Nearest Neighbor Classification (kNN)

C. Perez

Project & Data Description

This code applies the kNN algorithm to a speech recognition dataset offered in the UCI machine learning repository with the aim to correctly classify the language each set of features belongs to. The dataset is made up of 329 observations with 16 numerical variables, and because the variables are all Mel-frequency cepstral coefficients (**mfcc**), scaling is not necessary prior to implementation.

Implementation

It is important that the variable to be classified is a factor and that conversion takes place after loading the data as can be seen in the following code:

```
# Machine Learning - kNN Method

# clear global environment
rm(list = ls(all.names = TRUE))


# libraries
library(tidyverse)
library(caret)
library(class)
library(gmodels)
library(knitr)


# import data
kdata <- read_csv("~/Documents/R/RProjects-Public/Machine-Learning-Data/accent-mfcc-data-1.csv",
                  col_names = TRUE)
kdata$language <- as.factor(kdata$language) # convert to factor


# include comment below if data requires scaling
#kdata <- kdata%>% mutate_at(c(2:length(kdata)), ~(scale(.) %>% as.vector))
```

The kNN algorithm essentially conducts proximity matches to label new points in P-dimensional space, where "P" is the number of features present in the dataset. The importance of "k" is such that k determines the number of neighbors to consider in labeling unseen data points.The default value of k is 1, however really small values of k can lead to overfitting while really large values of k can lead to the opposite. This begs the question... how do I identify the "best" k?

In my research I have found that the best k is subjective and problem specific. For example, when using kNN to confirm a medical diagnosis sometimes a k value that produces higher accuracy can yield false negatives which is of greater concern here than say false negatives for customer segmentation. This means that the "best" k is actually a value that both maximizes accuracy and adheres to the constraints of the desired outcome. For the purpose of this exercise, maximizing accuracy is of the greatest concern and hence the question becomes: how do I choose the value of k that will yield better accuracy?

Some sources suggest that the square root of p serves as a good estimate for k. Seeing as this suggests a value of approx. 17, the following code tests k values ranging from 1 to 17. 100 iterations are run under different 80/20 data partitions and the measure of accuracy is stored for each iteration under each value of k tested. All 100 iterations are then used to create a data frame in which the last column shows the index of the highest accuracy achieved for that iteration (observation). The goal of this is to get an idea, from a statistical perspective, which value of k tends to produce better accuracy.

```r
# declare variable for loops
iterations <- c()
indices_list <- list()
mean_vec_list <- list()


for (i in 1:100){

  indices_list[[i]] <- as.vector(createDataPartition(kdata$language,
                                                     times = 1, p = .80, list = FALSE))
  indices_vec <- as.vector(indices_list[[i]])

  train <- kdata[indices_vec,]
  test <- kdata[-indices_vec,]

  mean_vec <- c()

  for (k in 1:17){

    predictions <- knn(train = train[-1], test = test[-1], cl = train$language, k = k)

    mean_vec[k] <- mean(predictions == test$language)

  }

  mean_vec_list[[i]] <- mean_vec

}

best_k <- c()

for(i in 1:length(mean_vec_list)){

  best_k[i] <- which.max(mean_vec_list[[i]])

}
```

The data frame is as follows:

```r
# isolate best k via summary statistics
df <- data.frame()
```

