hmwk3.1a

student

2023-09-04

K-NN

Summary

Find the best validation set by splitting data into train, validation and test data, using kknn function, and without caret library, wanted to write own loops.

The average performance during the cross-validation was approximately: 0.696521739130435 The test set accuracy: 0.683673469387755 These are close, indicating that best k = 8 model is likely good based on this credit card worthiness data set.

Begin Code

Load Packages

```
knitr::opts_chunk$set(echo = TRUE)

# Load packages
#library(conflicted)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(class)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v ggplot2   3.4.3     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(devtools)
```

```
## Loading required package: usethis
```

```r
library(kknn)
```

Load and View Data Structure

```r
# load data
data <- read.csv("C:\\Users\\Public\\Documents\\gatech\\hw1\\credit_card_with_headers.csv")

# check data structure for data frame
str(data)
```

```
## 'data.frame':    654 obs. of  11 variables:
##  $ A1 : int  1 0 0 1 1 1 1 0 1 1 ...
##  $ A2 : num  30.8 58.7 24.5 27.8 20.2 ...
##  $ A3 : num  0 4.46 0.5 1.54 5.62 ...
##  $ A8 : num  1.25 3.04 1.5 3.75 1.71 ...
##  $ A9 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ A10: int  0 0 1 0 1 1 1 1 1 1 ...
##  $ A11: int  1 6 0 5 0 0 0 0 0 0 ...
##  $ A12: int  1 1 1 0 1 0 0 1 1 0 ...
##  $ A14: int  202 43 280 100 120 360 164 80 180 52 ...
##  $ A15: int  0 560 824 3 0 0 31285 1349 314 1442 ...
##  $ R1 : int  1 1 1 1 1 1 1 1 1 1 ...
```

Split data into 70% training and 15% testing, 15% validation. The training set is used to train the machine learning model. The validation set is used to tune the model's hyperparameters, helps in selecting the best-performing one without touching the test set, and for checking if the model is overfitting to the training data.

```r
# set seed
set.seed(456)

# get 70%, 15%, 15%
train_percent <- 0.70
validation_percent <- 0.15
test_percent <- 0.15

# number of rows for train, validation and test data
total_rows <- nrow(data)
train_size <- round(train_percent * total_rows)
validation_size <- round(validation_percent * total_rows)
test_size <- total_rows - train_size - validation_size

# get indices for data
```

```r
indices <- sample(1:total_rows)

# split data into training, validation, and test sets
train_data <- data[indices[1:train_size], ]
validation_data <- data[indices[(train_size + 1):(train_size + validation_size)], ]
test_data <- data[indices[(train_size + validation_size + 1):total_rows], ]
```

Apply K-Fold Cross Validation to Training Data. This step asses how well a machine learning model will perform and for estimating performance metrics fo a more reliable performance estimation.

```r
# get number of folds
number_fold <- 10
train_row_total <- nrow(train_data)

# variable to store fold assignments
fold_assignments <- rep(NA, train_row_total)

# loop through number of folds
for (fold in 1:number_fold)
{
  fold_index <- seq(from = fold, to = train_row_total, by = number_fold)
  fold_assignments[fold_index] <- fold
}
```

Loop through folds: get a comprehensive view of the model's performance across the subsets of data, and perform hyperparameter tuning by looping through k, to find best k.

```r
# response variable
train_target <- 'R1'

# initialize best k and performance variables
best_k <- 1
best_performance <- 0

# possible k values for k-NN
k_values <- c(1, 2, 3, 5, 8, 13)

# Loop through each k value for k-NN
for (k_val in k_values) {

  # Initialize performance variable for this k value
  performance_metrics <- vector('numeric', number_fold)

  # Loop through each fold for cross-validation
  for (fold in 1:number_fold) {
    # Create training and validation sets for this fold
    fold_train_data <- train_data[fold_assignments != fold, ]
    fold_validation_data <- train_data[fold_assignments == fold, ]

    # Separate predictors and target for training and validation sets
    train_predictors <- fold_train_data[, !(names(fold_train_data) %in% 'R1')]
    train_target <- fold_train_data$R1
```

```r
    val_predictors <- fold_validation_data[, !(names(fold_validation_data) %in% 'R1')]
    val_target <- fold_validation_data$R1

    # Train the k-NN model and make predictions
    predictions <- knn(train = train_predictors, test = val_predictors, cl = train_target, k = k_val)

    # Calculate the accuracy for this fold
    accuracy <- sum(predictions == val_target) / length(val_target)

    # Store the performance metric for this fold
    performance_metrics[fold] <- accuracy
  }

  # Compute the average performance across all folds
  average_performance <- mean(performance_metrics)

  # Print the average performance for this k value
  print(paste("Average performance for k =", k_val, "is", average_performance))

  # Update best performance and best k value
  if (average_performance > best_performance) {
    best_performance <- average_performance
    best_k <- k_val
  }
}
```

```
## [1] "Average performance for k = 1 is 0.65304347826087"
## [1] "Average performance for k = 2 is 0.63975845410628"
## [1] "Average performance for k = 3 is 0.665990338164251"
## [1] "Average performance for k = 5 is 0.692270531400966"
## [1] "Average performance for k = 8 is 0.696521739130435"
## [1] "Average performance for k = 13 is 0.665893719806763"
```

```r
# Print the best k value and performance
print(paste("Best k is", best_k, "with average performance", best_performance))
```

```
## [1] "Best k is 8 with average performance 0.696521739130435"
```

Run k-NN (k-Nearest Neighbors) model with best k model on test data. This evaluates the model's ability to generalize its predictions to new, unseen data.

```r
# Train the best k-NN model on the entire training set using the best_k value
best_knn_model <- knn(train = train_data[, !(names(train_data) %in% 'R1')],
                      test = test_data[, !(names(test_data) %in% 'R1')],
                      cl = train_data$R1,
                      k = best_k)

# Evaluate the model on the test data
test_target <- test_data$R1  # Assuming test_data has the actual labels in a column named 'R1'
test_predictions <- best_knn_model  # knn() returns test predictions directly

# Calculate accuracy on the test set
```

```r
test_accuracy <- sum(test_predictions == test_target) / length(test_target)

# Print test accuracy
print(paste("Test accuracy with best k =", best_k, "is", test_accuracy))
```

```
## [1] "Test accuracy with best k = 8 is 0.683673469387755"
```

The average performance during the cross-validation was approximately 0.696521739130435 and the test set accuracy was** 0.683673469387755 These are very close, indicating that best k =** 8 **model is likely a good choice based on this credit card worthiness dataset.**