Credit risk scorecard

Group: 13

-Jordan Tobias, 11902127, e11902127@student.tuwien.ac.at -Trailovic Luka, 11918523, e11918523@stude

Nov. 11, 2023

Contents

#Introduction

##Abstract:

#Predictive analytics research methodology

#Empirical result

0.1 Loading libraries

```
library(scorecard)
```

0.2 Loading external data

Calculating the information value of the predictor variables for selecting "informative" variables

```
data("germancredit")
dimension_information <- iv(germancredit, y="creditability", order=TRUE)
head(dimension_information, n = 5)
```

```
##                               variable info_value
##                                 <char>      <num>
## 1: status.of.existing.checking.account  0.6660115
```

```
## 2:                    duration.in.month  0.3345035
## 3:                       credit.history  0.2932335
## 4:                         age.in.years  0.2596514
## 5:             savings.account.and.bonds  0.1960096
```

```r
data.df<-germancredit[,c("creditability","status.of.existing.checking.account",
"duration.in.month",
"credit.history",
"age.in.years",
"savings.account.and.bonds")]
```

Hint: Consider the different primitive data types in R, i.e. numeric (num), factor (Factor), character (chr) and integer (int).

Hint: Consider the different composed data types in R, i.e. vector, matrix, array, data frame (df) and list. Theses types will be indicated in the names of the variables, e.g. data.df is a data frame that contains the data.

0.3 Filtering data and transforming data types

Filtering for missing values, information values and identical values by using the var_filter() function

```r
data_f.df <- var_filter(data.df,
                        "creditability")
```

```
## v Variable filtering on 1000 rows and 5 columns in 00:00:00
## v 0 variables are removed in total
```

Hint: Consider the change of the "creditability" data type from "factor" to "integer"

0.4 Splitting filtered data into train and test samples

Splitting the filtered data frame into a list that contains data frames for the train and the test samples

```r
data_f.list <- split_df(data_f.df,
                        "creditability",
                        ratio = c(0.75,0.25))
class(data_f.list)
```

```
## [1] "list"
```

```r
lapply(data_f.list,class)
```

```
## $train
## [1] "data.table" "data.frame"
##
## $test
## [1] "data.table" "data.frame"
```

```r
lapply(data_f.list, dim)
```

```
## $train
## [1] 727   6
##
## $test
## [1] 273   6
```

```r
str(data_f.list$train)
```

```
## Classes 'data.table' and 'data.frame':   727 obs. of  6 variables:
##  $ status.of.existing.checking.account: Factor w/ 4 levels "... < 0 DM","0 <= ... < 200 DM",..: 1 1 
##  $ duration.in.month                  : num  6 42 24 36 24 12 30 12 15 24 ...
##  $ credit.history                     : Factor w/ 5 levels "no credits taken/ all credits paid back d
##  $ age.in.years                       : num  67 45 53 35 53 61 28 25 28 32 ...
##  $ savings.account.and.bonds          : Factor w/ 5 levels "... < 100 DM",..: 5 1 1 5 3 4 1 1 1 2 ..
##  $ creditability                      : int  0 0 1 0 0 0 1 1 0 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Hint: By default the splitting is 70/30 % for the train/test samples.It can be altered in the split_df() function by the argument "ratio=c(0.7,0.3)". The names of the splitted samples can be altered by the argument "name_dfs=c('train','test')."

Generating a list for response variable: Dummy variable for defaults

```r
#Can be left out
default.list = lapply(data_f.list, function(x) x$creditability)
```

1 Transforming the predictor variables 'x': WOE(x) and GRP(x)

1.1 Weight-Of-Evidence (WOE)-Binning

Grouping (binning) predictor variables with respect to (w.r.t.) the WOE metric

1.1.1 WOE-Binning of total data: bins.df

Grouping all, i.e. total data: Needed for total cross validation purposes

```r
# Weight-Of-Evidence for (WOE)-Binning of total data and train data
bins.df = woebin(data_f.df,
                 "creditability", method = "tree")
```

```
## v Binning on 1000 rows and 6 columns in 00:00:00
```

```r
bins.list = woebin(data_f.list$train,
                   "creditability",
                   save_breaks_list = "breaks.list", method = "tree")
```

```
## v Binning on 727 rows and 6 columns in 00:00:00
```

Hint: The default binning method is method="width". Other methods are "frequ" that support numerical variables as well as "tree" and "chimerge" supporting both, i.e. numerical and categorical variables which are used in the optimal binning approach.

1.1.2 WOE-Binning of train and test data: bins.list

Hint: The breaks list is saved, because it is needed to build a report for the scorecard modeling. As the break list is saved in a separate R-script file, this file has to be loaded and the break list is saved as a variable that can be used in the scorecard modeling report.

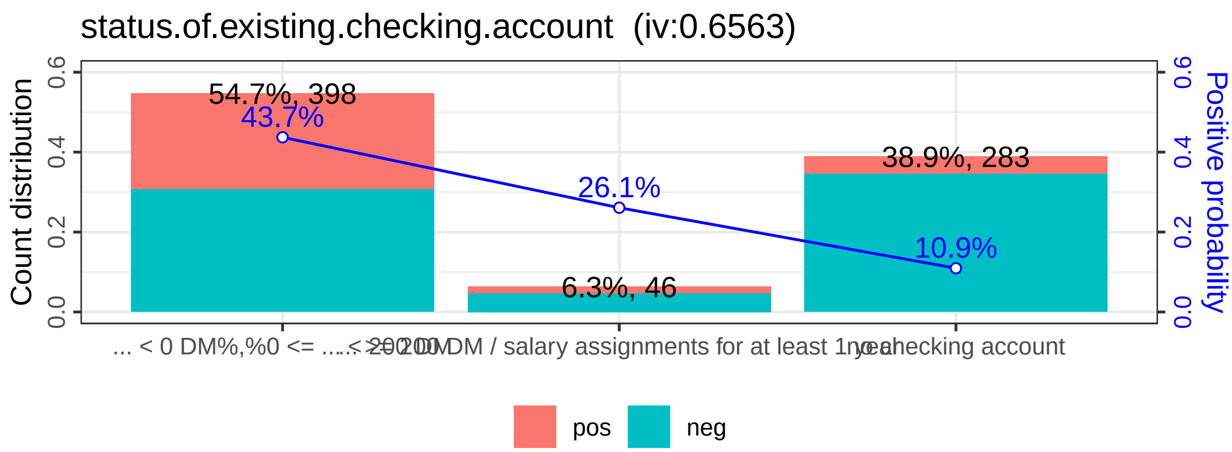
```r
breaks.list=list(
 credit.amount=c("1400", "4000", "5000", "9600"),
 duration.in.month=c("8", "16", "26", "44"),
```

