wine_exploration

Chris DiSerafino

4/14/2020

R Markdown

Research question: how well can we classify a wine as red or white using neural networks? Does the number of hidden units or layers matter? How does starting values or scaling effect our results? The risk function of neural networks is not convex. Which method of finding a global minima is most effective in this instance?

Explore Data

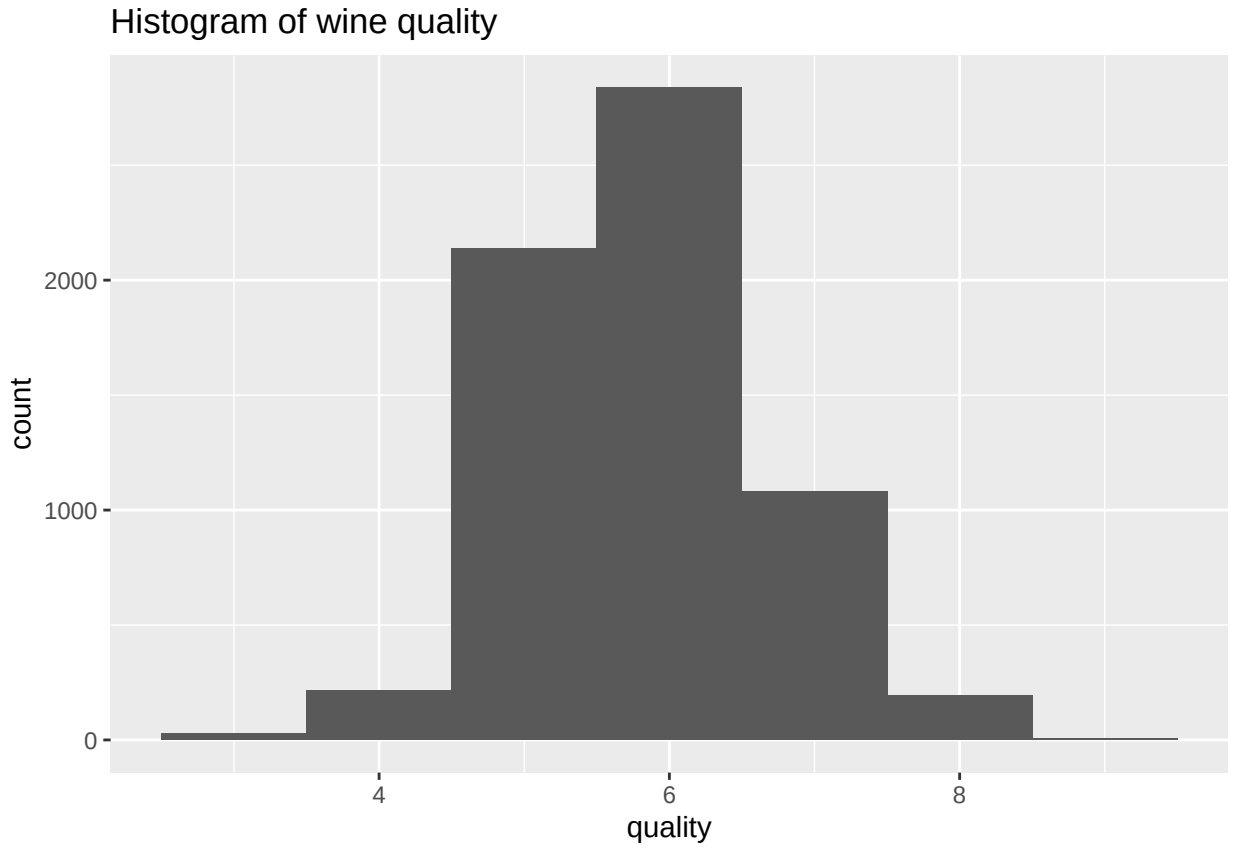
```
color_predict = color_wines[,2:12]
color_response = color_wines[,13]

quality_predict = quality_wines[,2:12]
quality_response = quality_wines[,13]

wines = wines[,2:13]

set.seed(13)
head(quality_wines)
```

```
##   X fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1 1           7.0             0.27        0.36           20.7     0.045
## 2 2           6.3             0.30        0.34            1.6     0.049
## 3 3           8.1             0.28        0.40            6.9     0.050
## 4 4           7.2             0.23        0.32            8.5     0.058
## 5 5           7.2             0.23        0.32            8.5     0.058
## 6 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  47                  186  0.9956 3.19      0.40     9.9
## 6                  30                   97  0.9951 3.26      0.44    10.1
##   quality
## 1       6
## 2       6
## 3       6
## 4       6
## 5       6
## 6       6
```

```
head(color_wines)
```

```
##   X fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1 1           7.0             0.27        0.36           20.7     0.045
## 2 2           6.3             0.30        0.34            1.6     0.049
## 3 3           8.1             0.28        0.40            6.9     0.050
## 4 4           7.2             0.23        0.32            8.5     0.058
## 5 5           7.2             0.23        0.32            8.5     0.058
## 6 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol type
## 1                  45                  170  1.0010 3.00      0.45     8.8    0
## 2                  14                  132  0.9940 3.30      0.49     9.5    0
## 3                  30                   97  0.9951 3.26      0.44    10.1    0
## 4                  47                  186  0.9956 3.19      0.40     9.9    0
## 5                  47                  186  0.9956 3.19      0.40     9.9    0
## 6                  30                   97  0.9951 3.26      0.44    10.1    0
```

