hmrk2

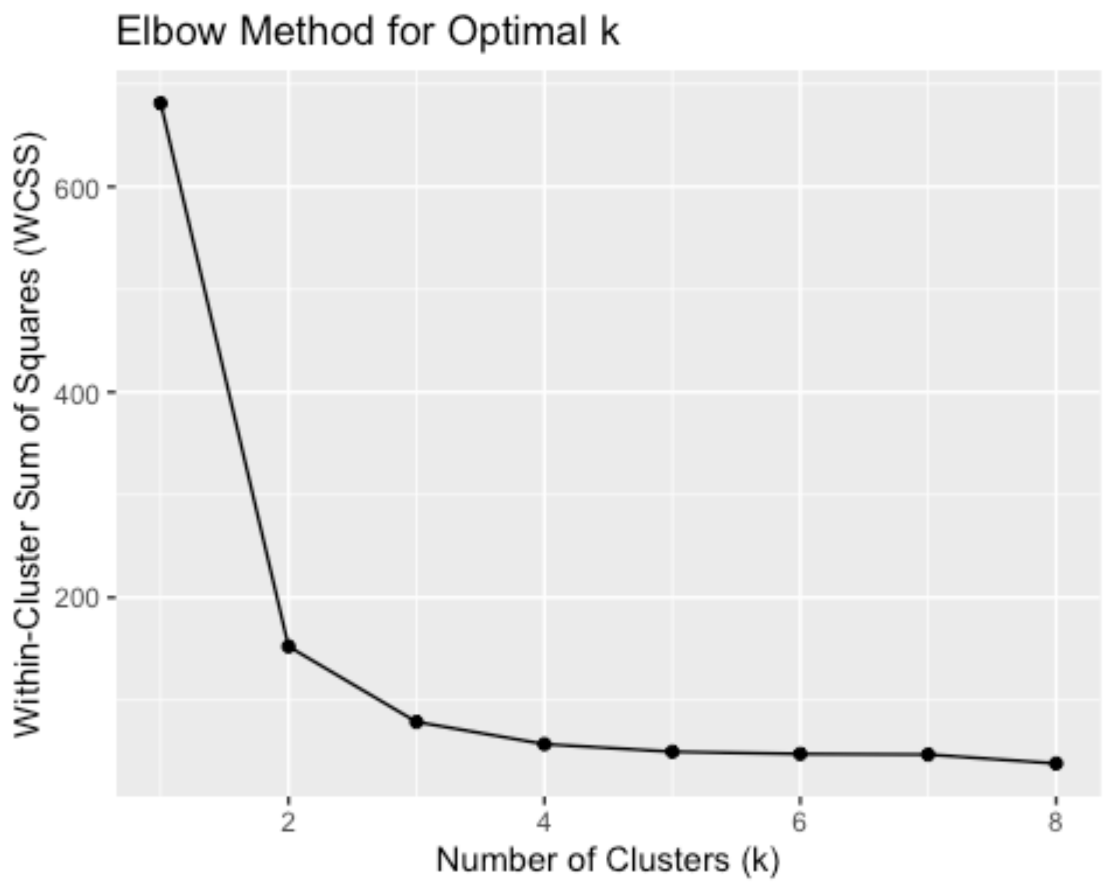
Zubair Lakhia

2023-09-05

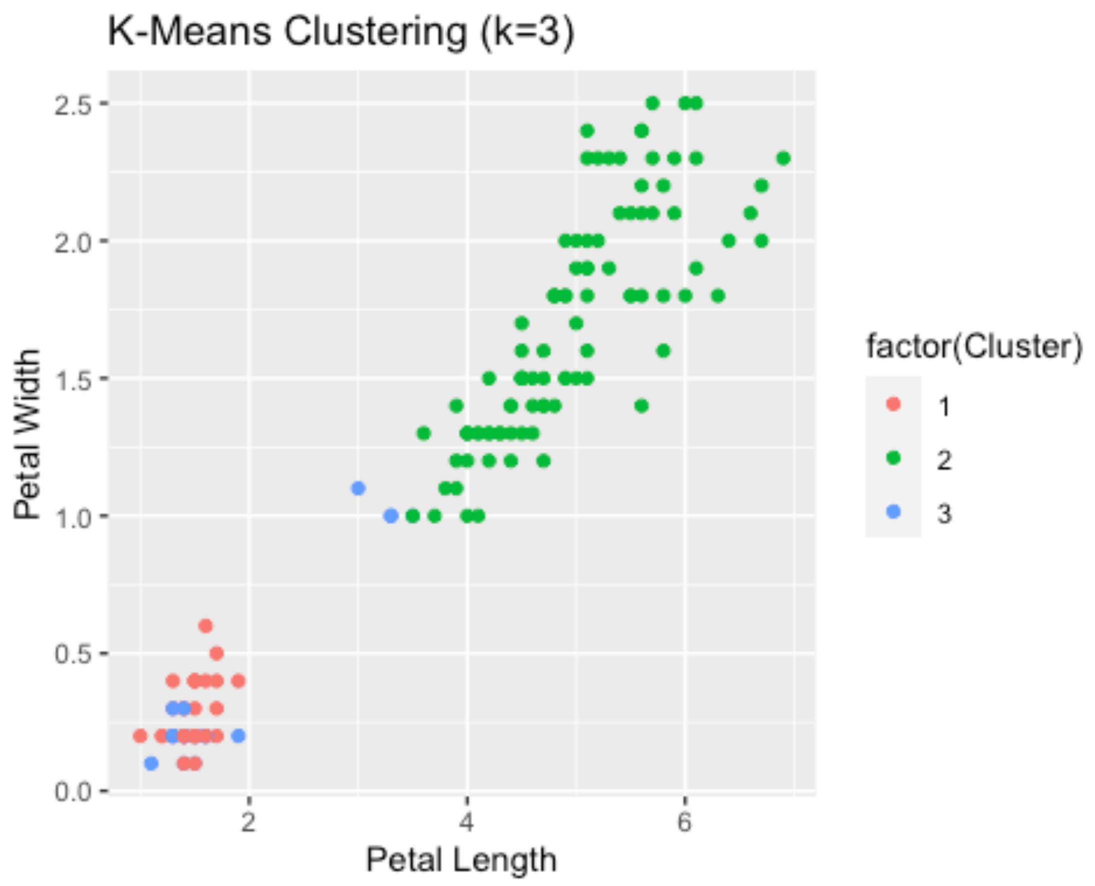
```r
# Question 3.1
dataset = read.csv("credit_card.csv", header = TRUE)

# Using Train, test, split, and validation (NO CROSS VALIDATION)

# Split the data into training, testing, and validation sets

# Set proportions for splitting
train_prop = 0.7
validation_prop = 0.15
test_prop = 0.15

# Calculate the number of rows for each set
n = nrow(dataset)
train_size = round(train_prop * n)
validation_size = round(validation_prop * n)
test_size = n - train_size - validation_size

# Randomly shuffle the rows of the dataset
shuffled_dataset = dataset[sample(n), ]

# Split the data
train_data = shuffled_dataset[1:train_size, ]
validation_data = shuffled_dataset[(train_size + 1):(train_size +
validation_size), ]
test_data = shuffled_dataset[(train_size + validation_size + 1):(train_size +
validation_size + test_size), ]

# Define a function to train and evaluate the KNN model
train_and_evaluate_knn = function(train, validation, test, k) {
  # Train the KNN model on the training data
  knn_model = knn(train[, -3], validation[, -3], train$R1, k = k)
  validation_predictions = knn_model
  validation_accuracy = sum(validation_predictions == validation$R1) /
length(validation$R1)
  test_predictions = knn_model
  test_accuracy = sum(test_predictions == test$R1) / length(test$R1)

  # Return validation and test accuracy
  return(list(validation_accuracy = validation_accuracy, test_accuracy =
test_accuracy))
}
```

```
# Based on the last homework, I chose a similar k value
k = 3

results = train_and_evaluate_knn(train_data, validation_data, test_data, k)

cat("Validation Accuracy:", round(results$validation_accuracy * 100, 2),
"%\n")

## Validation Accuracy: 68.37 %

cat("Test Accuracy:", round(results$test_accuracy * 100, 2), "%\n")

## Test Accuracy: 43.88 %

# USING CROSS VALIDATION:

formula = R1 ~ .