```
credit.history=c("no credits taken/ all credits paid back duly%,%all credits at this bank paid back dul
purpose=c("retraining%,%car (used)%,%radio/television", "furniture/equipment", "repairs%,%car (new)%,%
property=c("real estate", "building society savings agreement/ life insurance", "car or other, not in a
)
```

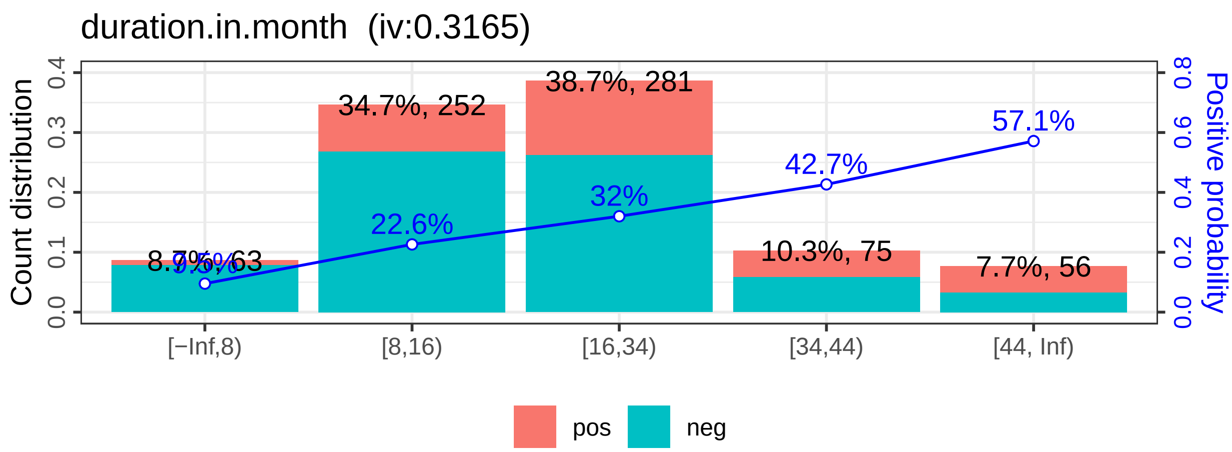
Plotting the bins (including bin statistics)

```
woebin_plot(bins.list)
```

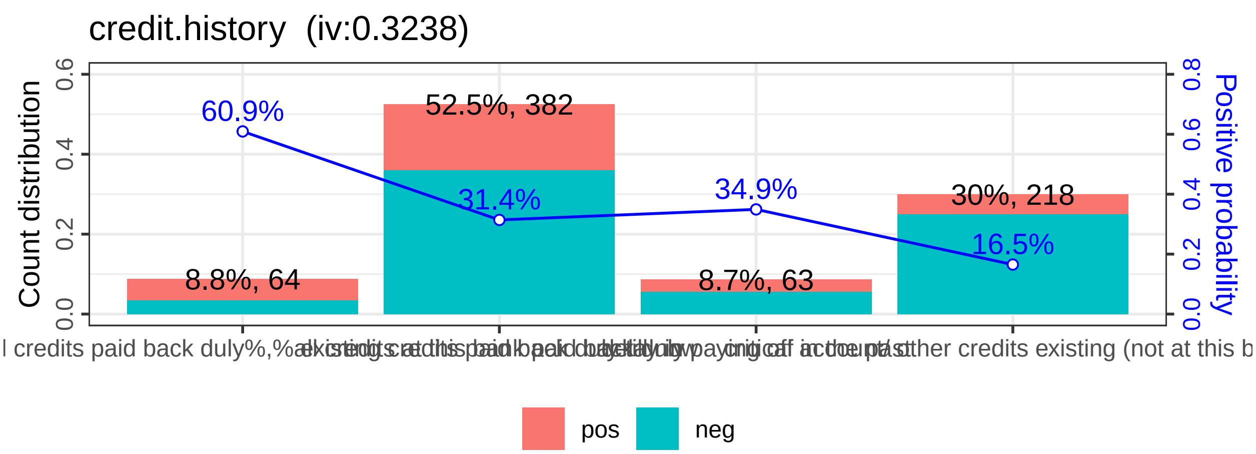
```
## $status.of.existing.checking.account
```



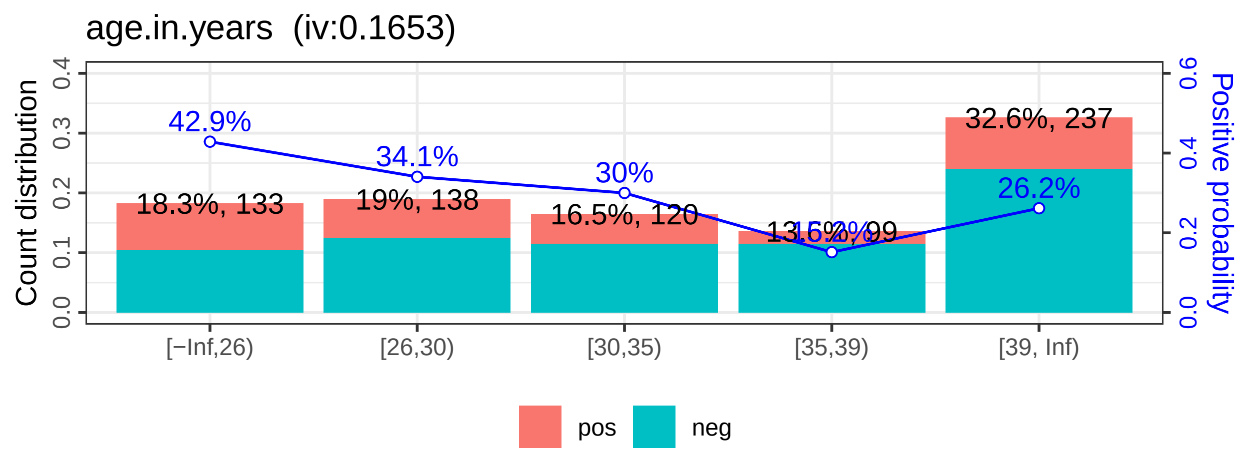

```
##
## $duration.in.month
```



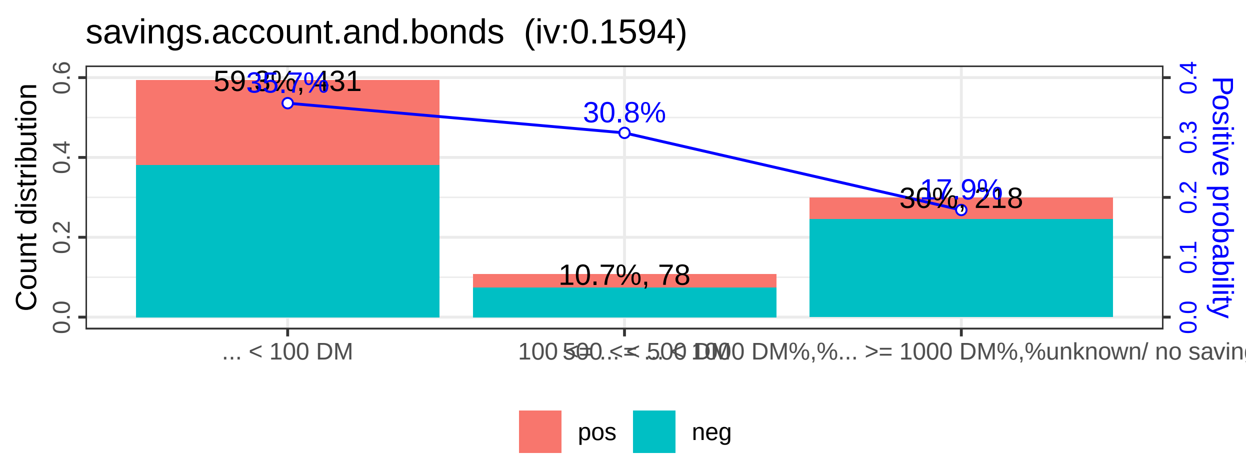

```
##
## $credit.history
```




```
##
## $age.in.years
```




```
##
## $savings.account.and.bonds
```



1.2 Bin-WOE transformation of predictor variables

1.2.1 Bin-WOE transformation of total data: data_woe.df

Transforming total sample: Needed for cross validation purposes

