Predicting Credit Risk for Loan Applicants

* [Quantitative Analysis Introduction](https://cojamalo.github.io/German_Credit_Analysis/report.html#quantitative-analysis-introduction)
  + [Analysis Scenario](https://cojamalo.github.io/German_Credit_Analysis/report.html#analysis-scenario)
  + [Data Source](https://cojamalo.github.io/German_Credit_Analysis/report.html#data-source)
* [Round One: Exploratory Data Analysis, Loss Function, and Initial Model Fit](https://cojamalo.github.io/German_Credit_Analysis/report.html#round-one-exploratory-data-analysis-loss-function-and-initial-model-fit)
  + [Load Packages](https://cojamalo.github.io/German_Credit_Analysis/report.html#load-packages)
  + [Load the Data](https://cojamalo.github.io/German_Credit_Analysis/report.html#load-the-data)
  + [Variable Classifications](https://cojamalo.github.io/German_Credit_Analysis/report.html#variable-classifications)
  + [Examing the Response Variable](https://cojamalo.github.io/German_Credit_Analysis/report.html#examing-the-response-variable)
  + [Binary Variables](https://cojamalo.github.io/German_Credit_Analysis/report.html#binary-variables)
  + [Non-binary Categorical Variables](https://cojamalo.github.io/German_Credit_Analysis/report.html#non-binary-categorical-variables)
  + [Continous Variables](https://cojamalo.github.io/German_Credit_Analysis/report.html#continous-variables)
  + [Loss Function](https://cojamalo.github.io/German_Credit_Analysis/report.html#loss-function)
  + [Exploring the Initial Model Space](https://cojamalo.github.io/German_Credit_Analysis/report.html#exploring-the-initial-model-space)
* [Round Two: Exploring Regularized Models](https://cojamalo.github.io/German_Credit_Analysis/report.html#round-two-exploring-regularized-models)
* [Round Three: Prediction Probability Cutoff Threshold Optimization](https://cojamalo.github.io/German_Credit_Analysis/report.html#round-three-prediction-probability-cutoff-threshold-optimization)
  + [Final Model Selection and Metrics](https://cojamalo.github.io/German_Credit_Analysis/report.html#final-model-selection-and-metrics)
* [Part Four: Identifying most relevant variables](https://cojamalo.github.io/German_Credit_Analysis/report.html#part-four-identifying-most-relevant-variables)
* [Conclusion](https://cojamalo.github.io/German_Credit_Analysis/report.html#conclusion)
* [Appendix A: Tuning Hyperparamaters](https://cojamalo.github.io/German_Credit_Analysis/report.html#appendix-a-tuning-hyperparamaters)

By: Shyamala

Quantitative Analysis Introduction

Analysis Scenario

A loan manager is requesting a statistical model to help her department determine which loan applicants are creditable, i.e. most likely to repay their loans. An applicant’s demographic and socio-economic profiles are considered by loan managers before a decision is made regarding his/her loan application. The manager’s goal is to minimize risk and maximize profits for the bank’s loan portfolio. The manager shares the information that for the type of loans the model would predict, if the borrower pays back the loan, the bank makes a profit of 35% the value of the loan. On the other hand, if the borrower defaults, the bank’s loss is 100%. The bank does not lose money for applicants who are rejected and the manager claims the model does not have to take into account the opportunity cost for applicants who would have repaid the loan but were rejected.

Upon receiving this request, I decided to develop a model for the manager that maximizes a profit-cost function given the provided data. The priority of the model fitting task will be prediction in this case as the manager has not specifically requested an interpretable model, but has requested a model with the best profit characteristics.

Data Source

The loan manager gives you access to a sample of her department’s loan data for 1000 applicants with the outcome of their loans included. She claims the dataset was prepared by another analyst with her input to be representative of the bank’s actual customers.

The data used in this project was originally provided by Dr. Hans Hofmann of the University of Hamburg and hosted by the UCI Machine Learning Repository. The specific version of the data used here (credit) was sourced from Penn State’s graduate-level Applied Data Mining and Statistical Learning 897D course.

Round One: Exploratory Data Analysis, Loss Function, and Initial Model Fit

To begin, the data must be loaded into the session and the variables checked to determine their appropriate types. Then, the variance of each predictor variable in relation to the response variable must be checked to ensure there are no zero variance predictors present. The profit/cost information must be programmed into a functional form that can be used to evaluate the fitted models. Finally, an initial set of models is fit to the data to determine how best to proceed.

Load Packages

**library**(data.table)

**library**(gmodels)

**library**(DT)

**library**(gridExtra)

**library**(pander)

**library**(stringr)

**library**(woe)

**library**(MASS)

**library**(randomForest)

**library**(xgboost)

**library**(caret)

**library**(tidyverse)

##Import Custom functions

**if**(!**exists**("createCVFolds", mode="function")) **source**("./R/createCVFolds.R")

**if**(!**exists**("get\_best\_cutoffs", mode="function")) **source**("./R/cutoff\_check.R")

Load the Data

Many of the variable names are invalid such as “Duration of Credit (month)”. The white spaces and invalid parenthesis characters will likely cause problems in any programmatic use of these predictors.

