LiteXploreR Demo

Terence Then

Version 1.0 December 31, 2017

Contents

1 How to Download this Package

As {LiteXploreR} is hosted on GitHub, you will need to use the function install_github() for the package {devtools} to install it.

To do that, you need to run the following codes in your R console:

"install.packages("devtools")" "library(devtools)" "install_github("Then-Terence/LiteXploreR")"

2 Setting Up the Workspace

First of all, I am going to load the package {LiteXploreR} and load a credit dataset obtained from Kaggle. I have split them into training, validation and testing set.

```
# Load the Package
library(LiteXploreR)
```

```
## Loading required package: data.table
```

```
## Warning: package 'data.table' was built under R version 3.4.3
```

```
# Load the Data
load("../Output/PartitionedData.RData")
```

This is how the workspace looks like:

```
ls()
```

```
## [1] "Test"  "Train" "Valid"
```

Let us start by looking at the training set.

```
head(Train)
```

```
##    Delinquency Utilization Age Due30    DebtRatio MonthlyIncome
## 1:           1 0.301491625  65     1    1.3824301          4575
## 2:           0 1.059940060  48     7 3620.0000000            NA
```

```
## 3:           0 0.155707651  39     0    0.5136848          3470
## 4:           0 0.172388507  42     0    0.4079970          2700
## 5:           0 0.620568038  39     0    0.3404613         10100
## 6:           0 0.009339405  95     0   24.0000000            NA
##    CreditAndLoans Due90 RealEstateLines Due60 Dependents
## 1:             20     0               2     1          0
## 2:             18     0               3     0          0
## 3:             15     0               1     0          2
## 4:              8     0               1     0          1
## 5:             11     0               1     0          4
## 6:              3     0               0     0          0
```

3 Demonstration of Functions

3.1 Overview of a Dataset: CovariateSummary()

In cases where the number of columns/ covariates is overwhelming, they can be screened out using the function CovariateSummary, which gives the AUC value of the raw covariates.

```
Overview <- CovariateSummary(Target = "Delinquency", Data = Train, Type = "Binary")
Overview
```

```
##           Covariate    Type Count of NA Count of 0    AUC
##  1:     Utilization numeric           0       6471  0.778
##  2:             Age integer           0          1 0.6337
##  3:           Due30 integer           0      75610 0.6904
##  4:       DebtRatio numeric           0       2443 0.5257
##  5:   MonthlyIncome integer       17789        976 0.5769
##  6:  CreditAndLoans integer           0       1143 0.5459
##  7:           Due90 integer           0      84985 0.6569
##  8: RealEstateLines integer           0      33796 0.5348
##  9:           Due60 integer           0      85454 0.6221
## 10:      Dependents integer        2353      52008 0.5472
```

The column "Type" serves as a check for the classes of the columns, i.e. whether they are imported correctly.

For the purpose of this demonstration, I will only use the covariates "Age", "Due30", "Due60", "Due90" in the following sections.

3.2 Exploring and Transforming a Numerical Covariate: NumericalTable(), CategorialTable() and GenerateLogit()

3.2.1 Covariate "Age""

First thing first, there is one observation with the age of 0. I will replace it with the second lowest value of age.

```
Train[Age == 0, Age := 21]
```

Here, a table is constructed by dividing "Age" into 20 categories of roughly equal sizes.

```
AgeTable <- NumericalTable(Target = "Delinquency", Covariate = "Age",
                           Data = Train, NumberOfBins = 20)
```

The table looks something like this:

```
AgeTable
```

```
##          Age Event Non Event Counts Probability       Logit
##  1:  [21,29]   612      4606   5218      0.1173  0.61426380
##  2:  (29,33]   551      4431   4982      0.1106  0.54800083
##  3:  (33,36]   411      3636   4047      0.1016  0.45260035
##  4:  (36,39]   431      4359   4790      0.0900  0.31875673
##  5:  (39,41]   331      3437   3768      0.0878  0.29241069
##  6:  (41,44]   488      5228   5716      0.0854  0.26117793
##  7:  (44,46]   365      3950   4315      0.0846  0.25107309
##  8:  (46,48]   359      4221   4580      0.0784  0.16814164
##  9:  (48,50]   353      4208   4561      0.0774  0.15437190
## 10:  (50,52]   335      4055   4390      0.0763  0.13907116
## 11:  (52,54]   319      4018   4337      0.0736  0.09929815
## 12:  (54,56]   257      3973   4230      0.0608 -0.10555408
## 13:  (56,58]   212      3880   4092      0.0518 -0.27435756
## 14:  (58,61]   303      5719   6022      0.0503 -0.30516984
## 15:  (61,63]   176      4225   4401      0.0400 -0.54564395
## 16:  (63,65]   111      3304   3415      0.0325 -0.76071234
## 17:  (65,68]   121      4250   4371      0.0277 -0.92623712
## 18:  (68,72]   113      4036   4149      0.0272 -0.94297497
## 19:  (72,78]   105      4430   4535      0.0232 -1.10954792
## 20: (78,109]    83      3998   4081      0.0203 -1.24206231
```