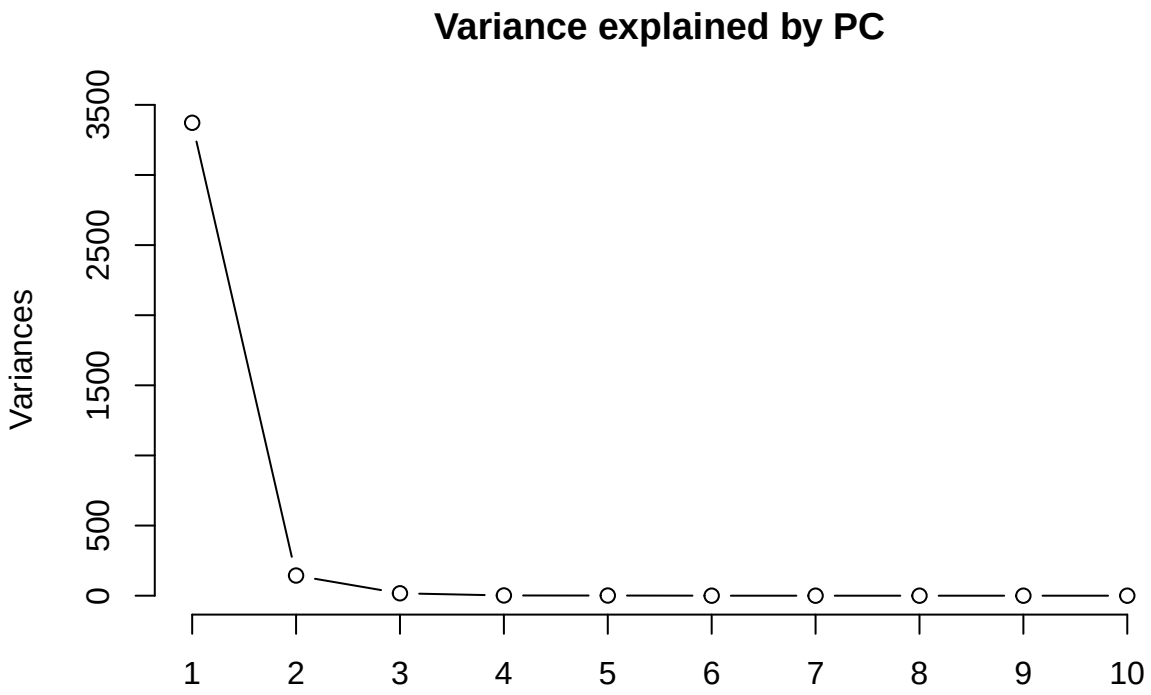
```
#look at distribution of wine quality
ggplot(wines) +
  geom_histogram(aes(x = quality), binwidth = 1) +
  ggtitle("Histogram of wine quality")
```



