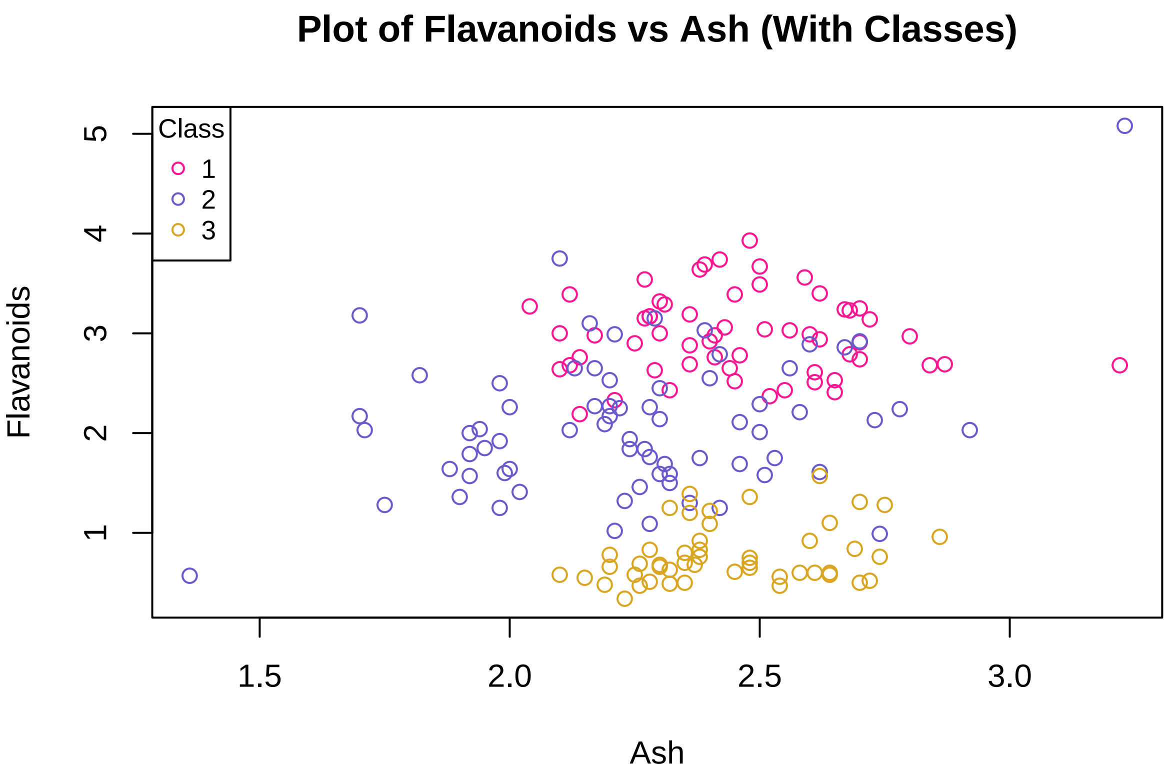
lab_06

2023-10-25

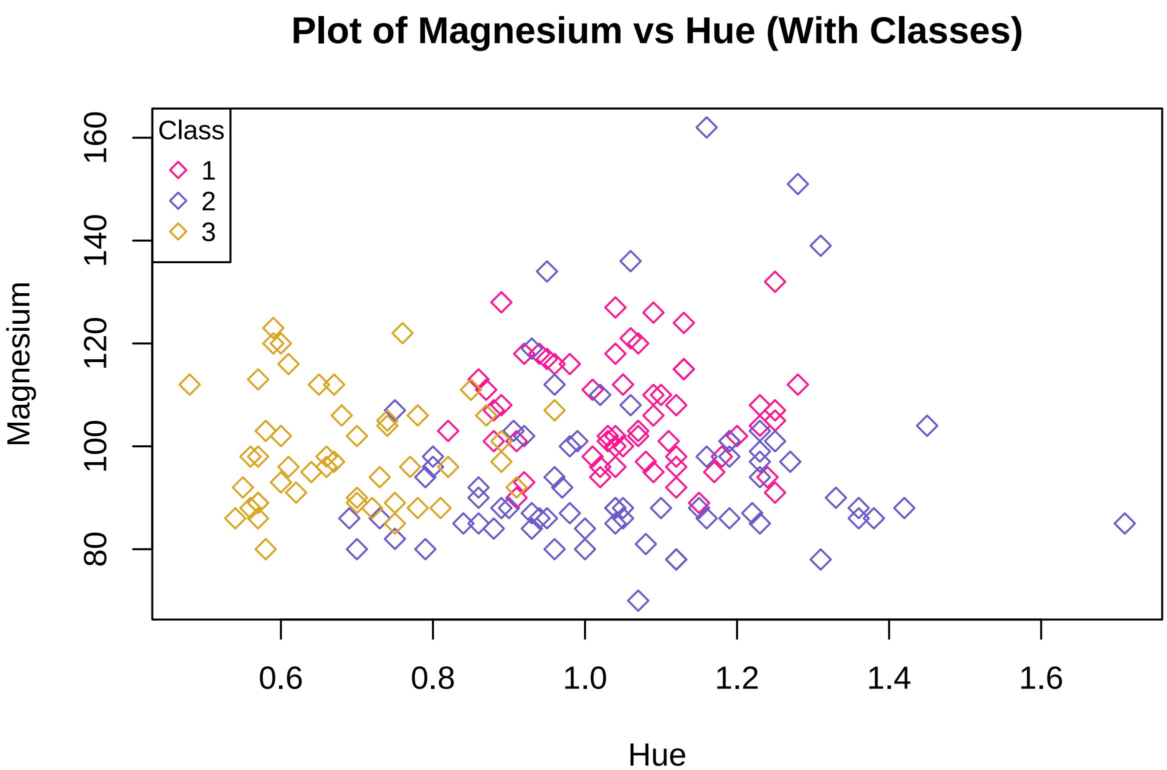
Data Exploration (Not as Fun as Wine Tasting)

