Essay_4

Wine Quality Prediction with KNN

Authors: Preston O'Connor, Anthony Yasan, Matthew Jacob, Khoa Dao, Nick Wierzbowski

Date: 4/14/2025

Introduction

Our model uses the K nearest neighbor method to see the relation between chemical properties of different wines and the quality ratings of wine experts. The data set is composed of these chemical properties and quality ratings of northern Portuguese 'Vinho Verde' wines, which seems to have been gathered for a research paper similar to ours by Cortez et al. in 2009. The relationship between the chemical properties of wine and the "subjective" flavor, encompassing taste and smell, actually seems to be pretty complex. According to Shapin, who wrote a paper detailing how the American wine tasting language and process was born out of contributions from wine experts, producers and chemists, the flavor or quality is not only affected by flavorants and odorants, which often only have an impact on the smell, but also much less reliable prior experience and expectations. In fact the current language popularized by people like Maynard Amerine and his associates in the post-war period in itself influences how we perceive different flavors, and potentially thus the impact of the chemical flavorants on the quality. All of that is to say that the taste of wine is influenced not just by measurable chemicals but also social factors, especially when considering the professional wine tasting industry and a common language used by them.

```
The packages we used are class, which has functions for categorization useful for KNN impleme
```

As for our results, the model seems to have been fairly successful at predicting wine quality when it was mid range, and more common. For the lower and higher ends of quality ratings the model performed worse, which can both be attributable to lower amounts of data for those categories and perhaps increasing deviation among those categories the further they get from an ‘average’ wine.

```r
#libraries given in Step 1 of the slides
# install.packages("class") # For KNN
# install.packages("tidyverse") # For visualization
# install.packages("corrplot") # correlation matrix visualization
# install.packages("ggplot2")# used to view correltation matrix
# install.packages("leaps") #for best subset sum
# install.packages("pROC") # for the ROC analaysys
library(class) # for KNN Implementation
library(tidyverse) # for visualization
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(corrplot)
```

```
corrplot 0.95 loaded
```

```r
library(ggplot2)
library(caret) # for KNN
```

```
Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift
```

```r
library(leaps)
library(pROC)
```

```
Type 'citation("pROC")' for a citation.

Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```r
#loading the data set
data <- read.csv("winequality-red.csv")
#names(wine)
summary(data)
```

```
 fixed.acidity   volatile.acidity  citric.acid    residual.sugar  
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900  
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900  
 Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200  
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539  
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600  
 Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500  
   chlorides       free.sulfur.dioxide total.sulfur.dioxide    density      
 Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901  
 1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956  
 Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968  
 Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967  
 3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978  
 Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037  
       pH          sulphates         alcohol          quality     
 Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000  
 1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000  
 Median :3.310   Median :0.6200   Median :10.20   Median :6.000  
 Mean   :3.311   Mean   :0.6581   Mean   :10.42   Mean   :5.636  
 3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000  
 Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000  
```

Data Description

Data source

The dataset used in this project is the Red Wine Quality dataset from the UCI Machine Learning Repository, also available on Kaggle (https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009). This dataset was compiled by Paulo Cortez et al. (2009) and is related to red Vinho Verde wine samples from the north of Portugal.

