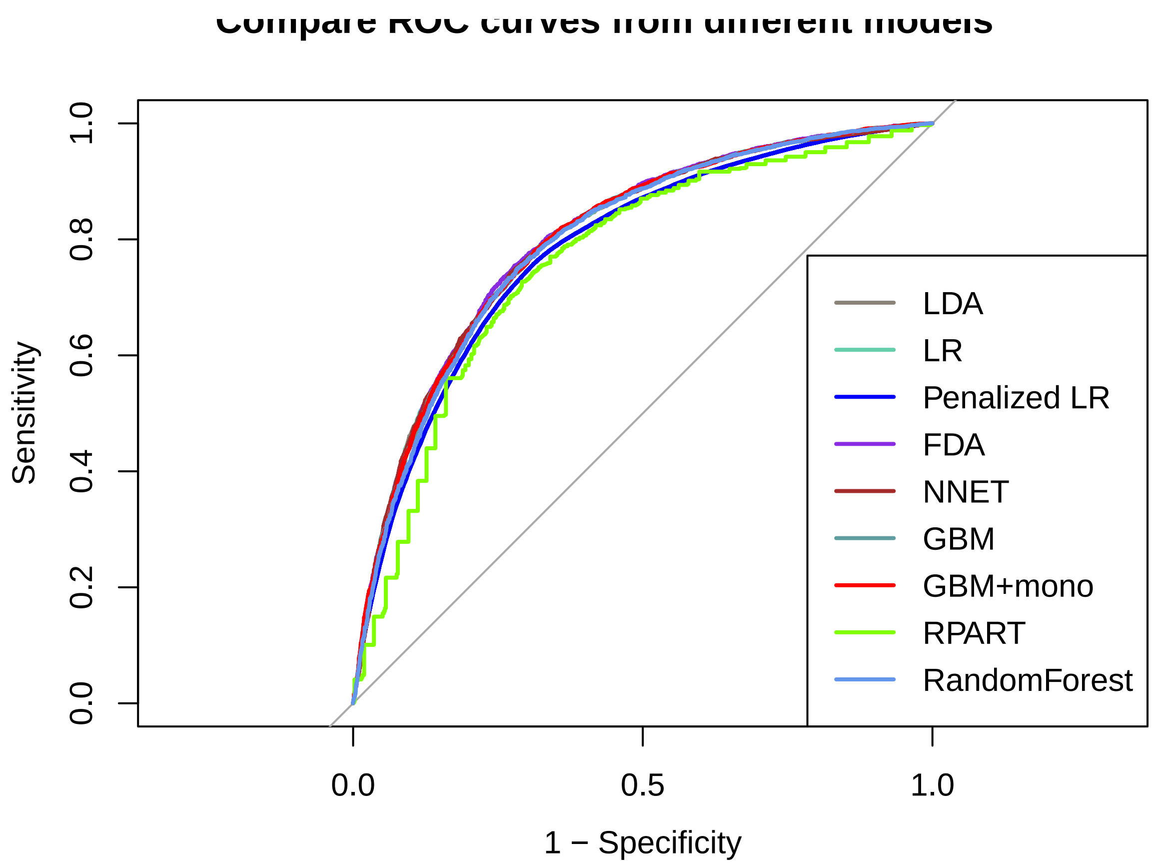
Results & Discussion

```
varImpRanks_df <- plyr::rbind.fill.matrix(fda_varImpRanks,gbm_varImpRanks)
# rbind fill removes row names
rownames(varImpRanks_df) <- c('fda','gbm')
# base::rbind preserves row names
varImpRanks_df <- rbind(varImpRanks_df, gbmMono_varImpRanks)
varImpRanks_df <- rbind(varImpRanks_df, lda_varImpRanks)
varImpRanks_df <- rbind(varImpRanks_df, lr_varImpRanks)
varImpRanks_df <- rbind(varImpRanks_df, lr_pen_varImpRanks)
varImpRanks_df <- rbind(varImpRanks_df, nn_varImpRanks)
varImpRanks_df <- rbind(varImpRanks_df, rpart_varImpRanks)
varImpRanks_df <- rbind(varImpRanks_df, rf_varImpRanks)
varImpRanks_df <- data.frame(varImpRanks_df)
# num of columns is hard to print, thus transpose for printing purposes
varImpRanks_df <- t(varImpRanks_df)
knitr::kable(varImpRanks_df) %>%
  kableExtra::kable_styling("striped", full_width = F) %>%
  kableExtra::row_spec(0, angle = -90)
```

```
plot(lda_modelRoc, type='s', col='antiquewhite4', legacy.axes=TRUE)
plot(logreg_modelRoc, type='s', col='aquamarine3', legacy.axes=TRUE, add=TRUE)
plot(logreg_penalized_modelRoc, type='s', col='blue', legacy.axes=TRUE, add=TRUE)
plot(fda_modelRoc, type='s', col='blueviolet', legacy.axes=TRUE, add=TRUE)
plot(nnet_modelRoc, type='s', col='brown', legacy.axes=TRUE, add=TRUE)
plot(gbm_modelRoc, type='s', col='cadetblue', legacy.axes=TRUE, add=TRUE)
plot(gbmMono_modelRoc, type='s', col='red', legacy.axes=TRUE, add=TRUE)
plot(rpart_modelRoc, type='s', col='chartreuse', legacy.axes=TRUE, add=TRUE)
plot(randomForest_modelRoc, type='s', col='cornflowerblue', legacy.axes=TRUE, add=TRUE)
```

