Homework 3

Austin Vanderlyn ajl745

8/4/2022

Homework 3

12.2. In Exercise 4.4, we described a data set which contained 96 oil samples each from one of seven types of oils (pumpkin, sunflower, peanut, olive, soybean, rapeseed, and corn). Gas chromatography was performed on each sample and the percentage of each type of 7 fatty acids was determined. We would like to use these data to build a model that predicts the type of oil based on a sample's fatty acid percentages.

(a) Like the hepatic injury data, these data suffer from extreme imbalance. Given this imbalance, should the data be split into training and test sets?

To be honest I'm not 100% sure what the question means by extreme imbalance (not normal distribution?) so I want to take a closer look at it first.

Libraries;

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version 4.1.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2

## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.3
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.1.2

## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(MASS)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.3

## Loading required package: Matrix

## Loaded glmnet 4.1-4
```

```
library(MLmetrics)
```

```
## Warning: package 'MLmetrics' was built under R version 4.1.3

##
## Attaching package: 'MLmetrics'

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE

## The following object is masked from 'package:base':
##
##     Recall
```

```
library(kernlab)
```

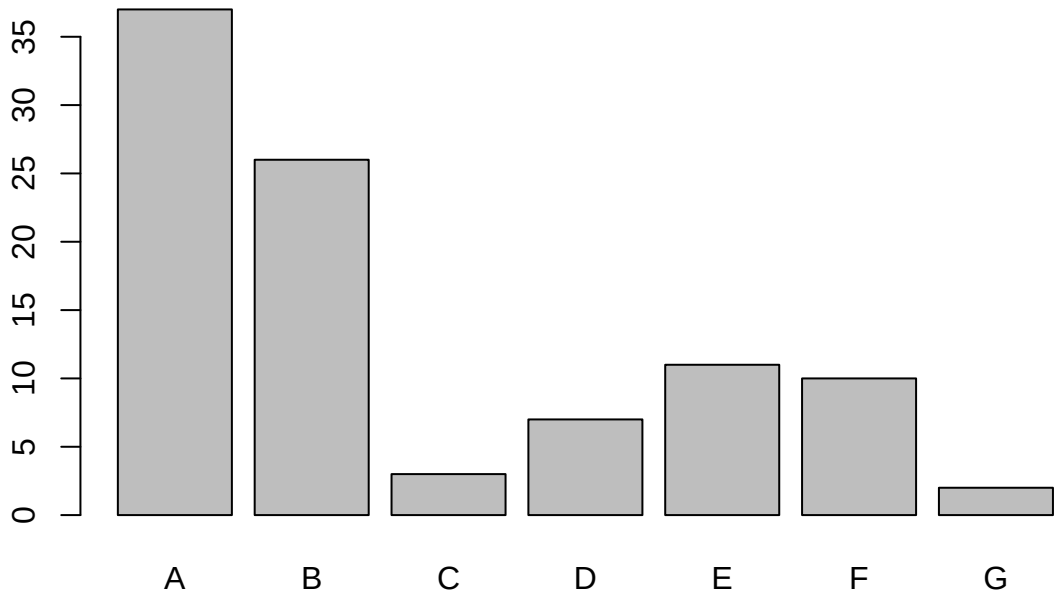
```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
data(oil)
```

plots

```
plot(oilType)
```



Yes, that is definitely not a normal distribution. I don't think that splitting into train and test would necessarily be appropriate here, because the number of observations is so small, and the outcome variable has seven different levels.

(b) Which classification statistic would you choose to optimize for this exercise and why?

Well, with a continuous response variable, accuracy is not generally the best metric, but since this is a classification problem, accuracy is a decent enough metric to use.

(c) Of the models presented in this chapter, which performs best on these data? Which oil type does the model most accurately predict? Least accurately predict?

The models presented in this chapter are; logistic regression, linear discriminant analysis, partial least squares discriminant analysis, penalized models, and nearest shrunken centroids, so those are the various types of models that I will to test this out on, except for logistic regression. That won't really work because this is a seven-level categorical response variable, not a binary one.

Create ctrl function;

```
set.seed(123)
ctrl = trainControl(method = "LGOCV",
                    classProbs = TRUE,
                    savePredictions = TRUE)
```

Fit least discriminant analysis regression model;

```
set.seed(123)
ldaTune = train(fattyAcids, oilType,
                method = "lda",
                preProcess = c("center", "scale"),
                metric = "Accuracy",
                trControl = ctrl)
ldaTune
```

```
## Linear Discriminant Analysis
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.958     0.9400536
```

LDA predictions;

```
testResults = data.frame(obs = oilType,
                         lda = predict(ldaTune, fattyAcids))
testResults
```

```
##    obs lda
## 1    A   A
## 2    A   A
## 3    A   A
## 4    A   A
## 5    A   A
## 6    A   A
## 7    A   A
## 8    A   A
## 9    A   A
## 10   A   A
## 11   A   A
## 12   A   A
## 13   A   A
## 14   A   A
## 15   A   E
## 16   A   A
## 17   A   A
## 18   A   A
## 19   B   B
## 20   B   B
## 21   B   B
## 22   B   B
## 23   B   B
## 24   B   B
```

```
## 25   E   E
## 26   E   E
## 27   G   G
## 28   F   F
## 29   C   C
## 30   B   B
## 31   B   B
## 32   B   B
## 33   B   B
## 34   B   B
## 35   B   B
## 36   A   A
## 37   A   B
## 38   A   B
## 39   A   A
## 40   A   A
## 41   A   A
## 42   D   D
## 43   D   D
## 44   E   E
## 45   E   E
## 46   E   E
## 47   E   E
## 48   E   E
## 49   E   E
## 50   E   E
## 51   E   E
## 52   D   D
## 53   D   D
## 54   D   D
## 55   D   D
## 56   D   D
## 57   F   F
## 58   F   F
## 59   F   F
## 60   F   F
## 61   C   C
## 62   C   C
## 63   A   A
## 64   A   A
## 65   A   A
## 66   A   A
## 67   A   A
## 68   A   A
## 69   A   A
## 70   A   A
## 71   A   A
## 72   A   A
## 73   A   A
## 74   A   A
## 75   A   A
## 76   B   B
## 77   B   B
## 78   B   B
```

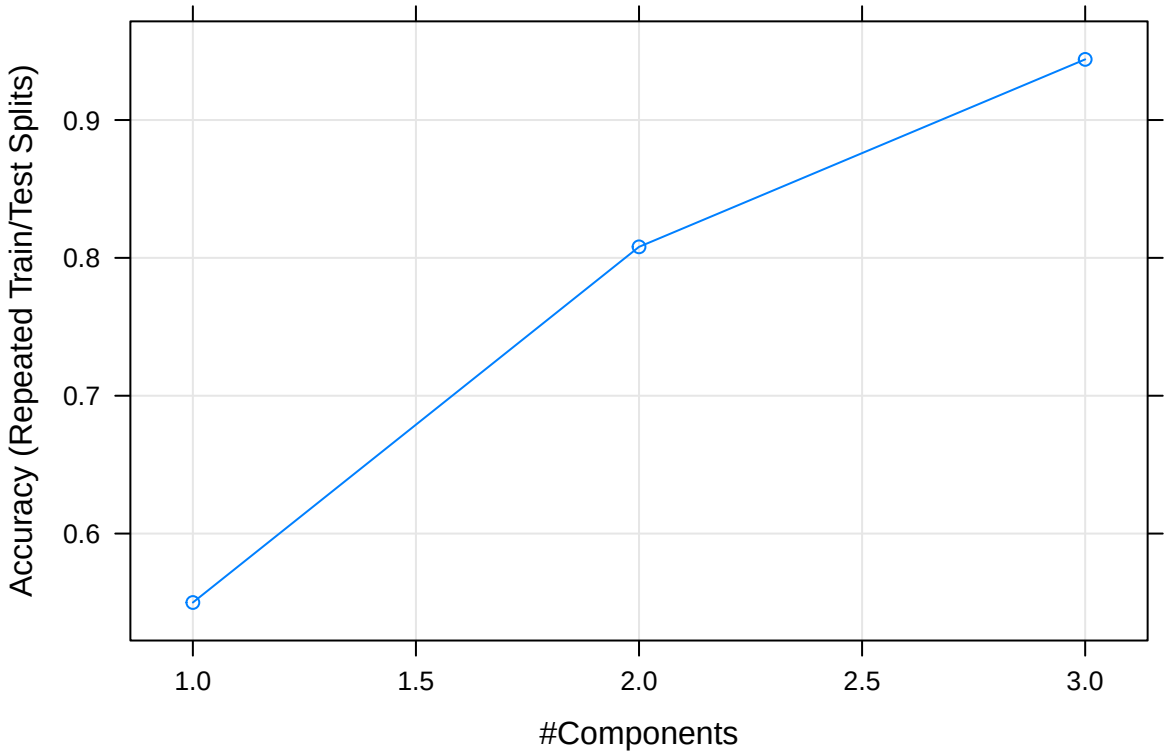
```
## 79   B   B
## 80   B   B
## 81   B   B
## 82   B   B
## 83   B   B
## 84   B   B
## 85   B   B
## 86   B   B
## 87   B   B
## 88   B   B
## 89   B   B
## 90   F   F
## 91   F   F
## 92   G   G
## 93   F   F
## 94   F   F
## 95   F   F
## 96   E   E
```

Fit PLSDA model;