```
data_woe.df <- woebin_ply(data_f.df, bins.df, to ='woe')
```

```
## v Woe transformating on 1000 rows and 5 columns in 00:00:00
```

```
data_grp.df <- woebin_ply(data_f.df, bins.df, to = 'bin')
```

```
## v Woe transformating on 1000 rows and 5 columns in 00:00:00
```

1.2.2 Bin-WOE transformation of train/test data: data_woe.list

Transforming splitted sample: Needed for train/test analysis

```
data_woe.list <- lapply(data_f.list,
                        function(x) woebin_ply(x, bins.list))
```

```
## v Woe transformating on 727 rows and 5 columns in 00:00:00
## v Woe transformating on 273 rows and 5 columns in 00:00:00
```

```
lapply(data_woe.list, class)
```

```
## $train
## [1] "data.table" "data.frame"
##
## $test
## [1] "data.table" "data.frame"
```

```
lapply(data_woe.list, dim)
```

```
## $train
## [1] 727   6
##
## $test
## [1] 273   6
```

```
head(data_woe.list$train[,1:6])
```

```
##    creditability status.of.existing.checking.account_woe duration.in.month_woe
##            <int>                                    <num>                 <num>
## 1:             0                                0.6019226            -1.3967784
## 2:             0                                0.6019226             0.5590492
## 3:             1                                0.6019226             0.1020496
## 4:             0                               -1.2409285             0.5590492
## 5:             0                               -1.2409285             0.1020496
## 6:             0                               -1.2409285            -0.3754349
##    credit.history_woe age.in.years_woe savings.account.and.bonds_woe
##                 <num>            <num>                         <num>
## 1:        -0.76597438       -0.1831382                    -0.6693108
## 2:         0.07366061       -0.1831382                     0.2674485
## 3:         0.23198376       -0.1831382                     0.2674485
## 4:         0.07366061       -0.8682532                    -0.6693108
## 5:         0.07366061       -0.1831382                    -0.6693108
## 6:         0.07366061       -0.1831382                    -0.6693108
```

1.3 Bin-Group (GRP) transformation of predictor variables

```r
data_grp.list = lapply(data_f.list,
                       function(x) woebin_ply(x, bins.list, to = 'bin'))
```

```
## v Woe transformating on 727 rows and 5 columns in 00:00:00
## v Woe transformating on 273 rows and 5 columns in 00:00:00
```

```r
lapply(data_grp.list, class)
```

```
## $train
## [1] "data.table" "data.frame"
##
## $test
## [1] "data.table" "data.frame"
```

```r
lapply(data_grp.list, dim)
```

```
## $train
## [1] 727   6
##
## $test
## [1] 273   6
```

```r
head(data_grp.list$train[,1:3])
```

```
##    creditability status.of.existing.checking.account_bin duration.in.month_bin
##            <int>                                  <char>                <char>
## 1:             0          ... < 0 DM%,%0 <= ... < 200 DM              [-Inf,8)
## 2:             0          ... < 0 DM%,%0 <= ... < 200 DM               [34,44)
## 3:             1          ... < 0 DM%,%0 <= ... < 200 DM               [16,34)
## 4:             0                     no checking account               [34,44)
## 5:             0                     no checking account               [16,34)
## 6:             0                     no checking account                [8,16)
```

```r
head(data_grp.list$train[,4])
```

```
##                                             credit.history_bin
##                                                         <char>
## 1: critical account/ other credits existing (not at this bank)
## 2:                    existing credits paid back duly till now
## 3:                             delay in paying off in the past
## 4:                    existing credits paid back duly till now
## 5:                    existing credits paid back duly till now
## 6:                    existing credits paid back duly till now
```

2 Generalized linear model (glm): Regressing response w.r.t. predictors

2.1 Logistic regression w.r.t. original predictors (data_f.list$train)

2.1.1 Logistic regression: Including original information of numeric, categorical and character predictors

```
data_f.glm <- glm(creditability ~ .,
                  family = binomial(),
                  data = data_f.list$train)
head(data_f.glm$model)
```

```
##   creditability status.of.existing.checking.account duration.in.month
## 1             0                            ... < 0 DM                 6
## 2             0                            ... < 0 DM                42
## 3             1                            ... < 0 DM                24
## 4             0                   no checking account                36
## 5             0                   no checking account                24
## 6             0                   no checking account                12
##                                               credit.history age.in.years
## 1 critical account/ other credits existing (not at this bank)           67
## 2                   existing credits paid back duly till now           45
## 3                            delay in paying off in the past           53
## 4                   existing credits paid back duly till now           35
## 5                   existing credits paid back duly till now           53
## 6                   existing credits paid back duly till now           61
##     savings.account.and.bonds
## 1 unknown/ no savings account
## 2                ... < 100 DM
## 3                ... < 100 DM
## 4 unknown/ no savings account
## 5         500 <= ... < 1000 DM
## 6              ... >= 1000 DM
```

Regression results

```
#summary(data_f.glm)
names(summary(data_f.glm))
```

```
##  [1] "call"           "terms"          "family"         "deviance"
##  [5] "aic"            "contrasts"      "df.residual"    "null.deviance"
##  [9] "df.null"        "iter"           "deviance.resid" "coefficients"
## [13] "aliased"        "dispersion"     "df"             "cov.unscaled"
## [17] "cov.scaled"
```

```
head(data_f.glm$coefficients)
```

```
##                                                                  (Intercept)
##                                                                    1.0761935
##                                       status.of.existing.checking.account0 <= ... < 200 DM
##                                                                   -0.3676314
## status.of.existing.checking.account... >= 200 DM / salary assignments for at least 1 year
##                                                                   -0.7515563
```

```
##                                  status.of.existing.checking.accountno checking account
##                                                                        -1.8341977
##                                                                  duration.in.month
##                                                                         0.0376293
##                                   credit.historyall credits at this bank paid back duly
##                                                                        -0.2651810
```

```
data_f.glm$aic
```

```
## [1] 743.6883
```

Analysis of regression via “analyses of variances”: anova()

```
anova(data_f.glm)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: creditability
##
## Terms added sequentially (first to last)
##
##
##                                      Df Deviance Resid. Df Resid. Dev
## NULL                                                   726     886.32
## status.of.existing.checking.account  3   94.790       723     791.53
## duration.in.month                    1   33.210       722     758.32
## credit.history                       4   24.432       718     733.89
## age.in.years                         1    4.784       717     729.11
## savings.account.and.bonds            4   13.420       713     715.69
```

Analysis of regression via “variance inflation factors”: vif()

```
#vif(data_f.glm, merge_coef = TRUE) - Use if ANOVA with std. deviations is not suitable
```

2.2 Logistic regression w.r.t. WOE-transformed predictors (data_woe.list$train)

The WOE-based logistic regression is the prefered regression approach as it delivers the most compact regression models.

2.2.1 WOE-based logistic regression: Including numeric, categorical and character predictors

```
data_woe.glm <- glm(creditability ~ .,
                    family = binomial(),
                    data = data_woe.list$train)
head(data_woe.glm$model)
```

```
##   creditability status.of.existing.checking.account_woe duration.in.month_woe
## 1             0                               0.6019226            -1.3967784
## 2             0                               0.6019226             0.5590492
## 3             1                               0.6019226             0.1020496
## 4             0                              -1.2409285             0.5590492
## 5             0                              -1.2409285             0.1020496
```

