Predictive Model for Credit Risk Assessment

Data Science Academy

Vinicius Biazon

Step 1 - Collecting the Data

```r
# Loading the dataset into a dataframe
credit.df <- read.csv("credit_dataset.csv", header = TRUE, sep = ",")
```

Step 2 - Normalizing the Data

```r
# Converting variables to factor type (categorical)
to.factors <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- as.factor(df[[variable]])
  }
  return(df)
}


## Normalization
scale.features <- function(df, variables){
  for (variable in variables){
    df[[variable]] <- scale(df[[variable]], center=T, scale=T)
  }
  return(df)
}


# Normalizing numerical variables
numeric.vars <- c("credit.duration.months", "age", "credit.amount")
credit.df <- scale.features(credit.df, numeric.vars)


# Factor-type variables
categorical.vars <- c('credit.rating', 'account.balance', 'previous.credit.payment.status',
                      'credit.purpose', 'savings', 'employment.duration', 'installment.rate',
                      'marital.status', 'guarantor', 'residence.duration', 'current.assets',
                      'other.credits', 'apartment.type', 'bank.credits', 'occupation',
                      'dependents', 'telephone', 'foreign.worker')

credit.df <- to.factors(df = credit.df, variables = categorical.vars)
```

Step 3 - Dividing data into training and testing

```
# Dividing data into training and testing - 60:40 ratio
indexes <- sample(1:nrow(credit.df), size = 0.6 * nrow(credit.df))
train.data <- credit.df[indexes,]
test.data <- credit.df[-indexes,]
```

Step 4 - Feature Selection

```
# Feature Selection
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

```
# Function for select variables
run.feature.selection <- function(num.iters=20, feature.vars, class.var){
  set.seed(10)
  variable.sizes <- 1:10
  control <- rfeControl(functions = rfFuncs, method = "cv",
                        verbose = FALSE, returnResamp = "all",
                        number = num.iters)
  results.rfe <- rfe(x = feature.vars, y = class.var,
                     sizes = variable.sizes,
                     rfeControl = control)
  return(results.rfe)
}


rfe.results <- run.feature.selection(feature.vars = train.data[,-1],
                                     class.var = train.data[,1])


# Viewing the results
rfe.results
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (20 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.7067 0.0000    0.01084  0.0000
##          2   0.7150 0.2005    0.07446  0.2179
##          3   0.7150 0.2058    0.05493  0.1492
##          4   0.7547 0.3640    0.09907  0.2717
##          5   0.7547 0.3570    0.09227  0.2480
##          6   0.7646 0.3658    0.08443  0.2628
##          7   0.7645 0.3610    0.08007  0.2400
##          8   0.7663 0.3529    0.08324  0.2471
##          9   0.7632 0.3626    0.06943  0.1960
##         10   0.7550 0.3466    0.06889  0.1959
##         20   0.7714 0.3550    0.06291  0.1848        *
##
## The top 5 variables (out of 20):
##    account.balance, credit.duration.months, previous.credit.payment.status, savings, credit.amount
```

```r
varImp((rfe.results))
```

```
##                                    Overall
## account.balance                 17.1230875
## credit.duration.months          11.3367916
## previous.credit.payment.status   9.5711988
## savings                          8.2236468
## credit.amount                    4.3876959
## credit.purpose                   4.0790940
## age                              2.5745197
## guarantor                        2.4414053
## apartment.type                   2.4071515
## marital.status                   1.9734464
## current.assets                   1.9659546
## employment.duration              1.8806548
## occupation                       1.5887135
## dependents                       1.4670447
## bank.credits                     1.3109975
## other.credits                    0.8185889
## telephone                        0.7248668
## residence.duration               0.4147860
## foreign.worker                  -0.5914106
## installment.rate                -0.5930717
```

Step 5 - Creating and Evaluating the Model

```r
# Creating and Evaluating the Model
library(caret)
```

```r
library(ROCR)


# Utility library for building graphics
source("plot_utils.R")


## Separating feature and class
test.feature.vars <- test.data[,-1]
test.class.var <- test.data[,1]


# Building a logistic regression model
formula.init <- "credit.rating ~ ."
formula.init <- as.formula(formula.init)
lr.model <- glm(formula = formula.init, data = train.data, family = "binomial")