```
set.seed(123)
plsdaTune = train(fattyAcids, oilType,
                  method = "pls",
                  preProcess = c("center", "scale"),
                  metric = "Accuracy",
                  trControl = ctrl)
plsdaTune
```

```
## Partial Least Squares
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results across tuning parameters:
##
##   ncomp  Accuracy  Kappa
##   1      0.550     0.2547986
##   2      0.808     0.7005337
##   3      0.944     0.9181376
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was ncomp = 3.
```

```
plot(plsdaTune)
```



store PLSDA predictions;

```
testResults$plsda = predict(plsdaTune, fattyAcids)
```

Fit penalized model;

```
glmnGrid = expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
                       .lambda = seq(.01, .2, length = 40))
set.seed(123)
glmnTune = train(fattyAcids, oilType,
                 method = "glmnet",
                 preProcess = c("center", "scale"),
                 metric = "Accuracy",
                 trControl = ctrl,
                 tuneGrid = glmnGrid)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
glmnTune
```

```
## glmnet
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda      Accuracy  Kappa
##   0.0    0.01000000  0.956     0.9371745
##   0.0    0.01487179  0.956     0.9371745
##   0.0    0.01974359  0.956     0.9371745
##   0.0    0.02461538  0.956     0.9371745
##   0.0    0.02948718  0.956     0.9371745
##   0.0    0.03435897  0.956     0.9371745
##   0.0    0.03923077  0.956     0.9371745
```

```
##   0.0    0.04410256  0.954     0.9342655
##   0.0    0.04897436  0.954     0.9342655
##   0.0    0.05384615  0.954     0.9342655
##   0.0    0.05871795  0.954     0.9342655
##   0.0    0.06358974  0.954     0.9342655
##   0.0    0.06846154  0.954     0.9342655
##   0.0    0.07333333  0.954     0.9342655
##   0.0    0.07820513  0.954     0.9342655
##   0.0    0.08307692  0.952     0.9313351
##   0.0    0.08794872  0.952     0.9313351
##   0.0    0.09282051  0.952     0.9313351
##   0.0    0.09769231  0.952     0.9313351
##   0.0    0.10256410  0.952     0.9313351
##   0.0    0.10743590  0.952     0.9313351
##   0.0    0.11230769  0.952     0.9313351
##   0.0    0.11717949  0.952     0.9313351
##   0.0    0.12205128  0.952     0.9313351
##   0.0    0.12692308  0.952     0.9313351
##   0.0    0.13179487  0.952     0.9313351
##   0.0    0.13666667  0.952     0.9313351
##   0.0    0.14153846  0.950     0.9284261
##   0.0    0.14641026  0.950     0.9284261
##   0.0    0.15128205  0.950     0.9284261
##   0.0    0.15615385  0.950     0.9284261
##   0.0    0.16102564  0.950     0.9284261
##   0.0    0.16589744  0.950     0.9284261
##   0.0    0.17076923  0.950     0.9284261
##   0.0    0.17564103  0.950     0.9284261
##   0.0    0.18051282  0.950     0.9284261
##   0.0    0.18538462  0.948     0.9254956
##   0.0    0.19025641  0.946     0.9225652
##   0.0    0.19512821  0.946     0.9225652
##   0.0    0.20000000  0.946     0.9225652
##   0.1    0.01000000  0.960     0.9429402
##   0.1    0.01487179  0.958     0.9400521
##   0.1    0.01974359  0.958     0.9400521
##   0.1    0.02461538  0.958     0.9400521
##   0.1    0.02948718  0.958     0.9400521
##   0.1    0.03435897  0.958     0.9400521
##   0.1    0.03923077  0.958     0.9400521
##   0.1    0.04410256  0.958     0.9400521
##   0.1    0.04897436  0.958     0.9400521
##   0.1    0.05384615  0.958     0.9400521
##   0.1    0.05871795  0.958     0.9400521
##   0.1    0.06358974  0.958     0.9400521
##   0.1    0.06846154  0.958     0.9400521
##   0.1    0.07333333  0.956     0.9371745
##   0.1    0.07820513  0.956     0.9371745
##   0.1    0.08307692  0.956     0.9371745
##   0.1    0.08794872  0.956     0.9371745
##   0.1    0.09282051  0.956     0.9371745
##   0.1    0.09769231  0.956     0.9371745
##   0.1    0.10256410  0.956     0.9371745
##   0.1    0.10743590  0.956     0.9371745
```