I have hidden the preliminaries since it is literally just reading the `.csv` file and we've seen that a billion times now.

Below, we have the plot of Flavanoids and Ash, with their classes shown.



The three classes of wine seem to be separated in "layers", with no overlap between classes 1 and 3 and slight overlap between classes 2 and 3. Class 1 and 2 seem to overlap greatly, and it looks like class 2 encompasses a wider range of Ash and Flavanoids, as seen by the point in the top right and bottom left corners.



Compared to the previous plot, the boundaries of the classes here are less obvious. There is a lot of overlap of classes 1 and 2, with some overlap of classes 1 and 3, and classes 2 and 3.

Standardising and Splitting

We use the standardise function from class and standardise all the columns below:

```
standardise <- function(x) {
  return((x - mean(x))/sd(x))
}

#so I can preserve the original data set
wine_su <- wine

#applying to all the columns except the class column
wine_su[,-1] <- apply(wine[,-1], 2, standardise)
```

Now, we split our data set:

```
set.seed(200)
nrows <- nrow(wine_su)
sample_rows <- sample(nrow(wine_su), size=nrows/2)
train <- wine_su[sample_rows, ]
test  <- wine_su[-sample_rows,]
```

Here, we have the first few rows of our training set:

```
##     Class     Alcohol Malic.Acid        Ash Alcalinity.of.Ash  Magnesium
## 166     3  0.89844565  1.8114477 -0.3882602         0.8998349 -0.8220960
## 114     2 -1.95930769 -1.4289521  0.4865539         0.4506745 -0.8220960
## 44      1  0.29486844  1.4712952 -0.2789084        -0.5973666  0.2281415
## 26      1  0.06082829 -0.2563213  3.1109961         1.6484357  1.6984740
```

```
## 64      2 -0.77678907 -1.0798483 -0.7527660        -0.1482061 -0.8921119
## 82      2 -0.34566248 -0.4711544 -0.6069637        -0.2080942 -0.9621277
##     Total.Phenols  Flavanoids Nonflavanoid.phenols Proanthocyanins
## 166    -1.6219712 -1.56105131            1.2707258      -0.7703189
## 114     0.2954180 -0.01929168            0.4672118      -0.2636438
## 44      0.5510698  0.60141674           -0.3363022       0.1207304
## 26      0.5350916  0.65147387            0.8689688       0.5749909
## 64      1.9251987  1.07195377           -1.3808704       0.4876331
## 82     -0.1519728  0.50130248           -0.8184106       0.3129175
##     Color.Intensity        Hue OD280.OD315.of.diulted.wines    Proline
## 166       0.6737349 -0.7763408                   -1.2136578 -0.7205077
## 114      -0.8532554  0.6236583                   -0.4249147 -0.9936038
## 44       -0.3011233 -0.6013409                    0.5469294 -0.2124219
## 26       -0.6375788  0.7549083                    0.8286233  0.2639084
## 64       -0.2623015  1.1486580                    0.3638283 -1.0380613
## 82       -0.4995458  0.8861582                    0.7441151 -0.1044537
```

Training (to be a Sommelier?)