*# Load the data to a tibble dataframe using data.table::fread*

credit = **fread**("german\_credit.csv") %>%

tbl\_df

*# Convert all the variable names to snake\_case by making them all lower case, removing invalid characters, and replacing spaces with an underscore*

**names**(credit) = **names**(credit) %>%

tolower %>%

**str\_replace\_all**("[ /]", "\_") %>%

**str\_replace\_all**("[(&)]", "")

*# Show a preview*

credit %>% **datatable**(style="bootstrap")

Showing 1 to 10 of 1,000 entries

Variable Classifications

The first step in exploring the data is to determine what type of variables are present in the data. Are the variables categorical or quantitative? If they are categorical are they binary or do they have multiple levels?

*# Force all variables to factors and count the number of unique levels present.*

credit\_types = credit %>%

**mutate\_all**(factor) %>%

**map**(levels) %>%

**map**(length) %>%

tbl\_df %>%

**gather**(variable, n\_unique) %>%

**arrange**(n\_unique) %>%

**mutate**(binary = n\_unique==2, categorical = n\_unique <=10, continuous = n\_unique > 10)

*# Preview the data*

**datatable**(credit\_types, style="bootstrap")

Show  entries

Search:

|  | **variable** | **n\_unique** | **binary** | **categorical** | **continuous** |
| --- | --- | --- | --- | --- | --- |
| 1 | creditability | 2 | true | true | false |
| 2 | no\_of\_dependents | 2 | true | true | false |
| 3 | telephone | 2 | true | true | false |
| 4 | foreign\_worker | 2 | true | true | false |
| 5 | guarantors | 3 | false | true | false |
| 6 | concurrent\_credits | 3 | false | true | false |
| 7 | type\_of\_apartment | 3 | false | true | false |
| 8 | account\_balance | 4 | false | true | false |
| 9 | instalment\_per\_cent | 4 | false | true | false |
| 10 | sex\_\_marital\_status | 4 | false | true | false |

Showing 1 to 10 of 21 entries

* [Previous](https://cojamalo.github.io/German_Credit_Analysis/report.html)
* [1](https://cojamalo.github.io/German_Credit_Analysis/report.html)
* [2](https://cojamalo.github.io/German_Credit_Analysis/report.html)
* [3](https://cojamalo.github.io/German_Credit_Analysis/report.html)
* [Next](https://cojamalo.github.io/German_Credit_Analysis/report.html)

In this case, variables with ten or less unique levels are considered categorical variables. This assumption will not be true for every data set as some categorical factors will have more than ten levels. However, this cutoff is appropriate for this particular data set as the largest categorical variable has 10 levels.

Four binary categorical variables are present including the response variable creditability.

Fourteen non-binary categorical variables are present.

Three quantitative variables are present.

Now, each variable type can be explored.

Examing the Response Variable

First, the response variable is considered separately as the distribution of the response variable must be determined.

**CrossTable**(credit$creditability)

##

##

## Cell Contents

## |-------------------------|

## | N |

## | N / Table Total |

## |-------------------------|

##

##

## Total Observations in Table: 1000

##

##

## | 0 | 1 |

## |-----------|-----------|

## | 300 | 700 |

## | 0.300 | 0.700 |

## |-----------|-----------|

##

##

##

##

creditability has two levels with 700 observations in the positive class and 300 observations in the negative class. If the data is representative of the loan applicants to the bank, about 70% of applicants can repay their loans and 30% cannot pay. The imbalance in the classes is noted for modeling as more information is available to classify the positive class, which may allow some models to result in good accuracy when predicting the positive class, but poor accuracy when predicting the negative class while having good overall accuracy.

The current type of the data treats the classes as either a 1 or a 0. Let’s refactor this variable so it is more interpretable.

*# Assign the positive class 1, the label "Good" and the negative class 0, the label"Poor"*

credit$creditability = **ifelse**(credit$creditability == 1, "Good", "Poor")

credit$creditability = **factor**(credit$creditability, levels=**c**("Good", "Poor"))

**CrossTable**(credit$creditability)

##

##

## Cell Contents

## |-------------------------|

## | N |

## | N / Table Total |

## |-------------------------|

##

##

## Total Observations in Table: 1000

##

##

## | Good | Poor |

## |-----------|-----------|

## | 700 | 300 |

## | 0.700 | 0.300 |

## |-----------|-----------|

##

##

##

##

Binary Variables

Now, the remaining three binary variables are analyzed.

*# Identify the names of the binary, categorical variables*

binary\_names = credit\_types %>%

**filter**(binary) %>%

.$variable

*# Factor these variables*

credit = credit %>% **mutate\_at**(binary\_names, factor)

The most crucial information to determine at this stage in the analysis is whether any of the predictors have zero variance. A predictor has zero variance when all of its values are identical with respect to the response. For instance, if all observations of a predictor have only “Good” creditability. In such a case, the values of the predictor do not distinguish between the two classes in the response. Many models will fail to fit with zero variance predictors. In addition, if any individual category of a predictor is zero in respect to the response, then the model fitting will fail as many models separate categorical variables into separate predictors. If any predictor category has zero instances with respect to the response, the model will fail to fit.

Any predictor categories with zero instances with respect to the response?

**length**(**checkConditionalX**(credit[,binary\_names] %>% **select**(-creditability), credit$creditability)) > 0

## [1] FALSE

*# Determine is any predictors has zero variance*

nzv = **nearZeroVar**(credit[,binary\_names], names = TRUE, saveMetrics = TRUE) %>% **select**(-percentUnique)

indexes = **rownames**(nzv)

**pandoc.table**(**data.frame**(indexes = indexes, nzv) %>% tbl\_df %>% **arrange**(**desc**(freqRatio)))

| **indexes** | **freqRatio** | **zeroVar** | **nzv** |
| --- | --- | --- | --- |
| foreign\_worker | 26.03 | FALSE | TRUE |
| no\_of\_dependents | 5.452 | FALSE | FALSE |
| creditability | 2.333 | FALSE | FALSE |
| telephone | 1.475 | FALSE | FALSE |

None of these binary predictors has zero variance, but foreign\_worker has near zero variance with one class 26x more prevalent than the other in respect to creditability. This predictor will remain in the data, but this near zero variance could make it a likely candidate for removal later in the analysis if regularization is needed for model building.

