Cross-Validation KNN ML Model's

2024-01-25

K Fold Validation - KNN Model

```
rm(list = ls())

# Call library 'kknn' for use.
library(kknn)

#  load 'credit_card_data.txt' file into a table without headers.
myData <- read.table("credit_card_data.txt", stringsAsFactors = FALSE, header = FALSE)

# Initialize a variable to store the count of the matches.
matches <- 0

# Set the seed.
set.seed(1)

# Define the number of folds for k-fold cross-validation.
k_fold_validation <- 10

# Create indices for splitting the data into 10 folds.
folds <- cut(seq(1, nrow(myData)), breaks = k_fold_validation, labels = FALSE)

# For loop to iterate through each fold.
for (i in 1:k_fold_validation) {
  # Split the data into training and testing data.
  test_row_indices <- which(folds == i, arr.ind = TRUE)
  testing_data <- myData[test_row_indices, ]
  training_data <- myData[-test_row_indices, ]

  # Create the knn model using the training data.
  myModel_knn <- kknn(V11 ~ .,
                      training_data,
                      testing_data,
                      k = 12,
                      distance = 2,
                      kernel = "optimal",
                      scale = TRUE)

  # Round the fitted values of the model to 0 or 1, to see if they match the data from myData.
  rounded_fitted_values <- round(fitted.values(myModel_knn))

  # Check if the rounded values match the data and add 1 to the matches if so.
```

```
  matches <- matches + sum(rounded_fitted_values == testing_data[, 11])
}

# Calculate the percentage of matches, (matches) / (total data points) * 100.
accuracy <- (matches / nrow(myData)) * 100
accuracy
```

```
## [1] 82.263
```

I am using a k-nearest-neighbors model to find a good classifier using cross validation. The data I am using is the 'credit_card_data.txt' dataset.

For this problem, I need to implement a cross validation technique, so I chose to utilize k-fold cross validation as it is a quick, simple, and well-known cross validation. Since 10-fold cross validation is popularly used, I decided to use 10 folds. I found it interesting that the accuracy percentage was highest for k=12 and k=15 in my last project with the same dataset, so I decided to use the same k value, k=12, for the knn model to test the accuracy using not only the training data, but the testing data as well, alongside the 10-fold cross validation technique.

I found that the accuracy is 82.263% when splitting the testing data and training data using 10-fold cross validation. Since I used 10 folds, this means that during each iteration, the testing data is 1/10 folds, or 10% of the data, whilst the training data is 9/10 folds, or 90% of the data. During the 10-fold cross validation however, it is not simply using one fold, but iteration through each fold to utilize one fold for the testing data, and the remaining 9 folds for the training data, but this is done for every fold of the data and eventually each 1 fold will go through the testing data to ensure that every data point is used for testing exactly once. This is the process of k-fold cross validation, which can be used for any number of folds (up to the number of total data points.)

Compare the 82.263% to my last knn model (k=12) percentage of 85.3%, I see that this model is not as accurate, but this is to be expected as I am not only simply using the training set of data, but using the training and testing set alongside the k-fold cross validation.

Train, Validation, Test Split - KNN Model

```
rm(list = ls())

#Set the seed.
set.seed(1)

# Call library 'kknn' for use.
library(kknn)

#  load 'credit_card_data.txt' file into a table without headers.
myData <- read.table("credit_card_data.txt", stringsAsFactors = FALSE, header = FALSE)

# Create variables for the percentages of training, validation, and testing data.
training_percentage <- 0.6
validation_percentage <- 0.2
testing_percentage <- 0.2

# Calculate the number of samples for each set based on percentages.
num_samples <- nrow(myData)
```

```
num_train <- round(training_percentage * num_samples)
num_validation <- round(validation_percentage * num_samples)
num_test <- num_samples - num_train - num_validation

# Create indices for splitting the data into sets.
indices <- sample(1:num_samples)

# Split the data into training, validation, and test sets.
train_data <- myData[indices[1:num_train], ]
validation_data <- myData[indices[(num_train + 1):(num_train + num_validation)], ]
test_data <- myData[indices[(num_train + num_validation + 1):num_samples], ]

# Create the knn model using the training set.
myModel_knn <- kknn(V11 ~ .,
                    train_data,
                    validation_data,
                    k = 12,
                    distance = 2,
                    kernel = "optimal",
                    scale = TRUE)

# Predict using the model on the validation set.
predictions_validation <- round(fitted.values(myModel_knn))

# Calculate accuracy for the validation set.
correct_predictions_validation <- sum(predictions_validation == validation_data[, 11])
accuracy_validation <- correct_predictions_validation / nrow(validation_data) * 100

# Accuracy for the validation set.
accuracy_validation
```

```
## [1] 83.96947
```

```
# Predict using the model on the test set.
predictions_test <- round(fitted.values(myModel_knn))

# Calculate accuracy for the test set.
correct_predictions_test <- sum(predictions_test == test_data[, 11])
accuracy_test <- correct_predictions_test / nrow(test_data) * 100
accuracy_test
```

```
## [1] 51.9084
```

I am using the kknn function to find a good classifier whilst splitting the data into training, validation, and testing data sets. The data I am using, again, is the 'credit_card_data.txt.'

For this problem, I am splitting the data into training, validation, and testing data sets. The reason I are doing this, and the importance of this split is the fact that training data must be used to build and train the model, validation data is used to evaluate and compare different model's and see which is best, and the testing data is used to test the model's accuracy on new, unseen data that has been split and essentially 'hidden' from the model. I chose to do a 60/20/20 split between the training/validation/testing splits of the data. It is generally a good idea to do either a 50/25/25,60/20/20, or 70/15/15 split. I simply chose to do the middle, as I would assume it is the median and safest recommended value split.

By simply using k-fold cross validation, and in this model, the accuracy_validation variable is found to be 83.96947% whilst the accuracy_test variable is found to be 51.9084%. When it comes to the percentages decreasing when the validation set of data is introduced, and again when the testing set of the data is introduced, it is to be expected that they would decrease along each time. The reasons for this I can only assume can be attributed to either randommness in the data splitting not being cross validated, the k value for the knn model needs to be tuned, or perhaps a better split would grant me better percentages due to the data having a small sample size.