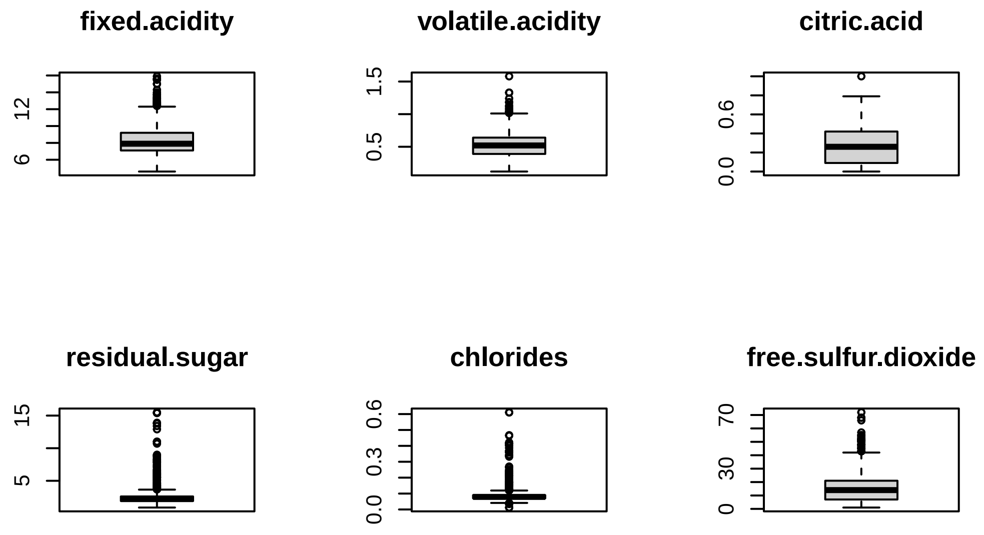
Data structure and Variables

The dataset contains 1599 observations (rows) and 12 variables (11 numeric input variables and 1 output variable: quality). The variables represent physicochemical attributes of the wine samples:

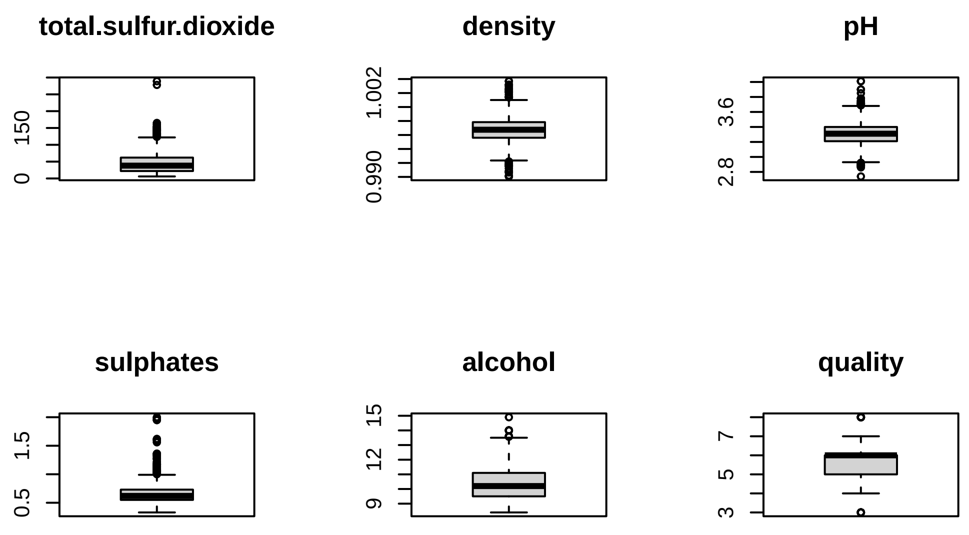
- **fixed acidity**: Refers primarily to tartaric acid, which is one of the main acids found naturally in grapes.
- **volatile acidity**: Measures the amount of acetic acid (vinegar) in the wine.
- **citric acid**: A natural acid found in small quantities in wine that can add freshness and flavor.
- **residual sugar**: Represents the amount of sugar remaining after fermentation.
- **chlorides**: Indicates the amount of salt in the wine, which can affect the taste and preservation.
- **free sulfur dioxide**: Refers to the part of sulfur dioxide (SO) that is not bound to other molecules and is available to act as an antioxidant and antimicrobial agent.
- **total sulfur dioxide**: Includes both free and bound forms of SO .
- **density**: The density of wine, which is influenced by the sugar and alcohol content.
- **pH**: Indicates how acidic or basic the wine is.
- **sulphates**: Sulfate compounds can contribute to the wine's flavor and preservation.
- **alcohol**: The percentage of ethanol by volume in the wine.
- **quality**: This is the response variable, a sensory score assigned by professional tasters ranging from 0 to 10. It reflects the overall quality of the wine sample based on taste, aroma, and balance.