```
##   0.1    0.11230769  0.956     0.9371745
##   0.1    0.11717949  0.956     0.9371745
##   0.1    0.12205128  0.956     0.9371745
##   0.1    0.12692308  0.956     0.9371745
##   0.1    0.13179487  0.956     0.9371745
##   0.1    0.13666667  0.956     0.9371745
##   0.1    0.14153846  0.956     0.9371745
##   0.1    0.14641026  0.956     0.9371745
##   0.1    0.15128205  0.956     0.9371745
##   0.1    0.15615385  0.954     0.9342441
##   0.1    0.16102564  0.952     0.9313351
##   0.1    0.16589744  0.952     0.9313351
##   0.1    0.17076923  0.952     0.9313351
##   0.1    0.17564103  0.952     0.9313351
##   0.1    0.18051282  0.950     0.9282972
##   0.1    0.18538462  0.948     0.9251724
##   0.1    0.19025641  0.948     0.9251724
##   0.1    0.19512821  0.948     0.9251724
##   0.1    0.20000000  0.946     0.9221346
##   0.2    0.01000000  0.960     0.9429402
##   0.2    0.01487179  0.958     0.9400521
##   0.2    0.01974359  0.958     0.9400521
##   0.2    0.02461538  0.958     0.9400521
##   0.2    0.02948718  0.958     0.9400521
##   0.2    0.03435897  0.958     0.9400521
##   0.2    0.03923077  0.958     0.9400521
##   0.2    0.04410256  0.958     0.9400521
##   0.2    0.04897436  0.958     0.9400521
##   0.2    0.05384615  0.958     0.9400521
##   0.2    0.05871795  0.958     0.9400521
##   0.2    0.06358974  0.958     0.9400521
##   0.2    0.06846154  0.958     0.9400521
##   0.2    0.07333333  0.958     0.9400521
##   0.2    0.07820513  0.958     0.9400521
##   0.2    0.08307692  0.958     0.9400521
##   0.2    0.08794872  0.958     0.9400521
##   0.2    0.09282051  0.958     0.9400521
##   0.2    0.09769231  0.958     0.9400521
##   0.2    0.10256410  0.958     0.9400521
##   0.2    0.10743590  0.958     0.9400521
##   0.2    0.11230769  0.958     0.9400521
##   0.2    0.11717949  0.958     0.9400521
##   0.2    0.12205128  0.958     0.9400521
##   0.2    0.12692308  0.958     0.9400521
##   0.2    0.13179487  0.958     0.9400521
##   0.2    0.13666667  0.958     0.9400521
##   0.2    0.14153846  0.958     0.9400521
##   0.2    0.14641026  0.958     0.9400521
##   0.2    0.15128205  0.958     0.9400521
##   0.2    0.15615385  0.958     0.9400521
##   0.2    0.16102564  0.958     0.9400521
##   0.2    0.16589744  0.950     0.9279437
##   0.2    0.17076923  0.948     0.9249362
##   0.2    0.17564103  0.948     0.9249362
```

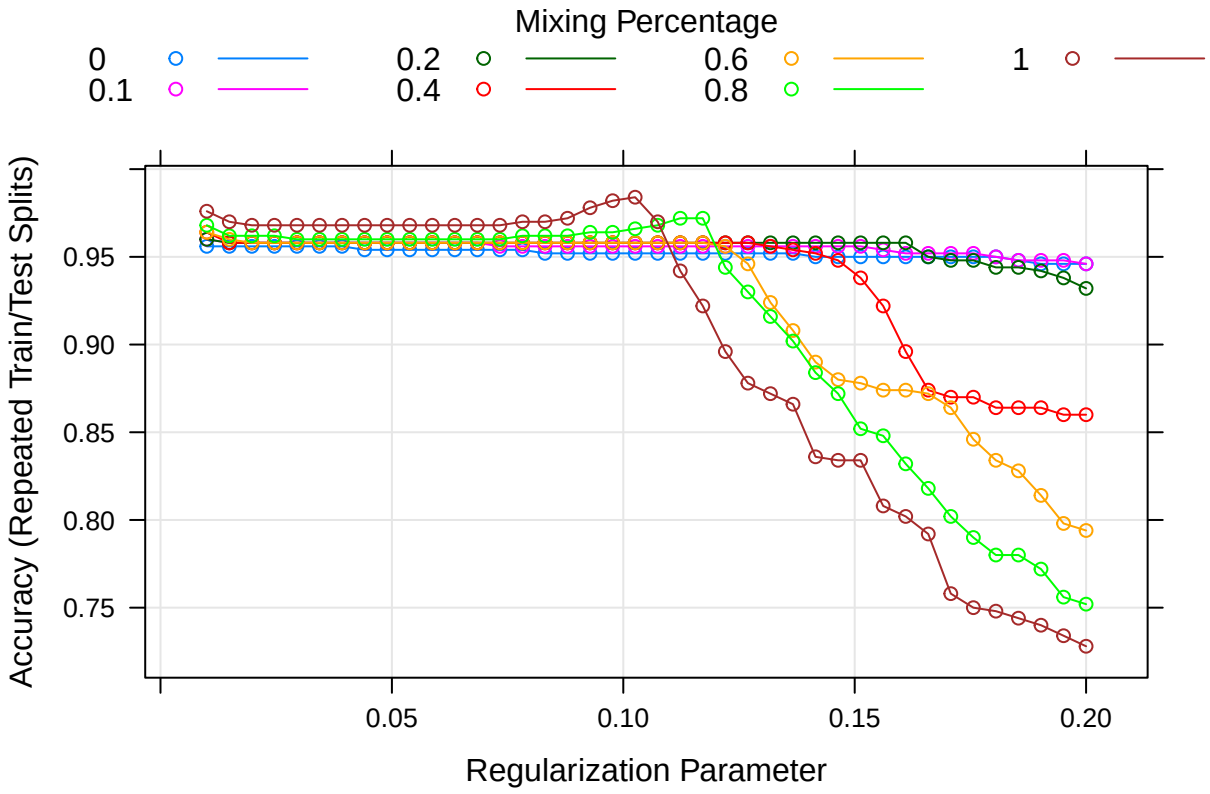
```
##   0.2    0.18051282  0.944     0.9188564
##   0.2    0.18538462  0.944     0.9188564
##   0.2    0.19025641  0.942     0.9158186
##   0.2    0.19512821  0.938     0.9097347
##   0.2    0.20000000  0.932     0.9005345
##   0.4    0.01000000  0.964     0.9486248
##   0.4    0.01487179  0.958     0.9400521
##   0.4    0.01974359  0.958     0.9400521
##   0.4    0.02461538  0.958     0.9400521
##   0.4    0.02948718  0.958     0.9400521
##   0.4    0.03435897  0.958     0.9400521
##   0.4    0.03923077  0.958     0.9400521
##   0.4    0.04410256  0.958     0.9400521
##   0.4    0.04897436  0.958     0.9400521
##   0.4    0.05384615  0.958     0.9400521
##   0.4    0.05871795  0.958     0.9400521
##   0.4    0.06358974  0.958     0.9400521
##   0.4    0.06846154  0.958     0.9400521
##   0.4    0.07333333  0.958     0.9400521
##   0.4    0.07820513  0.958     0.9400521
##   0.4    0.08307692  0.958     0.9400521
##   0.4    0.08794872  0.958     0.9400521
##   0.4    0.09282051  0.958     0.9400521
##   0.4    0.09769231  0.958     0.9400521
##   0.4    0.10256410  0.958     0.9400521
##   0.4    0.10743590  0.958     0.9400521
##   0.4    0.11230769  0.958     0.9400521
##   0.4    0.11717949  0.958     0.9400521
##   0.4    0.12205128  0.958     0.9400521
##   0.4    0.12692308  0.958     0.9400521
##   0.4    0.13179487  0.956     0.9370143
##   0.4    0.13666667  0.954     0.9339764
##   0.4    0.14153846  0.952     0.9309345
##   0.4    0.14641026  0.948     0.9249004
##   0.4    0.15128205  0.938     0.9096211
##   0.4    0.15615385  0.922     0.8847582
##   0.4    0.16102564  0.896     0.8439189
##   0.4    0.16589744  0.874     0.8091812
##   0.4    0.17076923  0.870     0.8025566
##   0.4    0.17564103  0.870     0.8018071
##   0.4    0.18051282  0.864     0.7918254
##   0.4    0.18538462  0.864     0.7918254
##   0.4    0.19025641  0.864     0.7918254
##   0.4    0.19512821  0.860     0.7852443
##   0.4    0.20000000  0.860     0.7847430
##   0.6    0.01000000  0.964     0.9486248
##   0.6    0.01487179  0.960     0.9429402
##   0.6    0.01974359  0.958     0.9400521
##   0.6    0.02461538  0.958     0.9400521
##   0.6    0.02948718  0.958     0.9400521
##   0.6    0.03435897  0.958     0.9400521
##   0.6    0.03923077  0.958     0.9400521
##   0.6    0.04410256  0.958     0.9400521
##   0.6    0.04897436  0.958     0.9400521
```

```
##   0.6    0.05384615  0.958     0.9400521
##   0.6    0.05871795  0.958     0.9400521
##   0.6    0.06358974  0.958     0.9400521
##   0.6    0.06846154  0.958     0.9400521
##   0.6    0.07333333  0.958     0.9400521
##   0.6    0.07820513  0.958     0.9400521
##   0.6    0.08307692  0.958     0.9400521
##   0.6    0.08794872  0.958     0.9400521
##   0.6    0.09282051  0.958     0.9400521
##   0.6    0.09769231  0.958     0.9400521
##   0.6    0.10256410  0.958     0.9400521
##   0.6    0.10743590  0.958     0.9400521
##   0.6    0.11230769  0.958     0.9400521
##   0.6    0.11717949  0.958     0.9400521
##   0.6    0.12205128  0.956     0.9370484
##   0.6    0.12692308  0.946     0.9219840
##   0.6    0.13179487  0.924     0.8876737
##   0.6    0.13666667  0.908     0.8625040
##   0.6    0.14153846  0.890     0.8340662
##   0.6    0.14641026  0.880     0.8179101
##   0.6    0.15128205  0.878     0.8143797
##   0.6    0.15615385  0.874     0.8080645
##   0.6    0.16102564  0.874     0.8075176
##   0.6    0.16589744  0.872     0.8039616
##   0.6    0.17076923  0.864     0.7904818
##   0.6    0.17564103  0.846     0.7596814
##   0.6    0.18051282  0.834     0.7388832
##   0.6    0.18538462  0.828     0.7283050
##   0.6    0.19025641  0.814     0.7035605
##   0.6    0.19512821  0.798     0.6754041
##   0.6    0.20000000  0.794     0.6682070
##   0.8    0.01000000  0.968     0.9544010
##   0.8    0.01487179  0.962     0.9458283
##   0.8    0.01974359  0.962     0.9458283
##   0.8    0.02461538  0.962     0.9458283
##   0.8    0.02948718  0.960     0.9429402
##   0.8    0.03435897  0.960     0.9429402
##   0.8    0.03923077  0.960     0.9429402
##   0.8    0.04410256  0.960     0.9429402
##   0.8    0.04897436  0.960     0.9429402
##   0.8    0.05384615  0.960     0.9429402
##   0.8    0.05871795  0.960     0.9429402
##   0.8    0.06358974  0.960     0.9429402
##   0.8    0.06846154  0.960     0.9429402
##   0.8    0.07333333  0.960     0.9429402
##   0.8    0.07820513  0.962     0.9458283
##   0.8    0.08307692  0.962     0.9458283
##   0.8    0.08794872  0.962     0.9458283
##   0.8    0.09282051  0.964     0.9487164
##   0.8    0.09769231  0.964     0.9487164
##   0.8    0.10256410  0.966     0.9516045
##   0.8    0.10743590  0.968     0.9543701
##   0.8    0.11230769  0.972     0.9597857
##   0.8    0.11717949  0.972     0.9592505
```