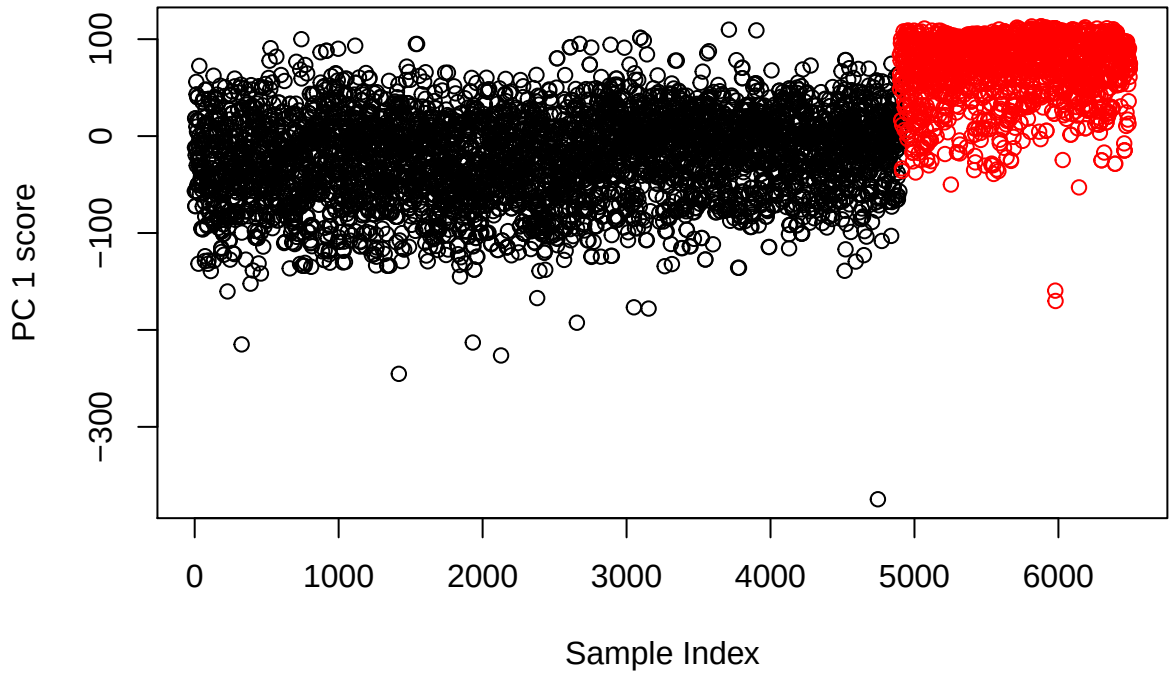

```
#how would we want to classify these wines?


#Run a PCA
pcs = prcomp(color_predict)

#Summarize the pcs
summary(pcs)
```

```
## Importance of components:
##                             PC1      PC2     PC3     PC4    PC5     PC6     PC7
## Standard deviation     58.0698 11.98513 4.13082 1.28066 1.0328 0.17714 0.14464
## Proportion of Variance  0.9538  0.04063 0.00483 0.00046 0.0003 0.00001 0.00001
## Cumulative Proportion   0.9538  0.99439 0.99921 0.99968 1.0000 0.99999 0.99999
##                            PC8    PC9    PC10      PC11
## Standard deviation     0.1211 0.1031 0.02787 0.0007517
## Proportion of Variance 0.0000 0.0000 0.00000 0.0000000
## Cumulative Proportion  1.0000 1.0000 1.00000 1.0000000
```