Data cleaning

```r
# Boxplots to detect outliers
par(mfrow = c(2, 3))
for (i in 1:6) {
  boxplot(data[[i]], main = names(data)[i])
}
```



```r
# Plot remaining 6 variables
par(mfrow = c(2, 3))
for (i in 7:12) {
  boxplot(data[[i]], main = names(data)[i])
}
```



```r
par(mfrow = c(1, 1))  # Reset

# Function to remove outliers beyond 1.5 * IQR
remove_outliers_IQR <- function(df, column) {
  #df <- df %>% filter(!is.na(df[[column]]))  # Remove missing data
  df <- df %>% filter(!is.na(!!sym(column)))
  Q1 <- quantile(df[[column]], 0.25, na.rm = T)
  Q3 <- quantile(df[[column]], 0.75, na.rm = T)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  df %>% filter(df[[column]] >= lower_bound & df[[column]] <= upper_bound)
}

# Remove outliers for all relevant columns
data_cleaned <- data
columns <- c("fixed.acidity", "volatile.acidity", "citric.acid",
             "residual.sugar", "chlorides", "free.sulfur.dioxide",
             "total.sulfur.dioxide", "density", "pH",
             "sulphates", "alcohol")

data_cleaned <- data
```

```
for (col in columns) {
  data_cleaned <- remove_outliers_IQR(data_cleaned, col)
}

# Summary of the cleaned data
summary(data_cleaned)
```

```
 fixed.acidity    volatile.acidity  citric.acid    residual.sugar 
 Min.   : 5.100   Min.   :0.1200   Min.   :0.000   Min.   :1.200  
 1st Qu.: 7.100   1st Qu.:0.3900   1st Qu.:0.080   1st Qu.:1.900  
 Median : 7.800   Median :0.5200   Median :0.240   Median :2.100  
 Mean   : 8.147   Mean   :0.5222   Mean   :0.246   Mean   :2.181  
 3rd Qu.: 9.000   3rd Qu.:0.6300   3rd Qu.:0.390   3rd Qu.:2.400  
 Max.   :12.300   Max.   :1.0050   Max.   :0.730   Max.   :3.600  
   chlorides      free.sulfur.dioxide total.sulfur.dioxide    density      
 Min.   :0.0420   Min.   : 1.00       Min.   :  6.00       Min.   :0.9926  
 1st Qu.:0.0690   1st Qu.: 8.00       1st Qu.: 22.00       1st Qu.:0.9955  
 Median :0.0780   Median :13.00       Median : 35.00       Median :0.9966  
 Mean   :0.0783   Mean   :14.79       Mean   : 40.95       Mean   :0.9966  
 3rd Qu.:0.0870   3rd Qu.:20.00       3rd Qu.: 54.00       3rd Qu.:0.9975  
 Max.   :0.1160   Max.   :40.00       Max.   :113.00       Max.   :1.0004  
       pH          sulphates         alcohol          quality     
 Min.   :2.980   Min.   :0.3300   Min.   : 8.70   Min.   :3.000  
 1st Qu.:3.230   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000  
 Median :3.330   Median :0.6100   Median :10.10   Median :6.000  
 Mean   :3.325   Mean   :0.6284   Mean   :10.35   Mean   :5.637  
 3rd Qu.:3.400   3rd Qu.:0.7000   3rd Qu.:11.00   3rd Qu.:6.000  
 Max.   :3.680   Max.   :0.9400   Max.   :13.00   Max.   :8.000  
```

```
# New size of the cleaned dataset
nrow(data_cleaned)
```

```
[1] 1135
```

The dataset contains 1135 observations (rows) after removing outliers.

