Finding the Ten Best and Ten Worst Wines

Set Up

```
library(dplyr)
library(readxl)
library(tidyverse)
library(ggplot2)

library(rpart)
library(rpart.plot)

wine_data <- read_csv("WineData.csv")
```

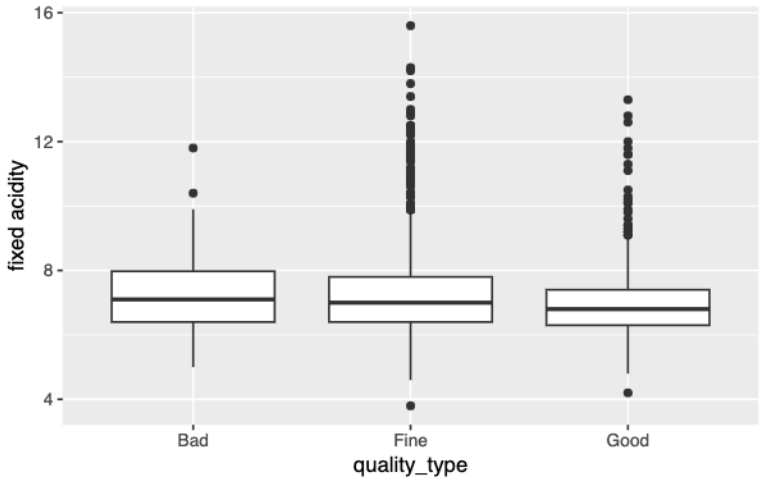
```
wine_data <- wine_data |>
  mutate(quality_type = case_when(quality == 3 ~ "Bad",
                                  quality == 4 ~ "Bad",
                                  quality == 7 ~ "Good",
                                  quality == 8 ~ "Good",
                                  quality == 9 ~ "Good",
                                  TRUE ~ "Fine"))
```

Visualizing the Data

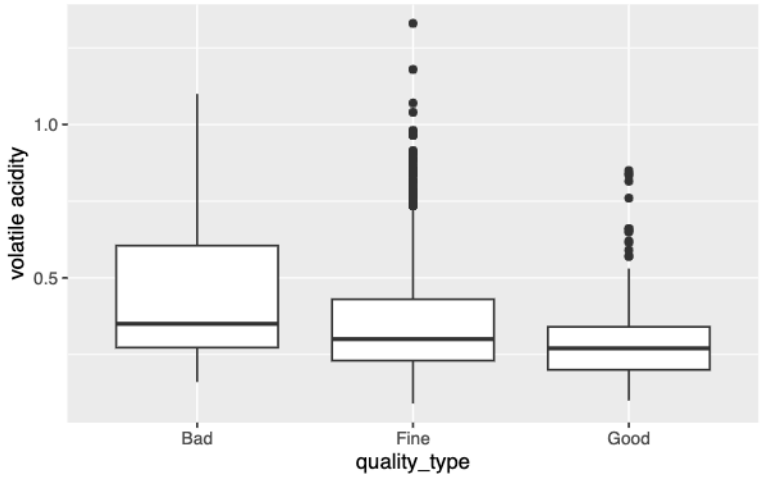
Before we can write a useful algorithm to predict wine quality, we need to decide what three properties we want to use from the wine data set. We need to find what variables have the best correlation with wine quality.

Before looking at the data it is important to remember that our main goal is to predict the ten best and ten worst wines, not to predict overall wine quality well. So we want to make sure that our chosen three properties are varied enough in terms of their relationship with wine quality that they can find good wines AND bad wines given the right algorithm.

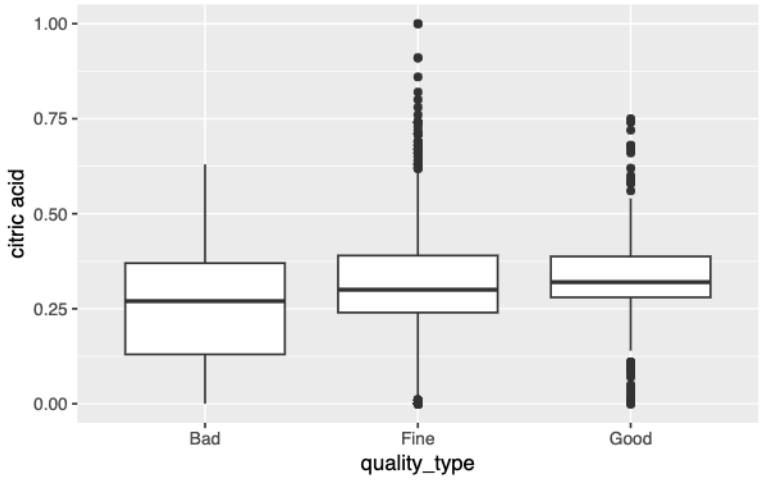
```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `fixed acidity`))
```