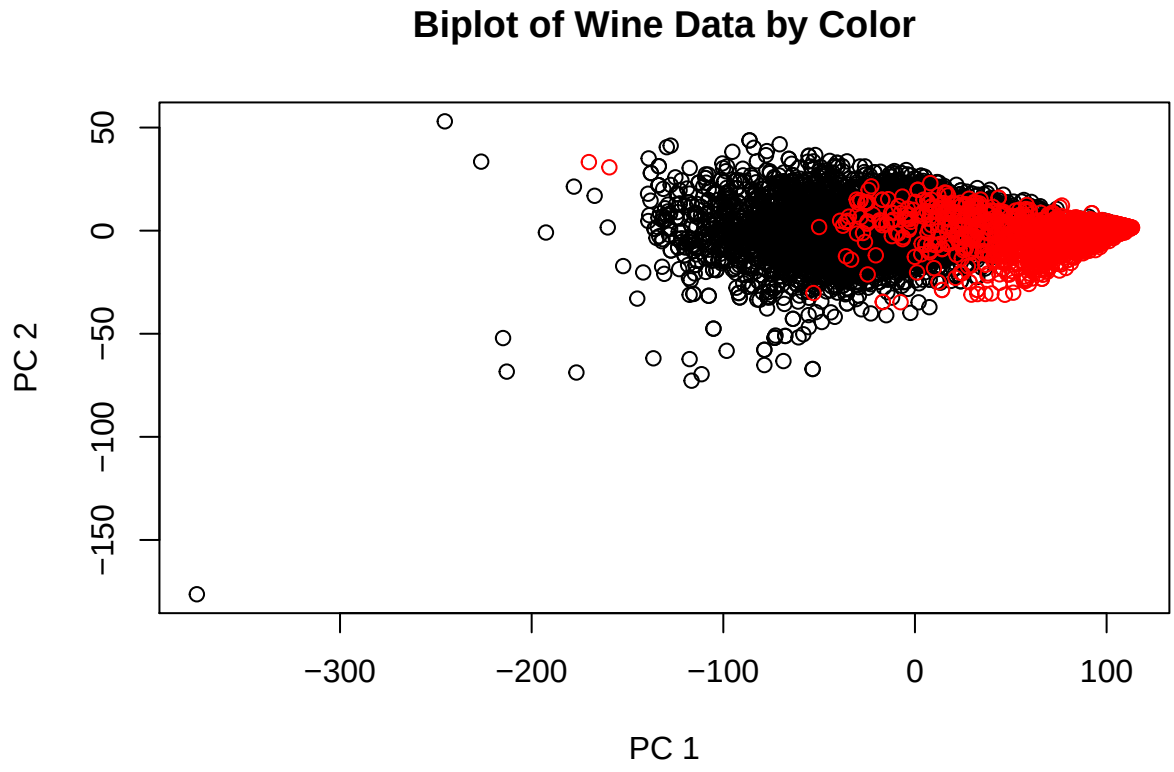
```r
screeplot(pcs, type = "lines", main = "Variance explained by PC")
```



