Homework 2 Stats 0218

Heyyy!

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ ggplot2   3.5.1     ✔ tibble    3.2.1
✔ lubridate 1.9.4     ✔ tidyr     1.3.1
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

```
wine_data <- read_csv("~/Downloads/Amuna_wine_data.csv")
```

```
Rows: 500 Columns: 5
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (1): type
dbl (4): wine_id, alcohol, volatile acidity, sulphates

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
wine_original <- read_csv("~/Downloads/Wine_Data.csv")
```

```
Rows: 5997 Columns: 13
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (1): type
dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
wine_data_long <- wine_original |>
  pivot_longer(cols = -c(quality, type), #changing it from wide to long to facet graphs
               names_to = "variable",
               values_to = "value")

wine_data_long |>
ggplot() +
  geom_point(aes(y=value, x=quality)) +
               facet_wrap(~variable, scales = "free_y") +
  labs(title = "Relationship Between Quality and Each Wine Property",
       x = "Quality",
       y = "Each Property Variable") +
  theme_minimal()
```

```
Warning: Removed 37 rows containing missing values or values outside the scale range
(`geom_point()`).
```

Here I am exploring the relationships between quality and each physiochemical property in order to choose the top three explanatory variables for my wine prediction model. Density, free sulfur dioxide, residual sugar, citric acid, fixed acidity, and pH seem to show weak relationships with quality in this dataset since there are very little variations in them across different quality of wines.

Potential variables with their reasoning: - Alcohol (most higher quality wine tend to have higher levels of alcohol) - Chlorides (most higher quality wine tend to have lower levels of chloride) - Sulphates (indicating a likely negative relationship with quality) - Volatile acidity (also indicating a likely negative relationship wth quality) - Total sulfur dioxide (also indicating a likely negative relationship with quality)

To be honest, I randomly chose three variables from the list to get the dataset since it is difficult to tell, but upon reflection, I decide to look at the correlation more closely.

Here let's calculate each property's correlation coefficient with quality. I can see that alcohol, density, volatile acidity, and chlorides have the strongest correlation with quality. For the remaining variables, the relationship is very tiny. I may have chosen density instead, but I think the variables I already chose should be good.

```
library(dplyr)

cor_with_quality <- wine_original |>
  select_if(is.numeric) |>
  cor(use = "complete.obs") |>
  as.data.frame() |>
  tibble::rownames_to_column(var = "variable") |>
  filter(variable != "quality") |>
  rename(correlation = quality)


cor_with_quality %>%
  ggplot(aes(x = reorder(variable, correlation), y = correlation)) +
  geom_col(fill = "red", alpha = 0.7) +
  geom_text(aes(label = round(correlation, 2))) +
  coord_flip() +
  labs(title = "Correlation of Wine Characteristics with Quality",
       x = "Wine Characteristics",
       y = "Correlation Coefficient") +
  theme_minimal()
```

Exploring the three variables:

```
wine_data |>
  ggplot() +
  geom_point(aes(x=`volatile acidity`, y=sulphates, color = alcohol)) +
              facet_wrap(~type, scales = "fixed" ) +
  scale_color_viridis_c()
```

```
wine_data |>
  ggplot() +
  geom_point(aes(x=`volatile acidity`, y=alcohol, color = sulphates)) +
              facet_wrap(~type, scales = "fixed" ) +
  scale_color_viridis_c()
```

Red and white wines exhibit distinct properties. White wine has lower levels of volatile acidity, while red wine varies in this regard. Both have similarly low sulphate levels: white wine (0.2–1.0) and red wine (0.4–1.0), with some high outliers. White wine shows greater variability in alcohol content, whereas red wine has slightly lower variance but is comparable overall.

!!The cluster in the lower left of the white wine data suggests that the k-nearest neighbors model may not be ideal due to the high density of data points in that region, which can make it difficult to distinguish the good wines from the bad ones.

Which model to choose?

It is hard for me to tell which model would be the best, so let's build all three and determine which one is the most accurate.