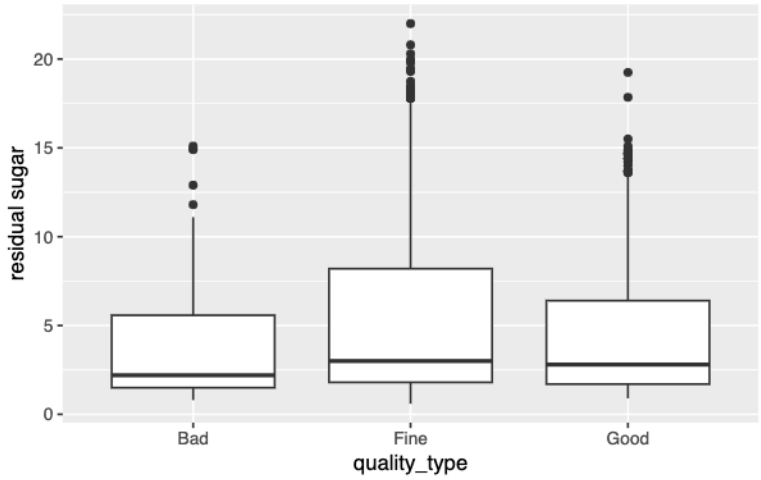

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `volatile acidity`))
```



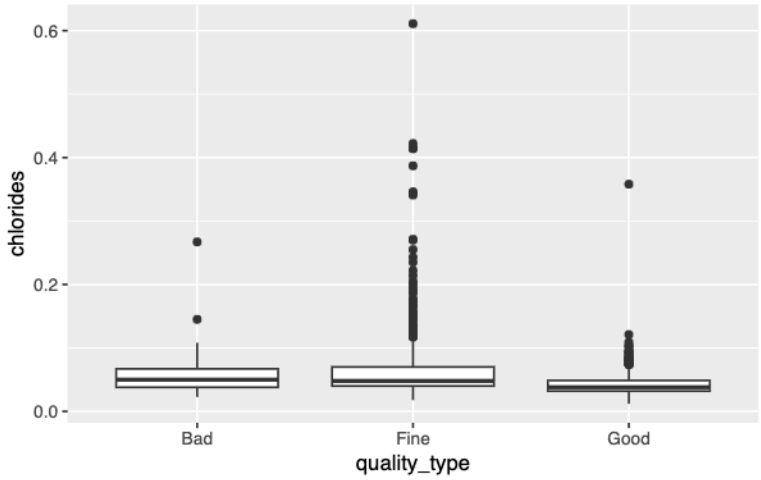

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `citric acid`))
```



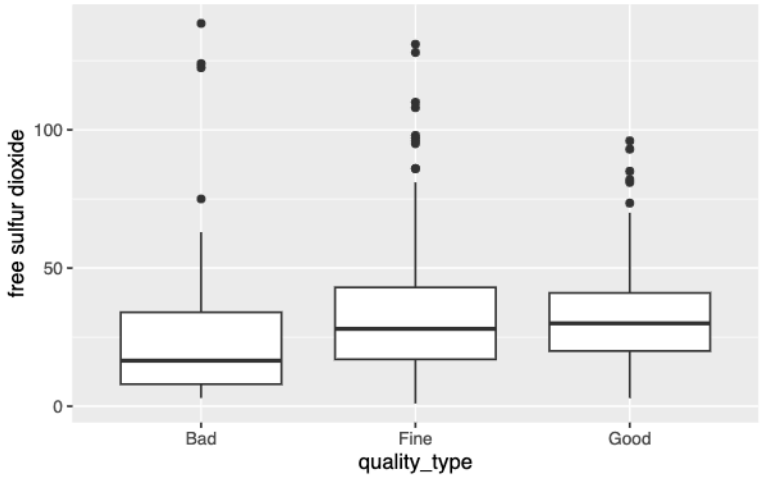

```
wine_data |>
  filter(`residual sugar` < 50) |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `residual sugar`))
```



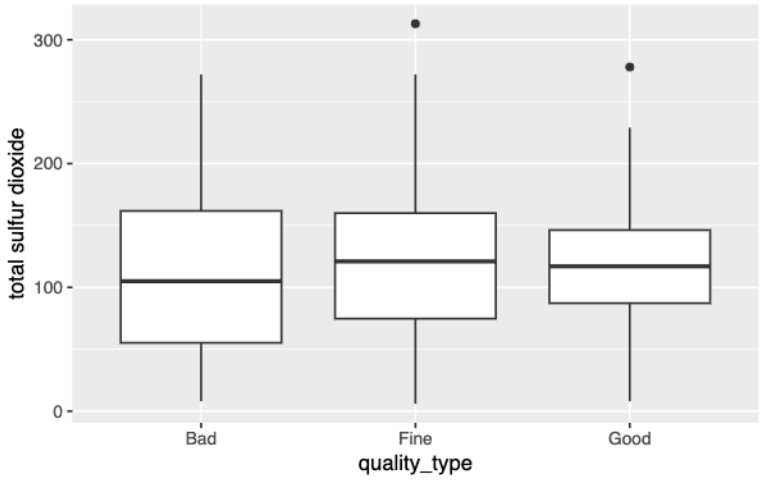

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = chlorides))
```



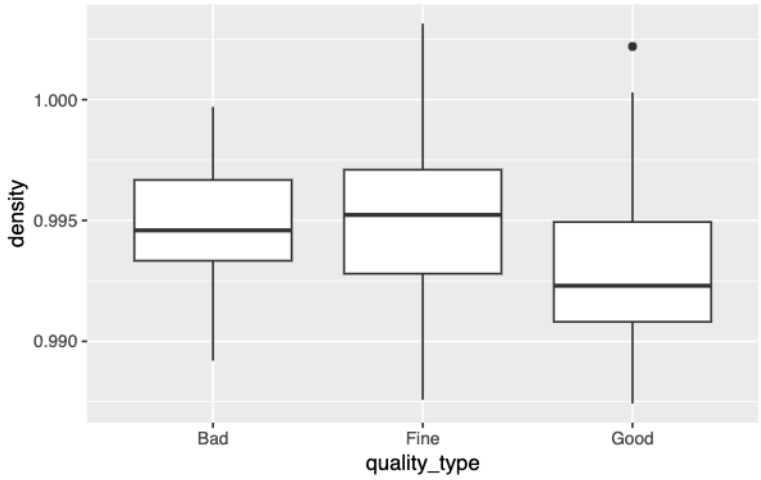

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `free sulfur dioxide`))
```