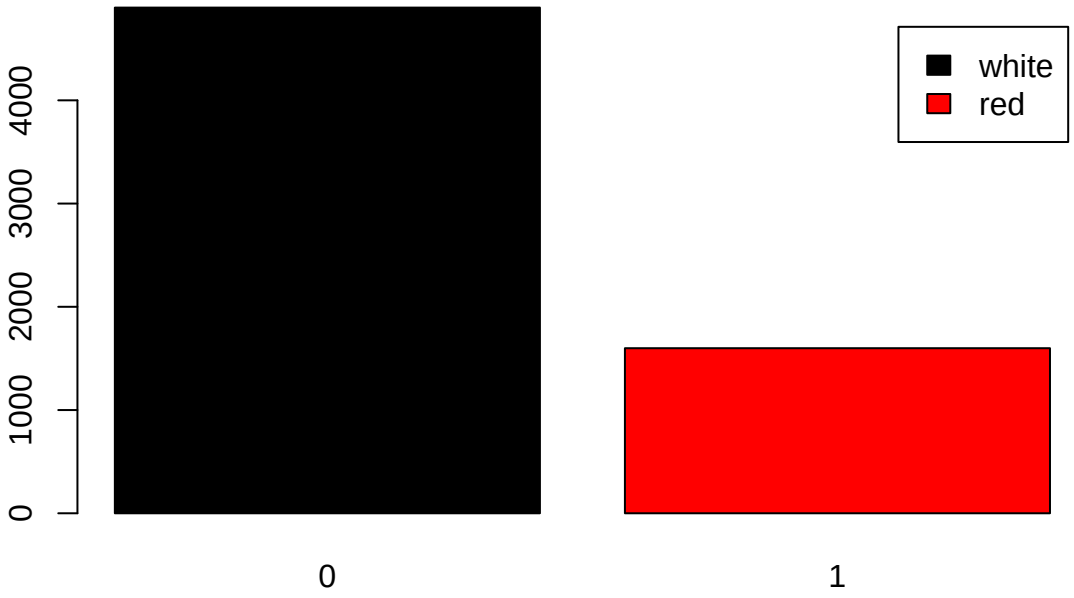

```r
plot(pcs$x[,1],
      main = paste("PC", eval(1)),
      xlab = "Sample Index",
      ylab = paste("PC", eval(1), "score"),
      col = color_response + 1)
```



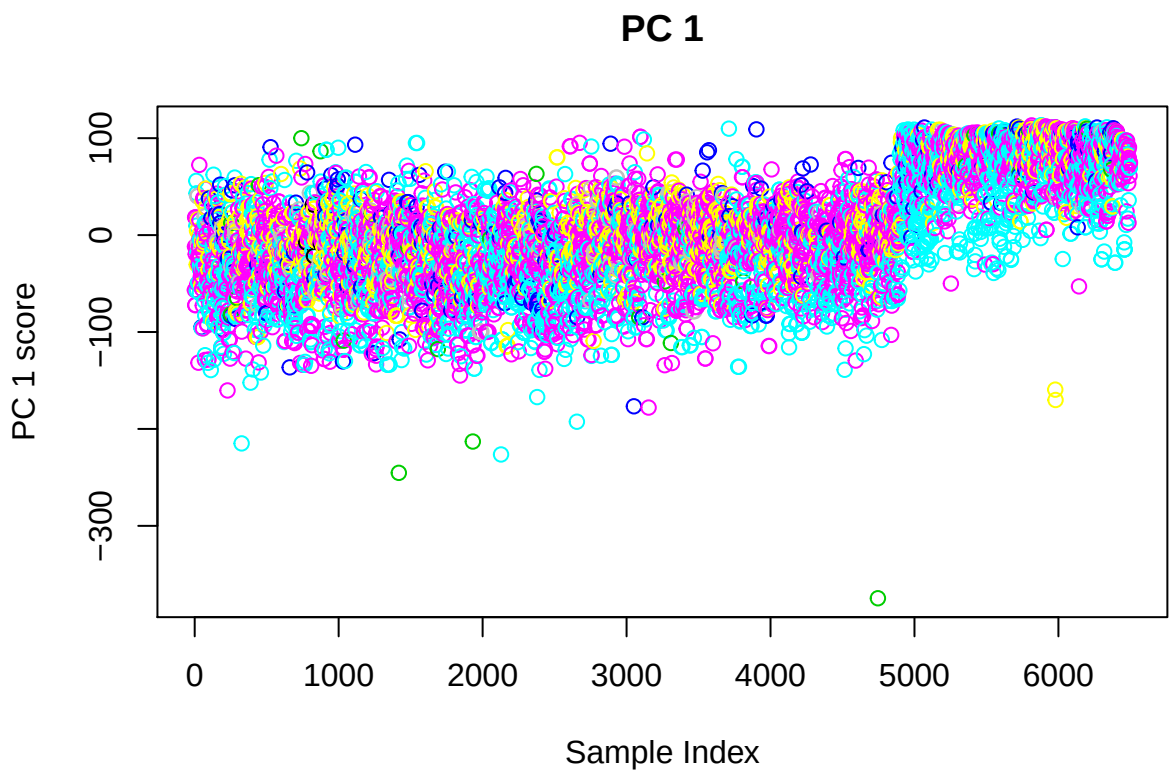

```
#first pc seems to do most of work in seperating by wine color

plot(pcs$x[,1:2],
     main = "Biplot of Wine Data by Color",
     xlab = "PC 1", ylab = "PC 2",
     col = color_response + 1)
```