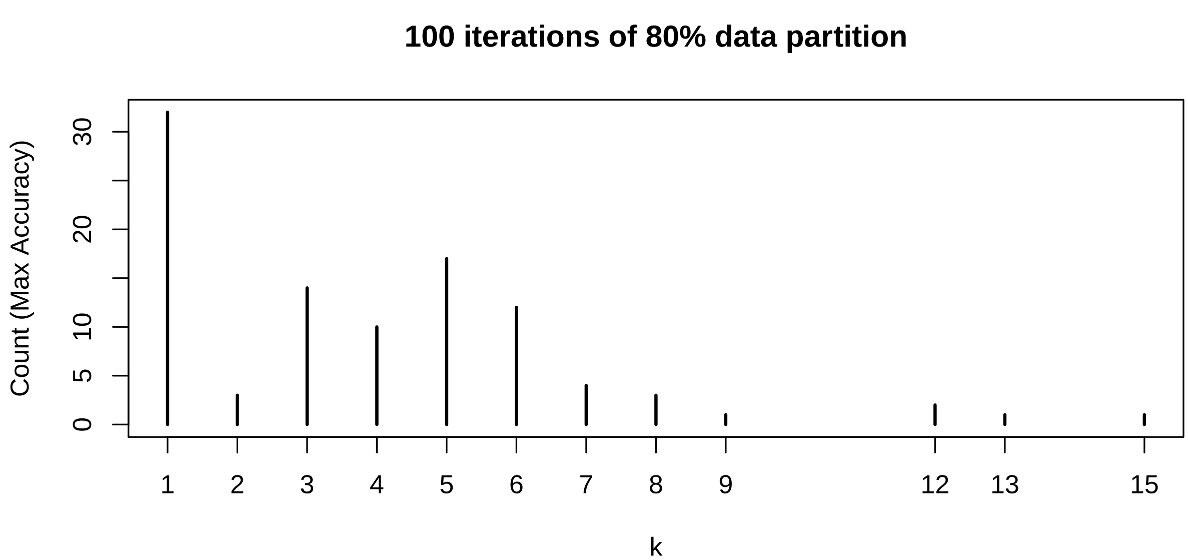
```r
for (i in 1:100){

  df <- rbind(df,mean_vec_list[[i]])

}

colnames(df) <- c(1:17) # rename column vectors to value of K
df <- df %>% mutate(`k (most accurate)` = best_k)

kable(df[1:5,c(1:7,18)], caption = "Full Size: 100 x 18")
```

Table 1: Full Size: 100 x 18

1	2	3	4	5	6	7	k (most accurate)
0.7384615	0.6923077	0.7384615	0.7384615	0.6769231	0.6923077	0.6923077	1
0.7538462	0.7384615	0.8307692	0.8000000	0.8307692	0.8153846	0.7846154	3
0.7384615	0.7384615	0.8153846	0.8461538	0.8153846	0.8307692	0.8307692	4
0.7538462	0.7230769	0.7538462	0.7538462	0.7846154	0.8615385	0.7384615	6
0.7846154	0.6923077	0.7384615	0.6923077	0.7384615	0.7384615	0.7692308	1

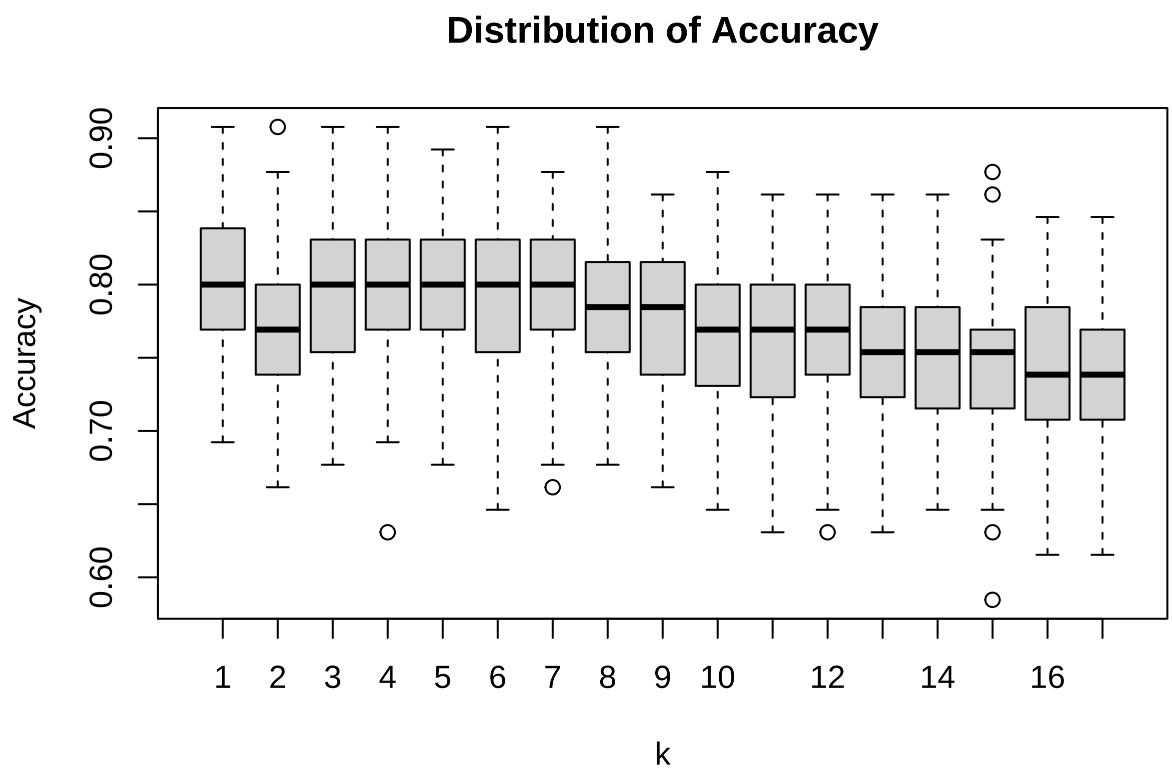
The last column shows the value of k that achieved the highest accuracy for the corresponding iteration. By creating the following plot over the 100 iterations, a rough likelihood that a specific k value will produce the greatest accuracy for a random data partition can be identified:

```r
plot(table(best_k), main = "100 iterations of 80% data partition",
     xlab = "k", ylab = "Count (Max Accuracy)")
```