Non-binary Categorical Variables

Next, the non-binary categorical variables are reviewed.

*# Identify the names of the non-binary, categorical variables*

nb\_names = credit\_types %>% **filter**(!binary,categorical) %>% .$variable

*# Factor these variables*

credit = credit %>% **mutate\_at**(nb\_names, factor)

These predictors are also checked for zero variance.

Any predictor categories with zero instances with respect to the response?

**length**(**checkConditionalX**(credit[,nb\_names], credit$creditability)) > 0

## [1] FALSE

*# Determine is any predictors has zero variance*

nzv = **nearZeroVar**(credit[,nb\_names], names = TRUE, saveMetrics = TRUE) %>% **select**(-percentUnique)

indexes = **rownames**(nzv)

**pandoc.table**(**data.frame**(indexes = indexes, nzv) %>% tbl\_df %>% **arrange**(**desc**(freqRatio)))

| **indexes** | **freqRatio** | **zeroVar** | **nzv** |
| --- | --- | --- | --- |
| guarantors | 17.44 | FALSE | FALSE |
| concurrent\_credits | 5.856 | FALSE | FALSE |
| type\_of\_apartment | 3.989 | FALSE | FALSE |
| value\_savings\_stocks | 3.295 | FALSE | FALSE |
| occupation | 3.15 | FALSE | FALSE |
| instalment\_per\_cent | 2.061 | FALSE | FALSE |
| no\_of\_credits\_at\_this\_bank | 1.901 | FALSE | FALSE |
| payment\_status\_of\_previous\_credit | 1.809 | FALSE | FALSE |
| sex\_\_marital\_status | 1.768 | FALSE | FALSE |
| account\_balance | 1.438 | FALSE | FALSE |
| duration\_in\_current\_address | 1.341 | FALSE | FALSE |
| length\_of\_current\_employment | 1.34 | FALSE | FALSE |
| purpose | 1.197 | FALSE | FALSE |
| most\_valuable\_available\_asset | 1.177 | FALSE | FALSE |

While guarantors has a high frequency ratio, none of these predictors is considered to have zero variance or near-zero variance.

Continous Variables

Finally, the continuous variables are summarized.

*# Identify the names of the non-binary, categorical variables*

quant\_names = credit\_types %>% **filter**(continuous) %>% .$variable

*# Helper function to summarize each predictor*

quant\_summary = **function**(data, vector\_name) {

data %>% **summarize\_at**(vector\_name, **funs**(MIN=min,Q1 = **quantile**(., 0.25), MEAN = mean, MEDIAN = median, Q3 = **quantile**(., 0.75), MAX = max, IQR = IQR, STDEV = sd)) %>%

**mutate**(SKEW = **ifelse**(MEAN > MEDIAN, "RIGHT", "LEFT"))

}

*# Output table*

**data.frame**(Predictor = quant\_names,

**bind\_rows**(

**quant\_summary**(credit, quant\_names[1]),

**quant\_summary**(credit, quant\_names[2]),

**quant\_summary**(credit, quant\_names[3]))) %>%

**pandoc.table**(split.tables=Inf)

| **Predictor** | **MIN** | **Q1** | **MEAN** | **MEDIAN** | **Q3** | **MAX** | **IQR** | **STDEV** | **SKEW** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| duration\_of\_credit\_month | 4 | 12 | 20.9 | 18 | 24 | 72 | 12 | 12.06 | RIGHT |
| age\_years | 19 | 27 | 35.54 | 33 | 42 | 75 | 15 | 11.35 | RIGHT |
| credit\_amount | 250 | 1366 | 3271 | 2320 | 3972 | 18424 | 2607 | 2823 | RIGHT |

All the three variables show marked positive skewness. A density plot of the predictors bears this out even more clearly.

left = **ggplot**(credit, **aes**(duration\_of\_credit\_month, fill=creditability, color=creditability)) + **geom\_density**(alpha=0.5)

middle = **ggplot**(credit, **aes**(age\_years, fill=creditability, color = creditability)) + **geom\_density**(alpha=0.5)

center = **ggplot**(credit, **aes**(credit\_amount, fill=creditability, color = creditability)) + **geom\_density**(alpha=0.5)

**grid.arrange**(left,middle,center, ncol=1)

Loss Function

After a basic exploratory analysis of the variables, the first round of models can be fit to the data. However, in order to evaluate the models, the profit-cost information must be functionalized.