```
## 6             0                              -1.2409285           -0.3754349
##   credit.history_woe age.in.years_woe savings.account.and.bonds_woe
## 1        -0.76597438       -0.1831382                    -0.6693108
## 2         0.07366061       -0.1831382                     0.2674485
## 3         0.23198376       -0.1831382                     0.2674485
## 4         0.07366061       -0.8682532                    -0.6693108
## 5         0.07366061       -0.1831382                    -0.6693108
## 6         0.07366061       -0.1831382                    -0.6693108
```

Summary of regression: summary()

```r
#summary(data_woe.glm)
names(summary(data_woe.glm))
```

```
##  [1] "call"           "terms"          "family"         "deviance"
##  [5] "aic"            "contrasts"      "df.residual"    "null.deviance"
##  [9] "df.null"        "iter"           "deviance.resid" "coefficients"
## [13] "aliased"        "dispersion"     "df"             "cov.unscaled"
## [17] "cov.scaled"
```

```r
data_woe.glm$coefficients
```

```
##                            (Intercept) status.of.existing.checking.account_woe
##                             -0.8551933                               0.8397136
##                     duration.in.month_woe                      credit.history_woe
##                              0.9462763                               0.7612796
##                          age.in.years_woe           savings.account.and.bonds_woe
##                              0.7536934                               0.7532095
```

```r
data_woe.glm$aic
```

```
## [1] 719.3701
```

Analyses of variance: anova()

```r
anova(data_woe.glm)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: creditability
##
## Terms added sequentially (first to last)
##
##
##                                         Df Deviance Resid. Df Resid. Dev
## NULL                                                      726     886.32
## status.of.existing.checking.account_woe  1   92.489       725     793.84
## duration.in.month_woe                    1   39.868       724     753.97
## credit.history_woe                       1   23.016       723     730.95
## age.in.years_woe                         1   13.690       722     717.26
## savings.account.and.bonds_woe            1    9.892       721     707.37
```

Variance inflation factors: vif()

```r
#vif(data_woe.glm, merge_coef = TRUE)
```

2.2.2 WOE-based logistic regression: Automatic selection of the best AIC-model

Automatic selection of optimal regression model via the Akaike Information Criteria (AIC)

```r
#glm_step <- step(data_woe.glm,
#                  direction="both",
#                  trace=FALSE)
#summary(eval(glm_step$call))
```

2.2.3 WOE-based logistic regression: Including numeric predictors only

2.3 Logistic regression w.r.t. GRP-transformed predictors (data_grp.list$train)

2.3.1 GRP-based logistic regression: Including numeric, categorical and character predictors

```r
data_grp.glm <- glm(creditability ~ .,
                    family = binomial(),
                    data = data_grp.list$train)
head(data_grp.glm$model)
```

```
##   creditability status.of.existing.checking.account_bin duration.in.month_bin
## 1             0          ... < 0 DM%,%0 <= ... < 200 DM             [-Inf,8)
## 2             0          ... < 0 DM%,%0 <= ... < 200 DM              [34,44)
## 3             1          ... < 0 DM%,%0 <= ... < 200 DM              [16,34)
## 4             0                     no checking account              [34,44)
## 5             0                     no checking account              [16,34)
## 6             0                     no checking account               [8,16)
##                                        credit.history_bin age.in.years_bin
## 1 critical account/ other credits existing (not at this bank)       [39, Inf)
## 2                  existing credits paid back duly till now       [39, Inf)
## 3                           delay in paying off in the past       [39, Inf)
## 4                  existing credits paid back duly till now         [35,39)
## 5                  existing credits paid back duly till now       [39, Inf)
## 6                  existing credits paid back duly till now       [39, Inf)
##                                     savings.account.and.bonds_bin
## 1 500 <= ... < 1000 DM%,%... >= 1000 DM%,%unknown/ no savings account
## 2                                                       ... < 100 DM
## 3                                                       ... < 100 DM
## 4 500 <= ... < 1000 DM%,%... >= 1000 DM%,%unknown/ no savings account
## 5 500 <= ... < 1000 DM%,%... >= 1000 DM%,%unknown/ no savings account
## 6 500 <= ... < 1000 DM%,%... >= 1000 DM%,%unknown/ no savings account
```

```r
#summary(data_grp.glm)
names(summary(data_grp.glm))
```

```
##  [1] "call"           "terms"          "family"         "deviance"
##  [5] "aic"            "contrasts"      "df.residual"    "null.deviance"
##  [9] "df.null"        "iter"           "deviance.resid" "coefficients"
## [13] "aliased"        "dispersion"     "df"             "cov.unscaled"
## [17] "cov.scaled"
```

```r
head(data_grp.glm$coefficients)
```

```
##                                                                                   (Intercept)
##                                                                                    -1.5142582
## status.of.existing.checking.account_bin... >= 200 DM / salary assignments for at least 1 year
##                                                                                    -0.5275276
##                                   status.of.existing.checking.account_binno checking account
##                                                                                    -1.5586997
##                                                                  duration.in.month_bin[16,34)
##                                                                                     1.4907985
##                                                                  duration.in.month_bin[34,44)
##                                                                                     1.9405486
##                                                                duration.in.month_bin[44, Inf)
##                                                                                     2.3257045
```

```r
data_grp.glm$aic
```

```
## [1] 737.3429
```

```r
anova(data_grp.glm)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: creditability
##
## Terms added sequentially (first to last)
##
##
##                                         Df Deviance Resid. Df Resid. Dev
## NULL                                                      726     886.32
## status.of.existing.checking.account_bin  2   92.489       724     793.84
## duration.in.month_bin                    4   40.292       720     753.54
## credit.history_bin                       3   23.344       717     730.20
## age.in.years_bin                         4   14.252       713     715.95
## savings.account.and.bonds_bin            2   10.604       711     705.34
```

2.3.2 Transforming categorical and character predictors into factors with the one_hot() function

```r
# Do not know yet if needed
data_grp_1h.list<-list()
data_grp_1h.list$train <- one_hot(data_grp.list$train,
                                  var_encode = c("status.of.existing.checking.account_bin",
                                                 "credit.history_bin","duration.in.month_bin")
                                  )
data_grp_1h.list$test <- one_hot(data_grp.list$test,
                                 var_encode = c("status.of.existing.checking.account_bin",
                                                "credit.history_bin","duration.in.month_bin")
                                 )
names(data_grp_1h.list)
```

```
## [1] "train" "test"
```

```r
lapply(data_grp_1h.list,dim)
```

```
## $train
## [1] 727  15
##
## $test
## [1] 273  15
```

3 Building the scorecard-model (scm) and calculating scorepoints

Scorepoints are calculated by combing scorecard-model, which combines bin and glm information, with individual data

3.1 Building the scorecard-model

Building the scorecard via bin and glm information resulting from train sample

```
scorecard.scm <- scorecard(bins.list,
                           data_woe.glm)
names(scorecard.scm)
```

```
## [1] "basepoints"                          "status.of.existing.checking.account"
## [3] "duration.in.month"                   "credit.history"
## [5] "age.in.years"                        "savings.account.and.bonds"
```

Investigating the content of the scorecard

```
scorecard.scm$basepoints
```

```
##      variable    bin    woe points
##        <char> <lgcl> <lgcl>  <num>
## 1: basepoints     NA     NA    449
```