# Viewing the model
summary(lr.model)
```

```
##
## Call:
## glm(formula = formula.init, family = "binomial", data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5419  -0.7045   0.4086   0.7050   1.9463
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       1.16628    1.06010   1.100 0.271264
## account.balance2                  0.31923    0.28665   1.114 0.265414
## account.balance3                  1.49616    0.28484   5.253  1.5e-07 ***
## credit.duration.months           -0.39587    0.14418  -2.746 0.006037 **
## previous.credit.payment.status2  0.79110    0.39429   2.006 0.044815 *
## previous.credit.payment.status3  0.97871    0.40949   2.390 0.016844 *
## credit.purpose2                  -1.49555    0.55953  -2.673 0.007521 **
## credit.purpose3                  -1.98682    0.55303  -3.593 0.000327 ***
## credit.purpose4                  -2.33613    0.53784  -4.343  1.4e-05 ***
## credit.amount                    -0.33074    0.17199  -1.923 0.054477 .
## savings2                          0.55392    0.38003   1.458 0.144956
## savings3                          0.52054    0.41976   1.240 0.214941
## savings4                          1.27146    0.39234   3.241 0.001192 **
## employment.duration2              0.01751    0.31030   0.056 0.954992
## employment.duration3              0.63109    0.37315   1.691 0.090789 .
## employment.duration4              0.34595    0.36343   0.952 0.341145
## installment.rate2                -0.41123    0.41581  -0.989 0.322671
## installment.rate3                -0.39323    0.44718  -0.879 0.379203
## installment.rate4                -0.72074    0.40134  -1.796 0.072520 .
## marital.status3                   0.64629    0.25393   2.545 0.010925 *
## marital.status4                   0.32874    0.41933   0.784 0.433064
## guarantor2                        0.56287    0.37730   1.492 0.135741
```

```
## residence.duration2            -0.77860    0.37506  -2.076 0.037901 *
## residence.duration3            -0.32924    0.44122  -0.746 0.455545
## residence.duration4            -0.57015    0.37925  -1.503 0.132749
## current.assets2                -0.22753    0.31908  -0.713 0.475780
## current.assets3                 0.14808    0.30600   0.484 0.628446
## current.assets4                -0.25448    0.54506  -0.467 0.640582
## age                             0.10919    0.12940   0.844 0.398769
## other.credits2                  0.34679    0.28425   1.220 0.222461
## apartment.type2                 0.76281    0.30253   2.521 0.011688 *
## apartment.type3                 0.18413    0.61236   0.301 0.763650
## bank.credits2                   0.13892    0.30148   0.461 0.644943
## occupation2                    -0.54620    0.79739  -0.685 0.493350
## occupation3                    -0.47442    0.77283  -0.614 0.539297
## occupation4                    -0.53380    0.83262  -0.641 0.521454
## dependents2                    -0.31941    0.34290  -0.931 0.351604
## telephone2                      0.16164    0.25346   0.638 0.523642
## foreign.worker2                 1.38209    0.83929   1.647 0.099612 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 726.13  on 599  degrees of freedom
## Residual deviance: 545.20  on 561  degrees of freedom
## AIC: 623.2
##
## Number of Fisher Scoring iterations: 5
```

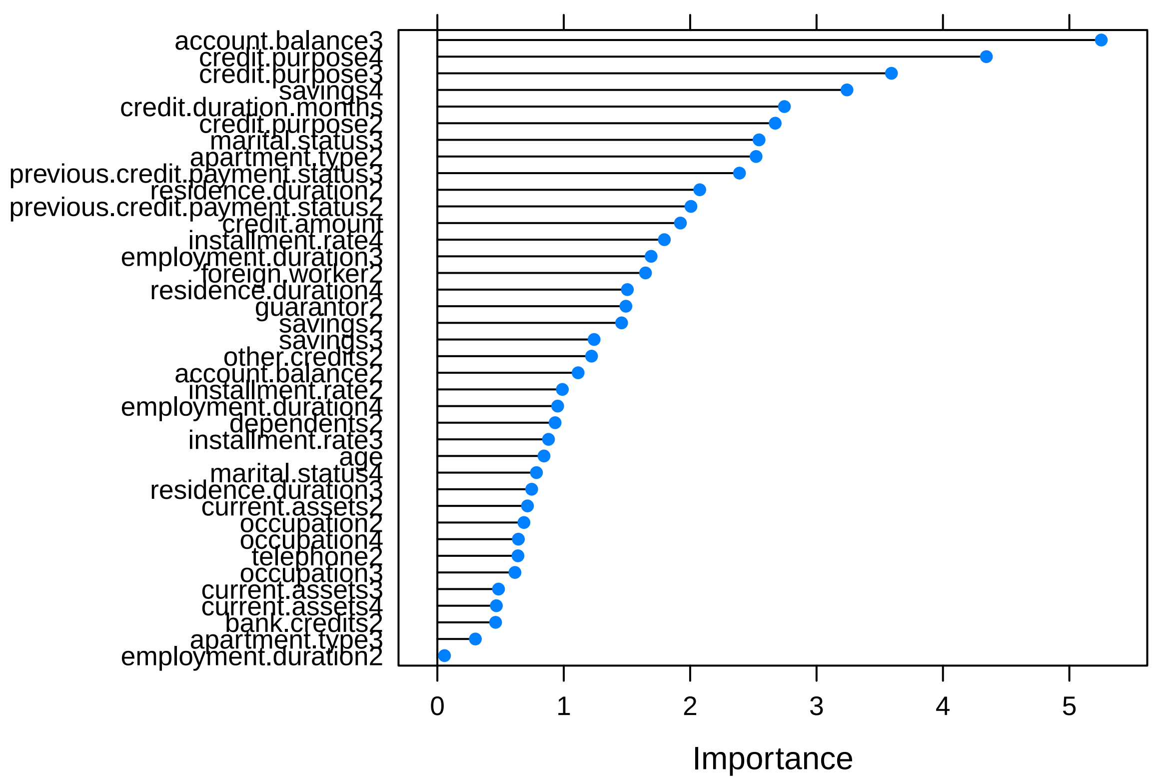
```r
# Testing the model on test data
lr.predictions <- predict(lr.model, test.data, type="response")
lr.predictions <- round(lr.predictions)