Let's split the wine into categories: best, worst, okay. We will use best and worst to find the top and worst 10 wines later.

```
wine_original |>
  count(quality)
```

```
# A tibble: 7 × 2
  quality     n
    <dbl> <int>
1       3    28
2       4   189
3       5  1976
4       6  2615
5       7  1006
6       8   178
7       9     5
```

```
wine_og_cat <- wine_original |>
  mutate(quality_cat = case_when(
    quality == 3 | quality == 4 ~ "Worst",
    quality == 5 | quality == 6 ~ "Okay",
    quality == 7 | quality == 8 | quality == 9 ~ "Best"))
```

kNN

kNN: For a given input of sulphates, alcohol, volatile acidity, and wine type, the model will predict the quality of the wine given the k nearest neighbors.

```
k<- 1

sulphates_input <- 0.8
alcohol_input <- 12
volatile_acidity_input <- 0.25

wine_og_cat |>
  mutate(distance = sqrt((sulphates_input - sulphates)^2 + (alcohol_input - alcohol)^2) +
         (volatile_acidity_input - `volatile acidity`)) |>
  arrange(distance) |>
  head(k) |>
  count(quality_cat) |>
  arrange(-n) |>
  head(1) |>
  pull(quality_cat)
```

```
[1] "Okay"

#Building my knn function along with wine type input because we will know the color of the
        wine, so I want to compare properties across the same type of wine.

wine_knn <- function(
    sulphates_input, alcohol_input, volatile_acidity_input, wine_type, k=3,
        wine_data_function)
{
wine_data_function |>
  filter(type == wine_type) |>
  mutate(distance = sqrt((sulphates_input - sulphates)^2 + (alcohol_input - alcohol)^2) +
        (volatile_acidity_input - `volatile acidity`)) |>
  arrange(distance) |>
  head(k) |>
  count(quality_cat) |>
  arrange(-n) |>
  head(1) |>
  pull(quality_cat) }

wine_knn(0.8, 12, 0.25, "red", k=2, wine_og_cat)

[1] "Okay"
```

Let's train our dataset!

```
k<- 1

set.seed(1)
training_wine_rows <- sample(1:nrow(wine_og_cat),
                         size = nrow(wine_og_cat)/2)


training_wine <- wine_og_cat[training_wine_rows, ]
testing_wine <- wine_og_cat[-training_wine_rows, ]

#For
preds<- NULL

for(row in 1:nrow(testing_wine))
  {
  preds[row] <- wine_knn(testing_wine[[row, "sulphates"]],
                         testing_wine[[row, "alcohol"]],
                         testing_wine[[row, "volatile acidity"]],
                         testing_wine[[row, "type"]],
                         k=13,
                         training_wine)
}

#Calculating the accuracy of the knn model
library(caret)

Loading required package: lattice


Attaching package: 'caret'

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

    lift
```

```
conf_matrix <- confusionMatrix(as.factor(preds), as.factor(testing_wine$quality_cat))

print(paste("Overall Accuracy:", round(conf_matrix$overall['Accuracy'] * 100, 2), "%"))
```

```
[1] "Overall Accuracy: 72.96 %"
```

```
table(preds, testing_wine$quality_cat)
```

```
preds   Best Okay Worst
  Best   185  184     8
  Okay   366 1998    95
  Worst   29  129     5
```

I learned about a new package for generating a confusion matrix that calculates overall accuracy, which I find very helpful when working with categorical variables like different wine classifications. The overall accuracy is relatively high at nearly 73% for k=13. However, I noticed that as k increases, although accuracy improves, the model struggles to correctly identify “Worst” wines, with only 5 correctly classified at k=13. This demonstrates that KNN is not effective for predicting the lowest-quality wines. This is because the majority of the initial data is classified as “okay” and is affecting the ability of the model to predict “worst” wine due to their significantly smaller representation in the dataset. On the other hand, it is better at predicting “best” wine comparatively.

LDA

I think LDA may not prove to be a useful model either since this initial dataset is not normally distributed, but let’s check.

