MAT00058H - Assignment 2

Matthew Knowles - 205006718

08/02/2021

Setting up

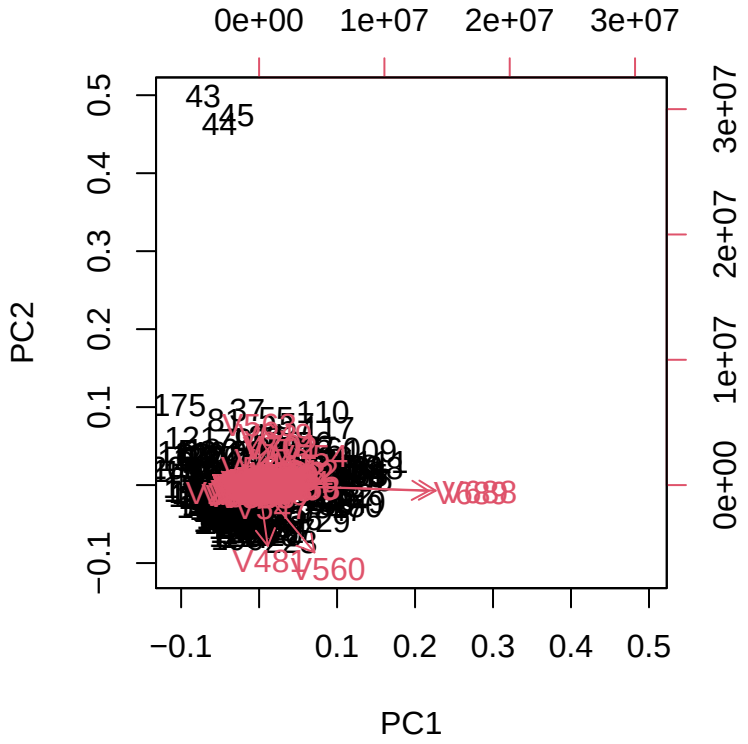
We begin by loading the data in the standard way.

```
beefData <- read.csv("/home/matthew/Documents/University/PDS/Datasets/dry_aged_beef.csv",
                     header = F)
```

Principal Component Analysis

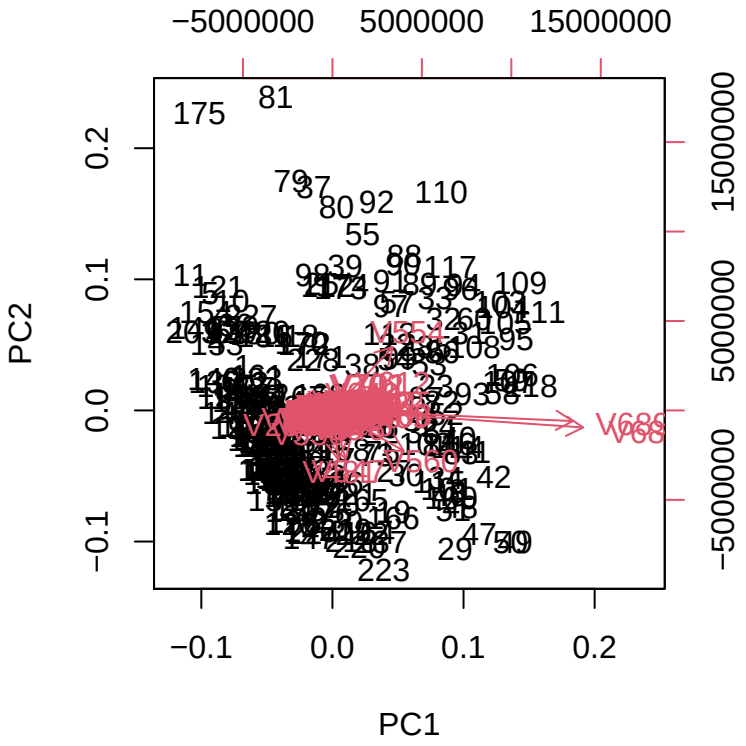
We now begin performing PCA. The aim is initially to identify any outliers.

```
beefPCA <- prcomp(beefData[,-1])
biplot(beefPCA)
```



We can very clearly see that V43, V44, V45 are outliers, so we will remove them.

```
beefData <- beefData[-(43:45),]
beefPCA <- prcomp(beefData[,-1])
biplot(beefPCA)
```



This clearly looks a lot better. Upon looking at it though one could argue that 175 and 81 are outliers, but I am going to leave them in on the reasoning that a) their value in PC1 is very in-line with a large cluster of the samples, and they don't exist very far outside the main clusters in PC2, whereas the 3 outliers we did remove were very extremely valued in PC2.