# Evaluating the model
confusionMatrix(table(data = lr.predictions, reference = test.class.var), positive = '1')
```

```
## Confusion Matrix and Statistics
##
##     reference
## data   0   1
##    0  48  27
##    1  76 249
##
##                Accuracy : 0.7425
##                  95% CI : (0.6967, 0.7847)
##     No Information Rate : 0.69
##     P-Value [Acc > NIR] : 0.01234
##
##                   Kappa : 0.3246
##
##  Mcnemar's Test P-Value : 2.25e-06
##
##             Sensitivity : 0.9022
##             Specificity : 0.3871
```

```
##          Pos Pred Value : 0.7662
##          Neg Pred Value : 0.6400
##              Prevalence : 0.6900
##          Detection Rate : 0.6225
##    Detection Prevalence : 0.8125
##       Balanced Accuracy : 0.6446
##
##        'Positive' Class : 1
##
```

Step 6 - Optimizing the Model

```
## Feature selection
formula <- "credit.rating ~ ."
formula <- as.formula(formula)
control <- trainControl(method = "repeatedcv", number = 10, repeats = 2)
model <- train(formula, data = train.data, method = "glm", trControl = control)
importance <- varImp(model, scale = FALSE)
plot(importance)
```




```
# Building the new model with the selected variables
formula.new <- "credit.rating ~ account.balance + credit.purpose + previous.credit.payment.status + savi
formula.new <- as.formula(formula.new)
lr.model.new <- glm(formula = formula.new, data = train.data, family = "binomial")
```

```r
# Viewing the new model
summary(lr.model.new)
```

```
##
## Call:
## glm(formula = formula.new, family = "binomial", data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5609  -0.8377   0.4852   0.7968   1.8340
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       0.5237     0.5228   1.002 0.316511
## account.balance2                  0.3046     0.2533   1.202 0.229214
## account.balance3                  1.4492     0.2563   5.655 1.55e-08 ***
## credit.purpose2                  -1.2090     0.4859  -2.488 0.012831 *
## credit.purpose3                  -1.3354     0.4636  -2.880 0.003971 **
## credit.purpose4                  -1.6652     0.4601  -3.619 0.000295 ***
## previous.credit.payment.status2   0.8127     0.3400   2.390 0.016841 *
## previous.credit.payment.status3   1.1839     0.3538   3.347 0.000818 ***
## savings2                          0.5487     0.3519   1.559 0.118980
## savings3                          0.5731     0.3885   1.475 0.140147
## savings4                          1.1605     0.3514   3.303 0.000958 ***
## credit.duration.months           -0.5285     0.1033  -5.114 3.15e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 726.13  on 599  degrees of freedom
## Residual deviance: 591.90  on 588  degrees of freedom
## AIC: 615.9
##
## Number of Fisher Scoring iterations: 5
```