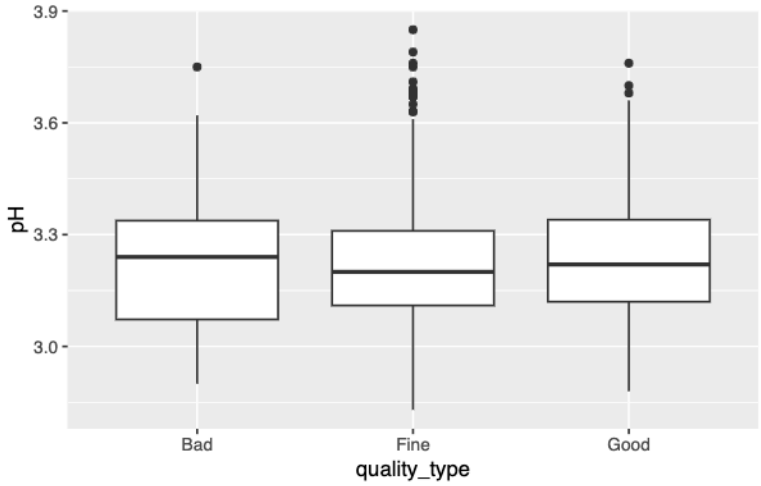

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `total sulfur dioxide`))
```



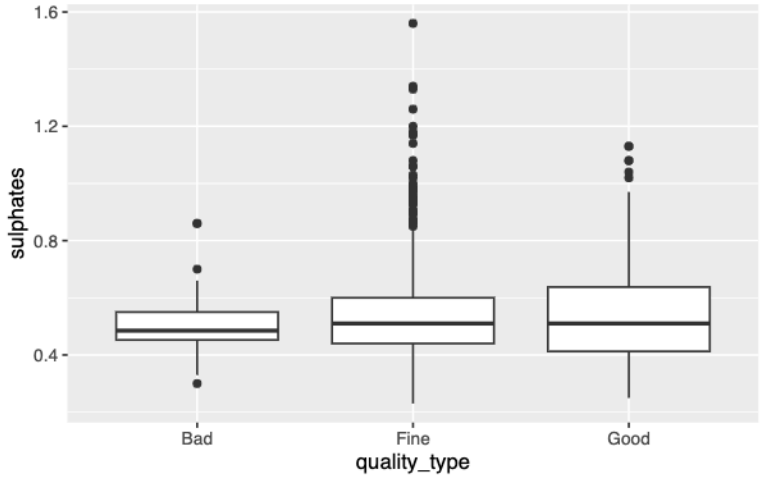

```
wine_data |>
  filter(density < 1.03) |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = density))
```



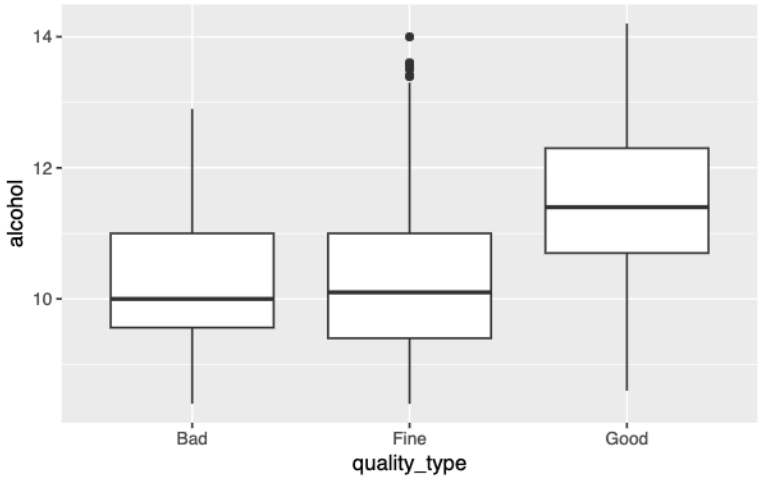

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = pH))
```




```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = sulphates))
```




```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = alcohol))
```

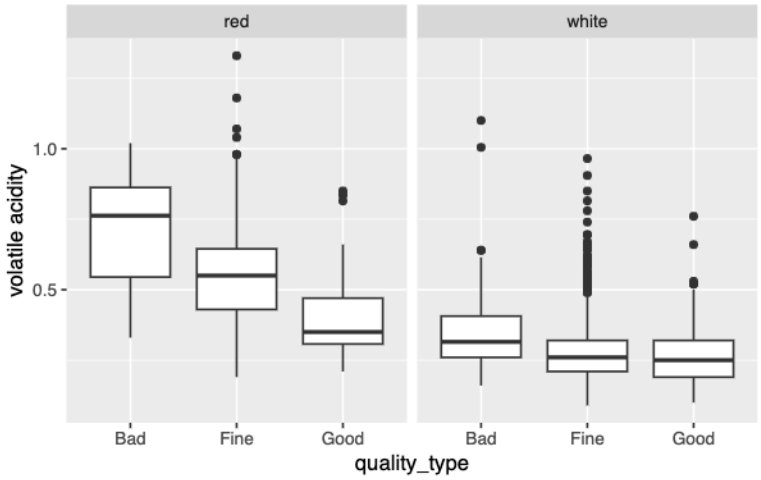


We can see from these box plots that there seems to be some sort of possible correlation between quality type and the following variables:

