# Logistic Regression Method in R

Gabriel Miranda

```
# Packages used

require(caret)
require(mlr)
```

```
# Everything I create, in my profile, in R I'll try to create as equal as possible in Python.

# Data from: https://www.kaggle.com/kabure/german-credit-data-with-risk
```

```
df = read.csv('german_credit_risk_target.csv')

head(df, n = 10)
```

```
##    X Age    Sex Job Housing Saving.accounts Checking.account Credit.amount
## 1  0  67   male   2     own            <NA>           little          1169
## 2  1  22 female   2     own          little         moderate          5951
## 3  2  49   male   1     own          little             <NA>          2096
## 4  3  45   male   2    free          little           little          7882
## 5  4  53   male   2    free          little           little          4870
## 6  5  35   male   1    free            <NA>             <NA>          9055
## 7  6  53   male   2     own      quite rich             <NA>          2835
## 8  7  35   male   3    rent          little         moderate          6948
## 9  8  61   male   1     own            rich             <NA>          3059
## 10 9  28   male   3     own          little         moderate          5234
##    Duration             Purpose Risk
## 1         6            radio/TV good
## 2        48            radio/TV  bad
## 3        12           education good
## 4        42 furniture/equipment good
## 5        24                 car  bad
## 6        36           education good
## 7        24 furniture/equipment good
## 8        36                 car good
## 9        12            radio/TV good
## 10       30                 car  bad
```

```
# Pre-processing

df$X = NULL

df$Sex = factor(df$Sex, levels = c('male', 'female'), labels = c(1, 2))
df$Housing = factor(df$Housing, levels = c('free', 'own', 'rent'), labels = c(1, 2, 3))
df$Saving.accounts = factor(df$Saving.accounts, levels = c('little', 'moderate', 'quite rich',
                                                           'rich'), labels = c(1, 2, 3, 4))
df$Checking.account = factor(df$Checking.account, levels = c('little', 'moderate', 'rich'),
                             labels = c(1, 2, 3))
df$Purpose = factor(df$Purpose, levels = c('radio/TV', 'education', 'furniture/equipment',
                                           'car', 'business', 'domestic appliances',
                                           'repairs', 'vacation/others'),
                    labels = c(1, 2, 3, 4, 5, 6, 7, 8))
df$Risk = factor(df$Risk, levels = c('bad', 'good'), labels = c(1, 2))

head(df, n = 10)
```

```
##    Age Sex Job Housing Saving.accounts Checking.account Credit.amount Duration
## 1   67   1   2       2            <NA>                1          1169        6
## 2   22   2   2       2               1                2          5951       48
## 3   49   1   1       2               1             <NA>          2096       12
## 4   45   1   2       1               1                1          7882       42
## 5   53   1   2       1               1                1          4870       24
## 6   35   1   1       1            <NA>             <NA>          9055       36
## 7   53   1   2       2               3             <NA>          2835       24
## 8   35   1   3       3               1                2          6948       36
## 9   61   1   1       2               4             <NA>          3059       12
## 10  28   1   3       2               1                2          5234       30
##    Purpose Risk
## 1        1    2
## 2        1    1
## 3        2    2
## 4        3    2
## 5        4    1
## 6        2    2
## 7        3    2
## 8        4    2
## 9        1    2
## 10       4    1
```

```
# Spliting train and test samples

inTrain = createDataPartition(df$Risk, p = 0.7, list = F)

train = df[inTrain, ]
test = df[-inTrain, ]
```

```
# Dealing with NA's

train = mlr::impute(train, target = "Risk",
                    cols = list(Saving.accounts = imputeLearner("classif.rpart"),
                                Checking.account = imputeLearner("classif.rpart")))
test = reimpute(test, train$desc)

train = train$data
```

```
# Training the model

lr = caret::train(Risk ~ ., data = train, method = 'glm')
prev_train = predict(lr, train)

confusionMatrix(train$Risk, prev_train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1  43 167
##          2  26 464
##
##                Accuracy : 0.7243
##                  95% CI : (0.6896, 0.7571)
##     No Information Rate : 0.9014
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1877
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62319
##             Specificity : 0.73534
##          Pos Pred Value : 0.20476
##          Neg Pred Value : 0.94694
##              Prevalence : 0.09857
##          Detection Rate : 0.06143
##    Detection Prevalence : 0.30000
##       Balanced Accuracy : 0.67926
##
##        'Positive' Class : 1
##
```

```
# Testing the model

prev_test = predict(lr, test)

confusionMatrix(test$Risk, prev_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1  14  76
##          2  11 199
##
##                Accuracy : 0.71
##                  95% CI : (0.6551, 0.7607)
##     No Information Rate : 0.9167
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.13
##
##  Mcnemar's Test P-Value : 6.813e-12
##
##             Sensitivity : 0.56000
##             Specificity : 0.72364
##          Pos Pred Value : 0.15556
##          Neg Pred Value : 0.94762
##              Prevalence : 0.08333
##          Detection Rate : 0.04667
##    Detection Prevalence : 0.30000
##       Balanced Accuracy : 0.64182
##
##        'Positive' Class : 1
##
```