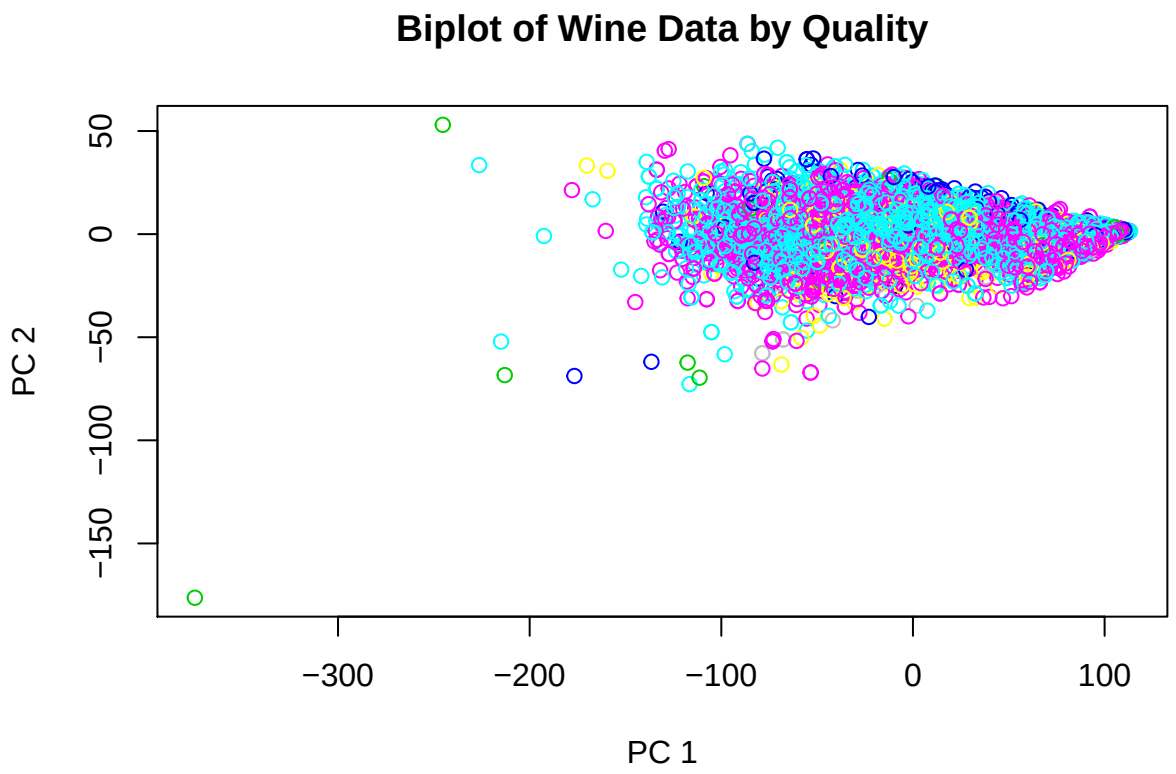

```
barplot(table(color_response), col = unique(color_response + 1), legend.text = c("white", "red"))
```