```
scorecard.scm$status.of.existing.checking.account
```

```
##                               variable
##                                 <char>
## 1: status.of.existing.checking.account
## 2: status.of.existing.checking.account
## 3: status.of.existing.checking.account
##                                                       bin count count_distr
##                                                    <char> <int>       <num>
## 1:                          ... < 0 DM%,%0 <= ... < 200 DM   398  0.54745530
## 2: ... >= 200 DM / salary assignments for at least 1 year    46  0.06327373
## 3:                                     no checking account   283  0.38927098
##      neg   pos   posprob        woe      bin_iv  total_iv
##    <int> <int>     <num>      <num>       <num>     <num>
## 1:   224   174 0.4371859  0.6019226 0.218273774 0.6562879
## 2:    34    12 0.2608696 -0.1869405 0.002124977 0.6562879
## 3:   252    31 0.1095406 -1.2409285 0.435889174 0.6562879
##                                                    breaks is_special_values
##                                                    <char>            <lgcl>
## 1:                          ... < 0 DM%,%0 <= ... < 200 DM             FALSE
## 2: ... >= 200 DM / salary assignments for at least 1 year             FALSE
## 3:                                     no checking account             FALSE
##    points
##     <num>
## 1:    -36
## 2:     11
## 3:     75
```

```
scorecard.scm$duration.in.month
```

```
##             variable       bin count count_distr   neg   pos   posprob
```

```
##                <char>    <char> <int>       <num> <int> <int>     <num>
## 1: duration.in.month  [-Inf,8)    63  0.08665750    57     6 0.0952381
## 2: duration.in.month    [8,16)   252  0.34662999   195    57 0.2261905
## 3: duration.in.month   [16,34)   281  0.38651994   191    90 0.3202847
## 4: duration.in.month   [34,44)    75  0.10316369    43    32 0.4266667
## 5: duration.in.month [44, Inf)    56  0.07702889    24    32 0.5714286
##           woe      bin_iv  total_iv breaks is_special_values points
##         <num>       <num>     <num> <char>             <lgcl>  <num>
## 1: -1.3967784 0.117489928 0.3165171      8              FALSE     95
## 2: -0.3754349 0.044932100 0.3165171     16              FALSE     26
## 3:  0.1020496 0.004106144 0.3165171     34              FALSE     -7
## 4:  0.5590492 0.035304912 0.3165171     44              FALSE    -38
## 5:  1.1421954 0.114683977 0.3165171    Inf              FALSE    -78
```

```r
scorecard.scm$credit.history
```

```
##          variable
##            <char>
## 1: credit.history
## 2: credit.history
## 3: credit.history
## 4: credit.history
##                                                                                          bin
##                                                                                       <char>
## 1: no credits taken/ all credits paid back duly%,%all credits at this bank paid back duly
## 2:                                              existing credits paid back duly till now
## 3:                                                       delay in paying off in the past
## 4:                            critical account/ other credits existing (not at this bank)
##    count count_distr   neg   pos   posprob         woe      bin_iv  total_iv
##    <int>       <num> <int> <int>     <num>       <num>       <num>     <num>
## 1:    64  0.08803301    25    39 0.6093750  1.29919919 0.169810394 0.3238461
## 2:   382  0.52544704   262   120 0.3141361  0.07366061 0.002892645 0.3238461
## 3:    63  0.08665750    41    22 0.3492063  0.23198376 0.004869416 0.3238461
## 4:   218  0.29986245   182    36 0.1651376 -0.76597438 0.146273629 0.3238461
##                                                                                       breaks
##                                                                                       <char>
## 1: no credits taken/ all credits paid back duly%,%all credits at this bank paid back duly
## 2:                                              existing credits paid back duly till now
## 3:                                                       delay in paying off in the past
## 4:                            critical account/ other credits existing (not at this bank)
##    is_special_values points
##               <lgcl>  <num>
## 1:             FALSE    -71
## 2:             FALSE     -4
## 3:             FALSE    -13
## 4:             FALSE     42
```

3.2 Calculating the scorepoints by combinig scorecard-model and individual data

Calculating the scorepoints (scores)

```r
score.df = scorecard_ply(data.df,
                         scorecard.scm,
                         only_total_score = FALSE)
```

```r
names(score.df)
```

```
## [1] "status.of.existing.checking.account_points"
## [2] "duration.in.month_points"
## [3] "credit.history_points"
## [4] "age.in.years_points"
## [5] "savings.account.and.bonds_points"
## [6] "score"
```

```r
head(score.df)
```

```
##    status.of.existing.checking.account_points duration.in.month_points
##                                         <num>                    <num>
## 1:                                        -36                       95
## 2:                                        -36                      -78
## 3:                                         75                       26
## 4:                                        -36                      -38
## 5:                                        -36                       -7
## 6:                                         75                      -38
##    credit.history_points age.in.years_points savings.account.and.bonds_points
##                    <num>               <num>                            <num>
## 1:                    42                  10                               36
## 2:                    -4                 -31                              -15
## 3:                    42                  10                              -15
## 4:                    -4                  10                              -15
## 5:                   -13                  10                              -15
## 6:                    -4                  47                               36
##    score
##    <num>
## 1:   596
## 2:   285
## 3:   587
## 4:   366
## 5:   388
## 6:   565
```

3.3 Calculating scorepoints for the splitted sample (train and test)

Generating a score list

```r
score.list <- lapply(data_f.list,
                     function(x) scorecard_ply(x, scorecard.scm))
names(score.list)
```

```
## [1] "train" "test"
```

Hint: The only_total_score = TRUE (= default argument) has to be used for providing two compatible lists for further processing.

3.4 Report (saved spreadsheet format): Scorecard modeling

```r
# {r include=FALSE}
y<-"creditability"
x<-c("status.of.existing.checking.account",
"duration.in.month",
"credit.history",
```

```r
"age.in.years",
"savings.account.and.bonds")
# breaks.list as defined in the woebin()
report(data_f.list,
       y,
       x,
       breaks.list,
       seed = NULL,
       save_report = "Report01")
```

Hint: Generated report file is stored in xlsx-format.
Hint: The R-chunk is not included (i.e. {r include=FALSE}) as otherwise the report would be included in the pdf-file as well.

4 Predicting (forecasting) probabilities and scorepoints

4.1 Probability prediction for the sub-samples (train and test)

```
predProb.list <- lapply(data_woe.list,
                        function(x) predict(data_woe.glm,
                                            type = 'response',
                                            x)
                        )
names(predProb.list)
```

```
## [1] "train" "test"
```

Hint: Due to the fact that the data_woe.glm was calibrated for the train sample two different types of prediction can be destinguished, i.e. the in-sample (IS) prediction by using the train sample in the predict()-function, and the out-of-sample (OoS) prediction by using the test sample in the predict()-function.

```
head(predProb.list$train) # In-Sample prediction
```

```
##          1          2          3          4          5          6
## 0.05231342 0.57413822 0.49670896 0.07794118 0.08418977 0.05527526
```

```
head(predProb.list$test)  # Out-of-Sample prediction
```

```
##          1          2          3          4          5          6
## 0.80468227 0.05884566 0.44581084 0.80468227 0.49493072 0.31585438
```

4.2 Scorepoint prediction for the sub-samples (train and test)

The prediction of the scorepoints is alread incorported in the built scorecard.