Best wine:

```
library(MASS)
```

```
Attaching package: 'MASS'

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

    select
```

```
lda_model_wine <- lda(quality_cat ~ `sulphates`+ `alcohol` + `volatile acidity`+`type`,
        data=training_wine)

testing_wine1 <- na.omit(testing_wine)
lda_wine <- predict(lda_model_wine, testing_wine1)

threshold <- 0.8
preds_on_threshold_best <- lda_wine$posterior[, "Best"] > threshold

table(preds_on_threshold_best, testing_wine1$quality_cat)
```

```
preds_on_threshold_best Best Okay Worst
                  FALSE  562 2294   107
                  TRUE    16    7     0
```

```
#accuracy 1
true_best_correct <- sum(preds_on_threshold_best == TRUE & testing_wine1$quality_cat ==
        "Best")
total_best_predictions <- sum(preds_on_threshold_best == TRUE)

best_wine_accuracy <- (true_best_correct / total_best_predictions) * 100
```

```
print(paste("Correctly Classified Best Wines:", round(best_wine_accuracy , 2), "%"))
```

```
[1] "Correctly Classified Best Wines: 69.57 %"
```

```
#accuracy 2
lda_wine$class <- factor(lda_wine$class, levels = levels(factor(testing_wine1$quality_cat)))
testing_wine1$quality_cat <- factor(testing_wine1$quality_cat)

conf_matrix_lda <- confusionMatrix(lda_wine$class, testing_wine1$quality_cat)

print(paste("LDA Model Accuracy - Confusion Matrix:",
        round(conf_matrix_lda$overall['Accuracy'] * 100, 2), "%"))
```

```
[1] "LDA Model Accuracy - Confusion Matrix: 77.29 %"
```

At 0.8 threshold, the model has the highest prediction accuracy of 69.57% for best wines. There is always a sacrifice. In this scenario, I am predicting at least 10 (16) best wine while minimizing the number of incorrect best wines; only 7 okay wines are misidentified as best. On the other hand, the model fails to capture the rest of the 562 best wines, which is a very high number. I think if we need to select the top 10 wine, we don't need to use threshold in this case, but rather rank purely based on the probability. Another problem is the method of assessing accuracy as there are number of ways to do so. In this case, I am only evaluating the model on its ability to correctly identify best wine out of all the wines, but do not include whether the model is classifying okay and worst wines correctly or not. However, if we used confusion matrix like we used for knn, the accuracy improves to be 77.29%. This makes me realize that there is no single way to determine the accuracy level of one model.

Tree

Now let's take a look at how decision tree would work for our initial sample data.

```
library(rpart)
library(rpart.plot)

tree_wine <- rpart(factor(quality_cat) ~ sulphates+ alcohol + `volatile acidity`+ type, data
        = training_wine,  cp = 0.005)
rpart.plot(tree_wine)
```

```
tree_preds<- predict(tree_wine, testing_wine, type = "class")
table(tree_preds, testing_wine$quality_cat)
```

```
tree_preds Best Okay Worst
     Best   169  147     6
     Okay   411 2164   102
     Worst    0    0     0
```

So, interestingly, the tree model is not able to identify the worst wine perhaps due to the very low number of worst wines.

```
actual_tree <- factor(testing_wine$quality_cat)
tree_preds1 <- factor(tree_preds, levels = levels(actual_tree))

conf_matrix_tree <- confusionMatrix(actual_tree, tree_preds1)
```

```
print(paste("Decision Tree Model Accuracy:", round(conf_matrix_tree$overall['Accuracy'] *
        100, 2), "%"))
```

```
[1] "Decision Tree Model Accuracy: 77.79 %"
```

The confusion matrix says that the model has an accuracy level of 77.83%, though it fails to capture the worst wines. This comes close to the accuracy level of both the LDA and the kNN.

500 Wine Dataset

Which method to choose?

The accuracy level for all three models are pretty similar to each other, so I will have to make a decision weighing the costs and benefits of each one.

I think I will choose LDA. kNN is very sensitive to tuning and it doesnt help that we have variables that have non-linear relationships and are clustered on one side. Also, our data is heavily skewed as we have way more okay wines than worst or best, which will certainly affect the predictions. For tree classification, the worst type of wine is not being captured which makes me believe its not a good option. Instead, I think LDA will help us lessen the dominance of okay wines, especially because it relies on probability distributions. I decided not to use threshold like I did earlier because we only care about the top 10 or worst 10 wines.

```
real_wine <- read_csv("~/Downloads/Amuna_wine_data.csv", show_col_types = FALSE)

lda_500_preds <- predict(lda_model_wine, real_wine)

real_wine$prob_best <- lda_500_preds$posterior[, "Best"]
real_wine$prob_worst <- lda_500_preds$posterior[, "Worst"]

top_10_best <- real_wine |>
  arrange(desc(prob_best)) |>  # Sort by highest probability of "Best"
  head(10)

top_10_best$wine_id
```

```
 [1]  21 257 275 309 370 336 436 353 497 132
```

```
top_10_worst <- real_wine |>
  arrange(desc(prob_worst)) |>  # Sort by highest probability of "Worst"
  head(10)

top_10_worst$wine_id
```

```
 [1] 226 170 255 141 339 480 364 193  75 318
```

This was hard…