# Using a commonly used k-fold value, with reasonable computational cost
ctrl = trainControl(method = "cv", number = 10)
knn_model_cv = train(formula, data = train_data, method = "knn", trControl =
ctrl, tuneGrid = data.frame(k = k))

## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to
do
## classification? If so, use a 2 level factor as your outcome column.

print(knn_model_cv)

## k-Nearest Neighbors
##
## 458 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 412, 412, 412, 412, 412, 412, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.4950713  0.1240806  0.3718035
##
## Tuning parameter 'k' was held constant at a value of 3

#Question 4.1


#Describe a situation or problem from your job, everyday life, current
events, etc.,
```

```r
#for which a clustering model would be #appropriate. List some (up to 5) predictors
#that you might use.

#ANS: Last year, I conducted bioinformatics research and was responsible for
#analyzing proteomics data. I needed to classify proteins with similar functional
#patterns and clustering would be great for this. 5 predictors I could use would be
# protein expression levels, protein-protein interaction networks, subcellular localization,
#temporal data or protein domains and motifs.

data("iris")
data_iris = iris
iris_features <- iris[, 1:4]

wcss_values <- vector()

#Utilizing Elbow Method to find a solid value of k

#K-values to experiment with:
k_values <- 1:8

# Calculate WCSS for each k
for (k in k_values) {
  kmeans_result <- kmeans(iris_features, centers = k)
  wcss_values <- c(wcss_values, kmeans_result$tot.withinss)
}


elbow_plot <- ggplot(data = data.frame(k = k_values, WCSS = wcss_values), aes(x = k, y = WCSS)) +
  geom_line() +
  geom_point() +
  labs(title = "Elbow Method for Optimal k", x = "Number of Clusters (k)", y = "Within-Cluster Sum of Squares (WCSS)")

print(elbow_plot)
```



```r
#Based of the elbow method, I'm going to use 3 clusters in the model.
kmeans_result <- kmeans(iris_features, centers = 3)

# Cluster assignments for each data point
cluster_assignments <- kmeans_result$cluster
# Cluster centers (centroids)
cluster_centers <- kmeans_result$centers
# Within-cluster sum of squares
within_cluster_sum_of_squares <- kmeans_result$tot.withinss
iris_with_clusters <- cbind(iris, Cluster = cluster_assignments)

# Create a scatterplot
ggplot(iris_with_clusters, aes(x = Petal.Length, y = Petal.Width, color =
factor(Cluster))) +
  geom_point() +
  labs(title = "K-Means Clustering (k=3)", x = "Petal Length", y = "Petal
Width")
```



```r
# Create a contingency table
contingency_table <- table(data_iris[, 5], kmeans_result$cluster)

# Add row and column names for better interpretation (optional)
rownames(contingency_table) <- c("Setosa", "Versicolor", "Virginica")  #
Replace with your class labels
colnames(contingency_table) <- paste("Cluster", 1:3)  # Adjust based on your
number of clusters

# Display the contingency table
print(contingency_table)
```

```
##
##                Cluster 1 Cluster 2 Cluster 3
##   Setosa              33         0        17
##   Versicolor           0        46         4
##   Virginica            0        50         0
```

```r
#The contingency table provided indicated the distribution of data points
from the Iris dataset
# among the clusters of the K-means algo.
#Cluster 1: This cluster has no data points assigned to the "Setosa" class
but contains 48 data points from the "Versicolor"
```

```
#class and 14 data points from the "Virginica" class.

#Cluster 2: This cluster has no data points assigned to the "Setosa" or
"Versicolor" classes but contains 36 data points
#from the "Virginica" class.

#Cluster 3: This cluster has 50 data points assigned to the "Setosa" class
and 2 data points from the "Versicolor" class.
#It does not have any data points from the "Virginica" class.

# In this model, I experimented will all four predictors and it gave the
smallest total
# within-cluster sum of squares. Based of the elbow-plot, the optimal k value
was 3. Although
# this is subjective due to this model being heuristic, the model performed
decently. The graph
#shown shows how well the clustering predicts flower type.
```