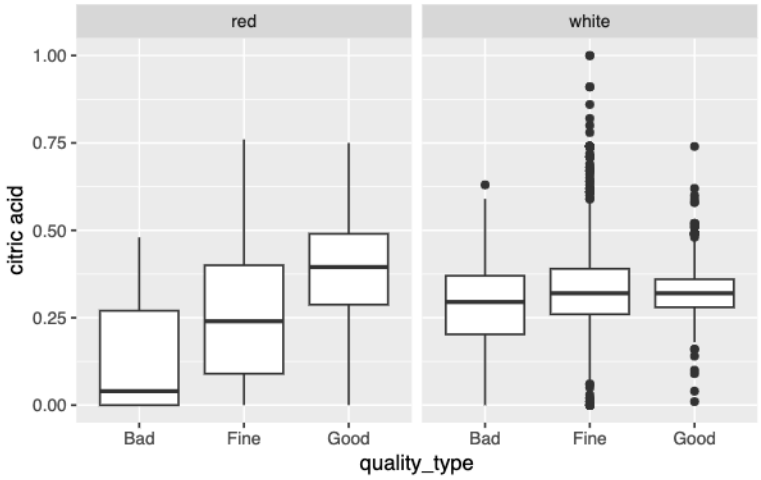
- Volatile acidity: The mean volatile acidity for bad wines is higher than that for fine and good wines. Also, the distribution is more spread out and includes many higher values that are not in any of the good wines (and very few of the fine wines). We should continue to look at this variable to see if there is any strong relationship that can help us.
- Citric acid: This has a similar box plot analysis to volatile acidity but it is the opposite– the bad wines tend to have lower values for citric acid.
- Free sulfur dioxide: There also seems to be a relationship between bad quality wines and free sulfur dioxide, so we should look into this variable as well.
- Density: Although bad wines and fine wines have similar looking plots, the box plot for good wine shows that on average, they have a lower density.
- Alcohol: The box plot for alcohol shows that good wines have a largely different alcohol content than fine and bad wines so this could be a good variable to look at to help us predict good wines.

There could also be an important difference between red and white wines and what chemicals make those wines better or worse. Below are the box plots of each variable we decided to look into above, but facet wrapped by wine type.

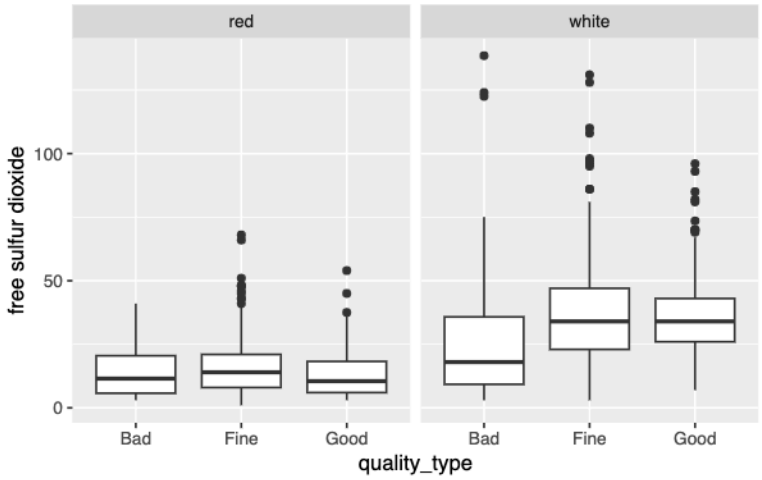
```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `volatile acidity`)) +
  facet_wrap(~type)
```



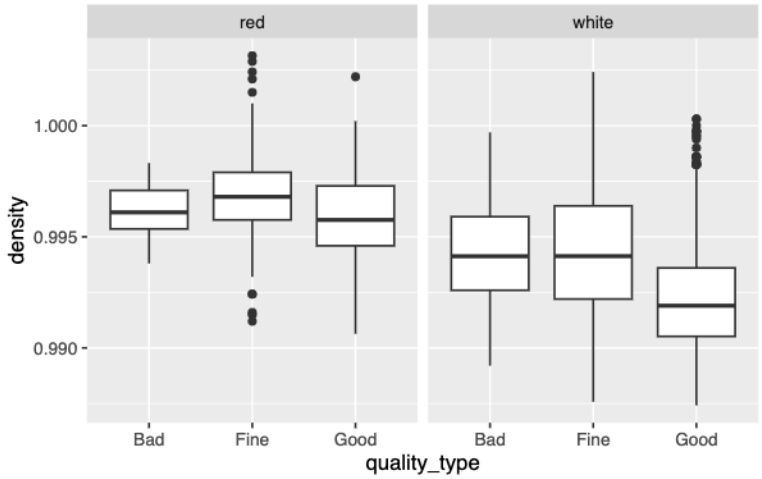

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `citric acid`)) +
  facet_wrap(~type)
```



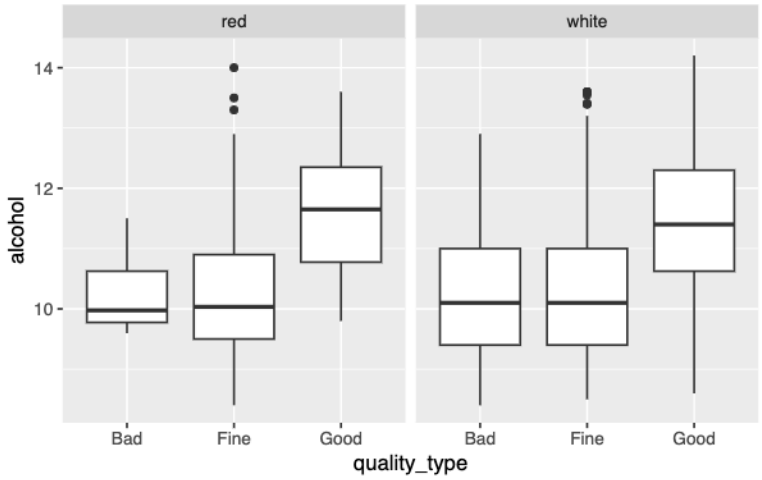

```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = `free sulfur dioxide`)) +
  facet_wrap(~type)
```




```
wine_data |>
  filter(density < 1.03) |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = density)) +
  facet_wrap(~type)
```