*# Profit35 calculates the average profit given the predicted class values and the actual class values*

Profit35 = **function**(actual, pred, positive=NULL) {

*# Generate a confusion matrix*

Confusion\_DF <- MLmetrics::**ConfusionDF**(pred, actual)

**if** (**is.null**(positive) == TRUE)

positive <- **as.character**(Confusion\_DF[1, 1])

*# Determine True Positive Rate*

TP <- **as.integer**(**subset**(Confusion\_DF, y\_true == positive &

y\_pred == positive)["Freq"])

*# Determine False Positive Rate*

FP <- **as.integer**(**sum**(**subset**(Confusion\_DF, y\_true != positive &

y\_pred == positive)["Freq"]))

*# Calculate average profit*

val\_35 = (TP / **sum**(Confusion\_DF$Freq))\*0.35 + (FP / **sum**(Confusion\_DF$Freq))\*-1

**return**(val\_35)

}

*# p35 is a wrapper for the caret package that allows the spcificity and accuracy to be included with the Profit35 value in the model evaluation results*

p35 = **function**(data, lev = NULL, model = NULL) {

p35\_val <- **Profit35**(data$obs, data$pred, lev[1])

spec = MLmetrics::**Specificity**(data$obs, data$pred)

acc = MLmetrics::**Accuracy**(data$pred, data$obs)

**return**(**c**(Accuracy = acc, Specificity= spec, P35 = p35\_val))

}

This cost function can be tested using hypothetical predictions from the no-information model, which always predicts the most common class, “Good” and a perfect model which predicts all of the classes exactly the same as what is observed in the data.

No-Information Model

**p35**(**data.frame**(obs = credit$creditability, pred = **factor**(**c**(**rep**("Good", 1000)), levels = **levels**(credit$creditability))))

## Accuracy Specificity P35

## 0.700 0.000 -0.055

Perfect Model

**p35**(**data.frame**(obs = credit$creditability, pred = credit$creditability))

## Accuracy Specificity P35

## 1.000 1.000 0.245

The no-information model has an accuracy of 70%, specificity of 0%, and a P35 value of -0.055. Thus, a dummy model that always issues loans regardless of creditability (labels all applicants as “Good”) would be expected to have a 0.055 unit loss. If the average loan amount is $10,000, then the total loss using this model would be $550,000 and the per applicant loss is $550. Note that specificity is included as the specificity (1 - False Positive Rate, or TN / sum(Predicted Negatives)) is going to be strongly related with the P35. This relationship exists because the cost function has a large penalty for false positives, the case when the model predicts an applicant is “Good” when they should be “Bad”. This example also indicates how a model’s accuracy can be decent at 70%, but expected unit profit is negative, a loss, reinforcing the importance of having both the specificity and P35 metrics included in the model’s evaluation.

The perfect model has an accuracy of 100%, a specificity of 100%, and a P35 of 0.245. Thus, if a perfect model existed, it would have a 0.245 unit profit. If the average loan amount is $10,000, then the total profit using this model would be $2,450,000 and the per applicant profit is $2450. In this case, the bank is only ever profiting, and thus, pockets a 35% profit on all loans. It is important that no model should ever exceed this hypothetical value and any model that comes close to it should be evaluated carefully as no model is perfect.

Absent other information, the most relevent null-model to use for comparison in this analysis is a model that represents randomly assigning “Good” or “Poor” creditability given the assumption that loan applicants are creditable at a 70-30 ratio. Thus, such a strategy always predicts 700 “Good” and 300 “Poor” applicants. However, the predictions are randomly assigned to applicants. Such a strategy would have varying performance depending on which applicants were randomly assigned to which category. In fact, one of the possibilities for such a strategy is one where all the predictions are 100% accurate. However, the chances of this occuring are minimal. To capture the model’s performance considering this random variation, the average performance of the model is used. This case is the one where the model predicts both categories at a 70-30 Good-Poor split.

Blind Proportional Model

blind = credit %>%

**select**(creditability) %>%

**mutate**(row\_num = 1:1000) %>%

**arrange**(creditability) %>%

**mutate**(predicted = **c**(**rep**("Good", 490), **rep**("Poor", 210), **rep**("Good", 210), **rep**("Poor", 90))) %>%

**arrange**(row\_num)

**p35**(**data.frame**(obs = blind$creditability, pred = blind$predicted))

## Accuracy Specificity P35

## 0.5800 0.3000 -0.0385

The blind proportional model has an accuracy of 58%, specificity of 30%, and a P35 value of -0.039. Thus, this strategy, on average, improves over the no-information model by about 0.016, but still represents an average loss for the bank.

Exploring the Initial Model Space

As prediction quality is the priority for this problem, a range of different model types is evaluated at once to see if any particular model produces better predictions. The included models are logistic regression, linear discriminant analysis, random forest, and xgboost gradient boosted trees.

Both random forest and xgboost have hyperparameters that can be tuned. The following code block includes the hyperparameters set to their optimal values. For the code used to find the optimal values, see Appendix A.

The models will be trained to maximize the per unit profit using the p35 function. In order to get a more accurate estimate of the per unit profit of each model, a 10x10 K-Fold Cross Validation (CV) process is used. Since multiple models are being compared, each model will be trained on the same set of folds.

**set.seed**(123)

repeatedResamples = **function**(y, k = 10, reps=10) {

**suppressWarnings**(

**for** (idx **in** 1:reps) {

*# Create custom indices: myFolds*

myFolds <- **createCVFolds**(y, k = k)

*# Create reusable trainControl object: myControl*

myControl <- **trainControl**(summaryFunction = p35,

classProbs = TRUE, *# IMPORTANT!*

verboseIter = FALSE,

savePredictions = TRUE,

index = myFolds

)

log\_fit = **train**(creditability ~ .,

method = 'glm',

family = 'binomial',

data = credit,

trControl = myControl,

metric = "P35")

lda\_fit = **train**(creditability ~ .,

method = 'lda',

show = FALSE,

data = credit,

trControl = myControl,

metric = "P35")

tuneGridRF = **data.frame**(

mtry=11

)

rf\_fit = **train**(creditability ~ .,

method = 'rf',

data = credit,

trControl=myControl,

tuneGrid=tuneGridRF,

metric = "P35")

tuneGridXGB <- **expand.grid**(

nrounds=300,

max\_depth = 2,

eta = 0.07,

gamma = 0.1,

colsample\_bytree = 1,

subsample = 1,

min\_child\_weight = 2)

xgb\_fit = **train**(creditability ~ .,

method = 'xgbTree',

data = credit,

trControl= myControl,

tuneGrid = tuneGridXGB,

metric = "P35")