```
##   0.8    0.12205128  0.944     0.9161633
##   0.8    0.12692308  0.930     0.8943547
##   0.8    0.13179487  0.916     0.8724006
##   0.8    0.13666667  0.902     0.8508435
##   0.8    0.14153846  0.884     0.8222238
##   0.8    0.14641026  0.872     0.8026495
##   0.8    0.15128205  0.852     0.7700774
##   0.8    0.15615385  0.848     0.7626054
##   0.8    0.16102564  0.832     0.7354331
##   0.8    0.16589744  0.818     0.7111845
##   0.8    0.17076923  0.802     0.6824537
##   0.8    0.17564103  0.790     0.6608431
##   0.8    0.18051282  0.780     0.6431565
##   0.8    0.18538462  0.780     0.6433107
##   0.8    0.19025641  0.772     0.6298140
##   0.8    0.19512821  0.756     0.6021826
##   0.8    0.20000000  0.752     0.5952021
##   1.0    0.01000000  0.976     0.9658619
##   1.0    0.01487179  0.970     0.9572891
##   1.0    0.01974359  0.968     0.9544010
##   1.0    0.02461538  0.968     0.9544010
##   1.0    0.02948718  0.968     0.9544010
##   1.0    0.03435897  0.968     0.9544010
##   1.0    0.03923077  0.968     0.9544010
##   1.0    0.04410256  0.968     0.9544010
##   1.0    0.04897436  0.968     0.9544010
##   1.0    0.05384615  0.968     0.9544010
##   1.0    0.05871795  0.968     0.9544010
##   1.0    0.06358974  0.968     0.9544010
##   1.0    0.06846154  0.968     0.9544010
##   1.0    0.07333333  0.968     0.9544010
##   1.0    0.07820513  0.970     0.9572891
##   1.0    0.08307692  0.970     0.9572891
##   1.0    0.08794872  0.972     0.9601772
##   1.0    0.09282051  0.978     0.9685656
##   1.0    0.09769231  0.982     0.9741310
##   1.0    0.10256410  0.984     0.9767099
##   1.0    0.10743590  0.970     0.9553312
##   1.0    0.11230769  0.942     0.9126273
##   1.0    0.11717949  0.922     0.8820401
##   1.0    0.12205128  0.896     0.8412352
##   1.0    0.12692308  0.878     0.8125735
##   1.0    0.13179487  0.872     0.8030438
##   1.0    0.13666667  0.866     0.7928679
##   1.0    0.14153846  0.836     0.7459677
##   1.0    0.14641026  0.834     0.7431485
##   1.0    0.15128205  0.834     0.7431485
##   1.0    0.15615385  0.808     0.7003179
##   1.0    0.16102564  0.802     0.6900510
##   1.0    0.16589744  0.792     0.6723960
##   1.0    0.17076923  0.758     0.6125302
##   1.0    0.17564103  0.750     0.5977161
##   1.0    0.18051282  0.748     0.5923965
##   1.0    0.18538462  0.744     0.5840453
```

```
##   1.0    0.19025641  0.740     0.5752718
##   1.0    0.19512821  0.734     0.5648384
##   1.0    0.20000000  0.728     0.5542727
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 1 and lambda = 0.1025641.
```

```
plot(glmnTune)
```



store glmn predictions;

```
testResults$glmn = predict(glmnTune, fattyAcids)
```

fit nearest shrunken centroid model;

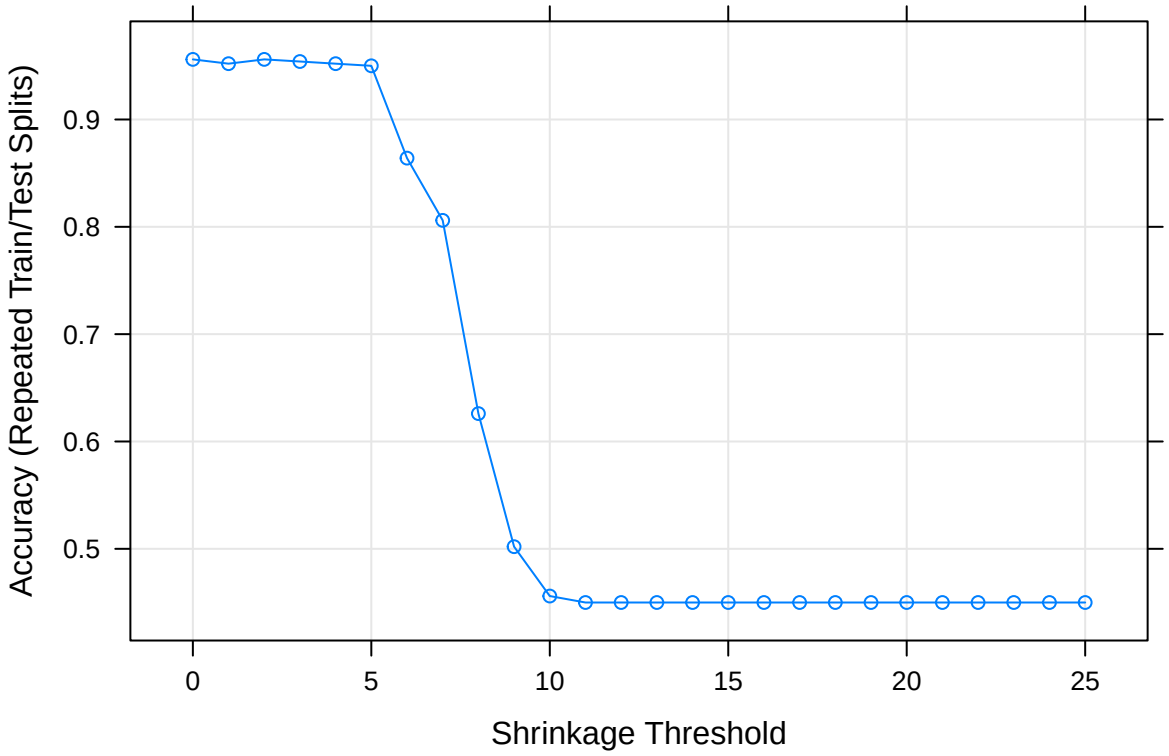
```
nscGrid = data.frame(.threshold = 0:25)
set.seed(123)
nscTune = train(fattyAcids, oilType,
                method = "pam",
                preProcess = c("center", "scale"),
                metric = "Accuracy",
                trControl = ctrl,
                tuneGrid = nscGrid)
```

```
## 1111111111111111111111111
```

```
nscTune
```

```
## Nearest Shrunken Centroids
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results across tuning parameters:
##
##   threshold  Accuracy  Kappa
##    0         0.956     0.93730639
##    1         0.952     0.93171370
##    2         0.956     0.93737292
##    3         0.954     0.93469373
##    4         0.952     0.93223788
##    5         0.950     0.92828617
##    6         0.864     0.80095718
##    7         0.806     0.71614608
##    8         0.626     0.41083152
##    9         0.502     0.15499730
##   10         0.456     0.01830508
##   11         0.450     0.00000000
##   12         0.450     0.00000000
##   13         0.450     0.00000000
##   14         0.450     0.00000000
##   15         0.450     0.00000000
##   16         0.450     0.00000000
##   17         0.450     0.00000000
##   18         0.450     0.00000000
##   19         0.450     0.00000000
##   20         0.450     0.00000000
##   21         0.450     0.00000000
##   22         0.450     0.00000000
##   23         0.450     0.00000000
##   24         0.450     0.00000000
##   25         0.450     0.00000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was threshold = 0.
```

```
plot(nscTune)
```




```
testResults$nsc = predict(nscTune, fattyAcids)
```

Compute Confusion matrices for all of the models;