```
wine_data |>
  ggplot() +
  geom_boxplot(aes(x = quality_type, y = alcohol)) +
  facet_wrap(~type)
```



These diagrams show us that there is an important distinction between white and red wine for many of the variables in our data set.

Deciding Our Three Properties

So what properties should we choose for our algorithm?

As stated earlier, our goal is to find properties that predict both good and bad wines. We can see that alcohol content has the strongest correlation with good wine quality no matter the wine type so it would be a good variable to choose from our data set. Density is also a good predictor of good wine but since we already have alcohol, it would not make sense to "use up" one of our property choices with a variable that will have a similar prediction effect as alcohol. Also, it has a strong correlation with good white wines, and a much weaker correlation with good red wines, so alcohol is the better variable choice for predicting good wine.

Predicting the bad wines gets a bit more complicated because there is not one singular variable that has a strong relationship with bad quality for both red and white wines. Since we can still choose two more properties, it may be helpful to pick one variable that predicts bad wines well for white wine, and one that predicts bad wines well for red wine.

For white wine, free sulfur dioxide has the strongest relationship with bad quality wine. Although the distribution is fairly large, the majority of free sulfur dioxide values are much lower that than for fine and good wines.

For red wine, citric acid and volatile acidity both seem to have a strong relationship with bad wine quality. When we look closer, volatile acidity has values that are more unique to bad wines (the distribution is smaller and more separated from fine/good wine quality values) than citric acid, which may be more helpful for us in our algorithm. Also, citric acid has no relationship with white wine quality that will help us in our analysis, but volatile acidity seems

to have at least some helpful relationship. For these reasons, we will choose volatile acidity as our third property.

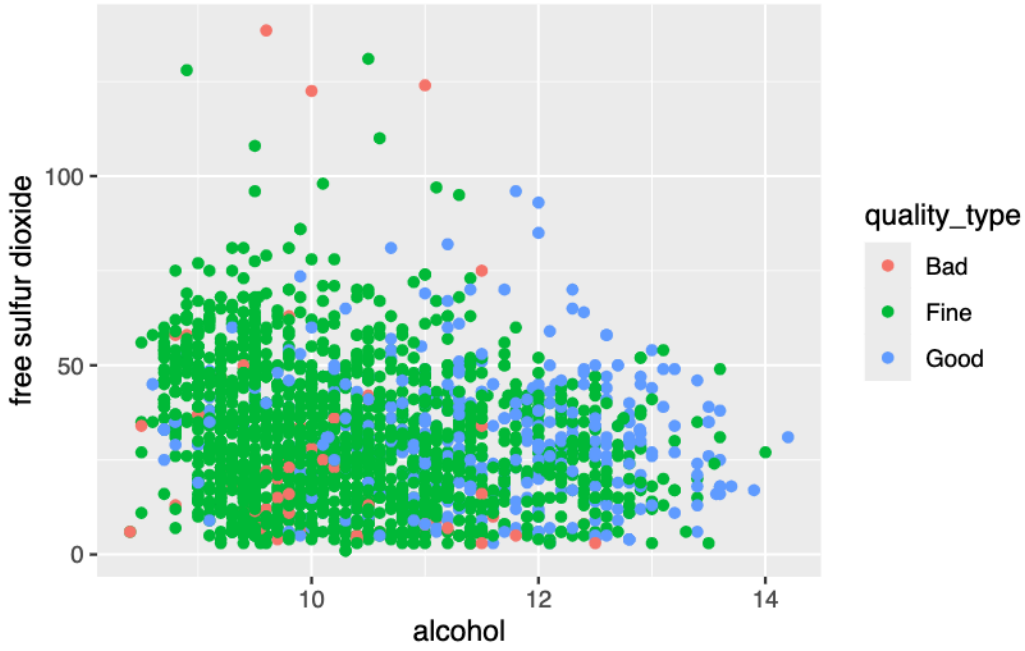
Our three properties: alcohol, volatile acidity, free sulfur dioxide.

Choosing the Algorithm Type

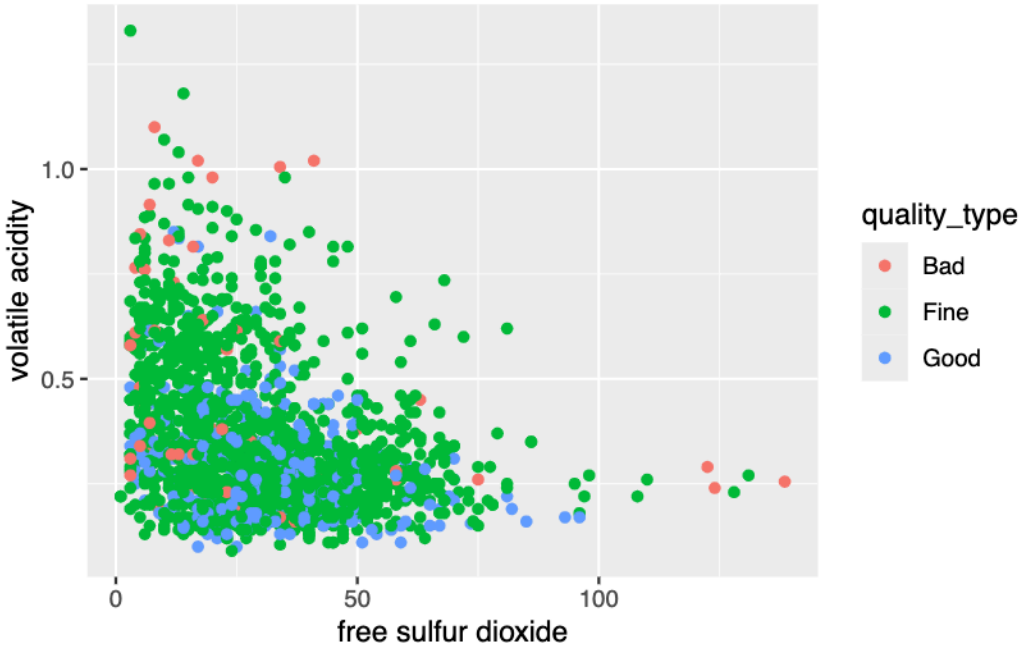
In class we have discussed three options for modeling this data: kNN, LDA, and tree-based algorithms.

First looking at LDA, when we graph our data with our chosen variables (using traces of what should be a 3D graph)...

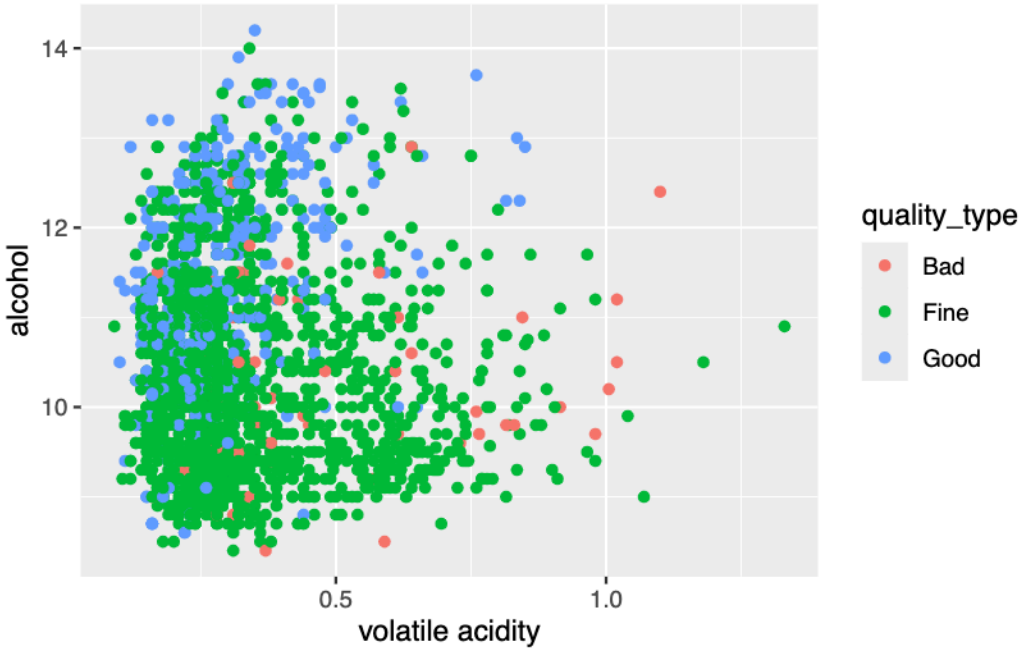
```
wine_data |>
    ggplot() +
  geom_point(aes(x = alcohol, y = `free sulfur dioxide`, color = quality_type))
```