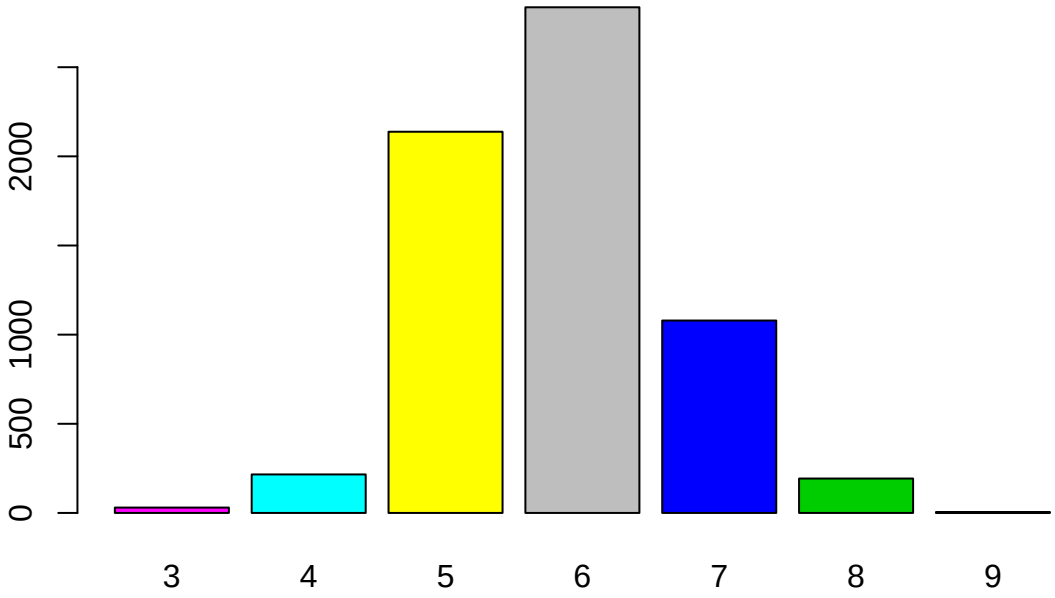

```
plot(pcs$x[,1],
     main = paste("PC", eval(1)),
     xlab = "Sample Index",
     ylab = paste("PC", eval(1), "score"),
     col = quality_response)
```



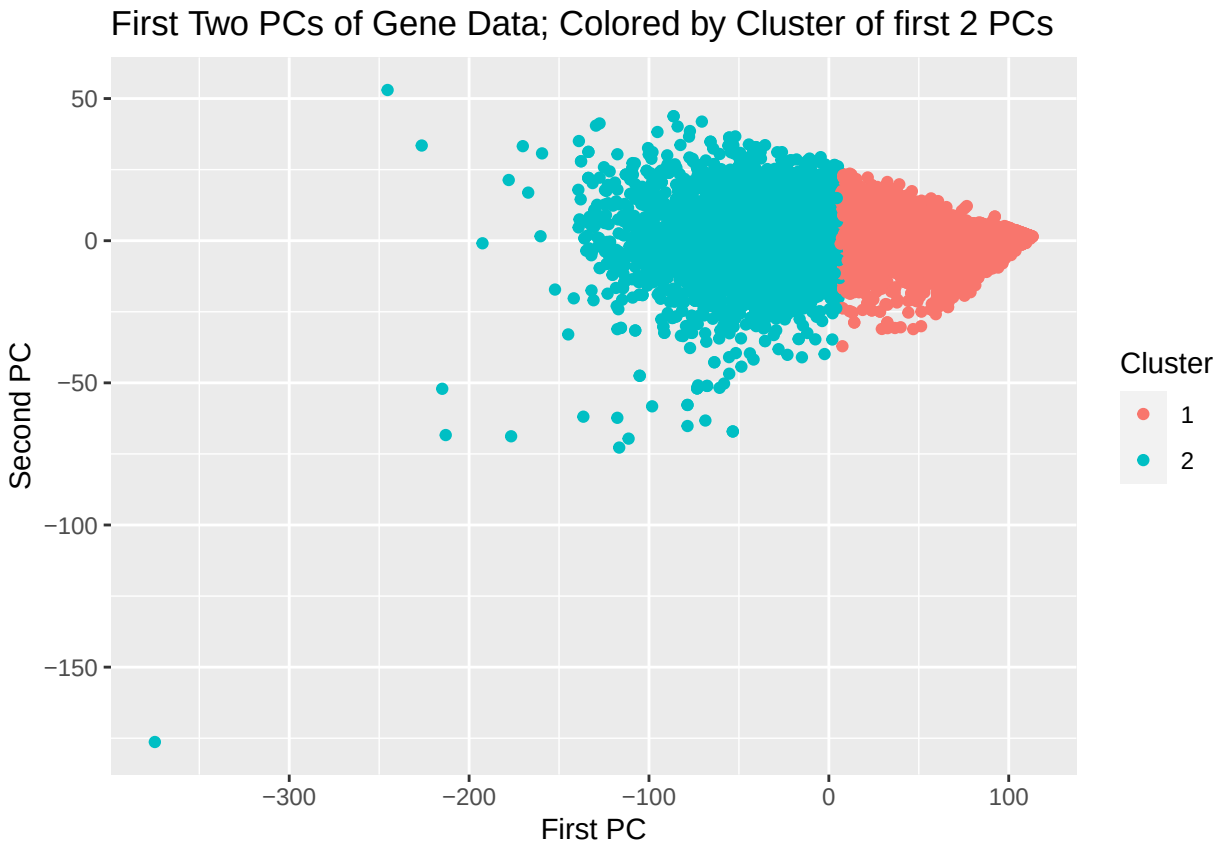

```
plot(pcs$x[,1:2],
     main = "Biplot of Wine Data by Quality",
     xlab = "PC 1", ylab = "PC 2",
     col = quality_response)
```



```r
barplot(table(quality_response), col = unique(quality_response))
```




```r
#only show first two pcs because they account for most of variance
k.pca = kmeans(pcs$x[,1:2], centers = 2)
cluster_update = as.factor(k.pca$cluster)
x <- pcs$x
ggplot() +
  geom_point(aes(x = x[,1], y = x[,2], col = cluster_update))+
  xlab("First PC") +
  ylab("Second PC") +
  scale_colour_discrete(name = "Cluster") +
  ggtitle("First Two PCs of Gene Data; Colored by Cluster of first 2 PCs")
```




```
#clustering using kmeans with 2 centers seems to be a good approximation of color
```