*# Create model\_list*

model\_list <- **list**(log = log\_fit, lda = lda\_fit, rf = rf\_fit, xgb = xgb\_fit)

**if** (idx == 1) {

*# Pass model\_list to resamples(): resamples*

resamples <- **resamples**(model\_list)

}

**else** {

current\_resample = **resamples**(model\_list)

resamples$values = **bind\_rows**(resamples$values, current\_resample$values)

}

}

)

**return**(resamples)

}

report\_dat = **repeatedResamples**(credit$creditability)

**summary**(report\_dat)

##

## Call:

## summary.resamples(object = report\_dat)

##

## Models: log, lda, rf, xgb

## Number of resamples: 100

##

## Accuracy

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

## log 0.67 0.72 0.750 0.7481 0.78 0.82 0

## lda 0.67 0.72 0.750 0.7513 0.78 0.84 0

## rf 0.66 0.72 0.750 0.7476 0.77 0.84 0

## xgb 0.66 0.73 0.755 0.7575 0.79 0.85 0

##

## P35

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

## log -0.0135 0.03650 0.05675 0.055890 0.077750 0.1135 0

## lda -0.0135 0.03700 0.05850 0.058440 0.077625 0.1175 0

## rf -0.0300 0.01725 0.03425 0.036605 0.055125 0.1045 0

## xgb -0.0170 0.03200 0.05375 0.053590 0.074000 0.1275 0

##

## Specificity

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

## log 0.2333333 0.4333333 0.5000000 0.4823333 0.5333333 0.6666667 0

## lda 0.2333333 0.4333333 0.5000000 0.4896667 0.5666667 0.6666667 0

## rf 0.1666667 0.3250000 0.3666667 0.3843333 0.4333333 0.6000000 0

## xgb 0.2333333 0.4000000 0.4500000 0.4536667 0.5083333 0.6666667 0

Overall, the four models improved over the performance of blind porportional model. The median of the metrics for each model’s performance over the CV folds is used for the metric estimate as the median does not depend on the tail behavior of the performance metric distribution and, thus, is highly resistant to aberrant data points created during the k-fold process.

The best model for this round is the xgboost model.

It may be appropriate to introduce regularization into the model fitting procedure. Regularization introduces penalties for model complexity in order to reduce overfitting. Overfitting causes increases in model variance for out-of-sample predictions, which reduces model quality and accuracy. Since so many of the variables have low average gain relative to the others, it is possible the model will overfit the data to some extent by attempting to draw relationships from variables that little to no relationship with the response. Models like xgboost and random forests are more robust to this issue. But, the logistic regression and linear discriminant analysis are not robust to such issue and may improve from regularization.

Round Two: Exploring Regularized Models

For round two, the goal will be to use regularization in the models that do not involve trees. The method “regLogistic” will be used for regularized logistic regression and “rlda” for regularized linear discriminant analysis.

**set.seed**(123)

repeatedResamples = **function**(y, data, k = 10, reps=10) {

**suppressWarnings**(

**for** (idx **in** 1:reps) {

*# Create custom indices: myFolds*

myFolds <- **createCVFolds**(y, k = k)

*# Create reusable trainControl object: myControl*

myControl <- **trainControl**(summaryFunction = p35,

classProbs = TRUE, *# IMPORTANT!*

verboseIter = FALSE,

savePredictions = TRUE,

index = myFolds

)

round2\_tuneGridRegLog = **expand.grid**(

cost = 1,

loss = "L1",

epsilon = 0.01

)

log\_reg\_fit = **train**(creditability ~ .,

method = 'regLogistic',

data = data,

trControl=myControl,

tuneGrid = round2\_tuneGridRegLog,

metric = "P35")

round2\_tuneGridrlda = **data.frame**(estimator = "Moore-Penrose Pseudo-Inverse")

rlda\_fit = **train**(creditability ~ .,

method = 'rlda',

data = data,

trControl=myControl,

tuneGrid = round2\_tuneGridrlda,

metric = "P35")

*# Create model\_list*

model\_list <- **list**(log\_reg = log\_reg\_fit, rlda = rlda\_fit)

**if** (idx == 1) {

*# Pass model\_list to resamples(): resamples*

resamples <- **resamples**(model\_list)

}

**else** {

current\_resample = **resamples**(model\_list)

resamples$values = **bind\_rows**(resamples$values, current\_resample$values)

}

}

)

**return**(resamples)

}

fit\_results = **repeatedResamples**(credit$creditability, credit)

**summary**(fit\_results)

##

## Call:

## summary.resamples(object = fit\_results)

##

## Models: log\_reg, rlda

## Number of resamples: 100

##

## Accuracy

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

## log\_reg 0.68 0.7300 0.76 0.7558 0.78 0.84 0

## rlda 0.58 0.6475 0.68 0.6779 0.71 0.79 0

##

## P35

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

## log\_reg 0.0035 0.04 0.05675 0.055075 0.071375 0.1435 0

## rlda 0.0315 0.07 0.08750 0.087935 0.104500 0.1510 0

##

## Specificity

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

## log\_reg 0.3000000 0.4000000 0.4666667 0.4643333 0.5 0.7666667 0

## rlda 0.5333333 0.7333333 0.7666667 0.7726667 0.8 0.9666667 0

The regularized logistic regression performed similarly to the logistic regression with a median per unit profit of 0.057 for the regularized logistic regression versus 0.055 for the logistic regression. The original logistic regression will be used going forward.