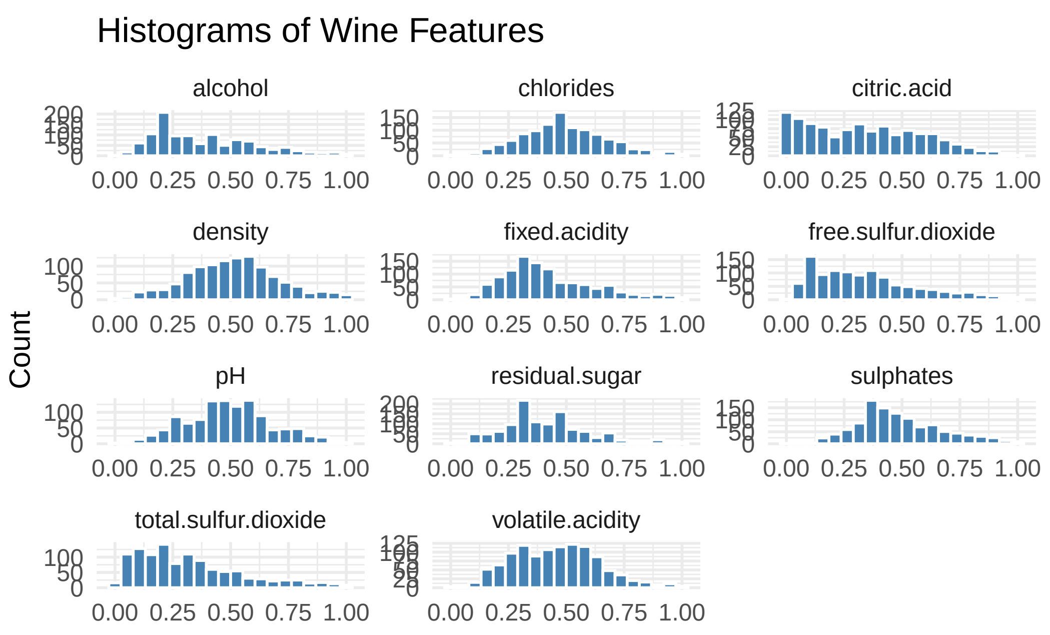
Normalize Data

```r
normalize <- function(x) { (x - min(x)) / (max(x) - min(x)) }

data_norm <- as.data.frame(lapply(data_cleaned[, 1:11], normalize))
head(data_norm)
```

```
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1     0.3194444        0.6553672  0.00000000      0.2916667 0.4594595
2     0.3750000        0.8587571  0.00000000      0.5833333 0.7567568
3     0.3750000        0.7231638  0.05479452      0.4583333 0.6756757
4     0.8472222        0.1807910  0.76712329      0.2916667 0.4459459
5     0.3194444        0.6553672  0.00000000      0.2916667 0.4594595
6     0.3194444        0.6101695  0.00000000      0.2500000 0.4459459
  free.sulfur.dioxide total.sulfur.dioxide   density        pH sulphates
1           0.2564103            0.2616822 0.6683673 0.7571429 0.3770492
2           0.6153846            0.5700935 0.5408163 0.3142857 0.5737705
3           0.3589744            0.4485981 0.5663265 0.4000000 0.5245902
4           0.4102564            0.5046729 0.6938776 0.2571429 0.4098361
5           0.2564103            0.2616822 0.6683673 0.7571429 0.3770492
6           0.3076923            0.3177570 0.6683673 0.7571429 0.3770492
    alcohol
1 0.1627907
2 0.2558140
3 0.2558140
4 0.2558140
5 0.1627907
6 0.1627907
```

Data visualization

```r
# Reshape data to long format
wine_long <- pivot_longer(data_norm, cols = 1:11, names_to = "variable",
 ↪  values_to = "value")

# Create faceted histogram plot
ggplot(wine_long, aes(x = value)) +
  geom_histogram(bins = 20, fill = "steelblue", color = "white") +
  facet_wrap(~variable, scales = "free", ncol = 3) +
  theme_minimal() +
  labs(title = "Histograms of Wine Features", x = "", y = "Count")
```