Creating Testing and Training Sets

We can use the *sample()* to generate two sets, one for training and one for testing.

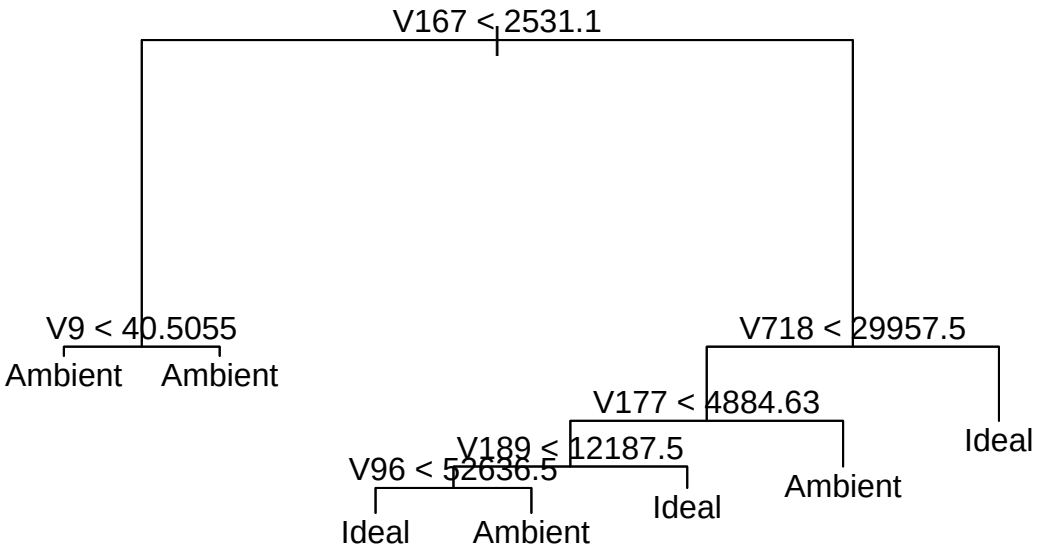
```
index = sample(1:nrow(beefData), size= round(0.7*nrow(beefData)), replace=FALSE)

trainData = beefData[index,]
testData = beefData[-index,]
```

Trees

With our training and test data all set up, we can now create a tree.

```
beefData$V1 <- as.factor(beefData$V1) #Data is non-numeric
beefTree <- tree(V1~., data=beefData)
plot(beefTree)
text(beefTree)
```



We can now check the accuracy of the model. First, for the training set:

```
tree.pred = predict(beefTree, trainData[,-1], type="class")
table(predicted = tree.pred, true= trainData[,1])
```

```
##           true
## predicted Ambient Ideal
##   Ambient      71     1
##   Ideal         0    92
```

From this we can actually calculate the acuracy.

```
(71+92)/dim(trainData)[1]
```

```
## [1] 0.9939024
```

We can see that this is 99% accurate. Let's now check for the test set:

```
tree.pred = predict(beefTree, testData[,-1], type="class")
table(predicted = tree.pred, true= testData[,1])
```

```
##           true
## predicted Ambient Ideal
##   Ambient      44     0
##   Ideal         0    27
```