```
wine_data |>
    ggplot() +
  geom_point(aes(x = `free sulfur dioxide`, y = `volatile acidity`, color = quality_type))
```




```
wine_data |>
    ggplot() +
  geom_point(aes(x = `volatile acidity`, y = alcohol, color = quality_type))
```



There does not seem to be a way that we could "make cuts" that would help us in consistently finding the best and worst wines, since all of the data is fairly clumped together. We could use LDA to predict wine quality, but looking at these graphs, it may not be the easiest and most accurate algorithm.

That leaves us with either kNN or a tree-based algorithm. We could choose to go with either but the combination of our data set being mostly quantitative and knowing that kNN will allow us to more easily manipulate how much of a variable will effect our prediction (which is important given our analysis of the variables we chose earlier), tells us that kNN is most likely the best choice given our approach.

Creating the Algorithm

First, let's write a kNN algorithm that uses all of our variables and gives us the wine quality type that is most common for our k nearest neighbors.

```
wine_quality_kNN_type <- function(k, volatileacidity_input, fsd_input, alcohol_input, data){
  data |>
  mutate(distance = sqrt(((`volatile acidity` - volatileacidity_input)*10)^2
                         + ((`free sulfur dioxide` - fsd_input)/10)^2
                         + ((alcohol - alcohol_input))^2)) |>
  arrange(distance) |>
  head(k) |>
  count(quality_type) |>
  arrange(-n) |>
  head(1) |>
  pull(quality_type)
}
```

For this first algorithm, multiplying the volatile acidity difference by ten and dividing the free sulfur dioxide by ten was just to make all three variables on similar scales but we will create more helpful scaling based on our analysis later.

Now, let's find our how accurate this algorithm is (using k = 10 for now):

```
set.seed(1)
training_rows <- sample(1:nrow(wine_data),
                        size = nrow(wine_data)/2)
train_data <- wine_data[training_rows, ]
test_data <- wine_data[-training_rows, ]

k <- 10
```

```
ans <- NULL
for(i in 1:nrow(test_data)){
   prediction <- wine_quality_kNN_type(k,
                                       test_data$`volatile acidity`[i],
                                       test_data$`density`[i],
                                       test_data$alcohol[i],
                                       train_data)
  ans[i] <- prediction == test_data$quality_type[i]
}
mean(ans)
```

```
[1] 0.781
```

This says that our algorithm is 78% accurate on our testing and training data. This is a pretty good percentage if we wanted to predict wine quality type for any given data set. However, given our context, this accuracy metric is actually unhelpful because it is possible that the algorithm is predicting fine wines well, and most of the predictions for bad and good wine are part of the 22% that are incorrect.

Before we try to find a better algorithm, let's first create a variable that can numerically account for type:

```
wine_data <- mutate(wine_data, type_num = case_when(type == "white" ~ 0,
                                                    type == "red" ~ 1))
```

Using the analysis of our variables and why we chose them earlier, we can write the following algorithms:

LOW QUALITY RED: For this algorithm, we want to separate red and white wine in our input data because the variables we choose have different relationships with the wine depending on its type. For red wine, we saw that volatile acid was really the only property that had a major effect on separating bad wines from fine wines. So we can create a kNN algorithm that takes the k nearest neighbors based on volatile acidity and wine type. We filtered by red wine so that our input data is only looking at the nearest red wine neighbors, and added the type as a factor to our distance equation so that the test data wine type will also be accounted for (i.e. `type_num` will always 1, so if type_input is also red: 1, then this will not make the distance any farther, but if it is white-0, the distance will be larger). Lastly, we filtered alcohol to be less than 11, because we know that if the alcohol level is higher than 12 it is most likely going to be a good wine. Since we want to focus on picking out our bad wines from neighbors that are fine wines, we can filter this out to get a more accurate algorithm.

```
wine_quality_kNN_lowRED <- function(k, va_input, type_input, data){
  data |>
  filter(type == "red") |>
  filter(alcohol < 11) |>
  mutate(distance = sqrt(((`volatile acidity` - va_input))^2
                         + (type_num - type_input)^2)) |>
  arrange(distance) |>
  head(k) |>
  summarize(mean(quality))
}
```

LOW QUALITY WHITE: This algorithm will look similar to the low red function because it is a similar situation except for now free sulfur dioxide is the best predictor for bad wine. We are going to keep volatile acidity as a factor because there is still some relationship between bad white wine and volatile acidity. However, we are not going to scale these variables or manipulate them in the distance equation because they already have values that will account for the amount they affect the distance function. Free sulfur dioxide is what we want to have the greatest effect and it already has higher values that will yield higher difference values. Volatile acidity has the opposite which makes sense for our data, and wine type is in the middle.

```
wine_quality_kNN_lowWHITE <- function(k, va_input, fsd_input, type_input, data){
  data |>
  filter(type == "white") |>
  filter(alcohol < 11) |>
  mutate(distance = sqrt(((`volatile acidity` - va_input))^2
                         + ((`free sulfur dioxide` - fsd_input)*10)^2
                         + ((type_num - type_input))^2)) |>
  arrange(distance) |>
  head(k) |>
  summarize(mean(quality))
}
```

HIGH QUALITY: Since white wine and red wine are both similarly effected by alcohol, we do not need to create separate functions filtered by type. We will use alcohol and volatile acidity because as we found earlier, those two qualities have the highest impact on quality for good wines.

```
wine_quality_kNN_high <- function(k, va_input, alc_input, data){
  data |>
  mutate(distance = sqrt(((`volatile acidity` - va_input))^2
```

```
                          + ((alcohol - alc_input))^2)) |>
  arrange(distance) |>
  head(k) |>
  summarize(mean(quality))
}
```

Finding the Best and Worst With Our Algorithm

Now that we have our three algorithms to predict wine quality, we need to create a way to get the ten best and ten worst wines out of our data set.

```
pred_best <- function(k, data_known, data_desired) {
  prediction_best <- data.frame(quality = 1:nrow(data_desired))
  prediction_best <- mutate(prediction_best, wine_number = 1:nrow(data_desired))

  for(i in 1:nrow(data_desired)){
     prediction <- wine_quality_kNN_high(k,
                                         data_desired$`volatile acidity`[i],
                                         data_desired$alcohol[i],
                                         data_known)
    prediction_best$quality[i] <- as.numeric(prediction)
  }

  prediction_best |>
    arrange(-quality) |>
    head(10)
}
```

In the code above we created a function that shows us the ten best wines in the data set, using the data we know (wine data).

Now let's do it for bad wines. Since we have two functions for bad wines, we need to take bad wines from both lists in order to get the ten worst wines. Our functions are different and we want to accurately represent both wine types in our data set, so we will find a proportion for the amount of white to red wines in our data set and take a proportionate amount from each worst wines list.

```
pred_worst <- function(k, data_known, data_desired) {
  prediction_worstRED <- data.frame(quality = 1:nrow(data_desired))
  prediction_worstRED <- mutate(prediction_worstRED, wine_number = 1:nrow(data_desired))
```

```
  for(i in 1:nrow(data_desired)){
     prediction <- wine_quality_kNN_lowRED(k,
                                           data_desired$`volatile acidity`[i],
                                           data_desired$type_num[i],
                                           data_known)
    prediction_worstRED$quality[i] <- as.numeric(prediction)
  }

  prediction_worstWHITE <- data.frame(quality = 1:nrow(data_desired))
  prediction_worstWHITE <- mutate(prediction_worstWHITE, wine_number = 1:nrow(data_desired))

  for(i in 1:nrow(data_desired)){
     prediction <- wine_quality_kNN_lowWHITE(k,
                                           data_desired$`volatile acidity`[i],
                                           data_desired$`free sulfur dioxide`[i],
                                           data_desired$type_num[i],
                                           data_known)
    prediction_worstWHITE$quality[i] <- as.numeric(prediction)
  }

  data_desired <- data_desired |>
  mutate(type_true = case_when(type == "white" ~ TRUE,
                               TRUE ~ FALSE))
  white_prop <- floor(10*(mean(data_desired$type_true)))
  red_prop <- (10 - white_prop)

  prediction_worst <- bind_rows((prediction_worstRED |> arrange(quality) |> head(red_prop)),

  prediction_worst
}
```

Testing the Algorithm

Now that we have a complete algorithm for predicting the ten best and ten worst wines, we should test the accuracy of our data using our wine data before applying it to the new data set. We will create a testing and training data set from our wine data, and use our algorithm to predict the ten best and ten worst wines.