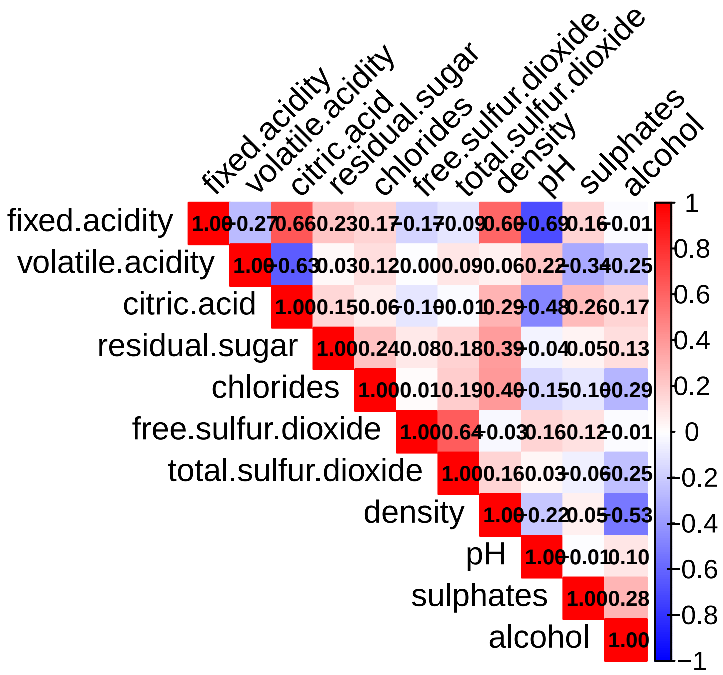
Correlation matrix

```
# Compute correlation matrix for numeric input variables
cor_matrix <- cor(data_norm[, 1:11])  # Exclude 'quality' if you're only
 ↪  interested in inputs
print(round(cor_matrix, 2))       # Round for readability
```

```
                     fixed.acidity volatile.acidity citric.acid residual.sugar
fixed.acidity                 1.00            -0.27        0.66           0.23
volatile.acidity             -0.27             1.00       -0.63           0.03
citric.acid                   0.66            -0.63        1.00           0.15
residual.sugar                0.23             0.03        0.15           1.00
chlorides                     0.17             0.12        0.06           0.24
free.sulfur.dioxide          -0.17             0.00       -0.10           0.08
total.sulfur.dioxide         -0.09             0.09       -0.01           0.18
density                       0.60             0.06        0.29           0.39
pH                           -0.69             0.22       -0.48          -0.04
sulphates                     0.16            -0.34        0.26           0.05
alcohol                      -0.01            -0.25        0.17           0.13
                     chlorides free.sulfur.dioxide total.sulfur.dioxide density
```

```
fixed.acidity             0.17               -0.17                -0.09    0.60
volatile.acidity          0.12                0.00                 0.09    0.06
citric.acid               0.06               -0.10                -0.01    0.29
residual.sugar            0.24                0.08                 0.18    0.39
chlorides                 1.00                0.01                 0.19    0.40
free.sulfur.dioxide       0.01                1.00                 0.64   -0.03
total.sulfur.dioxide      0.19                0.64                 1.00    0.16
density                   0.40               -0.03                 0.16    1.00
pH                       -0.15                0.16                 0.03   -0.22
sulphates                -0.10                0.12                -0.06    0.05
alcohol                  -0.29               -0.01                -0.25   -0.53
                        pH sulphates alcohol
fixed.acidity        -0.69      0.16   -0.01
volatile.acidity      0.22     -0.34   -0.25
citric.acid          -0.48      0.26    0.17
residual.sugar       -0.04      0.05    0.13
chlorides            -0.15     -0.10   -0.29
free.sulfur.dioxide   0.16      0.12   -0.01
total.sulfur.dioxide  0.03     -0.06   -0.25
density              -0.22      0.05   -0.53
pH                    1.00     -0.01    0.10
sulphates            -0.01      1.00    0.28
alcohol               0.10      0.28    1.00
```

```r
corrplot(cor_matrix, method = "color", type = "upper",
         tl.col = "black", tl.srt = 45,
         addCoef.col = "black", number.cex = 0.7,
         col = colorRampPalette(c("blue", "white", "red"))(200))
```