First, we borrow all the functions from class. I have hidden them since it's just the functions we wrote in class, but trust me, they are there.

Now, we train a k-NN classifier using the Hue and Magnesium predictors.

```r
#checking the ranges we need
range(train$Hue)
```

```
## [1] -1.826340  3.292407
```

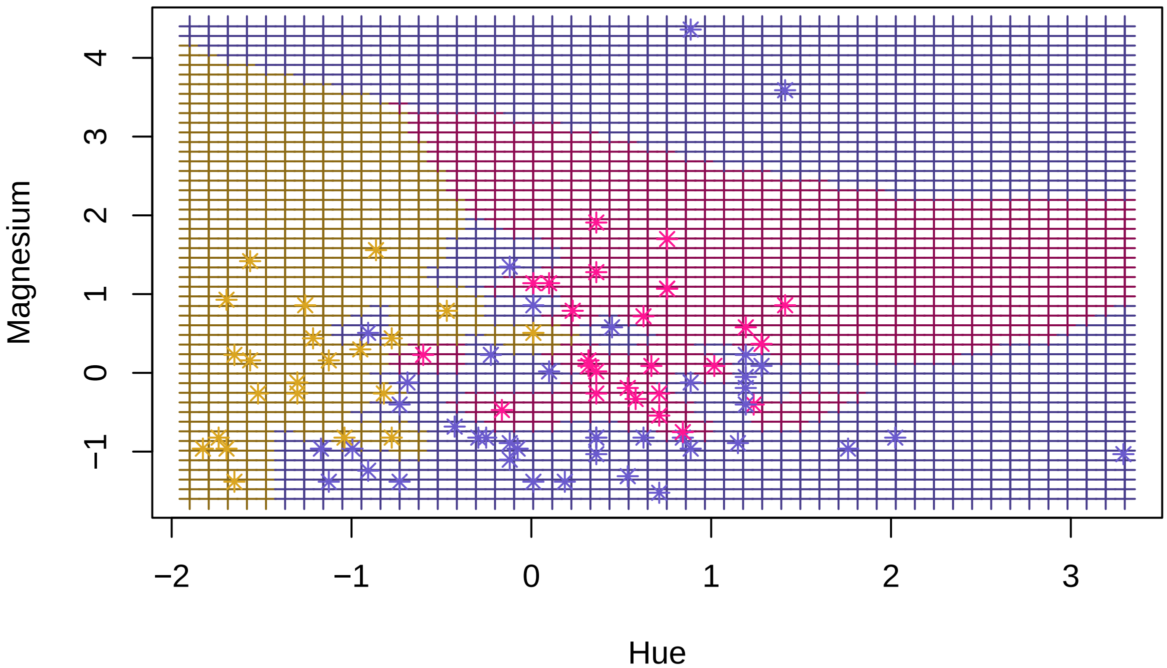
```r
range(train$Magnesium)
```

```
## [1] -1.522254  4.359076
```

