R Notebook

```
# Packages used

require(caret)
require(mlr)
```

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

```
# 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$X = NULL

df$Sex = as.integer(factor(df$Sex, levels = c('male', 'female'), labels = c(1, 2)))
df$Housing = as.integer(factor(df$Housing, levels = c('free', 'own', 'rent'),
                               labels = c(1, 2, 3)))
df$Saving.accounts = as.integer(factor(df$Saving.accounts,
                            levels = c('little', 'moderate', 'quite rich', 'rich'),
                            labels = c(1, 2, 3, 4)))
df$Checking.account = as.integer(factor(df$Checking.account,
                             levels = c('little', 'moderate', 'rich'),
                             labels = c(1, 2, 3)))
df$Purpose = as.integer(factor(df$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))

str(df)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Age             : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ Sex             : int  1 2 1 1 1 1 1 1 1 1 ...
##  $ Job             : int  2 2 1 2 2 1 2 3 1 3 ...
##  $ Housing         : int  2 2 2 1 1 1 2 3 2 2 ...
##  $ Saving.accounts : int  NA 1 1 1 1 NA 3 1 4 1 ...
##  $ Checking.account: int  1 2 NA 1 1 NA NA 2 NA 2 ...
##  $ Credit.amount   : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ Duration        : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ Purpose         : int  1 1 2 3 4 2 3 4 1 4 ...
##  $ Risk            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 2 2 2 1 ...
```

```
# Dealing with NA's and normalizing

imput = preProcess(df, method = "knnImpute")

df = predict(imput, df)
```

```r
head(df, n = 15)
```

```
##            Age        Sex        Job    Housing Saving.accounts
## 1   2.76507291 -0.6699448  0.1468757 -0.1336436     -0.06504672
## 2  -1.19080809  1.4911675  0.1468757 -0.1336436     -0.52516074
## 3   1.18272051 -0.6699448 -1.3830794 -0.1336436     -0.52516074
## 4   0.83108664 -0.6699448  0.1468757 -2.0159476     -0.52516074
## 5   1.53435438 -0.6699448  0.1468757 -2.0159476     -0.52516074
## 6  -0.04799802 -0.6699448 -1.3830794 -2.0159476     -0.29510373
## 7   1.53435438 -0.6699448  0.1468757 -0.1336436      1.77540936
## 8  -0.04799802 -0.6699448  1.6768308  1.7486604     -0.52516074
## 9   2.23762211 -0.6699448 -1.3830794 -0.1336436      2.92569442
## 10 -0.66335729 -0.6699448  1.6768308 -0.1336436     -0.52516074
## 11 -0.92708269  1.4911675  0.1468757  1.7486604     -0.52516074
## 12 -1.01499116  1.4911675  0.1468757  1.7486604     -0.52516074
## 13 -1.19080809  1.4911675  0.1468757 -0.1336436     -0.52516074
## 14  2.14971364 -0.6699448 -1.3830794 -0.1336436     -0.52516074
## 15 -0.66335729  1.4911675  0.1468757  1.7486604     -0.52516074
##    Checking.account Credit.amount   Duration     Purpose Risk
## 1        -0.9876081   -0.74475875 -1.2358595 -1.28575857    2
## 2         0.5275578    0.94934176  2.2470700 -1.28575857    1
## 3        -0.6845749   -0.41635407 -0.7382981 -0.67232414    2
## 4        -0.9876081    1.63342961  1.7495086 -0.05888971    2
## 5        -0.9876081    0.56638010  0.2568246  0.55454473    1
## 6        -0.3815418    2.04898375  1.2519473 -0.67232414    2
## 7         0.8305909   -0.15455142  0.2568246 -0.05888971    2
## 8         0.5275578    1.30254507  1.2519473  0.55454473    2
## 9         0.5275578   -0.07519582 -0.7382981 -1.28575857    2
## 10        0.5275578    0.69533296  0.7543859  0.55454473    1
## 11        0.5275578   -0.70012123 -0.7382981  0.55454473    1
## 12       -0.9876081    0.36728255  2.2470700  1.16797916    1
## 13        0.5275578   -0.60376084 -0.7382981 -1.28575857    2
## 14       -0.9876081   -0.73413077  0.2568246  0.55454473    1
## 15       -0.9876081   -0.66186049 -0.4895174  0.55454473    2
```

```r
# Spliting train and test samples

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

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

```r
# Training the model

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

confusionMatrix(as.factor(train$Risk), as.factor(prev_train))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1  56 154
```

```
##          2  33 457
##
##                Accuracy : 0.7329
##                  95% CI : (0.6984, 0.7653)
##     No Information Rate : 0.8729
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2386
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6292
##             Specificity : 0.7480
##          Pos Pred Value : 0.2667
##          Neg Pred Value : 0.9327
##              Prevalence : 0.1271
##          Detection Rate : 0.0800
##    Detection Prevalence : 0.3000
##       Balanced Accuracy : 0.6886
##
##        'Positive' Class : 1
##
```

```r
# Testing the model

prev_test = predict(knn, test)

confusionMatrix(test$Risk, prev_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1  20  70
##          2  32 178
##
##                Accuracy : 0.66
##                  95% CI : (0.6033, 0.7135)
##     No Information Rate : 0.8267
##     P-Value [Acc > NIR] : 1.0000000
##
##                   Kappa : 0.0794
##
##  Mcnemar's Test P-Value : 0.0002487
##
##             Sensitivity : 0.38462
##             Specificity : 0.71774
##          Pos Pred Value : 0.22222
##          Neg Pred Value : 0.84762
##              Prevalence : 0.17333
##          Detection Rate : 0.06667
##    Detection Prevalence : 0.30000
##       Balanced Accuracy : 0.55118
##
##        'Positive' Class : 1
```

##