```
head(score.list$train)
```

```
##    score
##    <num>
## 1:   596
## 2:   366
## 3:   388
## 4:   565
## 5:   559
## 6:   592
```

```
head(score.list$test)
```

```
##    score
##    <num>
## 1:   285
## 2:   587
## 3:   403
## 4:   285
## 5:   389
## 6:   443
```

5 Analyzing scoring results and testing forecasting accuracy

5.1 Stability of score distributions: Population stability index (PSI)

fig.width=5

```r
#Should be discussed if needed
psi.list <- perf_psi(score = score.list,
                     label = default.list,
                     return_distr_dat = TRUE)
names(psi.list)
```

```
## [1] "pic" "psi" "dat"
```

```r
names(psi.list$dat)
```

```
## [1] "score"
```

```r
head(psi.list$dat$score[,1:9])
```

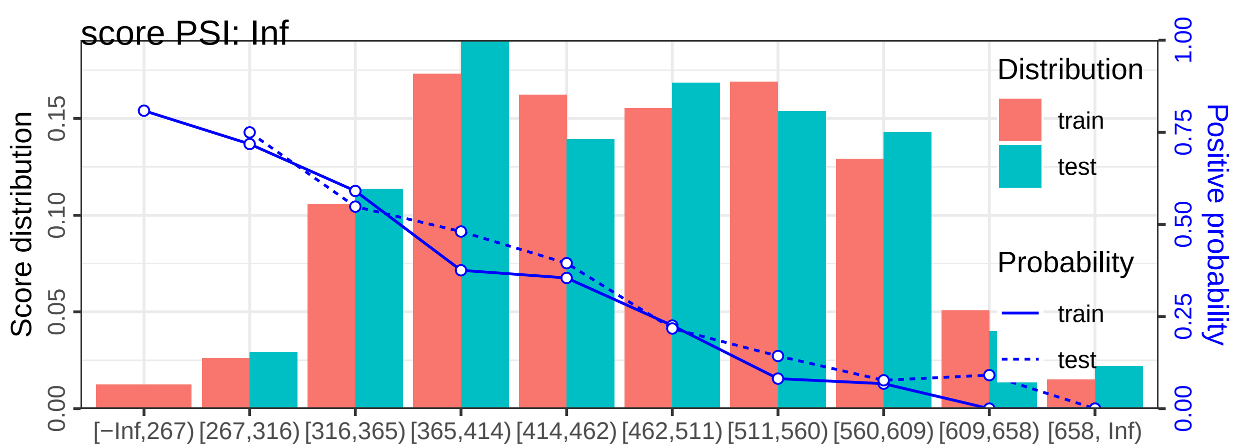
```
## Key: <datset>
##    datset         bin count cum_count   neg   pos cum_neg cum_pos count_distr
##    <fctr>      <fctr> <int>     <int> <int> <int>   <int>   <int>       <num>
## 1:  train [-Inf,267)     9         9     1     8       1       8      0.0124
## 2:  train  [267,316)    19        28     4    15       5      23      0.0261
## 3:  train  [316,365)    77       105    27    50      32      73      0.1059
## 4:  train  [365,414)   126       231    74    52     106     125      0.1733
## 5:  train  [414,462)   118       349    72    46     178     171      0.1623
## 6:  train  [462,511)   113       462    85    28     263     199      0.1554
```

```r
psi.list$psi
```

```
##    variable    dataset   psi
##      <char>     <char> <num>
## 1:    score train_test   Inf
```

```r
psi.list$pic
```

```
## $score
```



5.2 Statistical analysis of scoring system

```r
gains.df <- gains_table(score = score.list,
                        label = default.list,
                        method = "tree")
```

Investigating the gains dataframe

```r
head(gains.df)
```

```
## Key: <datset>
##    datset         bin count cum_count   neg   pos cum_neg cum_pos count_distr
##    <fctr>      <fctr> <int>     <int> <int> <int>   <int>   <int>       <num>
## 1:  train [-Inf,356)    66        66    18    48      18      48      0.0908
## 2:  train  [356,388)    79       145    34    45      52      93      0.1087
## 3:  train  [388,409)    67       212    39    28      91     121      0.0922
## 4:  train  [409,436)    77       289    48    29     139     150      0.1059
## 5:  train  [436,467)    63       352    41    22     180     172      0.0867
## 6:  train  [467,499)    83       435    60    23     240     195      0.1142
##    posprob cum_posprob rejected_rate approval_rate
##      <num>       <num>         <num>         <num>
## 1:  0.7273      0.7273        0.0908        0.9092
## 2:  0.5696      0.6414        0.1994        0.8006
## 3:  0.4179      0.5708        0.2916        0.7084
## 4:  0.3766      0.5190        0.3975        0.6025
## 5:  0.3492      0.4886        0.4842        0.5158
## 6:  0.2771      0.4483        0.5983        0.4017
```

```r
tail(gains.df)
```

```
## Key: <datset>
##    datset         bin count cum_count   neg   pos cum_neg cum_pos count_distr
##    <fctr>      <fctr> <int>     <int> <int> <int>   <int>   <int>       <num>
## 1:   test  [436,467)    25       130    18     7      67      63      0.0916
## 2:   test  [467,499)    39       169    32     7      99      70      0.1429
## 3:   test  [499,531)    18       187    14     4     113      74      0.0659
## 4:   test  [531,559)    26       213    23     3     136      77      0.0952
## 5:   test  [559,592)    28       241    25     3     161      80      0.1026
## 6:   test [592, Inf)    32       273    29     3     190      83      0.1172
##    posprob cum_posprob rejected_rate approval_rate
##      <num>       <num>         <num>         <num>
## 1:  0.2800      0.4846        0.4762        0.5238
## 2:  0.1795      0.4142        0.6190        0.3810
## 3:  0.2222      0.3957        0.6850        0.3150
## 4:  0.1154      0.3615        0.7802        0.2198
## 5:  0.1071      0.3320        0.8828        0.1172
## 6:  0.0938      0.3040        1.0000        0.0000
```

5.3 Cross validation of total and sub-samples

5.3.1 Cross validation w.r.t. total data

```r
cv.list_woe <- perf_cv(data_woe.df,
                  y='creditability',
                  no_folds = 5,
                  binomial_metric = 'gini')
```

```
cv.list_woe
```

```
## $gini
## Key: <datset>
##    datset     train validation
##    <char>     <num>      <num>
## 1:      1 0.5755396  0.5514096
## 2:      2 0.5554299  0.6269250
## 3:      3 0.5749480  0.5482914
## 4:      4 0.5738966  0.5490395
## 5:      5 0.5781260  0.5091701
```

```
cv.list_grp <- perf_cv(data_grp.df,
                   y='creditability',
                   no_folds = 5,
                   binomial_metric = 'gini')
cv.list_grp
```

```
## $gini
## Key: <datset>
##    datset     train validation
##    <char>     <num>      <num>
## 1:      1 0.5802946  0.5228027
## 2:      2 0.5635628  0.5966220
## 3:      3 0.5755175  0.5335912
## 4:      4 0.5865409  0.5164972
## 5:      5 0.5857871  0.4672120
```

5.3.2 Cross validation w.r.t. train data

```
cv.list_woe_train <- perf_cv(data_woe.list$train,
                   y='creditability',
                   no_folds = 5,
                   binomial_metric = 'gini')
cv.list_woe_train
```

```
## $gini
## Key: <datset>
##    datset     train validation
##    <char>     <num>      <num>
## 1:      1 0.6017153  0.5244856
## 2:      2 0.5972931  0.5323068
## 3:      3 0.6000276  0.5298246
## 4:      4 0.5721552  0.6435518
## 5:      5 0.5781943  0.6214354
```