```
confusionMatrix(testResults$lda, testResults$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 34  0  0  0  0  0  0
##          B  2 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  1  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 0.9688
##                  95% CI : (0.9114, 0.9935)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9585
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            0.9189   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   0.9714  1.00000  1.00000   0.9882   1.0000
## Pos Pred Value         1.0000   0.9286  1.00000  1.00000   0.9167   1.0000
## Neg Pred Value         0.9516   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3542   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3542   0.2917  0.03125  0.07292   0.1250   0.1042
## Balanced Accuracy      0.9595   0.9857  1.00000  1.00000   0.9941   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

```
confusionMatrix(testResults$plsda, testResults$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 35  0  0  1  0  0  0
##          B  2 26  0  0  0  0  2
##          C  0  0  0  0  0  0  0
##          D  0  0  3  5  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  1  0 10  0
##          G  0  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.9062
##                  95% CI : (0.8295, 0.9562)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8733
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            0.9459   1.0000  0.00000  0.71429   1.0000   1.0000
```

```
## Specificity            0.9831   0.9429  1.00000  0.96629   1.0000   0.9884
## Pos Pred Value         0.9722   0.8667      NaN  0.62500   1.0000   0.9091
## Neg Pred Value         0.9667   1.0000  0.96875  0.97727   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3646   0.2708  0.00000  0.05208   0.1146   0.1042
## Detection Prevalence   0.3750   0.3125  0.00000  0.08333   0.1146   0.1146
## Balanced Accuracy      0.9645   0.9714  0.50000  0.84029   1.0000   0.9942
##                      Class: G
## Sensitivity           0.00000
## Specificity           1.00000
## Pos Pred Value            NaN
## Neg Pred Value        0.97917
## Prevalence            0.02083
## Detection Rate        0.00000
## Detection Prevalence  0.00000
## Balanced Accuracy     0.50000
```

```
confusionMatrix(testResults$glmn, testResults$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 37  0  3  0  2  0  0
##          B  0 26  0  0  0  0  2
##          C  0  0  0  0  0  0  0
##          D  0  0  0  6  0  0  0
##          E  0  0  0  0  9  0  0
##          F  0  0  0  1  0 10  0
##          G  0  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.9167
##                  95% CI : (0.8424, 0.9633)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8851
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            1.0000   1.0000  0.00000  0.85714  0.81818   1.0000
## Specificity            0.9153   0.9714  1.00000  1.00000  1.00000   0.9884
## Pos Pred Value         0.8810   0.9286      NaN  1.00000  1.00000   0.9091
## Neg Pred Value         1.0000   1.0000  0.96875  0.98889  0.97701   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292  0.11458   0.1042
## Detection Rate         0.3854   0.2708  0.00000  0.06250  0.09375   0.1042
## Detection Prevalence   0.4375   0.2917  0.00000  0.06250  0.09375   0.1146
## Balanced Accuracy      0.9576   0.9857  0.50000  0.92857  0.90909   0.9942
##                      Class: G
```

```
## Sensitivity           0.00000
## Specificity           1.00000
## Pos Pred Value            NaN
## Neg Pred Value        0.97917
## Prevalence            0.02083
## Detection Rate        0.00000
## Detection Prevalence  0.00000
## Balanced Accuracy     0.50000
```

```
confusionMatrix(testResults$nsc, testResults$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 35  0  0  0  0  0  0
##          B  2 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 0.9792
##                  95% CI : (0.9268, 0.9975)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9722
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            0.9459   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   0.9714  1.00000  1.00000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9286  1.00000  1.00000   1.0000   1.0000
## Neg Pred Value         0.9672   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3646   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3646   0.2917  0.03125  0.07292   0.1146   0.1042
## Balanced Accuracy      0.9730   0.9857  1.00000  1.00000   1.0000   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

So, the best performing model here was the Nearest Shrunken Centroid. In terms of its accuracy in predicting each particular level of the response variable, we can take another look at the confusion matrix for that model;

```
confusionMatrix(testResults$nsc, testResults$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 35  0  0  0  0  0  0
##          B  2 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 0.9792
##                  95% CI : (0.9268, 0.9975)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9722
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            0.9459   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   0.9714  1.00000  1.00000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9286  1.00000  1.00000   1.0000   1.0000
## Neg Pred Value         0.9672   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3646   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3646   0.2917  0.03125  0.07292   0.1146   0.1042
## Balanced Accuracy      0.9730   0.9857  1.00000  1.00000   1.0000   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

The nearest shrunken centroid model had a 5-way for most accuract predictor; C, D, E, F, and G all have 100% prediction rate. The worst accuracy rate was for Class A, which had a 97.3% accuracy rate.

13.2. Use the fatty acid data from the previous exercise set (Exercise 12.2).

(a) Use the same data splitting approach (if any) and pre-processing steps that you did in the previous chapter. Using the same classification statistic as before, build models described in this chapter for these data. Which model has the best predictive ability? How does this optimal model's performance compare to the best linear model's performance? Would you infer that the data have nonlinear separation boundaries based on this comparison?

So again, I am going to use centering and scaling in the train function to preprocess the data, and not split the data into training/testing sets due to the nature of the response variable and the small sample size.

The models discussed in Chapter 13 are the quadratic discrimininant analysis (QDA), the regularized discriminant analysis (RDA), the mixture discriminant analysis (MDA), naive Bayes, k-nearest neighbors, neural networks, flexible discriminant analysis, and support vector machines. I had trouble getting the QDA to run and eventually gave up on it but was able to fit all the other models discussed using the FattyAcids data.

```
set.seed(123)
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:38)
FDATune = train(fattyAcids, oilType,
                method = "fda",
                preProcess = c("center", "scale"),
                metric = "Accuracy",
                trControl = ctrl,
                tuneGrid = marsGrid)
```

```
## Loading required package: earth
```

```
## Warning: package 'earth' was built under R version 4.1.3
```

```
## Loading required package: Formula
```

```
## Loading required package: plotmo
```

```
## Warning: package 'plotmo' was built under R version 4.1.3
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```
## Warning: package 'TeachingDemos' was built under R version 4.1.3
```

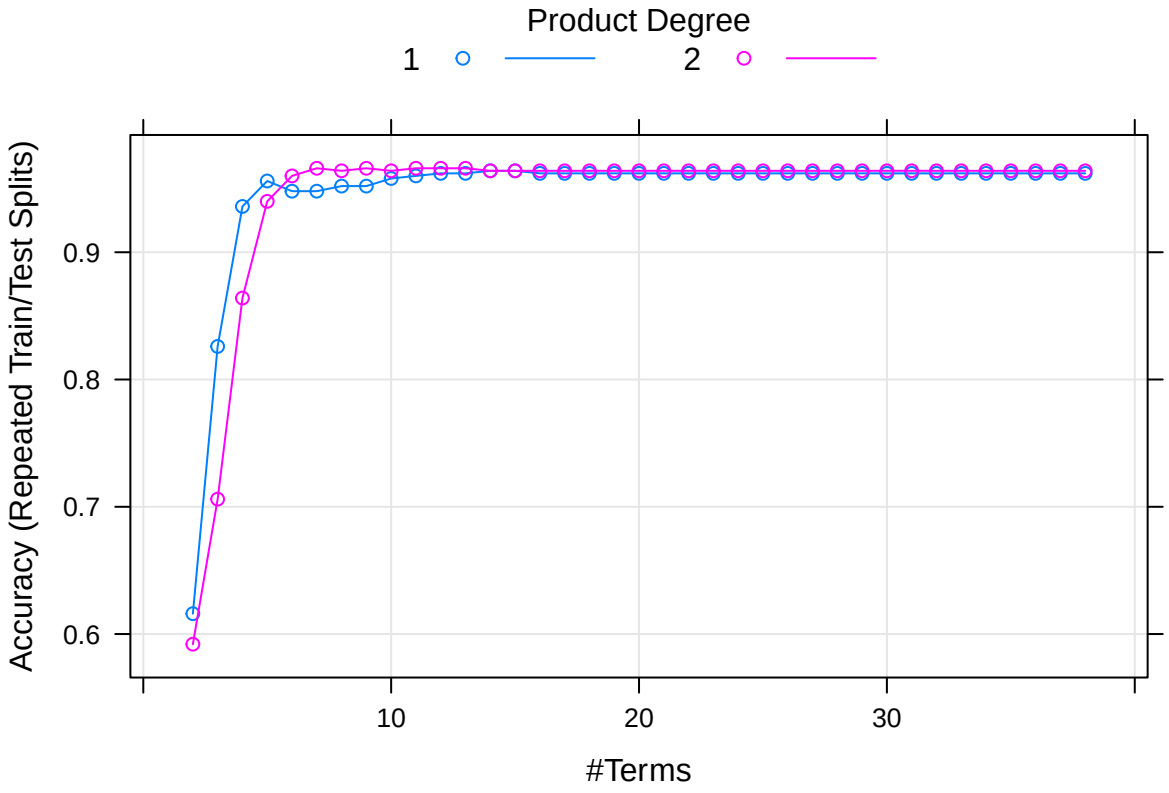
```
FDATune
```

```
## Flexible Discriminant Analysis
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results across tuning parameters:
##
```

```
##   degree  nprune  Accuracy  Kappa
##   1        2      0.616     0.3819750
##   1        3      0.826     0.7363712
##   1        4      0.936     0.9059240
##   1        5      0.956     0.9366040
##   1        6      0.948     0.9256167
##   1        7      0.948     0.9256081
##   1        8      0.952     0.9311278
##   1        9      0.952     0.9310618
##   1       10      0.958     0.9398106
##   1       11      0.960     0.9426366
##   1       12      0.962     0.9454829
##   1       13      0.962     0.9454829
##   1       14      0.964     0.9483920
##   1       15      0.964     0.9483920
##   1       16      0.962     0.9454742
##   1       17      0.962     0.9454742
##   1       18      0.962     0.9454742
##   1       19      0.962     0.9454742
##   1       20      0.962     0.9454742
##   1       21      0.962     0.9454742
##   1       22      0.962     0.9454742
##   1       23      0.962     0.9454742
##   1       24      0.962     0.9454742
##   1       25      0.962     0.9454742
##   1       26      0.962     0.9454742
##   1       27      0.962     0.9454742
##   1       28      0.962     0.9454742
##   1       29      0.962     0.9454742
##   1       30      0.962     0.9454742
##   1       31      0.962     0.9454742
##   1       32      0.962     0.9454742
##   1       33      0.962     0.9454742
##   1       34      0.962     0.9454742
##   1       35      0.962     0.9454742
##   1       36      0.962     0.9454742
##   1       37      0.962     0.9454742
##   1       38      0.962     0.9454742
##   2        2      0.592     0.3569905
##   2        3      0.706     0.5463414
##   2        4      0.864     0.7987660
##   2        5      0.940     0.9119916
##   2        6      0.960     0.9416300
##   2        7      0.966     0.9503348
##   2        8      0.964     0.9474257
##   2        9      0.966     0.9505841
##   2       10      0.964     0.9477051
##   2       11      0.966     0.9504809
##   2       12      0.966     0.9503867
##   2       13      0.966     0.9503867
##   2       14      0.964     0.9476912
##   2       15      0.964     0.9476912
##   2       16      0.964     0.9476912
##   2       17      0.964     0.9476912
```