The cutoffs can be obtained from the function quantile() in base R. Its minimum and maximum are replaced by 0 and Inf respectively.

```
AgeBreaks <- quantile(Train[, Age], probs = seq(0, 1, 0.05))
AgeBreaks <- c(0, AgeBreaks[-c(1, 21)], Inf)
```

Then, the table is updated with the thresholds, the covariate is discretized, and the logit scores are extracted from the table.

```
AgeTable <- NumericalTable("Delinquency", "Age", Train,
                           CustomBins = T,
                           CustomIntervals = AgeBreaks)
Train[, AgeCat := cut(Age, breaks = AgeBreaks, include.lowest = T)]
Train <- GenerateLogit(Data = Train, CrossTable = AgeTable, Covariate = "AgeCat")
```

The function generates another column with the logit scores and will be named by appending the original column name followed by “Logit”.

In this case, a new column named “AgeCatLogit” is generated.

```
head(unique(Train[, c("AgeCat", "AgeCatLogit")]))
```

```
##     AgeCat AgeCatLogit
## 1:  [0,29]   0.6142638
## 2: (29,33]   0.5480008
## 3: (33,36]   0.4526004
## 4: (36,39]   0.3187567
## 5: (39,41]   0.2924107
## 6: (41,44]   0.2611779
```

3.2.2 Covariate "Due30"

While the covariate "Due30" is numerical, it may be helpful for treating it as categorical in the first inspection. This is due to it having a limited number of unique values, and not being able to split into equal categories as most of the borrowers will have the value of 0.

```
Due30Table <- CategoricalTable(Target = "Delinquency", Covariate = "Due30",
                               Data = Train)
Due30Table
```

```
##     Due30 Event Non Event Counts Probability      Logit
##  1:     0  3025     72585  75610      0.0400 -0.5452006
##  2:     1  1459      8168   9627      0.1516  0.9101738
##  3:     2   733      2021   2754      0.2662  1.6184446
##  4:     3   383       685   1068      0.3586  2.0512627
##  5:     4   191       247    438      0.4361  2.3755317
##  6:     5    90       110    200      0.4500  2.4319759
##  7:     6    46        44     90      0.5111  2.6770984
##  8:     7    14        16     30      0.4667  2.4991152
##  9:     8     4        12     16      0.2500  1.5340343
## 10:     9     3         4      7      0.4286  2.3449645
## 11:    10     3         0      3      1.0000         Inf
## 12:    12     1         1      2      0.5000  2.6326466
## 13:    13     1         0      1      1.0000         Inf
## 14:    96     2         1      3      0.6667  3.3257938
## 15:    98    81        70    151      0.5364  2.7786005
```

There are very little observations having values greater than 5. As such, they will be grouped together.

The exceptions are 96 and 98, which may indicate a special status, they will be grouped separately.

Then, the table is updated.

```
Due30Breaks <- c(-Inf, 0:4, 95, Inf)
Due30Table <- NumericalTable(Target = "Delinquency", Covariate = "Due30",
                             Data = Train, CustomBins = T,
                             CustomIntervals = Due30Breaks)
Due30Table
```

```
##        Due30 Event Non Event Counts Probability      Logit
## 1:  [-Inf,0]  3025     72585  75610      0.0400 -0.5452006
## 2:     (0,1]  1459      8168   9627      0.1516  0.9101738
## 3:     (1,2]   733      2021   2754      0.2662  1.6184446
## 4:     (2,3]   383       685   1068      0.3586  2.0512627
## 5:     (3,4]   191       247    438      0.4361  2.3755317
## 6:    (4,95]   162       187    349      0.4642  2.4891343
## 7: (95, Inf]    83        71    154      0.5390  2.7888073
```

