Homework-4

Group 4

6/4/2020

Problem 2: Predicting Housing Median Prices k-NN

The file BostonHousing.xlsx contains information on over 500 census tracts in Boston, where for each tract 14 variables are recorded. The last column (CAT.MEDV) was derived from MEDV, such that it obtains the value 1 if MEDV > 30 and 0 otherwise. Consider the goal of predicting the median value (MEDV) of a tract, given the information in the first 13 columns. Partition the data into training (60%) and validation (40%) sets.

(a) Perform a k-NN prediction with all 13 predictors (ignore the CAT.MEDV column), trying values of k from 1 to 5. Make sure to normalize the data (click "normalize input data"). What is the best k chosen? What does it mean?

```
# Import the required packages
library(dplyr)
library(readxl)
library(FNN)
library(caret)

# Import the BostonHousing.xlsx file
BostonHousing <- read_xlsx("BostonHousing.xlsx", sheet = "Data")

# Define the normalize function
normalize <- function(x) {
  return((x - min(x))) / (max(x) - min(x))
}

# Normalize the dataframe
df.norm <- as.data.frame(lapply(BostonHousing[1:13], normalize))

# Generate the training data indices
indices <- sample(seq_len(nrow(df.norm)), size = floor(0.6 * nrow(df.norm)))

# Get training and validation data
train_data <- df.norm[indices, ]
validation_data <- df.norm[-indices, ]

# Create a dataframe to keep track of k vs error
error.df <- data.frame("k" = 1:5, "error" = rep(0, 5))

# Loop for K = 1 to 5
for (i in 1:5) {
```

```
  model <- knnreg(x = train_data[, 1:12], y = train_data[, 13], k = i)
  error.df[i, 2] <- RMSE(validation_data[, 13], predict(model, validation_data[, 1:12]))
}

# Get the K-value with the lowest RMSE error
best_k <- filter(error.df, error == min(error.df$error))$k
cat("The model with the best K is:", best_k, "\n")
```

```
## The model with the best K is: 4
```

(b) Predict the MEDV for a tract with the following information, using the best k:

```
# Let's get the model with the best K
model <- knnreg(x = train_data[, 1:12], y = train_data[, 13], k = best_k)

# Create a dataframe for the given record
df <- data.frame(
  "CRIM" = 0.2, "ZN" = 0, "INDUS" = 7,
  "CHAS" = 0, "NOX" = 0.538, "RM" = 6,
  "AGE" = 62, "DIS" = 4.7, "RAD" = 4,
  "TAX" = 307, "PTRATIO" = 21, "LSTAT" = 10
)

# Normalize the dataframe
df.norm <- as.data.frame(lapply(df[1:12], normalize))

# Predict the MEDV value for the new record.
prediction <- predict(model, df.norm)
cat("The MEDV prediction for the above record is:", prediction, "\n")
```

```
## The MEDV prediction for the above record is: 22.925
```

(c) Why is the error of the training data zero?

The error for training data will be zero at K = 1, since the single closest neighbor to the training sample vector will be itself. Hence the error will be zero.

```
# Train the model with k = 1
model <- knnreg(x = train_data[, 1:12], y = train_data[, 13], k = 1)

# print
cat(
  "Error for Training Data at k = 1:",
  RMSE(train_data[, 13], predict(model, train_data[, 1:12])),
  "\n"
)
```

```
## Error for Training Data at k = 1: 0
```

```
# remove all env variables
rm(list = ls())
```

(d) Why is the validation data error overly optimistic compared to the error rate when applying this k-NN predictor to new data?

We have chosen a model which can can perform best on the validation data. So it makes total sense that the error of the model on validation data is more optimistic as compared to it's error on new data.

(e) If the purpose is to predict MEDV for several thousands of new tracts, what would be the disadvantage of using k-NN prediction? List the operations that the algorithm goes through in order to produce each prediction.

- The disadvantage of using K-NN prediction for several thousand new tracts would be that the KNN algorithm will be slow. Calculating the euclidean distances for several thousand vectors would take a long time. Also, more sophisticated methods like Regression Trees and Neural Networks would perform better on such a large dataset.

The algorithm goes through the following operations in order to produce each operation -

For each record in the dataset, the algorithm -

1. Calculates the euclidean distance of that vector(record) with every other record in the dataset.
2. Sorts the euclidean distances from the lowest to the highest.
3. Takes the top K neighbors and then -
    - If it is a classification problem, it takes the classes of each of the K neighbors and assigns the majority class to the current record.
    - If it is a regression problem, it takes the average of the output variable of each of the K neighbors and assigns it to the current record.