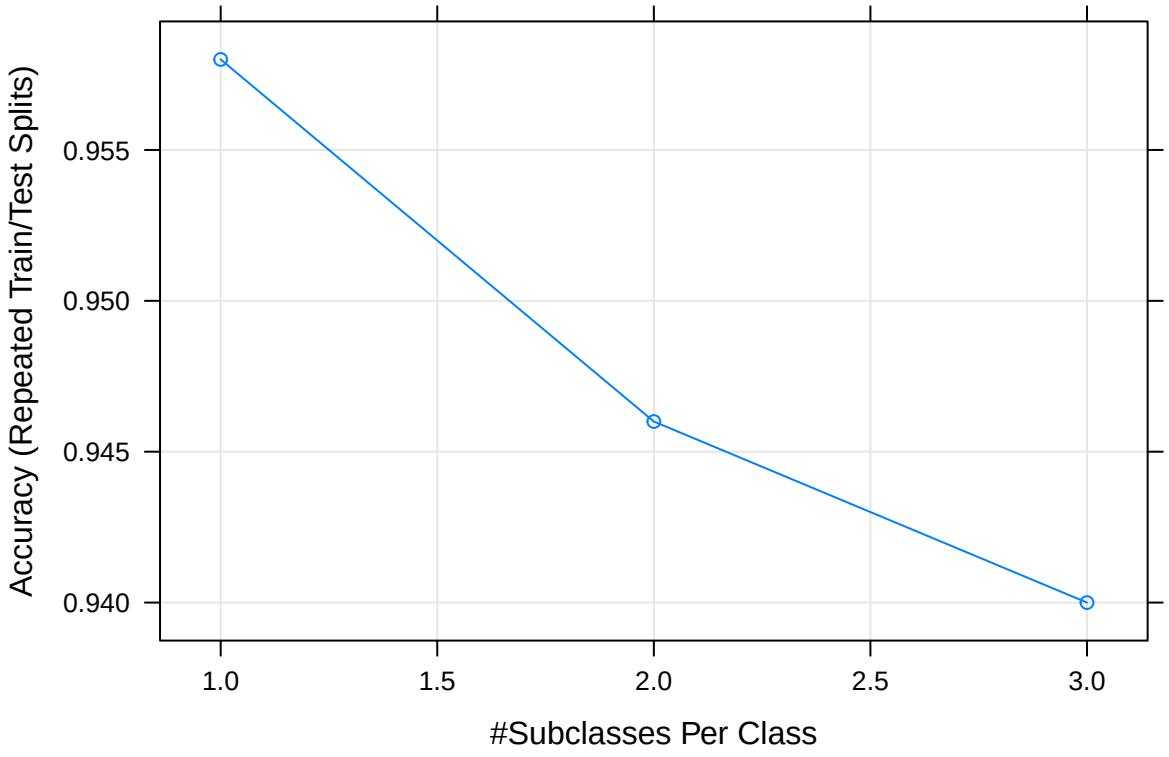
```
##    2       18      0.964     0.9476912
##    2       19      0.964     0.9476912
##    2       20      0.964     0.9476912
##    2       21      0.964     0.9476912
##    2       22      0.964     0.9476912
##    2       23      0.964     0.9476912
##    2       24      0.964     0.9476912
##    2       25      0.964     0.9476912
##    2       26      0.964     0.9476912
##    2       27      0.964     0.9476912
##    2       28      0.964     0.9476912
##    2       29      0.964     0.9476912
##    2       30      0.964     0.9476912
##    2       31      0.964     0.9476912
##    2       32      0.964     0.9476912
##    2       33      0.964     0.9476912
##    2       34      0.964     0.9476912
##    2       35      0.964     0.9476912
##    2       36      0.964     0.9476912
##    2       37      0.964     0.9476912
##    2       38      0.964     0.9476912
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 2 and nprune = 7.
```

```
plot(FDATune)
```



MDA model;

```
set.seed(123)
MDATune = train(fattyAcids, oilType,
                method = "mda",
                preProcess = c("center", "scale"),
                tuneGrid = expand.grid(.subclasses = 1:3),
                metric = "Accuracy",
                trControl = ctrl)
plot(MDATune)
```



Naive Bayes;

```
set.seed(123)
grid_nb = expand.grid(.usekernel = TRUE,
                      .fL = 2,
                      .adjust = TRUE)
NBTune = train(fattyAcids, oilType,
                method = "nb",
                preProcess = c("center", "scale"),
                metric = "Accuracy",
                trControl = ctrl,
                tuneGrid = grid_nb)
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
```

```
## observation 10

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 10

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 18

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 13

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 18

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 18

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 18

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 16

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 16

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 5
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 17

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 17

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 19

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 19

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 16

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 16

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 8
```

```
NBTune
```

```
## Naive Bayes
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.954     0.9356861
##
## Tuning parameter 'fL' was held constant at a value of 2
## Tuning
##  parameter 'usekernel' was held constant at a value of TRUE
## Tuning
##  parameter 'adjust' was held constant at a value of TRUE
```

Fit K-nearest neighbors model;