```
cv.list_grp_train <- perf_cv(data_grp.list$train,
                   y='creditability',
                   no_folds = 5,
                   binomial_metric = 'gini')
cv.list_grp_train
```

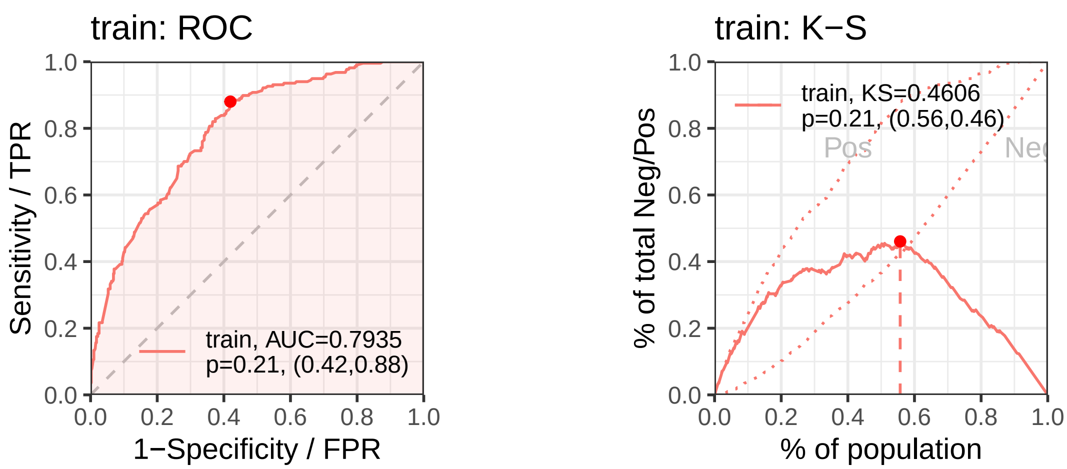
```
## $gini
## Key: <datset>
##    datset     train validation
```

```
##     <char>      <num>       <num>
## 1:       1 0.6112027  0.4927984
## 2:       2 0.5965793  0.5033213
## 3:       3 0.6103731  0.4842105
## 4:       4 0.5809770  0.6063425
## 5:       5 0.5815261  0.5954067
```

5.4 Probability prediction accuracy (single dataset): In-Sample and Out-of-Sample testing

5.4.1 IS-testing of predicted probabilities via train sample

```r
#I guess it can be removed but still keeping it in the first version
perf_eva(pred = predProb.list$train,
         label = data_woe.list$train$creditability,
         title = 'train',
         show_plot=c("roc","ks"),
         confusion_matrix = TRUE)
```

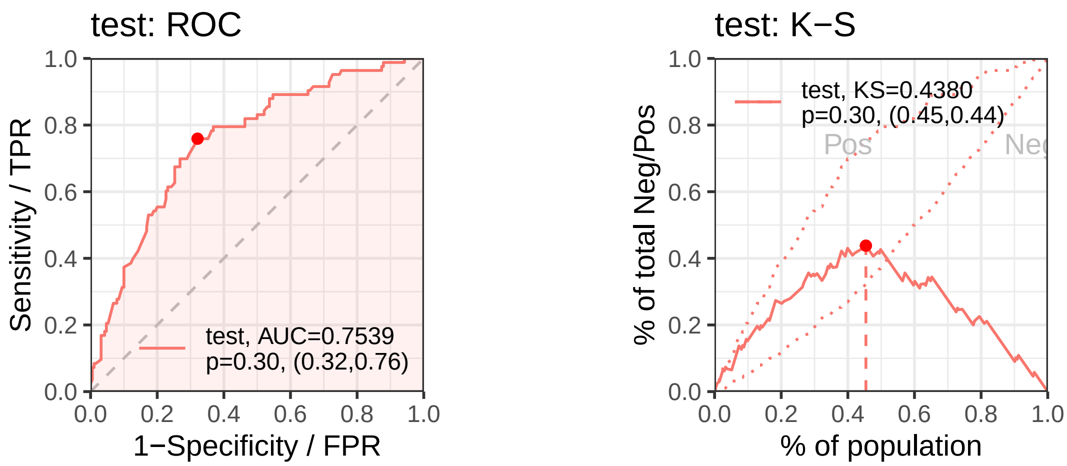



```
## $binomial_metric
## $binomial_metric$train
##          MSE      RMSE   LogLoss        R2        KS       AUC      Gini
##        <num>     <num>     <num>     <num>     <num>     <num>     <num>
## 1: 0.1620372 0.4025385 0.4864994 0.2261554 0.4605765 0.7935077 0.5870155
##
##
## $confusion_matrix
## $confusion_matrix$train
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    296    214 0.4196078
## 2:      1     26    191 0.1198157
## 3:  total    322    405 0.3301238
##
##
## $pic
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
```

```
## 2 2 (1-1,2-2) arrange gtable[layout]
```

```
### OoS-testing of predicted probabilities via test sample

perf_eva(pred = predProb.list$test,
         label = data_woe.list$test$creditability,
         title = 'test',
         show_plot=c("roc","ks"),
         confusion_matrix = TRUE)
```



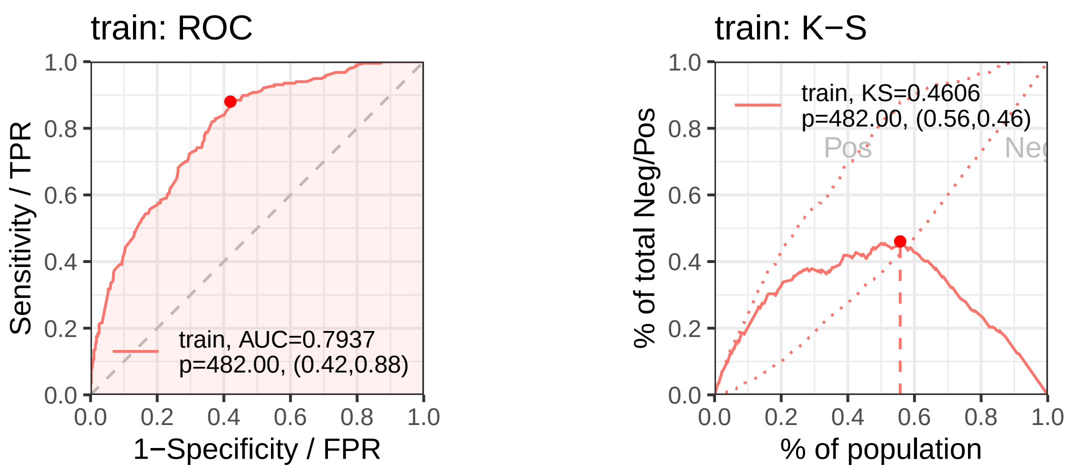

```
## $binomial_metric
## $binomial_metric$test
##          MSE      RMSE   LogLoss        R2        KS       AUC     Gini
##        <num>     <num>     <num>     <num>     <num>     <num>    <num>
## 1: 0.1760701 0.4196071 0.5286325 0.1678927 0.4379835 0.7539315 0.507863
##
##
## $confusion_matrix
## $confusion_matrix$test
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    129     61 0.3210526
## 2:      1     20     63 0.2409639
## 3:  total    149    124 0.2967033
##
##
## $pic
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name          grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

```
## Scorepoint prediction accuracy (single dataset): In-Sample and Out-of-Sample testing

### IS-testing of predicted scores via train sample

perf_eva(pred = score.list$train,
         label = data_f.list$train$creditability,
         title = 'train',
         binomial_metric = "gini",
```

```r
         show_plot=c("roc","ks"),
         confusion_matrix = TRUE)
```