Analysis

Best Subset

```
data_combined <- cbind(data_norm, quality = data_cleaned$quality)
 nrow(data_norm)
```

```
[1] 1135
```

```
# due to 2 instances of the value of qaulity 3 remove the data
data_combined <- data_combined %>%
  filter(quality !=3)
 # remove the data with the two points of classification as three




nrow(data_combined)
```

```
[1] 1133
```

Analysis

Train and Test Split with the optimized K

- We used an 80:20 ratio for our training and test sets to provide the model with a sizable amount of data to train on, while still maintaining a good mechanism for evaluating its performance. This yielded a training set of 906 and a testing set of 227. Using all the columns of the data set allows the KNN model to consider every feature that could potentially have an impact on wine quality, resulting in more accurate and informed predictions. Using the full set of variables, we are not in danger of leaving out any important information.

```
# First Implement the Seed and split the Training and Testing Data
set.seed(12)
#selected_cols <- c(2,10,11) # this received 61.25% with k =20
#selected_cols <- c(2,3,4,5, 7, 8, 9,10,11) # this received 61.87% with k =
↪  31
index <- sample(1:nrow(data_combined), 0.8 * nrow(data_combined))

train_data <- data_combined[index, 1:11]   # Features: columns 1 to 11
test_data <- data_combined[-index, 1:11]

cat("Number of training samples:", nrow(train_data), "\n")
```

```
Number of training samples: 906
```

```
cat("Number of testing samples:", nrow(test_data), "\n")
```

```
Number of testing samples: 227
```

```
train_labels <- data_combined[index, 12]   # Target: column 12
test_labels <- data_combined[-index, 12]

# The numbers are treated as categorical not to calculate the average from
train_labels <- as.factor(train_labels)
test_labels <- as.factor(test_labels)

k_values <- 2:60

# Train KNN with k = 2-60
```

```r
accuracy_list <- c()

for (k in k_values) {
  knn_pred <- knn(train = train_data, test = test_data, cl = train_labels, k
 ↪  = k)
  acc <- mean(knn_pred == test_labels)
  accuracy_list <- c(accuracy_list, acc)
}

# Find the best k
best_k <- k_values[which.max(accuracy_list)]
best_acc <- max(accuracy_list)

cat("The Best k:", best_k, "\n")
```

```
The Best k: 6
```

```r
cat("The Best Accuracy:", round(best_acc, 4), "\n")
```

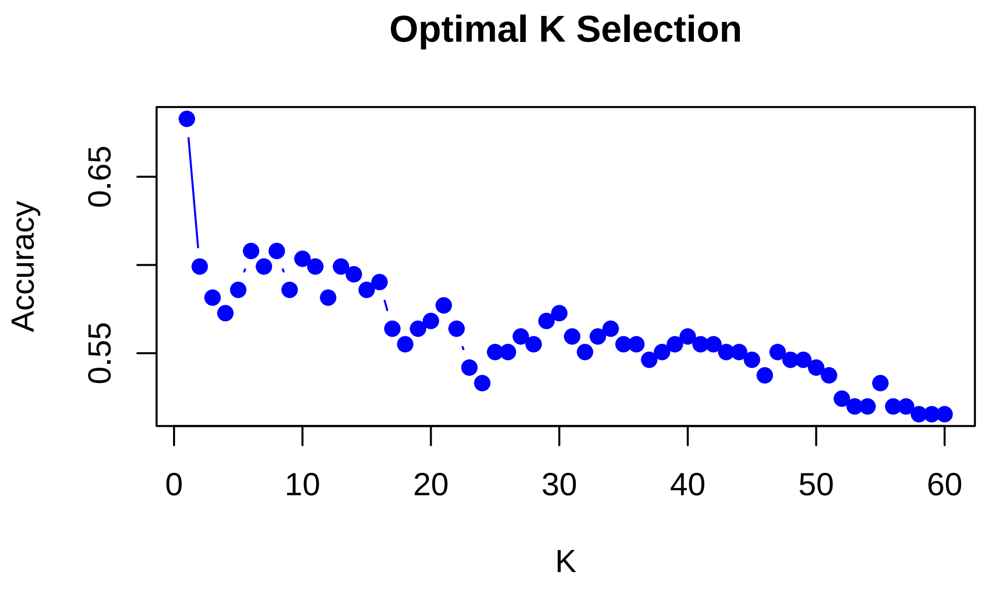
```
The Best Accuracy: 0.6211
```

```r
# Final model with best k
final_knn <- knn(train = train_data, test = test_data, cl = train_labels, k =
 ↪  best_k)
```