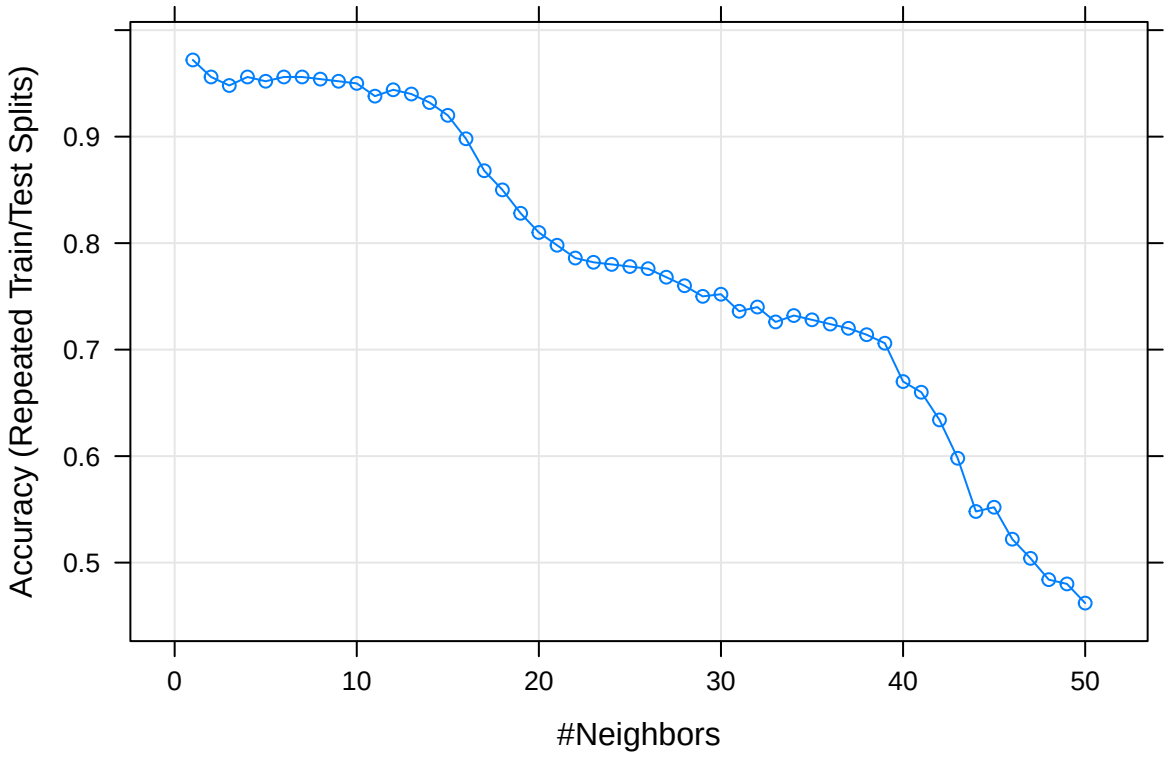
```
set.seed(123)
KNNTune = train(fattyAcids, oilType,
                method = "knn",
                preProcess = c("center", "scale"),
```

```
                metric = "Accuracy",
                trControl = ctrl,
                tuneGrid = data.frame(.k = 1:50))
KNNTune
```

```
## k-Nearest Neighbors
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy  Kappa
##    1  0.972     0.95963542
##    2  0.956     0.93687697
##    3  0.948     0.92610745
##    4  0.956     0.93741809
##    5  0.952     0.93178237
##    6  0.956     0.93749952
##    7  0.956     0.93760078
##    8  0.954     0.93407500
##    9  0.952     0.93073527
##   10  0.950     0.92841474
##   11  0.938     0.91048200
##   12  0.944     0.91919154
##   13  0.940     0.91361684
##   14  0.932     0.90150792
##   15  0.920     0.88184561
##   16  0.898     0.84729933
##   17  0.868     0.79960631
##   18  0.850     0.77139837
##   19  0.828     0.73469278
##   20  0.810     0.70494297
##   21  0.798     0.68499830
##   22  0.786     0.66468595
##   23  0.782     0.65792129
##   24  0.780     0.65486786
##   25  0.778     0.65177905
##   26  0.776     0.64864180
##   27  0.768     0.63566689
##   28  0.760     0.62344418
##   29  0.750     0.60685230
##   30  0.752     0.60868621
##   31  0.736     0.58321555
##   32  0.740     0.58960300
##   33  0.726     0.56707990
##   34  0.732     0.57658351
##   35  0.728     0.56955155
##   36  0.724     0.56136114
##   37  0.720     0.55464962
```

```
##    38  0.714     0.54198598
##    39  0.706     0.52222730
##    40  0.670     0.45725910
##    41  0.660     0.43255518
##    42  0.634     0.38554217
##    43  0.598     0.31544642
##    44  0.548     0.22165798
##    45  0.552     0.22609932
##    46  0.522     0.16861485
##    47  0.504     0.13342321
##    48  0.484     0.09364995
##    49  0.480     0.08537915
##    50  0.462     0.05066991
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
plot(KNNTune)
```



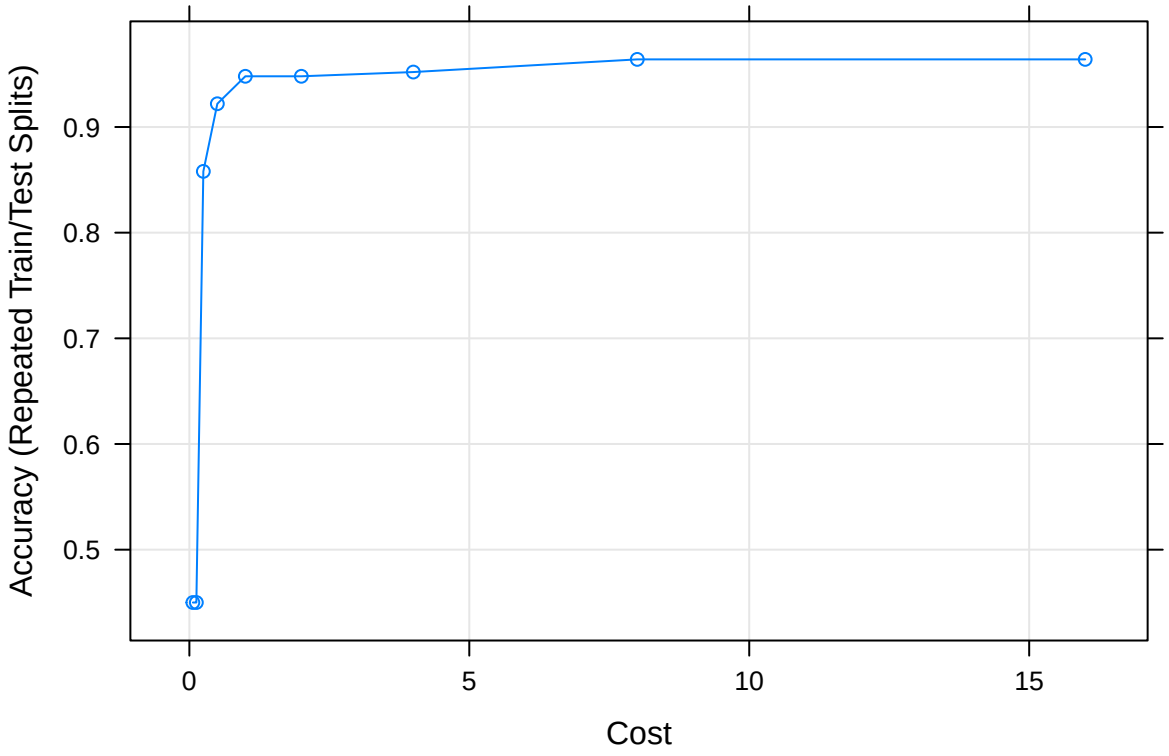
Fit SVM model;

```
set.seed(123)
sigmaRangeReduced = sigest(as.matrix(fattyAcids))
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1],.C = 2^(seq(-4, 4)))
ctrl2 = trainControl(method = "LGOCV",
                     summaryFunction = defaultSummary)
```

```
SVMTune = train(fattyAcids, oilType,
                method = "svmRadial",
                preProcess = c("center", "scale"),
                metric = "Accuracy",
                trControl = ctrl2,
                tuneGrid = svmRGridReduced)
SVMTune
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results across tuning parameters:
##
##   C        Accuracy  Kappa
##    0.0625  0.450     0.0000000
##    0.1250  0.450     0.0000000
##    0.2500  0.858     0.7782424
##    0.5000  0.922     0.8843903
##    1.0000  0.948     0.9259851
##    2.0000  0.948     0.9259851
##    4.0000  0.952     0.9314047
##    8.0000  0.964     0.9488038
##   16.0000  0.964     0.9485997
##
## Tuning parameter 'sigma' was held constant at a value of 0.03096983
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.03096983 and C = 8.
```

```
plot(SVMTune)
```