Analysis: One PC explains over 95 percent of the variance, and two PCs explain over 99 percent of the variance! This is something we should definitely mention, we can plot to two dimensions and still have most of the variance accounted for

Create Neural Network

```
max = apply(color_wines, 2 , max)
min = apply(color_wines, 2 , min)
wines = as.data.frame(scale(color_wines, center = min, scale = max - min))

training_size = round(.75 * nrow(wines))
indices = sample(1:nrow(wines), training_size)
training_set = wines[indices,]
testing_set = wines[-(indices),]


NN = neuralnet(type ~ ., training_set, hidden = 3 , linear.output = F )

# plot neural network
plot(NN)
```

```
predict_testNN = compute(NN, testing_set[,c(1:12)])
predict_testNN = predict_testNN$net.result

predicted_labels = c();
```

```
predicted_labels = (predict_testNN[,1] >= 0.5) *1

# Calculate Risk
nn_risk = sum(testing_set$type == predicted_labels)/length(predicted_labels)
if (nn_risk >= 0.5) {
  nn_risk = 1 - nn_risk
}
nn_risk_test = nn_risk

predict_testNN = compute(NN, training_set[,c(1:12)])
predict_testNN = predict_testNN$net.result

predicted_labels = c();
predicted_labels = (predict_testNN[,1] >= 0.5) *1

# Calculate Risk
nn_risk = sum(training_set$type == predicted_labels)/length(predicted_labels)
if (nn_risk >= 0.5) {
  nn_risk = 1 - nn_risk
}
nn_risk_train = nn_risk

model = c(1, 1)
error = c(nn_risk_train, nn_risk_test)
data = c("train", "test")
total_error = data.frame(cbind(model, error, data))

ggplot(total_error) +
  geom_boxplot(aes( x = model, y = error, color = data)) +
  ggtitle("Testing and Training Error Rate of Models")
```


Testing and Training Error Rate of Models
error
0
1
model
data
test
train