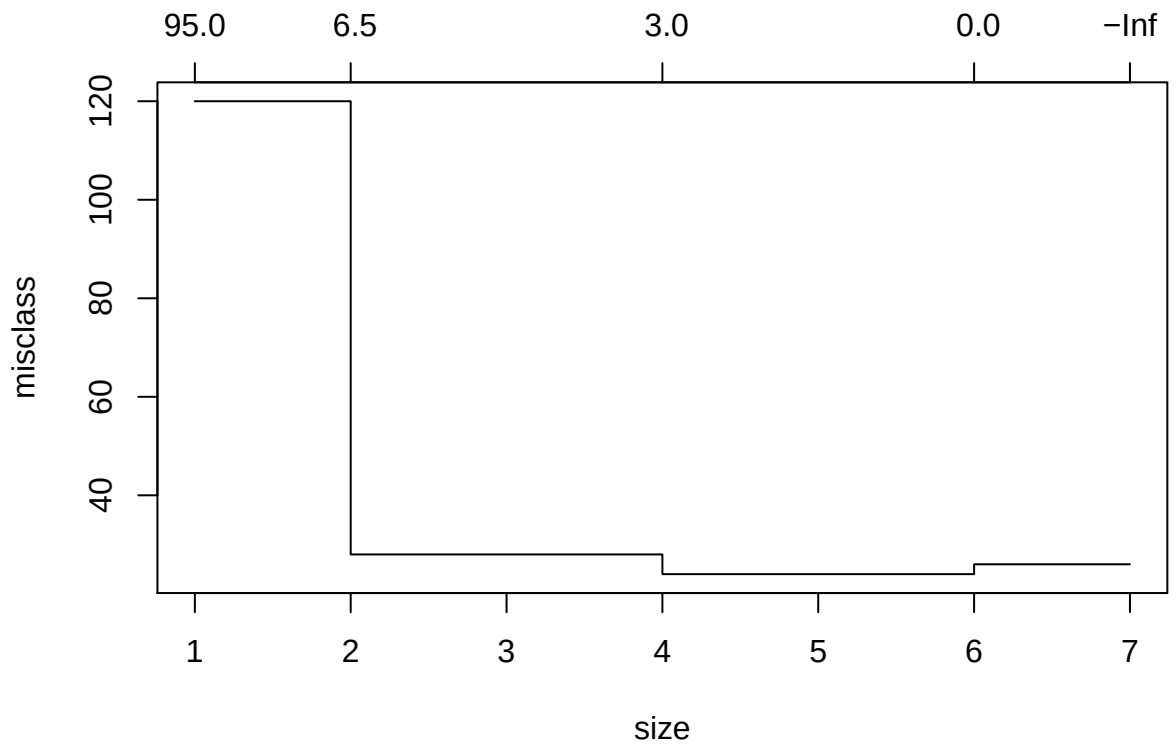
Again, the accuracy is:

```
(44+27)/dim(testData)[1]
```

```
## [1] 1
```

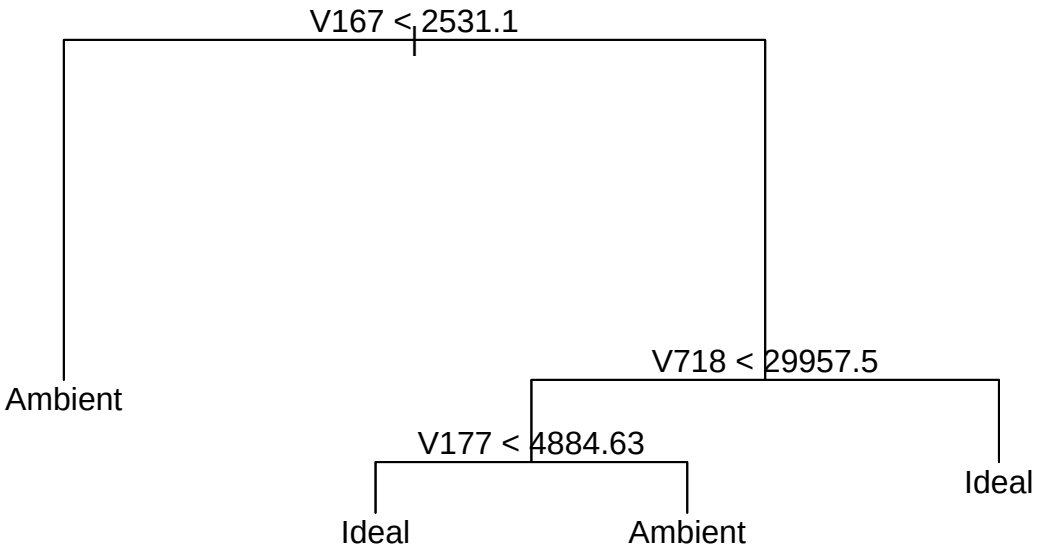
So this tells us the test set is perfectly accurate. We now move on to cross-validation. We can run k-fold cross validation on the tree using *cv.tree* as below:

```
beefCV = cv.tree(beefTree, FUN=prune.misclass)
plot(beefCV)
```



We next run the pruning code, chosing to use *best=4* based on the above CV graph.

```
pruneBeef = prune.misclass(beefTree, best=4)
plot(pruneBeef)
text(pruneBeef)
```