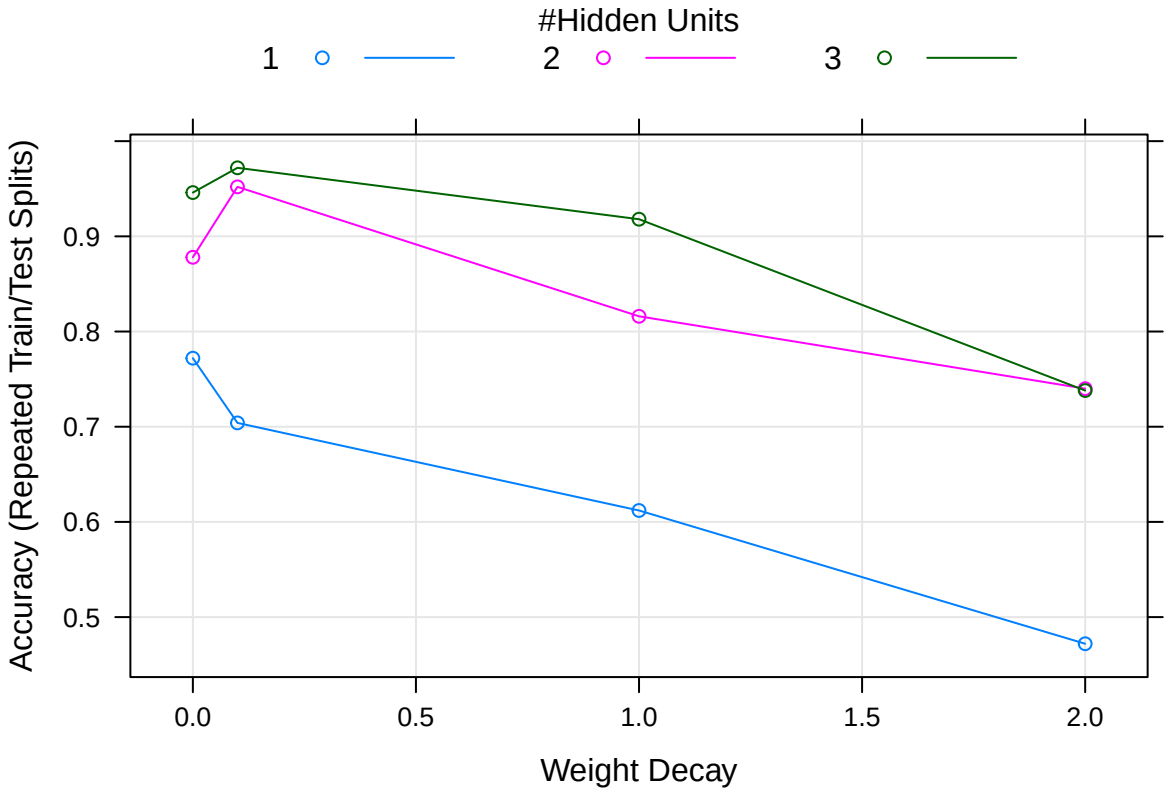
Fit Neural Network;

```
set.seed(123)
nnetGrid = expand.grid(.size = 1:3,
                       .decay = c(0, .1, 1, 2))
nnetTune = train(fattyAcids, oilType,
                 method = "nnet",
                 metric = "Accuracy",
                 preProc = c("center", "scale"),
                 tuneGrid = nnetGrid,
                 trControl = ctrl,
                 trace = FALSE)
nnetTune
```

```
## Neural Network
##
## 96 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## Pre-processing: centered (7), scaled (7)
## Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
## Summary of sample sizes: 76, 76, 76, 76, 76, 76, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy  Kappa
```

```
##    1     0.0    0.772     0.65309120
##    1     0.1    0.704     0.53507867
##    1     1.0    0.612     0.35908000
##    1     2.0    0.472     0.04417336
##    2     0.0    0.878     0.82227212
##    2     0.1    0.952     0.93000634
##    2     1.0    0.816     0.71207230
##    2     2.0    0.740     0.57630110
##    3     0.0    0.946     0.92367039
##    3     0.1    0.972     0.95948755
##    3     1.0    0.918     0.87735972
##    3     2.0    0.738     0.57484437
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.1.
```

```
plot(nnetTune)
```



(b) Which oil type does the optimal model most accurately predict? Least accurately predict?

Make predictions for each of the different models;

```
testResults2 = data.frame(obs = oilType,
                          mda = predict(MDATune, fattyAcids))
```

```
testResults2$fda = predict(FDATune, fattyAcids)
testResults2$nb = predict(NBTune, fattyAcids)
testResults2$knn = predict(KNNTune, fattyAcids)
testResults2$svm = predict(SVMTune, fattyAcids)
testResults2$nnet = predict(nnetTune, fattyAcids)
```

Confusion matrices;

```
confusionMatrix(testResults2$mda, testResults2$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 34  0  0  0  0  0  0
##          B  2 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  1  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 0.9688
##                  95% CI : (0.9114, 0.9935)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9585
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            0.9189   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   0.9714  1.00000  1.00000   0.9882   1.0000
## Pos Pred Value         1.0000   0.9286  1.00000  1.00000   0.9167   1.0000
## Neg Pred Value         0.9516   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3542   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3542   0.2917  0.03125  0.07292   0.1250   0.1042
## Balanced Accuracy      0.9595   0.9857  1.00000  1.00000   0.9941   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

```
confusionMatrix(testResults2$fda, testResults2$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 37  0  0  0  0  0  0
##          B  0 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9623, 1)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Balanced Accuracy      1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

```
confusionMatrix(testResults2$nb, testResults2$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 37  0  0  0  0  0  0
```

```
##          B  0 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9623, 1)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Balanced Accuracy      1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

```
confusionMatrix(testResults2$knn, testResults2$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 37  0  0  0  0  0  0
##          B  0 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
```

```
##
##                Accuracy : 1
##                  95% CI : (0.9623, 1)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Balanced Accuracy      1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

```
confusionMatrix(testResults2$svm, testResults2$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 36  0  0  0  0  0  0
##          B  1 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 0.9896
##                  95% CI : (0.9433, 0.9997)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9861
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            0.9730   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   0.9857  1.00000  1.00000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9630  1.00000  1.00000   1.0000   1.0000
## Neg Pred Value         0.9833   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3750   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3750   0.2812  0.03125  0.07292   0.1146   0.1042
## Balanced Accuracy      0.9865   0.9929  1.00000  1.00000   1.0000   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

```
confusionMatrix(testResults2$nnet, testResults2$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 37  0  0  0  0  0  0
##          B  0 26  0  0  0  0  1
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  1
##          G  0  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.9792
##                  95% CI : (0.9268, 0.9975)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.972
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   0.9857  1.00000  1.00000   1.0000   0.9884
## Pos Pred Value         1.0000   0.9630  1.00000  1.00000   1.0000   0.9091
```

```
## Neg Pred Value         1.0000   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Prevalence   0.3854   0.2812  0.03125  0.07292   0.1146   0.1146
## Balanced Accuracy      1.0000   0.9929  1.00000  1.00000   1.0000   0.9942
##                      Class: G
## Sensitivity           0.00000
## Specificity           1.00000
## Pos Pred Value            NaN
## Neg Pred Value        0.97917
## Prevalence            0.02083
## Detection Rate        0.00000
## Detection Prevalence  0.00000
## Balanced Accuracy     0.50000
```

There's a couple of models here that wound up with 100% accuracy, but I don't think that that means it's a better model, that seems like there's some overfitting going on, so I think the best model here is the SVM, which had the highest non-100% accuracy rate.

```
confusionMatrix(testResults2$svm, testResults2$obs)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E  F  G
##          A 36  0  0  0  0  0  0
##          B  1 26  0  0  0  0  0
##          C  0  0  3  0  0  0  0
##          D  0  0  0  7  0  0  0
##          E  0  0  0  0 11  0  0
##          F  0  0  0  0  0 10  0
##          G  0  0  0  0  0  0  2
##
## Overall Statistics
##
##                Accuracy : 0.9896
##                  95% CI : (0.9433, 0.9997)
##     No Information Rate : 0.3854
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9861
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity            0.9730   1.0000  1.00000  1.00000   1.0000   1.0000
## Specificity            1.0000   0.9857  1.00000  1.00000   1.0000   1.0000
## Pos Pred Value         1.0000   0.9630  1.00000  1.00000   1.0000   1.0000
## Neg Pred Value         0.9833   1.0000  1.00000  1.00000   1.0000   1.0000
## Prevalence             0.3854   0.2708  0.03125  0.07292   0.1146   0.1042
## Detection Rate         0.3750   0.2708  0.03125  0.07292   0.1146   0.1042
```

```
## Detection Prevalence   0.3750   0.2812  0.03125  0.07292   0.1146   0.1042
## Balanced Accuracy      0.9865   0.9929  1.00000  1.00000   1.0000   1.0000
##                      Class: G
## Sensitivity           1.00000
## Specificity           1.00000
## Pos Pred Value        1.00000
## Neg Pred Value        1.00000
## Prevalence            0.02083
## Detection Rate        0.02083
## Detection Prevalence  0.02083
## Balanced Accuracy     1.00000
```

Several of the oil types here have 100% prediction accuracy, and the lowest oil type prediction is for Class A.

In general, all of the non-linear models performed better than the linear models, so yes, it is reasonable to assume that there are some non-linear separation boundaries.