```r
# Testing the new model on test data
lr.predictions.new <- predict(lr.model.new, test.data, type = "response")
lr.predictions.new <- round(lr.predictions.new)
```

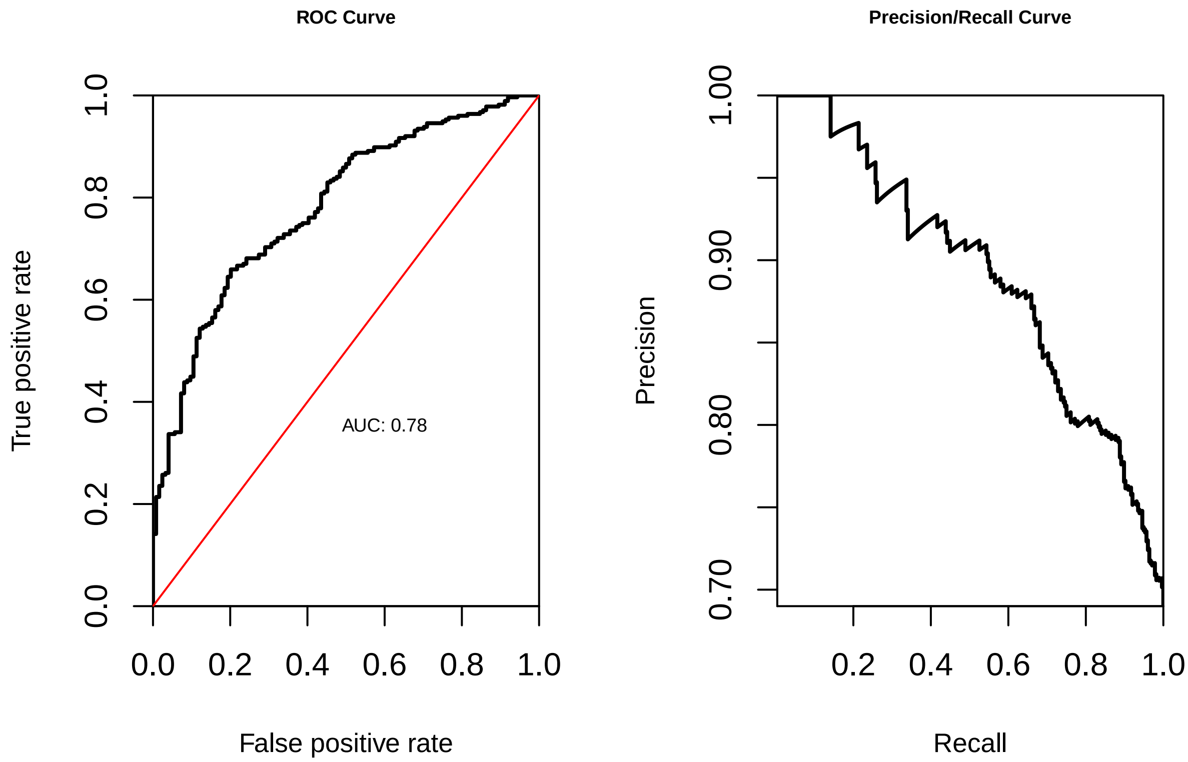
```r
# Evaluating the new model
confusionMatrix(table(data = lr.predictions.new, reference = test.class.var), positive = '1')
```

```
## Confusion Matrix and Statistics
##
##     reference
## data   0   1
##    0  38  24
##    1  86 252
##
```

```
##                Accuracy : 0.725
##                  95% CI : (0.6784, 0.7682)
##     No Information Rate : 0.69
##     P-Value [Acc > NIR] : 0.07107
##
##                   Kappa : 0.2545
##
##  Mcnemar's Test P-Value : 6.023e-09
##
##             Sensitivity : 0.9130
##             Specificity : 0.3065
##          Pos Pred Value : 0.7456
##          Neg Pred Value : 0.6129
##              Prevalence : 0.6900
##          Detection Rate : 0.6300
##    Detection Prevalence : 0.8450
##       Balanced Accuracy : 0.6097
##
##        'Positive' Class : 1
##
```

Step 7 - Evaluating performance: ROC Curve

```r
# Evaluating performance


# Creating ROC curves
lr.model.best <- lr.model
lr.prediction.values <- predict(lr.model.best, test.feature.vars, type = "response")
predictions <- prediction(lr.prediction.values, test.class.var)
par(mfrow = c(1,2))
plot.roc.curve(predictions, title.text = "ROC Curve")
plot.pr.curve(predictions, title.text = "Precision/Recall Curve")
```


ROC Curve
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC: 0.78
False positive rate
Precision/Recall Curve
Precision
0.70
0.80
0.90
1.00
0.2
0.4
0.6
0.8
1.0
Recall