By viewing the box plot for each k value over the 100 iterations, the distribution of accuracy can be compared across k values. This allows me to select k with some level of certainty that it can, and also has the highest likelihood of, producing the greatest accuracy.

```r
boxplot(df[,1:17], main = "Distribution of Accuracy",
        xlab = "k", ylab = "Accuracy")
```



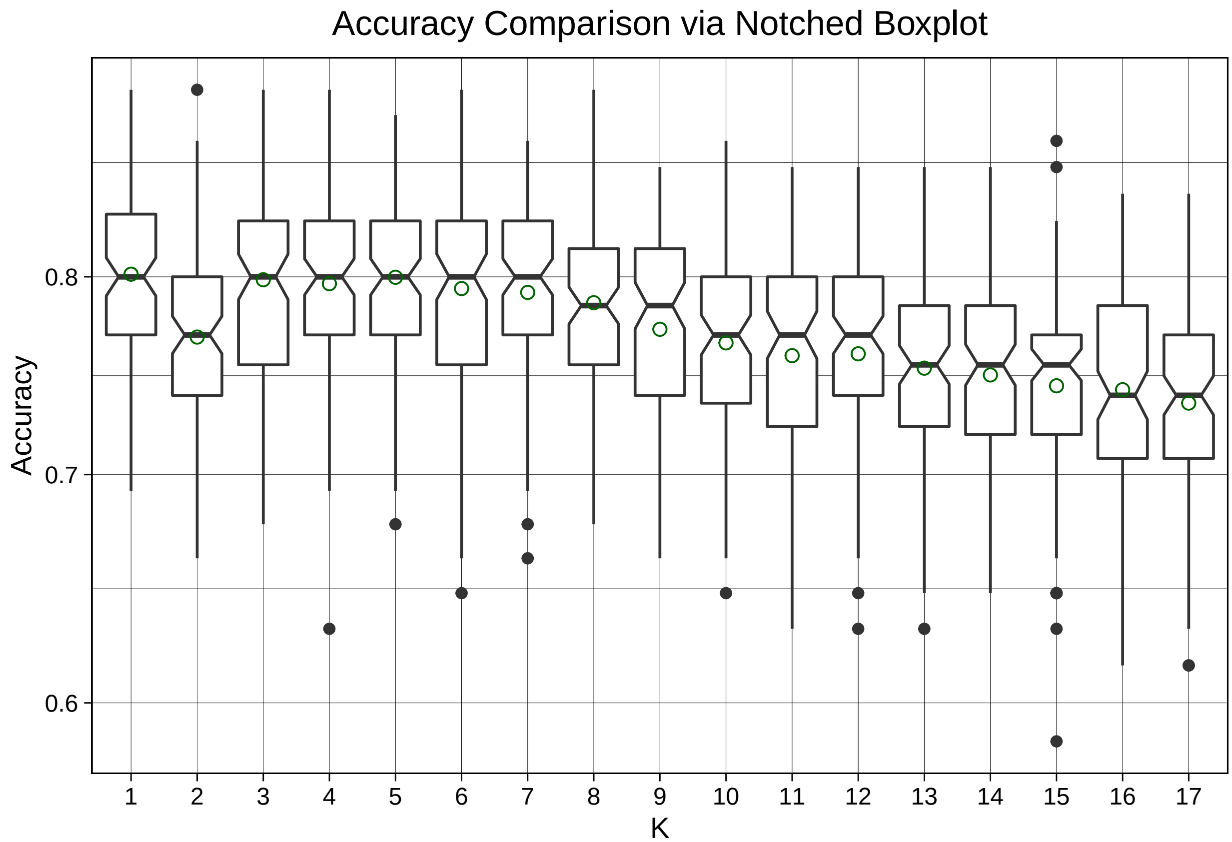
Running a pairwise mean(median) comparison test, and adjusting for the p-value, we would be able to see that there is a significant difference in the level of accuracy achieved between k values. An easier way to identify this is to show the notched box-plot in which the confidence intervals do not overlap for some k-values. This indicates a statistical difference across median values.

```r
dist_df <- mean_vec_list[[1]]

for (i in 2:100){

  dist_df <- as.data.frame(rbind(dist_df,mean_vec_list[[i]]))

}

rownames(dist_df) <- NULL

summary(dist_df)
```

```
       V1               V2               V3               V4        
 Min.   :0.6923   Min.   :0.6615   Min.   :0.6769   Min.   :0.6308  
 1st Qu.:0.7692   1st Qu.:0.7385   1st Qu.:0.7538   1st Qu.:0.7692  
 Median :0.8000   Median :0.7692   Median :0.8000   Median :0.8000  
 Mean   :0.8028   Mean   :0.7697   Mean   :0.7998   Mean   :0.7977  
```

```
 3rd Qu.:0.8346   3rd Qu.:0.8000   3rd Qu.:0.8308   3rd Qu.:0.8308  
 Max.   :0.9077   Max.   :0.9077   Max.   :0.9077   Max.   :0.9077  
       V5               V6               V7               V8        
 Min.   :0.6769   Min.   :0.6462   Min.   :0.6615   Min.   :0.6769  
 1st Qu.:0.7692   1st Qu.:0.7538   1st Qu.:0.7692   1st Qu.:0.7538  
 Median :0.8000   Median :0.8000   Median :0.8000   Median :0.7846  
 Mean   :0.8012   Mean   :0.7955   Mean   :0.7931   Mean   :0.7874  