Model Evaluation

The K is optimized

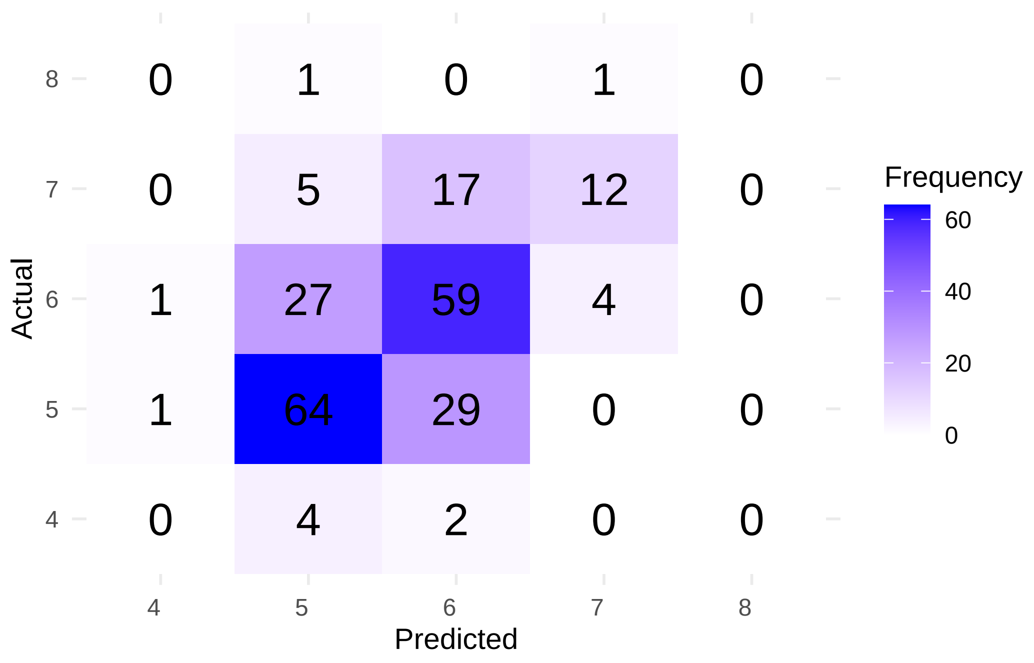
```r
set.seed(13)
k_values <- 1:60
accuracies <- sapply(k_values, function(k) {
pred <- knn(train_data, test_data, cl = train_labels, k = k)
mean(pred == test_labels)
})
# Plot accuracy vs. K
plot(k_values, accuracies, type = "b", col = "blue", pch = 19,
xlab = "K", ylab = "Accuracy", main = "Optimal K Selection")
```



- The plot illustrates the accuracy of the KNN model for different values of K, with optimal accuracy at low values of K, namely in the range of K equal to 5. As K increases, the model's accuracy has a tendency to fall, meaning larger neighborhoods reduce the model's ability to make correct predictions. Therefore, a lower value of K is more appropriate for this data set, sacrificing sensitivity to local trends for overall performance.