On the other hand, the regularized linear discriminant analysis resulted in a per unit profit of 0.088 versus the linear discriminate analysis which had a per unit profit of 0.054. The “rlda” model’s significant performance is related to its high specificity of 0.77, resulting in a lower false positive rate which means fewer applicants with bad credit are classified as having good credit by the model.

Round Three: Prediction Probability Cutoff Threshold Optimization

One of the assumptions that the analysis up to this point has made is that the cutoff threshold for determining whether a prediction belongs to the positive class or negative class is 0.5. This value means that if the model gives a prediction probability of greater than 0.5 then the prediction will be “Good” and vice versa. This threshold changes the sensitivity and specificity of the model’s performance. Since the profit-cost function is sensitive to a model’s specificity, which is inversely proportional to its False Positive Rate, the next step will be to modify the most promising models’ prediction cutoff probabilities to favor a higher specificity. In effect, if the prediction probability cutoff value is increased above 0.5, then the model will be less likely to predict “Good” creditability, potentially reducing the number of “Good” predictions that are in fact “Poor” and increasing the per unit profit of the model.

The functions for this step are found in the cutoff\_check.R file and involve a 10x10 K-Fold Cross Validation similar to the validation procedures used previously in this analysis. The procedure evaluates cutoff values down to a 0.05 difference in values (0.50, 0.55, 0.6, etc).

*# Determine best cutoff using 10x10 K-Fold Cross Validation*

best\_cutoff = **get\_best\_cutoffs**(formula = creditability ~ ., data = credit, method = "glm", family="binomial", verbose = FALSE)

best\_cutoff %>% **pandoc.table**()

| **method** | **best\_cutoff** | **best\_metric\_median** |
| --- | --- | --- |
| glm | 0.7 | 0.08825 |

*# Save final results for comparison to other models*

final\_results = **train\_cv**(formula = creditability ~ ., data = credit, , method = "glm", family="binomial", verbose=FALSE, cutoff = 0.7)

log\_perf = **data.frame**(value = final\_results$P35, group="Logistic")

Starting with the logistic regression model, the best cutoff is 0.7, which increases the estimate of the model’s per unit profit from 0.057 to 0.088.

*# Determine best cutoff using 10x10 K-Fold Cross Validation*

round1\_tuneGridRF = **data.frame**(mtry=11)

best\_cutoff = **get\_best\_cutoffs**(formula = creditability ~ ., data = credit, method = "rf", tuneGrid = round1\_tuneGridRF, verbose=FALSE)

best\_cutoff %>% **pandoc.table**()

| **method** | **best\_cutoff** | **best\_metric\_median** |
| --- | --- | --- |
| rf | 0.7 | 0.0865 |

*# Save final results for comparison to other models*

final\_results = **train\_cv**(formula = creditability ~ ., data = credit, , method = "rf", tuneGrid = round1\_tuneGridRF, verbose=FALSE, cutoff = 0.7)

rf\_perf = **data.frame**(value = final\_results$P35, group="Random Forest")

The per unit profit estimate for the Random Forest model is improved from 0.034 to 0.087 using a cutoff of 0.7.

*# Determine best cutoff using 10x10 K-Fold Cross Validation*

round1\_tuneGridXGB <- **expand.grid**(

nrounds=300,

max\_depth = 2,

eta = 0.07,

gamma = 0.1,

colsample\_bytree = 1,

subsample = 1,

min\_child\_weight = 2)

best\_cutoff = **get\_best\_cutoffs**(formula = creditability ~ ., data = credit, method = "xgbTree", tuneGrid = round1\_tuneGridXGB, verbose=FALSE)

best\_cutoff %>% **pandoc.table**()

| **method** | **best\_cutoff** | **best\_metric\_median** |
| --- | --- | --- |
| xgbTree | 0.7 | 0.092 |

*# Save final results for comparison to other models*

final\_results = **train\_cv**(formula = creditability ~ ., data = credit, method = "xgbTree", tuneGrid=round1\_tuneGridXGB, verbose=FALSE, cutoff = 0.7)

xgb\_perf = **data.frame**(value = final\_results$P35, group="Xgboost")

The per unit profit estimate for the xgboost model is improved from 0.054 to 0.092 using a cutoff of 0.7. This outcome is the best median CV performance seen so far.

*# Determine best cutoff using 10x10 K-Fold Cross Validation*

round2\_tuneGridrlda = **data.frame**(estimator = "Moore-Penrose Pseudo-Inverse")

best\_cutoff = **get\_best\_cutoffs**(formula = creditability ~ ., data = credit, method = "rlda", tuneGrid=round2\_tuneGridrlda, verbose=FALSE)

best\_cutoff %>% **pandoc.table**()

| **method** | **best\_cutoff** | **best\_metric\_median** |
| --- | --- | --- |
| rlda | 0.5 | 0.092 |

*# Save final results for comparison to other models*

final\_results = **train\_cv**(formula = creditability ~ ., data = credit, method = "rlda", tuneGrid=round2\_tuneGridrlda, verbose=FALSE, cutoff = 0.7)

rlda\_perf = **data.frame**(value = final\_results$P35, group="Regularized LDA")

The per unit profit estimate for the regularized LDA model is negligibly improved from 0.088 to 0.092 using a cutoff of 0.5. The small difference in performance 0f 0.004 between the models for the same cutoff value (since 0.5 is the default cutoff) is due to the variance caused by a different set of folds used in the 10x10 K-Fold CV for the best cutoff search process. This outcome gives evidence to the validity of the CV procedure as the performance estimates for two entirely different sets of CV folds had similar median per unit profit values.