	fda	gbm	gbmMono	lda	lr	lr_penalized	nn	RPart	RF
ExternalRiskEstimate	1	1	4	7	6	5	5	8	3
NumSatisfactoryTrades	2	5	1	1	2	1	1	1	1
MSinceMostRecentInqexcl7days	3	2	20	8	16	13	14	18	20
NetFractionRevolvingBurden	4	3	21	11	17	18	19	5	19
AverageMInFile	5	4	8	6	9	8	11	7	4
MSinceMostRecentDelq	6	7	2	9	1	4	6	17	10
NumRevolvingTradesWBalance	7	10	16	21	13	16	13	9	13
PercentInstallTrades	8	9	9	10	15	12	15	12	5
NumInqLast6M	9	13	17	18	14	14	16	2	6
PercentTradesNeverDelq	10	6	3	2	4	3	4	3	2
MaxDelq2PublicRecLast12M	NA	19	13	4	11	9	9	16	14
MaxDelqEver	NA	20	12	12	8	10	10	13	16
MSinceMostRecentTradeOpen	NA	15	15	20	19	20	17	6	17
MSinceOldestTradeOpen	NA	8	10	16	7	7	7	10	15
NetFractionInstallBurden	NA	17	5	13	3	2	2	15	9
NumBank2NatlTradesWHighUtilization	NA	14	11	17	20	15	21	14	12
NumInstallTradesWBalance	NA	16	19	15	21	21	20	4	21
NumTotalTrades	NA	11	18	19	18	19	18	11	18
NumTrades90Ever2DerogPubRec	NA	21	7	14	5	11	8	20	8
NumTradesOpeninLast12M	NA	18	6	5	10	6	3	20	11
PercentTradesWBalance	NA	12	14	3	12	17	12	20	7

```r
legend_ <- c('LDA', 'LR', 'Penalized LR','FDA', 'NNET','GBM','GBM+mono', 'RPART','RandomForest')
colors_ <-c('antiquewhite4',
            'aquamarine3',
            'blue',
            'blueviolet',
            'brown',
            'cadetblue',
            'red',
            'chartreuse',
            'cornflowerblue')
legend('bottomright', legend=legend_,
       col=colors_, lwd=2)
title(main = 'Compare ROC curves from different models', outer = TRUE)
```




```r
# gather all model AUCs in 1 list
aucs <- c(lda_modelRoc$auc,
          logreg_modelRoc$auc,
          logreg_penalized_modelRoc$auc,
          fda_modelRoc$auc,
          nnet_modelRoc$auc,
          gbm_modelRoc$auc,
          gbmMono_modelRoc$auc,
          rpart_modelRoc$auc,
          randomForest_modelRoc$auc)

scoreboard <- modelScoreBoard(testResults)
# add AUC list as a column
scoreboard$AUC <- aucs
scoreboard
```

```
##                          Accuracy Precision Sensitivity Specificity        F1
## lda_model               0.7228977 0.7178649   0.6958817   0.7478092 0.7067024
## log_reg_model           0.7218845 0.7163043   0.6958817   0.7458617 0.7059454
## logreg_penalized_model 0.7284701 0.7236126   0.7022175   0.7526777 0.7127546
## fda_model               0.7315096 0.7201690   0.7201690   0.7419669 0.7201690
## nnet_model              0.7218845 0.7177243   0.6927138   0.7487829 0.7049973
## gbm_model               0.7279635 0.7242888   0.6990496   0.7546251 0.7114455
## gbmMono_model           0.7284701 0.7231270   0.7032735   0.7517040 0.7130621
## rpart_model             0.7158055 0.7044492   0.7022175   0.7283350 0.7033316
## randomForest_model      0.7325228 0.7346025   0.6927138   0.7692308 0.7130435
##                              AUC
```

```
## lda_model              0.7998810
## log_reg_model          0.8002390
## logreg_penalized_model 0.7820838
## fda_model              0.8032575
## nnet_model             0.8001313
## gbm_model              0.7998948
## gbmMono_model          0.8003082
## rpart_model            0.7554376
## randomForest_model     0.7950744
```

REFERENCES:

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. New York: Springer.