# Numeric Prediction

Blake Pappas

2023-12-17

## Numeric Prediction in R

## Load the following packages:

```
library(caret)
library(class)
library(dplyr)
library(glmnet)
```

## In this exercise, we use the "laptop.csv" file. The goal is to predict the "Price" of a laptop based on its attributes.

## P1: Import the dataset. Split it to 80% training and 20% testing.

```
laptop = read.csv("laptop.csv")

train_rows = createDataPartition(y = laptop$Price,
                                 p = 0.80, list = FALSE)

# Training Dataset
laptop_train = laptop[train_rows, ]

# Testing Dataset
laptop_test = laptop[-train_rows, ]
```

## P2: Build a K-NN Model

### Do you need to normalize the data?

### Answer: Yes, the data needs to be normalized.

```
normalize = function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
  }

laptop_normalized = laptop %>% mutate_at(2:8, normalize)
laptop_normalized_train = laptop_normalized[train_rows, ]
laptop_normalized_test = laptop_normalized[-train_rows, ]
```

### Examine the normalized dataset. What went wrong? Why?

### Answer: Because the only value for the field "Screen.Size" is 15, there is a divide by zero error and all the normalized values resulted in a value of NaN.

### Write the code below to fix it:

```
# Don't include the "Screen.Size" field in the normalization function.
laptop_normalized = laptop %>% mutate_at(3:8, normalize)
laptop_normalized_train = laptop_normalized[train_rows, ]
laptop_normalized_test = laptop_normalized[-train_rows, ]
```

## P3: Build a K-NN classifier with a k value of 50

```
pred_knn = knnregTrain(train = laptop_normalized_train[, 3:8],
                        test = laptop_normalized_test[, 3:8],
                        y = laptop_normalized_train[, 1],
                        k = 50)
```

## P4: Evaluate the performance of the K-NN model

```
actual = laptop_normalized_test$Price
error = pred_knn - actual

# Average Error
```

```r
KNN_AE = mean(error)
KNN_AE
```

```
## [1] 1.926922
```

```r
# Mean Absolute Error
KNN_MAE = mean(abs(error))
KNN_MAE
```

```
## [1] 12.90081
```

```r
# Mean Absolute Percentage Error
KNN_MAPE = mean(abs(error / actual))
KNN_MAPE
```

```
## [1] 0.02721691
```

```r
# Root Mean Squared Error
KNN_RMSE = sqrt(mean(error^2))
KNN_RMSE
```

```
## [1] 19.24014
```

```r
# Total Sum of Squared Error
KNN_SSE = sum(error^2)
KNN_SSE
```

```
## [1] 588591
```

## P5: Now, repeat P3 and P4 with different k values (e.g., 50-60). Use a loop. Report the RMSE.

```r
cv = createFolds(y = laptop$Price, k = 60)

for (test_row in cv) {

  laptop_normalized = laptop %>% mutate_at(3:8, normalize)
  laptop_train = laptop_normalized[-test_row, ]
  laptop_test = laptop_normalized[test_row, ]

  pred_knn = knnregTrain(train = laptop_train[, 3:8],
                         test = laptop_test[, 3:8],
                         y = laptop_train[, 1],
                         k = 60)

  # RMSE
  rmse = sqrt(mean((pred_knn - laptop_test[, 1])^2))

  print(rmse)
}
```

```
## [1] 18.1591
## [1] 19.76281
## [1] 20.16131
## [1] 18.73482
## [1] 19.40419
## [1] 18.71618
## [1] 21.80249
## [1] 19.71056
## [1] 17.45733
## [1] 18.4461
## [1] 18.94631
## [1] 19.14078
## [1] 18.35375
## [1] 17.23189
## [1] 18.89389
## [1] 19.35365
## [1] 18.42411
## [1] 19.32266
## [1] 18.90913
## [1] 20.6505
## [1] 19.36423
## [1] 16.96404
## [1] 21.47797
## [1] 18.0963
## [1] 19.91855
## [1] 16.94643
## [1] 18.77599
## [1] 20.36921
## [1] 19.0613
## [1] 17.8734
## [1] 20.31572
## [1] 19.3392
## [1] 19.28582
## [1] 18.8486
## [1] 18.8965
## [1] 17.43267
## [1] 20.58754
## [1] 19.47707
## [1] 16.60578
## [1] 19.71162
## [1] 20.58979
## [1] 17.82
## [1] 16.59068
## [1] 19.95856
## [1] 18.96204
## [1] 19.8268
## [1] 16.92591
## [1] 20.99563
## [1] 20.74548
## [1] 20.63612
## [1] 20.9891
## [1] 19.88239
## [1] 20.49446
## [1] 20.11526
```

```
## [1] 20.57336
## [1] 19.4961
## [1] 16.56597
## [1] 22.21314
## [1] 19.46517
## [1] 18.07051
```

## P6: Build a linear regression model. Use cross-validation. Report the mean RMSE.

```
cv = createFolds(y = laptop$Price, k = 5)

rmse_cv = c()

for (test_row in cv) {

  laptop_train = laptop[-test_row, ]
  laptop_test = laptop[test_row, ]

  lm_model = lm(Price ~ ., data = laptop_train)

  pred_lm = predict(lm_model, laptop_test)

  rmse = sqrt(mean((pred_lm - laptop_test[, 1])^2))

  rmse_cv = c(rmse_cv, rmse)
}

# Mean RMSE
print(mean(rmse_cv))
```

```
## [1] 16.50833
```