Final Model Selection and Metrics

With the best cutoffs selected for the models, all four of the final model candidates have similar performance metrics with the xgboost and regularized LDA models showing the best per unit profit estimates and the logistic regression and random forest models showing slightly worse results. However, the range of per unit profit estimates is only 0.005. Thus, these models have similar performance characteristics.

In order to make the final selection, the distribution of per unit profit estimates for each of the ten repeated CV folds for each model with the optimum cutoff values is plotted together in a density plot. The density plot shows the distribution of values that occurred during the 10x10 K-Fold CV. Since the performance value produced from this procedure is an estimate of model performance, one can treat the probability density of each model’s per unit profit estimates as representative of a sampling distribution of the actual model performance.

*# Combine the four results dataframes from the previous section*

all\_perf = **bind\_rows**(log\_perf, rf\_perf, xgb\_perf, rlda\_perf)

*# Plot the distributions*

all\_perf %>% **ggplot**(**aes**(x = value, fill=group, color=group)) +

**geom\_density**(alpha=0.3) +

**theme**(legend.position=**c**(0.2, 0.8), plot.title = **element\_text**(hjust = 0.5)) +

**labs**(title = "Density of Per Unit Profit by Model Type",

y = "P(Per Unit Profit)",

x = "Per Unit Profit",

fill = "Model Type",

color = "Model Type")

Looking at this plot it is clear that no one model is significantly better than another due to the high overlap in distributions. However, if one had to choose a model, the xgboost model has the relatively most favorable model with a greater density over higher per unit profit estimate values. The relatively worse model would be the Logistic Regression as it as a greater variance in values than the other models and is, thus, expected to have a more unpredictable performance when used on out-of-sample data.

The final model is fit to the data using 10x10 K-Fold CV. In order to use the model on new predictions, a conditional transformation using the model’s predictions probabilities for the class “Good” would have to be performed using a cutoff threshold of 0.7.

**set.seed**(123)

*# Create custom indices: myFolds*

myFolds <- **createCVFolds**(credit$creditability, k = 10)

*# Create reusable trainControl object: myControl*

myControl <- **trainControl**(summaryFunction = p35,

classProbs = TRUE, *# IMPORTANT!*

verboseIter = FALSE,

savePredictions = TRUE,

index = myFolds

)

final\_fit = **train**(creditability ~ .,

data = credit,

method = "xgbTree",

tuneGrid = round1\_tuneGridXGB,

trControl = myControl,

metric = "P35")

Part Four: Identifying most relevant variables

While the model selection process is complete, I thought it would be useful to give the manager a report of what variables in her data provided above average information for predicting creditworthiness. In addition to having her model, she may be able to use this information to make better decisions for collecting future data, especially in brainstorming related applicant information to the high-quality variables or for developing new variables that achieve more detail in the areas that have the strongest predictive quality for determining creditability.

First, each categorical variable in the data set is split into separate binary predictors called dummy variables.

dummies <- **dummyVars**(creditability ~ ., data = credit)

dummies = **predict**(dummies, newdata = credit) %>% tbl\_df

dummies$creditability = credit$creditability

Then, two methods of determining variable predictive information are combined to determine the predictive quality of each variable:

1. From the woe package, the Information Value (IV) is concept used in risk management to assess predictive power of variable calculated from the WoE (Weight of Evidence).
2. From the xgboost package, the xgb.importance function calculates the gain for each predictor variable. The gain gives you an indication of how important a predictor is in how it causes branches of a decision tree such as those produced by the “xgbTree” method.

Both the IV and the Gain are scaled and normalized and then averaged. Either of these methods is adequate on its own, but the process can confirm each other when combined together to increase confidence in the results.

**set.seed**(123)

## Find Information Value for each variable

**invisible**(**capture.output**(IV <- **iv.mult**(**as.data.frame**(dummies),"creditability", summary = TRUE)))

## Find Gain for each varaible

X = dummies %>% dplyr::**select**(-creditability)

**library**(xgboost)

*# Prep data for xgboost*

x\_num = **as.matrix**(X) %>% **apply**(2,as.numeric)

x\_label = **as.numeric**(**as.character**(**ifelse**(dummies$creditability == "Good", 1, 0)))

x\_matrix = xgboost::**xgb.DMatrix**(data = x\_num, label = x\_label)

*# Fit the model using xgboost settings from analysis*

bst <- **xgboost**(data = x\_matrix,

nround = 300,

eta = 0.07,

max.depth = 2,

gamma = 0.1,

min\_child\_weight = 2,

subsample = 1,

colsample\_bytree = 1,

objective = "binary:logistic",

verbose = 0)

*# Determine most important predictors*

xgb\_import = **xgb.importance**(**colnames**(x\_num, do.NULL = TRUE, prefix = "col"), model = bst)