The covariate "Due30" is discretized and the logit scores are extracted.

```
Train[, Due30Cat := cut(Due30, breaks = Due30Breaks, include.lowest = T)]
Train <- GenerateLogit(Data = Train, CrossTable = Due30Table,
                       Covariate = "Due30Cat")
```

```
head(unique(Train[, c("Due30Cat", "Due30CatLogit")]))
```

```
##    Due30Cat Due30CatLogit
## 1: [-Inf,0]    -0.5452006
## 2:    (0,1]     0.9101738
```

```
## 3:    (1,2]     1.6184446
## 4:    (2,3]     2.0512627
## 5:    (3,4]     2.3755317
## 6:   (4,95]     2.4891343
```

3.2.3 Covariate "Due60"

The same procedures are repeated for the covariates "Due60" and "Due90". Feel free to skip this part and the next.

```
Due60Table <- CategoricalTable(Target = "Delinquency", Covariate = "Due60",
                               Data = Train)
Due60Table
```

```
##     Due60 Event Non Event Counts Probability      Logit
##  1:     0  4366     81088  85454      0.0511 -0.2890411
##  2:     1  1065      2360   3425      0.3109  1.8369598
##  3:     2   362       324    686      0.5277  2.7435473
##  4:     3   102        79    181      0.5635  2.8881716
##  5:     4    32        29     61      0.5246  2.7310867
##  6:     5    16         9     25      0.6400  3.2080107
##  7:     6     7         2      9      0.7778  3.8854096
##  8:     7     2         1      3      0.6667  3.3257938
##  9:     8     0         1      1      0.0000       -Inf
## 10:    11     1         0      1      1.0000        Inf
## 11:    96     2         1      3      0.6667  3.3257938
## 12:    98    81        70    151      0.5364  2.7786005
```

There are very little observations having values greater than 2. As such, they will be grouped together. Again, 96 and 98 will be grouped together separately.

Then, the table is updated.

```
Due60Breaks <- c(-Inf, 0:2, 95, Inf)
Due60Table <- NumericalTable(Target = "Delinquency", Covariate = "Due60",
                             Data = Train, CustomBins = T,
                             CustomIntervals = Due60Breaks)
Due60Table
```

```
##        Due60 Event Non Event Counts Probability      Logit
## 1:  [-Inf,0]  4366     81088  85454      0.0511 -0.2890411
## 2:     (0,1]  1065      2360   3425      0.3109  1.8369598
## 3:     (1,2]   362       324    686      0.5277  2.7435473
## 4:    (2,95]   160       121    281      0.5694  2.9120299
## 5: (95, Inf]    83        71    154      0.5390  2.7888073
```

The covariate "Due60" is discretized and the logit scores are extracted.

```
Train[, Due60Cat := cut(Due60, breaks = Due60Breaks, include.lowest = T)]
Train <- GenerateLogit(Data = Train, CrossTable = Due60Table,
                       Covariate = "Due60Cat")
```

```
head(unique(Train[, c("Due60Cat", "Due60CatLogit")]))
```

```
##     Due60Cat Due60CatLogit
## 1:  [-Inf,0]    -0.2890411
## 2:     (0,1]     1.8369598
## 3:     (1,2]     2.7435473
```

```
## 4:    (2,95]     2.9120299
## 5: (95, Inf]     2.7888073
```

3.2.4 Covariate "Due90"

```
Due90Table <- CategoricalTable(Target = "Delinquency", Covariate = "Due90",
                               Data = Train)
Due90Table
```

```
##     Due90 Event Non Event Counts Probability      Logit
##  1:     0  3948     81037  84985      0.0465 -0.3890501
##  2:     1  1049      2100   3149      0.3331  1.9385466
##  3:     2   475       470    945      0.5026  2.6432287
##  4:     3   244       178    422      0.5782  2.9480313
##  5:     4   115        47    162      0.7099  3.5274311
##  6:     5    53        23     76      0.6974  3.4674443
##  7:     6    30        21     51      0.5882  2.9893215
##  8:     7    15         5     20      0.7500  3.7312589
##  9:     8     9         2     11      0.8182  4.1367240
## 10:     9     9         1     10      0.9000  4.8298712
## 11:    10     3         3      6      0.5000  2.6326466
## 12:    11     1         2      3      0.3333  1.9394994
## 13:    12     0         1      1      0.0000       -Inf
## 14:    13     0         1      1      0.0000       -Inf
## 15:    14     1         1      2      0.5000  2.6326466
## 16:    15     0         1      1      0.0000       -Inf
## 17:    17     1         0      1      1.0000        Inf
## 18:    96     2         1      3      0.6667  3.3257938
## 19:    98    81        70    151      0.5364  2.7786005
```