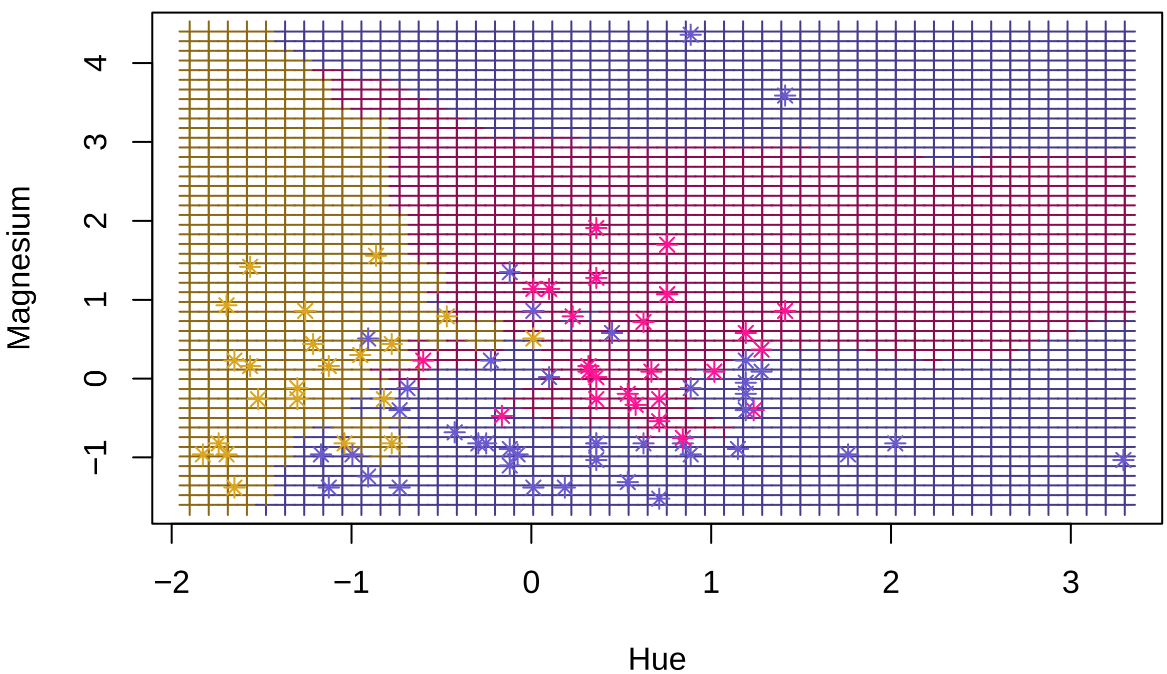
```r
#setting up the axes
hue_grid <- seq(-1.9, 3.3, length=50)
magnesium_grid <- seq(-1.6, 4.4, length=50)

#setting up the grid
two_variable_grid <- expand.grid(Hue = hue_grid, Magnesium = magnesium_grid)

classes_1 <- classify_k(data = train[, c("Hue", "Magnesium", "Class")],
         points = two_variable_grid, 1)
classes_3 <- classify_k(data = train[, c("Hue", "Magnesium", "Class")],
         points = two_variable_grid, 3)
classes_5 <- classify_k(data = train[, c("Hue", "Magnesium", "Class")],
         points = two_variable_grid, 5)
colours_alt <- c("deeppink4", "slateblue4", "goldenrod4")
```

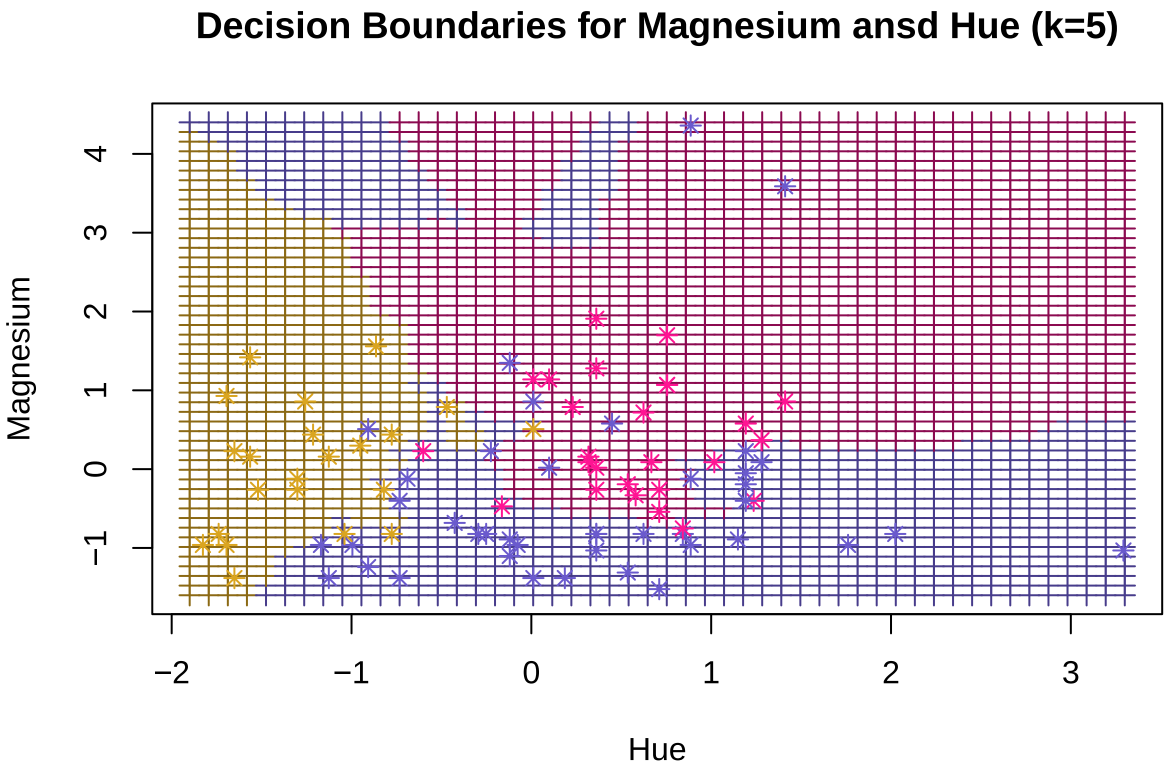
Here are the 3 plots required, with the training data overlaid on the decision boundary plots:

Decision Boundaries for Magnesium ansd Hue (k=1)



Decision Boundaries for Magnesium ansd Hue (k=3)





We find that:

```
## [1] "When k is 5, the accuracy is 74.2%."
```

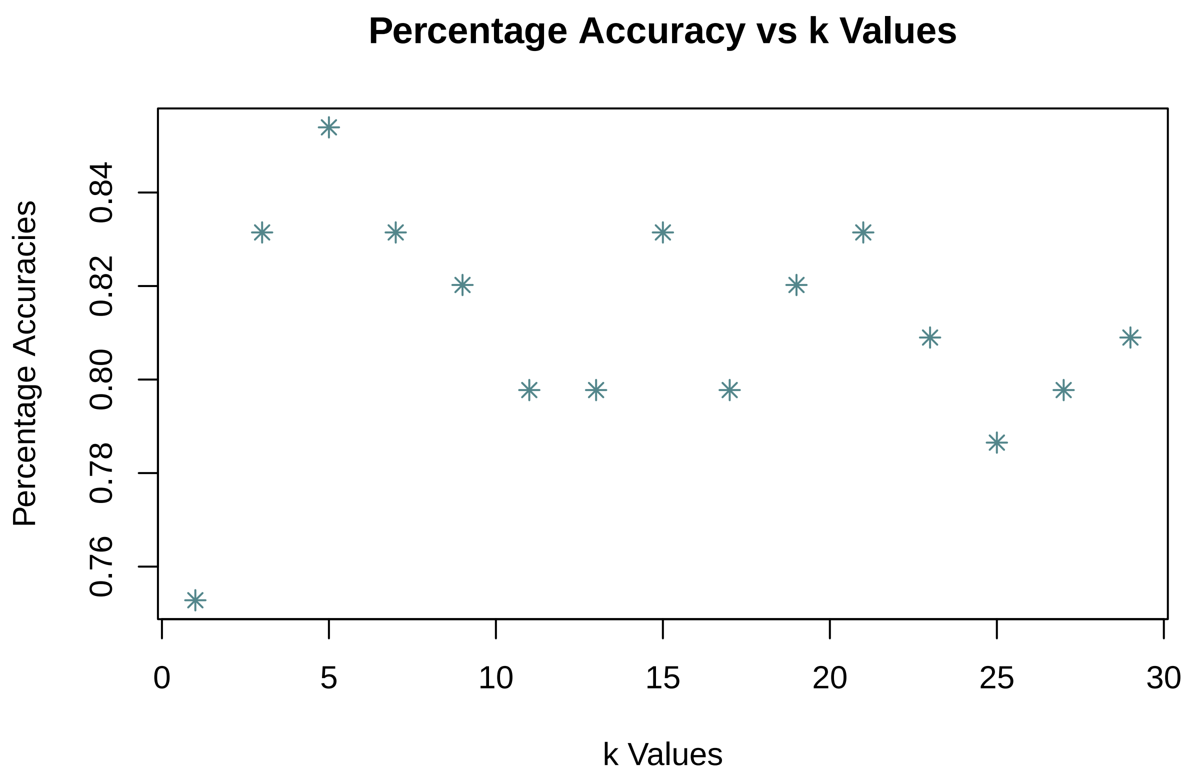
More is Better?

```
## [1] "For more predictors, when k is 5, the accuracy is 85.4%."
```

It seems that with more predictors and the same choice of k, the accuracy of the classifier increases.

```
k_breaks <- seq(1, 29, by=2)
accuracies <- numeric(length(k_breaks))

for (k in k_breaks) {
  accuracies[(k+1)/2] <- evaluate_k(train[,c("Alcalinity.of.Ash", "Malic.Acid",
                                             "Hue", "Magnesium", "Class")],
                                    test[,c("Alcalinity.of.Ash", "Malic.Acid",
                                            "Hue", "Magnesium", "Class")], k)
}
```

Here is the plot of the percentage accuracy versus k:



From the graph, there seems to be about 3 local maxima and 1 global maxima. The global maxima looks like it occurs around when k `=5`. It looks like the accuracy of the classifier drops after that, with some small increases in accuracy that never reach the same accuracy as before.