 3rd Qu.:0.8308   3rd Qu.:0.8308   3rd Qu.:0.8308   3rd Qu.:0.8154  
 Max.   :0.8923   Max.   :0.9077   Max.   :0.8769   Max.   :0.9077  
       V9              V10              V11              V12        
 Min.   :0.6615   Min.   :0.6462   Min.   :0.6308   Min.   :0.6308  
 1st Qu.:0.7385   1st Qu.:0.7346   1st Qu.:0.7231   1st Qu.:0.7385  
 Median :0.7846   Median :0.7692   Median :0.7692   Median :0.7692  
 Mean   :0.7735   Mean   :0.7666   Mean   :0.7603   Mean   :0.7611  
 3rd Qu.:0.8154   3rd Qu.:0.8000   3rd Qu.:0.8000   3rd Qu.:0.8000  
 Max.   :0.8615   Max.   :0.8769   Max.   :0.8615   Max.   :0.8615  
      V13              V14              V15              V16        
 Min.   :0.6308   Min.   :0.6462   Min.   :0.5846   Min.   :0.6154  
 1st Qu.:0.7231   1st Qu.:0.7192   1st Qu.:0.7192   1st Qu.:0.7077  
 Median :0.7538   Median :0.7538   Median :0.7538   Median :0.7385  
 Mean   :0.7537   Mean   :0.7502   Mean   :0.7449   Mean   :0.7426  
 3rd Qu.:0.7846   3rd Qu.:0.7846   3rd Qu.:0.7692   3rd Qu.:0.7846  
 Max.   :0.8615   Max.   :0.8615   Max.   :0.8769   Max.   :0.8462  
      V17        
 Min.   :0.6154  
 1st Qu.:0.7077  
 Median :0.7385  
 Mean   :0.7362  
 3rd Qu.:0.7692  
 Max.   :0.8462  
```

```r
# tidy data frame
dist_df_tidy <- gather(data = dist_df, key = "K", value = "Accuracy")
dist_df_tidy$K <- str_replace(dist_df_tidy$K, "V", "")
dist_df_tidy$K <- factor(dist_df_tidy$K, ordered = TRUE, levels = c(1:17))

ggplot(data = dist_df_tidy, mapping = aes(x = K, y = Accuracy))+
  geom_boxplot(notch = TRUE, orientation = "x")+ # median
  stat_summary(fun=mean, geom="point", shape=21, size=2, color = "darkgreen")+ # mean
  scale_y_log10()+
  labs(title = "Accuracy Comparison via Notched Boxplot")+
  theme_linedraw()+
  theme(plot.title = element_text(hjust = .5))
```



