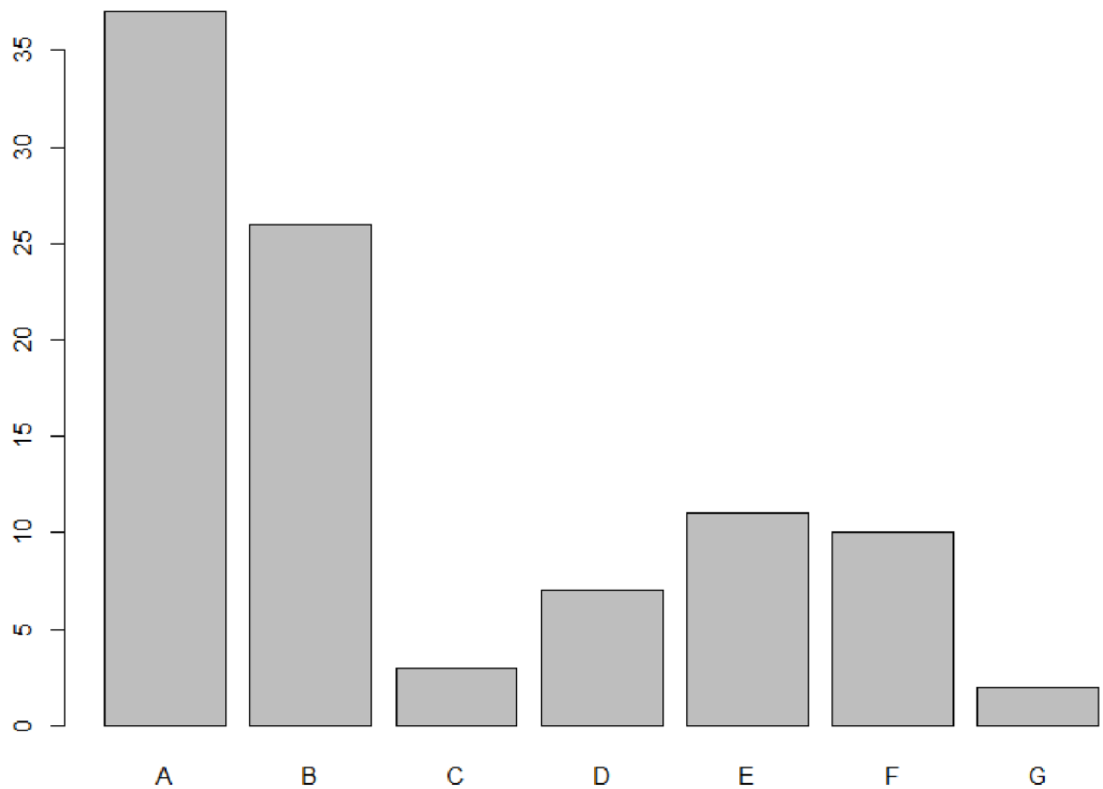
HOMEWORK6

Supriya Bachal
MICHIGAN TECHNOLOGICAL UNIVERSITY



(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.

a)oilType

A	B	C	D	E	F	G
0.38541667	0.27083333	0.03125000	0.07291667	0.11458333	0.10416667	0.02083333

The data is not evenly distributed among all the classes. Classes A and B have the highest no of occurrences whereas C and G have the least no of occurrence.It is a multinomial problem since there are 5 categories of Oil so the data set can be summarized as follows No of predictors(p): 7 No of classes :5