```
training_rows <- sample(1:nrow(wine_data),
                        size = nrow(wine_data)/2)
```

```
train_data <- wine_data[training_rows, ]
test_data <- wine_data[-training_rows, ]
```

We also need to determine which values to use for k. For high quality wines, since there are many of them in this data set and they are pretty clustered together, we can assume that our relationship between quality and our kNN function is more general and works well for a higher kNN. For low quality wines we may want a smaller value because there is a more complex relationship and also a smaller amount of bad wines in our data set, so we want to take a smaller number of nearest neighbors. Since our training data set is pretty large, let's set k = 30 for our high quality function and k = 15 for our low quality function. We can adjust these values later if needed.

Lastly, when we are testing, we will need to create a new function nearly identical to the ones we created in the last section, but that include the true wine quality for our test data. This will allow us to see if the wines we are predicting for our ten best and ten worst are actually good or bad.

```
pred_best_test <- function(k, data_known, data_desired) {
  prediction_best <- data.frame(quality = 1:nrow(data_desired))
  prediction_best <- mutate(prediction_best, real_quality = 1:nrow(data_desired))
  prediction_best <- mutate(prediction_best, wine_number = 1:nrow(data_desired))

  for(i in 1:nrow(data_desired)){
     prediction <- wine_quality_kNN_high(k,
                                         data_desired$`volatile acidity`[i],
                                         data_desired$alcohol[i],
                                         data_known)
    prediction_best$quality[i] <- as.numeric(prediction)
    prediction_best$real_quality[i] <- data_desired$quality[i]
  }

  prediction_best |>
    arrange(-quality) |>
    head(10)
}

pred_best_test(30, train_data, test_data)
```

```
    quality real_quality wine_number
1  6.766667            7         449
2  6.733333            6         352
3  6.733333            6         725
```

```
4  6.700000            7          59
5  6.700000            6         354
6  6.700000            7         401
7  6.700000            7         421
8  6.700000            6         553
9  6.700000            8         695
10 6.666667            6         142
```

```r
pred_worst_test <- function(k, data_known, data_desired) {
  prediction_worstRED <- data.frame(quality = 1:nrow(data_desired))
  prediction_worstRED <- mutate(prediction_worstRED, real_quality = 1:nrow(data_desired))
  prediction_worstRED <- mutate(prediction_worstRED, wine_number = 1:nrow(data_desired))

  for(i in 1:nrow(data_desired)){
     prediction <- wine_quality_kNN_lowRED(k,
                                           data_desired$`volatile acidity`[i],
                                           data_desired$type_num[i],
                                           data_known)
    prediction_worstRED$quality[i] <- as.numeric(prediction)
    prediction_worstRED$real_quality[i] <- data_desired$quality[i]
  }

  prediction_worstWHITE <- data.frame(quality = 1:nrow(data_desired))
  prediction_worstWHITE <- mutate(prediction_worstWHITE, real_quality = 1:nrow(data_desired))
  prediction_worstWHITE <- mutate(prediction_worstWHITE, wine_number = 1:nrow(data_desired))

  for(i in 1:nrow(data_desired)){
     prediction <- wine_quality_kNN_lowWHITE(k,
                                           data_desired$`volatile acidity`[i],
                                           data_desired$`free sulfur dioxide`[i],
                                           data_desired$type_num[i],
                                           data_known)
    prediction_worstWHITE$quality[i] <- as.numeric(prediction)
    prediction_worstWHITE$real_quality[i] <- data_desired$quality[i]
  }

  data_desired <- data_desired |>
  mutate(type_true = case_when(type == "white" ~ TRUE,
                               TRUE ~ FALSE))
  white_prop <- floor(10*(mean(data_desired$type_true)))
  red_prop <- (10 - white_prop)
```

```
  prediction_worst <- bind_rows((prediction_worstRED |> arrange(quality) |> head(red_prop)),

  prediction_worst
}

pred_worst_test(15, train_data, test_data)
```

```
    quality real_quality wine_number
1  5.066667            4         124
2  5.066667            4         177
3  5.066667            5         745
4  5.066667            4           1
5  5.066667            5          14
6  5.066667            4          16
7  5.066667            6          22
8  5.066667            6          23
9  5.066667            5          24
10 5.066667            5          25
```

We can see from our tests that majority of the wines in the ten best are good wines and majority of the wines in the ten worst are bad wines. This shows us that our algorithm is pretty good at predicting good and bad wines for our data set.

Also, after using the functions above and changing values of k around a bit, these values seem to still be a good fit for our data set.

Applying our Algorithm and Conclusion

We are finally able to use our algorithm to predict the ten best and ten worst wines for our new data set.

```
new_wine_data <- read_csv("Molly_wine_data.csv")
new_wine_data <- mutate(new_wine_data, type_num = case_when(type == "white" ~ 0,
                                                            type == "red" ~ 1))

pred_best(30, wine_data, new_wine_data)
```

```
    quality wine_number
1  6.766667         266
2  6.766667         352
```

```
3  6.733333          24
4  6.733333         278
5  6.733333         315
6  6.733333         348
7  6.700000         135
8  6.700000         460
9  6.666667          14
10 6.666667          22
```

```
pred_worst(15, wine_data, new_wine_data)
```

```
    quality wine_number
1  4.866667         232
2  4.866667         274
3  4.866667         384
4  4.866667          10
5  4.866667          18
6  4.866667          29
7  4.866667          36
8  4.866667         115
9  4.866667         132
10 4.866667         139
```

We can come to the conclusion that given our algorithm:

- Ten best wines: 266, 352, 24, 278, 315, 348, 135, 460, 14, 22
- Ten worst wines: 232, 274, 384, 10, 18, 29, 36, 115, 132, 139