The Confusion Matrix

```
conf_matrix <- table(Predicted = final_knn, Actual = test_labels)


conf_matrix_df <- as.data.frame(as.table(conf_matrix))

ggplot(data = conf_matrix_df, aes(x = Predicted, y = Actual, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "black", size = 6) +
  scale_fill_gradient(low = "white", high = "blue") +
  theme_minimal() +
  labs(x = "Predicted", y = "Actual", fill = "Frequency") +
  theme(axis.text.x = element_text(hjust = 1))
```



- Based on the confusion matrix, the KNN model is moderately accurate with the majority of the correct predictions clustered around classes 5 and 6. Class 6 has the maximum true positive value (61), indicating that the model is most confident in classifying this class correctly; however, there is a large misclassification between neighboring classes: a majority of class 5 wines are forecast as class 6 (64), and the class 6 wines are largely forecast as 5 (24) or 7 (6). Class 7 forecasts are everywhere, and the model performs very badly with classes 4 and 8, correctly classifying none within these groups. This means that the model is most powerful at detecting common, central classes, but less sensitive to less common or edge-case classes. Overall, the model gives general trends regarding wine quality, but may be assisted by algorithms that provide improved performance on underrepresented classes.

Multiple of Class Precision, Recall, and F1 Score

```r
conf_matrix <- confusionMatrix(final_knn, test_labels)
# print(conf_matrix) # for debugging purposes

cm_by_class <- conf_matrix$byClass

if (is.null(dim(cm_by_class))) {
  precision <- cm_by_class["Pos Pred Value"]
  recall <- cm_by_class["Sensitivity"]
  f1 <- 2 * (precision * recall) / (precision + recall)
  metrics_df <- data.frame(
    Class = levels(test_labels),
    Precision = round(precision, 3),
    Recall = round(recall, 3),
    F1_Score = round(f1, 3)
  )
} else {
  precision <- cm_by_class[, "Pos Pred Value"]
  recall <- cm_by_class[, "Sensitivity"]
  f1 <- 2 * (precision * recall) / (precision + recall)
  metrics_df <- data.frame(
    Class = rownames(cm_by_class),
    Precision = round(precision, 3),
    Recall = round(recall, 3),
    F1_Score = round(f1, 3)
  )
}

print(metrics_df)
```

```
            Class Precision Recall F1_Score
Class: 4 Class: 4     0.000  0.000      NaN
Class: 5 Class: 5     0.634  0.681    0.656
Class: 6 Class: 6     0.551  0.648    0.596
Class: 7 Class: 7     0.706  0.353    0.471
Class: 8 Class: 8       NaN  0.000      NaN
```

- Class 4: The model did not predict any wines as quality 4. Precision and recall are both 0, indicating that it completely missed this class.
- Class 5: For class 5, the model is most accurate, with 65.3% precision and 68.1% recall, meaning it correctly identifies most wines with this quality.
- Class 6: Class 6 predictions are good with precision 56.5% and recall 67%, indicating moderate accuracy and good coverage.
- Class 7: The model performs badly on class 7 with only 50% accuracy and 29.4% recall, thus missing most of the true class 7 wines.
- Class 8: The model doesn't predict class 8 at all. Both precision and recall are 0, which means it completely fails to predict this class.
- delete this point is so i know i submitted the right draft on 4/11

Conclusion & Summary

In this project, we used the K-Nearest Neighbors (KNN) algorithm to predict the quality of Portuguese ‘Vinho Verde’ wines based on a range of properties, including acidity, sugar content, and pH. After some data cleaning, normalization, and visualization, we trained and evaluated our model using an 80:20 training/test split. We determined the optimal values for K via different techniques, and then tested our optimized model using a confusion matrix. Our model achieved reasonable accuracy in predicting mid-range wine qualities, especially for the more common quality ratings (classes 5 and 6). However, the model faced challenges with predicting extreme quality classes (classes 4, 7, and 8). This may be as a result of there being less representation for particularly low quality and high quality wines. Overall, the model is able to accurately suggest certain trends regarding wine quality, particularly for mid-range wines, but it needs bolstering via algorithms that can better incorporate underrepresented classes into the model. This model could also be improved by including additional variables that account for social, environmental, and historical factors that may impact wine quality.

Sources

- Shapin, Steven. “A Taste of Science: Making the Subjective Objective in the California Wine World.” Social Studies of Science 46, no. 3 (2016): 436–60. http://www.jstor.org/stable/26099849.
- P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.
- Choudhury, P. (2020, June 26). Best way to learn KNN algorithm using R programming. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/