*# Combine the output of both types of predictor quality measures*

combo = IV %>% **left\_join**(xgb\_import, by = **c**("Variable" = "Feature"))

combo[**is.na**(combo)] = 0

*# Normalize and combine the two measures*

combo.out = combo %>% dplyr::**select**(Variable, Strength, InformationValue, Gain) %>% **mutate**(InformationValue = **as.numeric**(**scale**(InformationValue)), Gain = **as.numeric**(**scale**(Gain)), Avg\_Predict\_Qual = InformationValue + Gain / 2) %>% **arrange**(Variable)

*# Plot the results*

**library**(RColorBrewer)

fills <- **rev**(**brewer.pal**(6, "Blues"))

Variable <- InformationValue <- Strength <- NULL

**ggplot**(data = combo.out %>% **filter**(Avg\_Predict\_Qual > 0)) + **geom\_bar**(**aes**(x = **reorder**(Variable, Avg\_Predict\_Qual),

y = Avg\_Predict\_Qual, fill = Strength), stat = "identity") +

**coord\_flip**() + **scale\_fill\_manual**(values = fills) + **theme**(panel.grid.major.y = **element\_blank**(),

panel.grid.major.x = **element\_line**(linetype = "dashed",

colour = "grey"), panel.grid.minor = **element\_blank**(),

panel.background = **element\_blank**(), axis.ticks.x = **element\_blank**(),

axis.ticks.y = **element\_blank**()) + **xlab**("Variable") +

**ylab**("Prediction Value") + **ggtitle**("Prediction Value Summary for Predictors with Above Average Values")

The plot above is one of the potential deliverables to the manager that summarizes the results nicely and compares the difference in prediction strength. The combo.out table could also be shared to give even more detailed results.

It appears the fourth category of account\_balance has the highest predictive value for determining creditability. Notably, the variables not on this list do not have a strong predictive value for creditability.

Conclusion

The best model to predict creditability uses the xgboost algorithm to fit a gradient boosted tree classification model to the data. The final model, when using a 0.7 cutoff threshold, is expected produce a per unit profit of 0.092. This expected performance is an increase of 0.131 in expected per unit profit over a blind proportional model that predicts “Good” and “ppoor creditability at a 70-30 split and has an expected per unit profit of -0.039. If the model meets this performance estimate, the bank should expect to earn a per applicant profit of $920 if the average loan amount is $10,000.

One advantage of the xgboost model is it is portable to another language like Python. This benefit means the model could be coded into a predictive program that loan managers could run on applicant information. This tool could also be constructed using the R Shiny package as well.

Appendix A: Tuning Hyperparamaters

Both random forests and xgboost trees have tunable hyperparameters.

For random forest, the hyperparameter is “mtry”. To determine the best mtry, a standard three value search is performed for a small, medium, and large value of mtry.

1. Set eta = 0.1, max\_depth = 5, min\_child\_weight = 1, gamma = 0, subsample = 0.8, colsample\_bytree = 0.8

a. Find the best nrounds by using seq(from = 50, to = 950, by = 50)

2. Set nrounds equal to best value from 1a

a. Find the best max\_depth by using seq(1,10,1) and min\_child\_weight by suing seq(1,6,1) in the same call to train

3. Set max\_depth and min\_child\_weight to best value from 2a

a. Find the best gamma using seq(0,0.5,0.1)

b. Find the best nrounds by using seq(from = 50, to = 950, by = 50)

4. Set gamma to best value from 3a and set nrounds to the best value from 3b

a. Find the best subsample and colsample\_bytree using seq(0.6,1,0.05) in the same call to train

5. Set subsample and colsample\_bytree to best values from 4a

a. IF REGULARIZING ONLY- Find best alpha or lambda values using seq(0,1,0.1)

b. Find best eta using seq(0.01,0.1,0.01)

6. Set alpha and lambda to best values from 5a and eta to best value from 5b

The following code determines the best settings using 10X10 fold cross validation.

## trainControl settings for hyperparameter tunning

myControl <- **trainControl**(summaryFunction = p35,

classProbs = TRUE, *# IMPORTANT!*

verboseIter = TRUE,

savePredictions = TRUE,

method= "repeatedcv",

number = 5,

repeats = 5

)

### Unique model tuning grid for tunable parameters

tuneGridXGB <- **expand.grid**(

nrounds=300,

max\_depth = 2,

eta = 0.01,

gamma = 0.1,

colsample\_bytree = 1,

subsample = 1,

min\_child\_weight = 2)

## Tuning model for tunable hyperparameters in caret

**set.seed**(123)

**train**(creditability ~ .,

data = credit,

method = "rlda",

*# tuneGrid = tuneGridXGB,*

trControl = myControl)

## Tuning model for non-tunable hyperparameters in caret

**set.seed**(123)

results\_list = **list**()

**for** (value **in** **seq**(0,1, 0.1)) {

fit = **train**(creditability ~ .,

data = credit,

method = "xgbTree",

tuneGrid = tuneGridXGB,

trControl = myControl,

alpha = 0.7,

lambda = 0.7)

results\_list[[**as.character**(value)]] = fit$results

}

*# Record of Best Results*

round1\_tuneGridRF = **data.frame**(mtry=11)

round1\_tuneGridXGB <- **expand.grid**(

nrounds=300,

max\_depth = 2,

eta = 0.07,

gamma = 0.1,

colsample\_bytree = 1,

subsample = 1,

min\_child\_weight = 2)

round2\_tuneGridRegLog = **expand.grid**(

cost = 1,

loss = "L1",

epsilon = 0.01

)

round2\_tuneGridrlda = **data.frame**(estimator = "Moore-Penrose Pseudo-Inverse")