There are values greater than 3 are grouped together, with the exceptions of 96 and 98.

```
Due90Breaks <- c(-Inf, 0:3, 95, Inf)
Due90Table <- NumericalTable(Target = "Delinquency", Covariate = "Due90",
                             Data = Train, CustomBins = T,
                             CustomIntervals = Due90Breaks)
Due90Table
```

```
##        Due90 Event Non Event Counts Probability      Logit
## 1:  [-Inf,0]  3948     81037  84985      0.0465 -0.3890501
## 2:     (0,1]  1049      2100   3149      0.3331  1.9385466
## 3:     (1,2]   475       470    945      0.5026  2.6432287
## 4:     (2,3]   244       178    422      0.5782  2.9480313
## 5:    (3,95]   237       108    345      0.6870  3.4185755
## 6: (95, Inf]    83        71    154      0.5390  2.7888073
```

The covariate "Due90" is discretized and the logit scores are extracted.

```
Train[, Due90Cat := cut(Due90, breaks = Due90Breaks, include.lowest = T)]
Train <- GenerateLogit(Data = Train, CrossTable = Due90Table,
                       Covariate = "Due90Cat")
```

```
head(unique(Train[, c("Due90Cat", "Due90CatLogit")]))
```

```
##     Due90Cat Due90CatLogit
## 1:  [-Inf,0]    -0.3890501
```

```
## 2:     (0,1]     1.9385466
## 3:     (1,2]     2.6432287
## 4:     (2,3]     2.9480313
## 5:    (3,95]     3.4185755
## 6: (95, Inf]     2.7888073
```

3.3 Assessing the Performance of a Model: AUROC(), CovariateWeights(), and LogLoss()

Before model building, it is a standard practice to standardize the dataset in the field of credit scoring.

I have simplified the dataset to only include the target "Delinquency" and all the logit scores, i.e. columns with the term "Logit".

```
Train <- data.table(cbind(Train[, Delinquency]),
                    scale(Train[, grepl("Logit", names(Train)),
                                  with = F]))
setnames(Train, "V1", "Delinquency")
```

Over here, I have built a simple model using logistic regression, employing just the four covariates listed above.

```
ScoringModel <- glm(Delinquency ~ AgeCatLogit + Due30CatLogit + Due60CatLogit +
                      Due90CatLogit, data = Train, family = binomial())
```

While the relative importance of the covariates can be obtained from the coefficients using summary(), it can be a process that gives a bit of hassle.

```
summary(ScoringModel)
```

```
##
## Call:
## glm(formula = Delinquency ~ AgeCatLogit + Due30CatLogit + Due60CatLogit +
##     Due90CatLogit, family = binomial(), data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3888  -0.3120  -0.2780  -0.1992   2.9277
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.093464   0.018001 -171.85   <2e-16 ***
## AgeCatLogit    0.421514   0.018340   22.98   <2e-16 ***
## Due30CatLogit  0.436852   0.011075   39.45   <2e-16 ***
## Due60CatLogit  0.235644   0.009549   24.68   <2e-16 ***
## Due90CatLogit  0.422884   0.008825   47.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 44277  on 89999  degrees of freedom
## Residual deviance: 35019  on 89995  degrees of freedom
## AIC: 35029
##
## Number of Fisher Scoring iterations: 6
```

The function CovariateWeights() can be used to provide the relative importance of the covariates, from a scale of 0 to 100.

```
CovariateWeights(ScoringModel)
```

```
##   AgeCatLogit Due30CatLogit Due60CatLogit Due90CatLogit
##         27.79         28.80         15.53         27.88
```

Other than that, the function AUROC() computes the area under the curve, much faster than auc() from the package pROC(). Compared to pROC::auc(), the downside of using LiteXploreR::AUROC() is that it only provides the area under the curve itself, rather than returning a list of the details on the computation.

```
AUROC(Train[, Delinquency], ScoringModel$fitted.values)
```

```
## [1] 0.8163974
```

```
auc(Train[, Delinquency], ScoringModel$fitted.values)
```

```
## Area under the curve: 0.8164
```

```
system.time(AUROC(Train[, Delinquency], ScoringModel$fitted.values))
```

```
##    user  system elapsed
##    0.01    0.00    0.02
```

```
system.time(auc(Train[, Delinquency], ScoringModel$fitted.values))
```

```
##    user  system elapsed
##    0.78    0.12    0.91
```

In addition, a function for logarithmic loss is included in this package as well. This is another commonly used function not covered by base R. The function LogLoss() in this package only supports binary predictions, as opposed to predictions involving more than two classes.

```
LogLoss(Train[, Delinquency], ScoringModel$fitted.values)
```

```
## [1] 0.1945473
```