Now with our simpler tree, let us retest the accuracy, first for the training set.

```
tree.pred = predict(pruneBeef, testData[,-1], type="class")
table(predicted = tree.pred, true= testData[,1])
```

```
##           true
## predicted Ambient Ideal
##   Ambient      41     0
##   Ideal         3    27
```

```
(41+27)/dim(trainData)[1]
```

```
## [1] 0.4146341
```

This is considerably less accurate than last time. What about the test set?

```
tree.pred = predict(pruneBeef, trainData[,-1], type="class")
table(predicted = tree.pred, true= trainData[,1])
```

```
##          true
## predicted Ambient Ideal
##   Ambient      68     1
##   Ideal         3    92
```

```
(68+92)/dim(trainData)[1]
```

```
## [1] 0.9756098
```

On the test set, this is less accurate than last time, but much less accurate than for the training set. This would imply the model has not overfitted, since the test prediction would be much lower than the training prediction- whoch we have not seen in this case.

Forests

We can now build a random forest classifier. We will use the same training set as last time.

```
set.seed(1234)
trainData$V1 <- as.factor(trainData$V1)
beefForest = randomForest(V1~., data=trainData, mtry=35, ntree=500)
beefForest
```

```
##
## Call:
##  randomForest(formula = V1 ~ ., data = trainData, mtry = 35, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 35
##
##         OOB estimate of  error rate: 7.32%
## Confusion matrix:
##         Ambient Ideal class.error
## Ambient      62     9  0.12676056
## Ideal         3    90  0.03225806
```

With this we can read straight from the summary that the *out of bag* error estamate is 7.32% for the training set. When we first trained the model had an accuracy of 99%, so this is a bit worse. But before calling it quits we should see how well it predicts training and test data and make that comparoson.

```
beefForestPred = predict(beefForest, trainData[,-1], type="class")
table(predicted=beefForestPred, true = trainData[,1])
```

```
##          true
## predicted Ambient Ideal
##   Ambient      71     0
##   Ideal         0    93
```

```
(71+93)/dim(trainData)[1]
```

```
## [1] 1
```

Which is perfectly accurate, as was the case before. What about the test set though?

```
beefForestPred = predict(beefForest, testData[,-1], type="class")
table(predicted=beefForestPred, true = testData[,1])
```

```
##          true
## predicted Ambient Ideal
##   Ambient      35     0
##   Ideal         9    27
```

```
(35+27)/dim(testData)[1]
```

```
## [1] 0.8732394
```

So the random forest has an accuracy of 100% for training data nd 87% for test data.

```
100*(1-(35+27)/dim(testData)[1])
```

```
## [1] 12.67606
```

From the above calculation, we can see that the error for the test set is higher than the OOB. This combined with the fact that the training accuracy was much higher than for the test set suggests that this model is overfitting. The fact this model is overfitting suggests why this model is less accurate than our best decision tree when it comes to predicting the test set, which we were able to predict at ~97% accuracy with a pruned decision tree in the prior section.

Improving the forest

We tried 500 trees last time and it resulted in over-fitting, what about with 100?

```
set.seed(1234)
trainData$V1 <- as.factor(trainData$V1)
beefForestN = randomForest(V1~., data=trainData, mtry=35, ntree=100)
beefForestN
```

```
##
## Call:
##  randomForest(formula = V1 ~ ., data = trainData, mtry = 35, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 35
##
##         OOB estimate of  error rate: 8.54%
## Confusion matrix:
##         Ambient Ideal class.error
## Ambient      61    10  0.14084507
## Ideal         4    89  0.04301075
```

We can see the OOB error rate is slightly higher, but what about accuracy for test and training? We repeat the process as before:

```
beefForestPredN = predict(beefForestN, trainData[,-1], type="class")
table(predicted=beefForestPredN, true = trainData[,1])
```

```
##          true
## predicted Ambient Ideal
##   Ambient      71     0
##   Ideal         0    93
```

```
(71+93)/dim(trainData)[1]
```

```
## [1] 1
```

So there is no difference for training.

```
beefForestPred = predict(beefForest, testData[,-1], type="class")
table(predicted=beefForestPred, true = testData[,1])
```

```
##          true
## predicted Ambient Ideal
##   Ambient      35     0
##   Ideal         9    27
```

```
(35+27)/dim(testData)[1]
```

```
## [1] 0.8732394
```

Which is 2% higher than last time, but still much lower than training, and so is still over classifying.

Concluding Remarks

To conclude, we can see that the best classifier here for predicting the test set was the pruned decision tree, coming in at 97% accuracy.