KNN 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
```

```
# Reading the data

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

str(df)
```

```
## 'data.frame':    1000 obs. of  11 variables:
##  $ X               : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Age             : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ Sex             : Factor w/ 2 levels "female","male": 2 1 2 2 2 2 2 2 2 2 ...
##  $ Job             : int  2 2 1 2 2 1 2 3 1 3 ...
##  $ Housing         : Factor w/ 3 levels "free","own","rent": 2 2 2 1 1 1 2 3 2 2 ...
##  $ Saving.accounts : Factor w/ 4 levels "little","moderate",..: NA 1 1 1 1 NA 3 1 4 1 ...
##  $ Checking.account: Factor w/ 3 levels "little","moderate",..: 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         : Factor w/ 8 levels "business","car",..: 6 6 4 5 2 4 5 2 6 2 ...
##  $ Risk            : Factor w/ 2 levels "bad","good": 2 1 2 2 1 2 2 2 2 1 ...
```

```
# 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))

df[, 1] = scale(df[, 1])
```

```r
df[, 7:8] = scale(df[, 7:8])

head(df, n = 10)
```

```
##            Age Sex Job Housing Saving.accounts Checking.account Credit.amount
## 1   2.76507291   1   2       2            <NA>                1   -0.74475875
## 2  -1.19080809   2   2       2               1                2    0.94934176
## 3   1.18272051   1   1       2               1             <NA>   -0.41635407
## 4   0.83108664   1   2       1               1                1    1.63342961
## 5   1.53435438   1   2       1               1                1    0.56638010
## 6  -0.04799802   1   1       1            <NA>             <NA>    2.04898375
## 7   1.53435438   1   2       2               3             <NA>   -0.15455142
## 8  -0.04799802   1   3       3               1                2    1.30254507
## 9   2.23762211   1   1       2               4             <NA>   -0.07519582
## 10 -0.66335729   1   3       2               1                2    0.69533296
##      Duration Purpose Risk
## 1  -1.2358595       1    2
## 2   2.2470700       1    1
## 3  -0.7382981       2    2
## 4   1.7495086       3    2
## 5   0.2568246       4    1
## 6   1.2519473       2    2
## 7   0.2568246       3    2
## 8   1.2519473       4    2
## 9  -0.7382981       1    2
## 10  0.7543859       4    1
```

```r
# Spliting train and test samples

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

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

```r
# Dealing with NA's

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

```
## Functional features have been converted to numerics
## Functional features have been converted to numerics
```

```r
test = reimpute(test, train$desc)

train = train$data
```

```r
# Training the model

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

confusionMatrix(train$Risk, prev_train)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   1   2
##          1  57 153
##          2  23 467
##
##                Accuracy : 0.7486
##                  95% CI : (0.7147, 0.7803)
##     No Information Rate : 0.8857
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2727
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71250
##             Specificity : 0.75323
##          Pos Pred Value : 0.27143
##          Neg Pred Value : 0.95306
##              Prevalence : 0.11429
##          Detection Rate : 0.08143
##    Detection Prevalence : 0.30000
##       Balanced Accuracy : 0.73286
##
##        '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  17  73
##          2  17 193
##
##                Accuracy : 0.7
##                  95% CI : (0.6447, 0.7513)
##     No Information Rate : 0.8867
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1313
##
##  Mcnemar's Test P-Value : 6.731e-09
##
##             Sensitivity : 0.50000
##             Specificity : 0.72556
##          Pos Pred Value : 0.18889
##          Neg Pred Value : 0.91905
##              Prevalence : 0.11333
##          Detection Rate : 0.05667
##    Detection Prevalence : 0.30000
```

```
##       Balanced Accuracy : 0.61278
##
##        'Positive' Class : 1
##
```