By viewing the distribution of accuracy, the clear competitors are 1,5, and 7. Odd numbers reduce the possibility of a tie vote among neighbors and as such are better suited for k. I chose k=5 as it is better in my opinion to not have 1 neighbor with 100% voting power, and by reducing it to 20% allows for a reasonable voting process in order to classify. This is at the expense of the negligible difference in the minimums as seen from the box-plot. Had these minimums differed substantially, I might be inclined to reduce/increase the value of k.

```
# choose k = 5

indices <- as.vector(createDataPartition(kdata$language,times = 1, p = .80, list = FALSE))

train <- kdata[indices,]
test <- kdata[-indices,]

predictions <- knn(train = train[-1], test = test[-1], cl = train$language, k = 5)

mean(predictions == test$language)
```

```
[1] 0.8307692
```

The following table, while originally intended to display chi-square metrics for correlation of nominal features, provides metrics on how well the classification performed for the given test set.

```
CrossTable(x = predictions, y = test$language, prop.chisq = FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  65


             | test$language
 predictions |        ES |        FR |        GE |        IT |        UK |        US | Row Total |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
          ES |         4 |         0 |         0 |         0 |         0 |         0 |         4 |
             |     1.000 |     0.000 |     0.000 |     0.000 |     0.000 |     0.000 |     0.062 |
             |     0.800 |     0.000 |     0.000 |     0.000 |     0.000 |     0.000 |           |
             |     0.062 |     0.000 |     0.000 |     0.000 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
          FR |         0 |         4 |         0 |         0 |         0 |         1 |         5 |
             |     0.000 |     0.800 |     0.000 |     0.000 |     0.000 |     0.200 |     0.077 |
             |     0.000 |     0.667 |     0.000 |     0.000 |     0.000 |     0.030 |           |
             |     0.000 |     0.062 |     0.000 |     0.000 |     0.000 |     0.015 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
          GE |         0 |         0 |         5 |         1 |         0 |         0 |         6 |
             |     0.000 |     0.000 |     0.833 |     0.167 |     0.000 |     0.000 |     0.092 |
             |     0.000 |     0.000 |     0.833 |     0.167 |     0.000 |     0.000 |           |
             |     0.000 |     0.000 |     0.077 |     0.015 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
          IT |         0 |         0 |         0 |         4 |         0 |         0 |         4 |
             |     0.000 |     0.000 |     0.000 |     1.000 |     0.000 |     0.000 |     0.062 |
             |     0.000 |     0.000 |     0.000 |     0.667 |     0.000 |     0.000 |           |
             |     0.000 |     0.000 |     0.000 |     0.062 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
          UK |         0 |         0 |         0 |         0 |         6 |         1 |         7 |
             |     0.000 |     0.000 |     0.000 |     0.000 |     0.857 |     0.143 |     0.108 |
             |     0.000 |     0.000 |     0.000 |     0.000 |     0.667 |     0.030 |           |
             |     0.000 |     0.000 |     0.000 |     0.000 |     0.092 |     0.015 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
          US |         1 |         2 |         1 |         1 |         3 |        31 |        39 |
             |     0.026 |     0.051 |     0.026 |     0.026 |     0.077 |     0.795 |     0.600 |
             |     0.200 |     0.333 |     0.167 |     0.167 |     0.333 |     0.939 |           |
             |     0.015 |     0.031 |     0.015 |     0.015 |     0.046 |     0.477 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
Column Total |         5 |         6 |         6 |         6 |         9 |        33 |        65 |
             |     0.077 |     0.092 |     0.092 |     0.092 |     0.138 |     0.508 |           |
-------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
```

Conclusion

The top value along the diagonal of the above table shows the percentage of accuracy in correctly classifying each language based on the k value chosen. Here it was easy to try to identify a statistical of way of coming to a good choice for k, however thist strategy needs to be modified for larger datasets as it will become time consuming and is not practical for applications that require quick classification, such as sign recognition in autonomous vehicles.