Machine Learning Assignment 1

Agustín Valencia

11/19/2019

Assignment 1. Spam classification with nearest neighbors

1. Importing the data:

```
data <- read.xlsx("data/spambase.xlsx")
n = dim(data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = data[id,]
test = data[-id,]
```

2. Use logistic regression to classify the training and test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.5$, otherwise $\hat{Y} = 0$ and report the confusion matrices and the misclassification rates for train and test data. Analyze the obtained results.

```
# util function
get_performance <- function(targets, predictions) {
    t <- table(targets, predictions)
    print("Confusion Matrix")
    print(t)
    tn <- t[1,1]
    tp <- t[2,2]
    fp <- t[1,2]
    fn <- t[2,1]
    total <- dim(test)[1]
    tpr <- tp/total * 100
    tnr <- tn/total * 100
    fpr <- fp/total * 100
    fnr <- fn/total * 100

    cat("Classification performance:\n")
    cat("TPR = ", tpr, "%\n")
    cat("TNR = ", tnr, "%\n")
    cat("FPR = ", fpr, "%\n")
    cat("FNR = ", fnr, "%\n")
    cat("Misclassification Rate = ", (fp+fn)/total * 100, "%\n")
}

# fit the model
fit <- glm(Spam ~ . , data = train, family = "binomial")

# performance on training data
pred_train <- predict(fit, newdata = train)
pred_train_at_05 <- as.integer(pred_train > 0.5)
targets <- train$Spam
get_performance(targets, pred_train_at_05)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
##       0 875  56
##       1 156 283
## Classification performance:
## TPR =  20.65693 %
## TNR =  63.86861 %
## FPR =  4.087591 %
## FNR =  11.38686 %
## Misclassification Rate =  15.47445 %
```

```r
# performance on test data
pred_test <- predict(fit, newdata = test)
pred_test_at_05  <- as.integer(pred_test > 0.5)
targets <- test$Spam
get_performance(targets, pred_test_at_05)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
##       0 865  86
##       1 162 257
## Classification performance:
## TPR =  18.75912 %
## TNR =  63.13869 %
## FPR =  6.277372 %
## FNR =  11.82482 %
## Misclassification Rate =  18.10219 %
```

3. Use logistic regression to classify the test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.8$, otherwise $\hat{Y} = 0$

```r
# performance on train data
pred_train_at_08  <- as.integer(pred_train > 0.8)
get_performance(targets, pred_train_at_08)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
##       0 771 180
##       1 343  76
## Classification performance:
## TPR =  5.547445 %
## TNR =  56.27737 %
## FPR =  13.13869 %
## FNR =  25.0365 %
## Misclassification Rate =  38.17518 %
```

```r
# performance on test data
pred_test_at_08  <- as.integer(pred_test > 0.8)
get_performance(targets, pred_test_at_08)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
```

```
##       0 892  59
##       1 229 190
## Classification performance:
## TPR =  13.86861 %
## TNR =  65.10949 %
## FPR =  4.306569 %
## FNR =  16.71533 %
## Misclassification Rate =  21.0219 %
```

4. Use standard kknn() with K = 30 from package *kknn*, report the misclassification rates for the training and test data and compare the results with step 2.

```r
# Train KNN K=30
knn_model <- train.kknn(Spam ~ . , data = train, ks = 30)

# performance on training data
knn_fit <- predict(knn_model, train)
results <- as.integer(knn_fit > 0.5)
target <- train$Spam
get_performance(target, results)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
##       0 779 152
##       1  77 362
## Classification performance:
## TPR =  26.42336 %
## TNR =  56.86131 %
## FPR =  11.09489 %
## FNR =  5.620438 %
## Misclassification Rate =  16.71533 %
```

```r
# performance on test data
knn_fit <- predict(knn_model, test)
results <- as.integer(knn_fit > 0.5)
target <- test$Spam
get_performance(target, results)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
##       0 702 249
##       1 180 239
## Classification performance:
## TPR =  17.44526 %
## TNR =  51.24088 %
## FPR =  18.17518 %
## FNR =  13.13869 %
## Misclassification Rate =  31.31387 %
```

5. Repeat step 4 for K=1 and compare results with step 4. What effects does the decrease of K lead to and why?

```r
# Train KNN K=1
knn_model <- train.kknn(Spam ~ . , data = train, ks = 1)

# performance on training data
knn_fit <- predict(knn_model, train)
results <- as.integer(knn_fit > 0.5)
target <- train$Spam
get_performance(target, results)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
##       0 931   0
##       1   0 439
## Classification performance:
## TPR =  32.0438 %
## TNR =  67.9562 %
## FPR =  0 %
## FNR =  0 %
## Misclassification Rate =  0 %
```

```r
# performance on test data
knn_fit <- predict(knn_model, test)
results <- as.integer(knn_fit > 0.5)
target <- test$Spam
get_performance(target, results)
```

```
## [1] "Confusion Matrix"
##        predictions
## targets   0   1
##       0 644 307
##       1 185 234
## Classification performance:
## TPR =  17.08029 %
## TNR =  47.0073 %
## FPR =  22.40876 %
## FNR =  13.50365 %
## Misclassification Rate =  35.91241 %
```

If we assign k=1 training misclassification is 0%, this means we are overfitting our model, thus the misclassification for the testing set may be bigger than other scenarios.