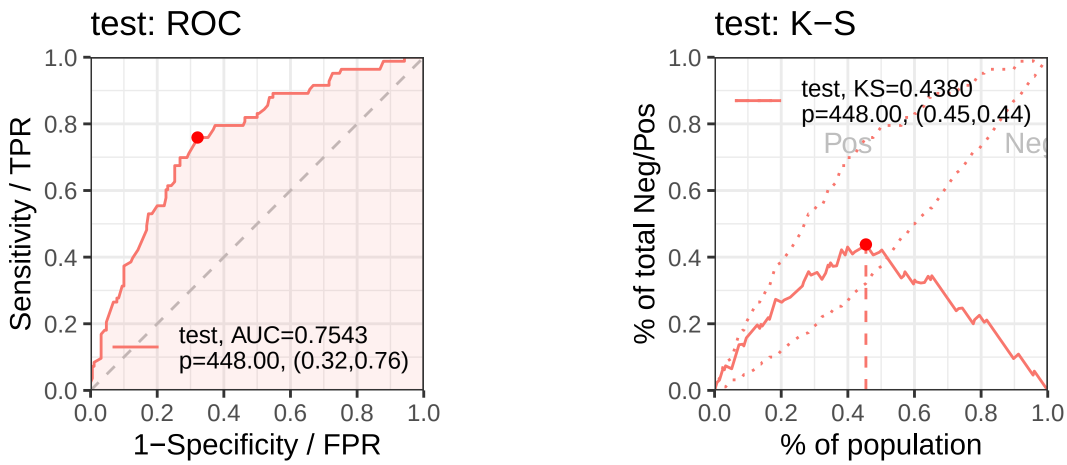



```
## $binomial_metric
## $binomial_metric$train
##         Gini
##        <num>
## 1: 0.5873227
##
##
## $confusion_matrix
## $confusion_matrix$train
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    296    214 0.4196078
## 2:      1     26    191 0.1198157
## 3:  total    322    405 0.3301238
##
##
## $pic
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

```r
#Hint: Threshold of confusion matrix relates to the scores.

### OoS-testing of predicted scores via test sample

perf_eva(pred = score.list$test,
         label = data_f.list$test$creditability,
         title = 'test',
         binomial_metric = "gini",
         show_plot=c("roc","ks"),
         confusion_matrix = TRUE)
```

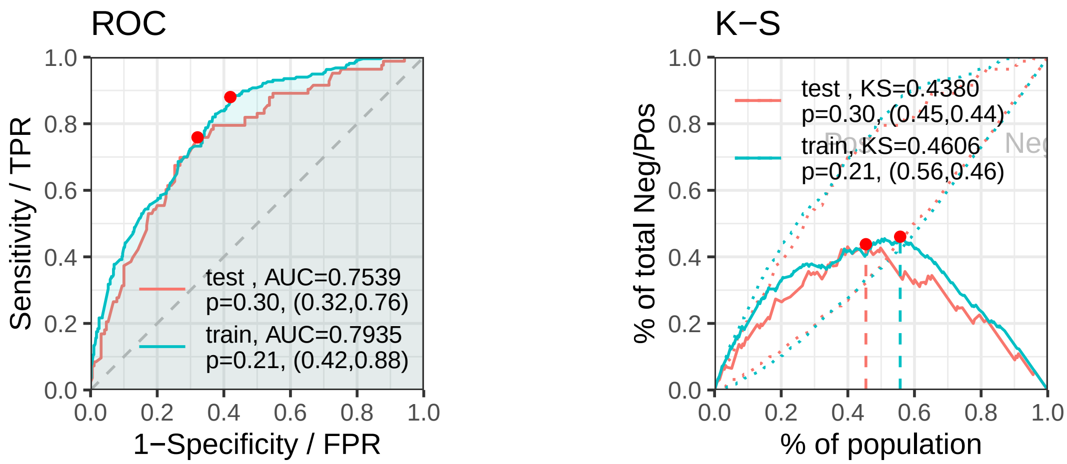



```
## $binomial_metric
## $binomial_metric$test
##        Gini
##       <num>
## 1: 0.508624
##
##
## $confusion_matrix
## $confusion_matrix$test
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    129     61 0.3210526
## 2:      1     20     63 0.2409639
## 3:  total    149    124 0.2967033
##
##
## $pic
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

5.5 Prediction accuracy (multiple dataset): IS- and OoS-Testing in one

Predicted probabilities

```r
perf_eva(pred = predProb.list,
         label = default.list,
         binomial_metric = "gini",
         show_plot=c("roc","ks"),
         confusion_matrix = TRUE)
```

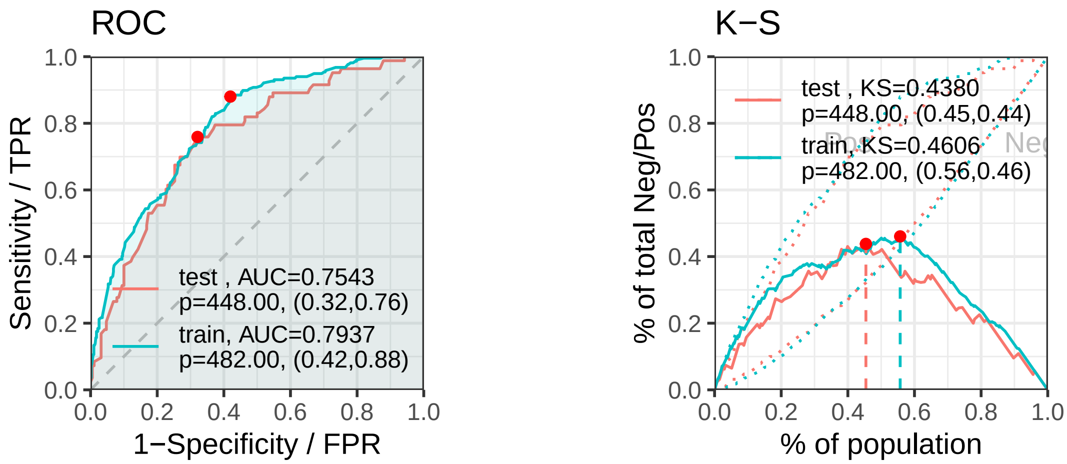



```
## $binomial_metric
## $binomial_metric$train
##         Gini
##        <num>
## 1: 0.5870155
##
## $binomial_metric$test
##        Gini
##       <num>
## 1: 0.507863
##
##
## $confusion_matrix
## $confusion_matrix$train
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    296    214 0.4196078
## 2:      1     26    191 0.1198157
## 3:  total    322    405 0.3301238
##
## $confusion_matrix$test
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    102     88 0.4631579
## 2:      1     15     68 0.1807229
## 3:  total    117    156 0.3772894
##
##
## $pic
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

Predicted scores

```
perf_eva(pred = score.list,
         label = default.list,
         binomial_metric = "gini",
```

```r
        show_plot=c("roc","ks"),
        confusion_matrix = TRUE)
```




```
## $binomial_metric
## $binomial_metric$train
##         Gini
##        <num>
## 1: 0.5873227
##
## $binomial_metric$test
##        Gini
##       <num>
## 1: 0.508624
##
##
## $confusion_matrix
## $confusion_matrix$train
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    296    214 0.4196078
## 2:      1     26    191 0.1198157
## 3:  total    322    405 0.3301238
##
## $confusion_matrix$test
##     label pred_0 pred_1     error
##    <char>  <num>  <num>     <num>
## 1:      0    102     88 0.4631579
## 2:      1     15     68 0.1807229
## 3:  total    117    156 0.3772894
##
##
## $pic
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

#Summary