KNN, Linear regression, and multilinear regression

Elizabeth Daniela Monsalve Forero

2023-10-07

```
library(tidyverse)
library(caret)
library(MASS)
library(glmnet)
library(boot)
```

This code block loads a number of packages in R, allowing the functions and tools provided by those packages to be used in data analysis. Each package has its own set of functionality:

1) caret (Classification and Regression Training): provides a unified set of functions and tools for training and evaluating supervised machine learning models. It facilitates the construction of classification and regression models by providing a common interface for many machine learning algorithms.
2) MASS (Modern Applied Statistics with S): Contains a wide variety of functions and data sets commonly used in applied statistics. It includes advanced statistical methods and functions for fitting linear models, logistic regression, discriminant analysis, among others.
3) glmnet: is a library used for fitting linear and logistic regression models with L1 and L2 regularization (lasso and ridge, respectively).
4) boot: It is used to perform resampling analysis and construction of confidence intervals using the bootstrapping technique. This library is useful to perform robust statistical inference and to evaluate uncertainty in parameter estimates and statistics.

```
data <- read.csv("diabetes_012_health_indicators_BRFSS2015.csv")
```

This line of code reads the database CSV file (diabetes_012_health_indicators_BRFSS2015) in R and loads its data into an object called data

```
set.seed(150)
sample <- data %>% sample_frac(0.01)
sample$Binary <- make.names(factor(ifelse(sample$Diabetes_012 > 0, 1, 0)))
sample$Binary <- factor(sample$Binary, levels = c("X0", "X1"))
```

First, a seed is set for the random number generator in R. By setting a specific seed (in this case, 150), the random numbers generated in subsequent operations will be the same each time the code is run, which facilitates the reproducibility of the analysis.

Subsequently, a random sample of 1% of the original data is selected and stored in a new object called ¨sample¨, a column called Diabetes_012 is binarized depending on whether the values are greater than 0, and then a new categorical variable called Binary is created with levels “N0” and “S1” representing the presence or absence of diabetes.

```
BMI <- sample
Menthlth <- sample
Physhtlth <- sample
```

3 new data vectors are created from the original ¨sample¨ dataset.

```
set.seed(150)
Index <- createDataPartition(sample$Binary, p = 0.8, list = FALSE, times = 1)
Data <- sample[Index, ]
Test <- sample[-Index, ]
```

This code block divides the data into a training set ¨Data¨ and a test set ¨Test¨ using the createDataPartition function, subsequently, it is specified that 80% of the data will be used to train models and 20% to evaluate their performance.

```
Control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction = twoClassSummary
```

This line of code is setting up a control variable called ¨Control¨ that will be used when training and evaluating a classification model. The ¨Control¨ object specifies that a 10-fold cross-validation will be performed, classification probabilities will be generated, and the twoClassSummary function will be used to summarize the results of the model evaluation.

```
set.seed(20)
KNN <- train(Binary ~ ., data = Data, method = "knn", trControl = Control, tuneLength = 10)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
print(KNN)
```

```
## k-Nearest Neighbors
##
## 2031 samples
##   22 predictor
##    2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1827, 1829, 1827, 1827, 1829, 1828, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.8907213  0.9681378  0.4467914
##    7  0.9080109  0.9728716  0.4114082
##    9  0.9120083  0.9728472  0.3935829
##   11  0.9105237  0.9740446  0.3578431
##   13  0.9086344  0.9710929  0.3190731
##   15  0.9052983  0.9728611  0.2712121
##   17  0.9066787  0.9746328  0.2621212
##   19  0.9022420  0.9746363  0.2442959
##   21  0.9043376  0.9787644  0.2266488
```

```
##   23  0.9050209  0.9799443  0.2238859
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
predictions <- predict(KNN, newdata = Test)
confusionMatrix(predictions, Test$Binary)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  X0  X1
##         X0 407  39
##         X1  16  44
##
##                Accuracy : 0.8913
##                  95% CI : (0.8609, 0.9171)
##     No Information Rate : 0.836
##     P-Value [Acc > NIR] : 0.0002696
##
##                   Kappa : 0.554
##
##  Mcnemar's Test P-Value : 0.0030123
##
##             Sensitivity : 0.9622
##             Specificity : 0.5301
##          Pos Pred Value : 0.9126
##          Neg Pred Value : 0.7333
##              Prevalence : 0.8360
##          Detection Rate : 0.8043
##    Detection Prevalence : 0.8814
##       Balanced Accuracy : 0.7461
##
##        'Positive' Class : X0
##
```

This code block is a complete process from initial setup to final evaluation of the KNN model:

1) Setting the Random Seed: Setting a random seed means that, although the KNN model uses randomness in its process, the results will be consistent in every execution of the code.

2) Training the KNN Model: The train() function is used to predict the binary variable "Binary" using all the other variables in the Data training dataset.

3) Parameter tuning: A parameter search is performed by testing different values of "k" in the range of 1 to 10. This is done using cross-validation, which means that the dataset is split into parts to evaluate the performance of the model with different values of "k". The goal is to find the optimal value of "k" that provides the best performance on the training dataset.

4) Model Evaluation: After training the KNN model with the optimal value of "k", the ¨Test¨ data is used to make predictions. These predictions are compared with the actual ¨Test¨ class labels to compute a confusion matrix. The confusion matrix shows the number of true positives, true negatives, false positives and false negatives.

```
BMI2 <- lm(BMI ~ ., data = BMI)
BMI_results <- cv.glm(data = BMI, glmfit = BMI2, K = 10)

Menthlth2 <- lm(MentHlth ~ ., data = Menthlth)
Menthlth_results <- cv.glm(data = Menthlth, glmfit = Menthlth2, K = 10)

Physhtlth2 <- lm(PhysHlth ~ ., data = Physhtlth)
Physhtlth_results <- cv.glm(data = Physhtlth, glmfit = Physhtlth2, K = 10)
```

This code is used to fit linear regression models and evaluate their performance using cross-validation on three different data sets ("BMI", "MentHlth" and "PhysHlth"). It helps to determine how well the model fits the data and how well it generalizes to new data.

1) Linear Regression: A linear regression model is being fitted where the dependent variables are: "BMI", "MentHlth" and "PhysHlth" and each set is used as the data source. The symbol ~ . means that all other variables are being used as independent variables to predict the dependent variable.

2) Cross Validation: Cross validation is performed (in this case, with 10 folds) to evaluate the performance of the fitted linear regression model for the dependent variables. The function cv.glm() takes as arguments the data set, the fitted model and the number of folds (K = 10) for cross-validation. This evaluates how well the model generalizes to unseen data and returns cross-validation results.