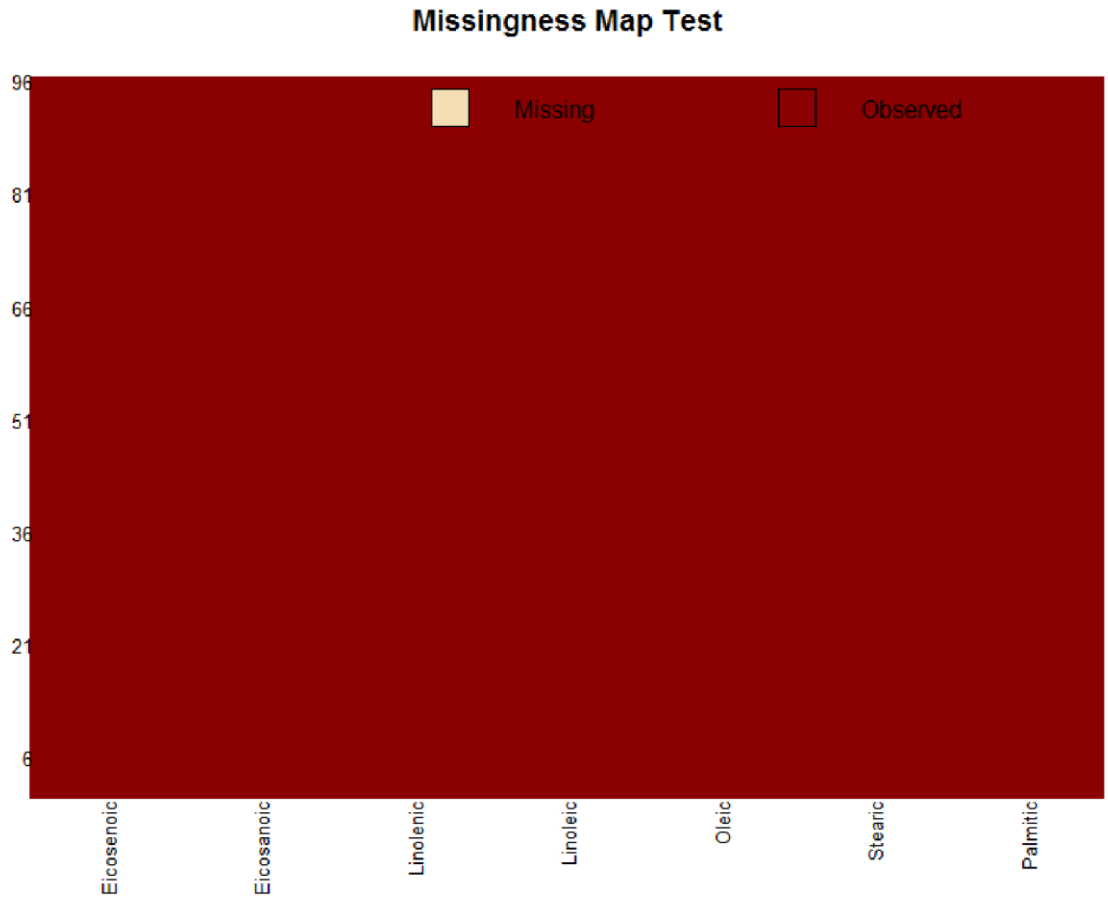
No of instances(n): 96

Preprocessing of Data:

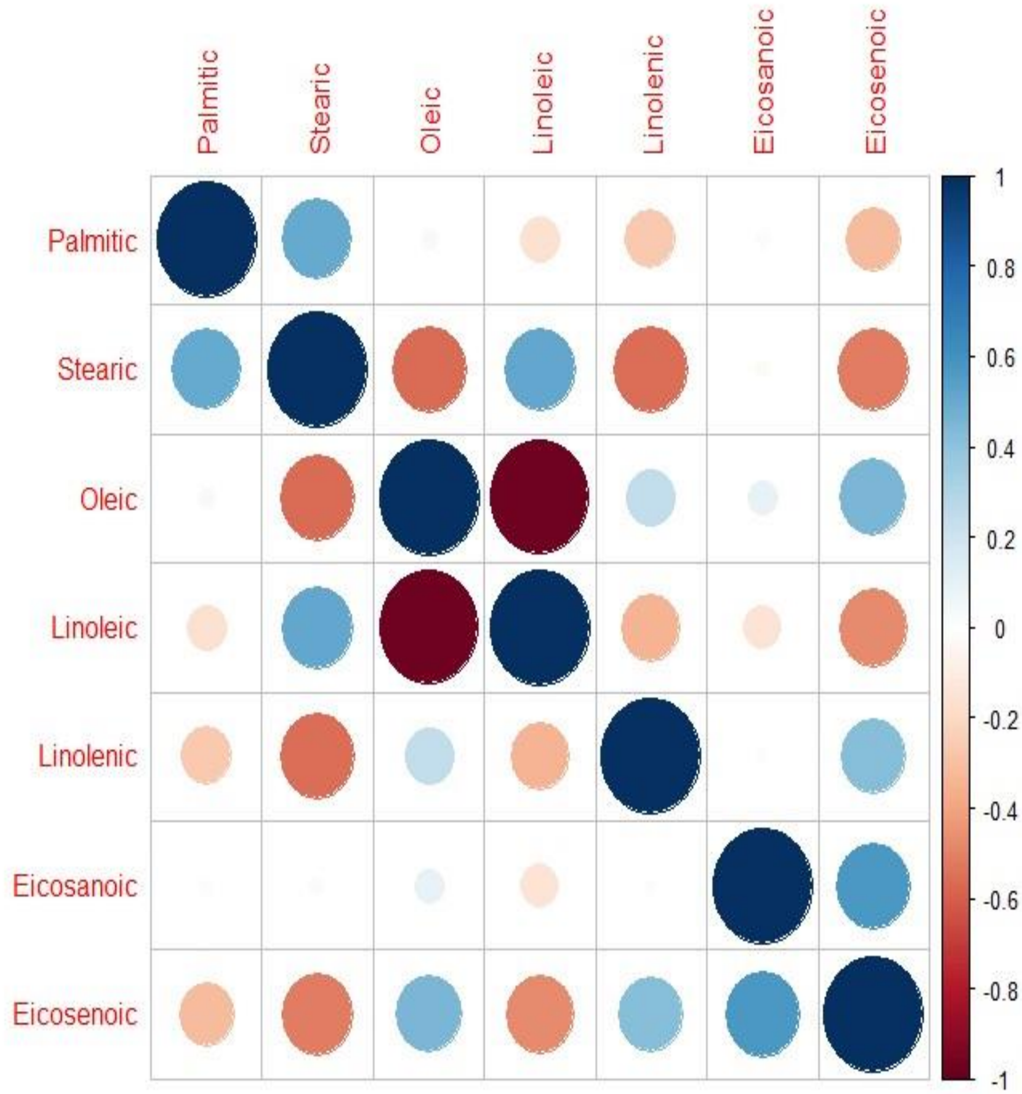
Near Zero Variance mitigation:

"Dropping 0 zero variance columns from 7 (fraction= 0.000000)"

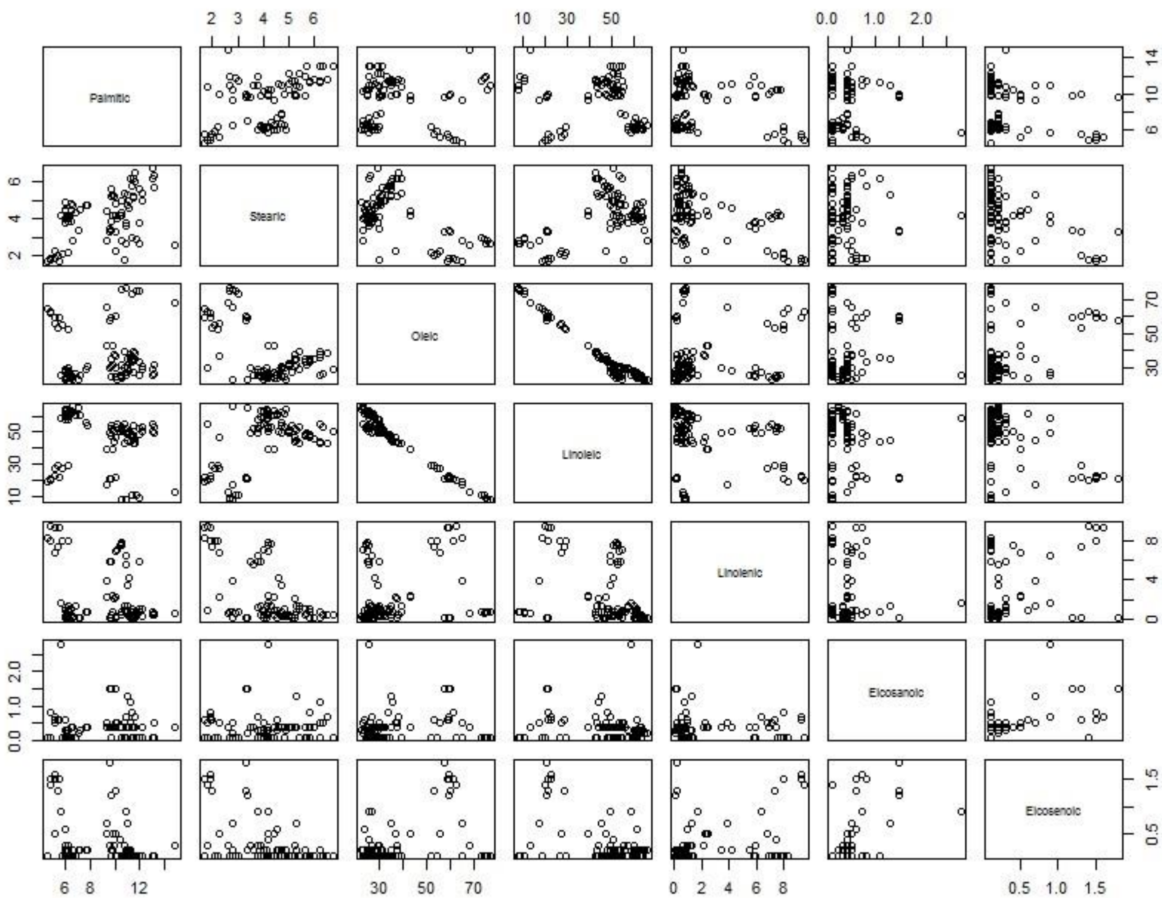
Missing values:



Linear Correlations:



As we can see from the plot linoleic and oleic show a very high negative correlation. And hence the 4th one is eliminated.



We may want to center and scale the data for better model performances. As required.

Data Splitting and Resampling:

Dividing the Data into Training and testing is not the best approach for this data set as the data set is very small. Unless we upsample the data to get more instances of minority class. However we have decided **not to split the data**.

ctrl <- trainControl(method = 'LGOCV',classProbs = TRUE,savePredictions = TRUE, summaryFunction = multiClassSummary)

MODEL SELECTION:

Kappa is used as metric since there are more than two classes hence roc cannot be used, and kappa works best with imbalanced classes since it takes it account the distribution of classes.

Mixture Discriminant Analysis

Mixture Discriminant Analysis

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

No pre-processing

Resampling: Repeated Train/Test Splits Estimated (4 reps, 75%)

Summary of sample sizes: 76, 76, 76, 76

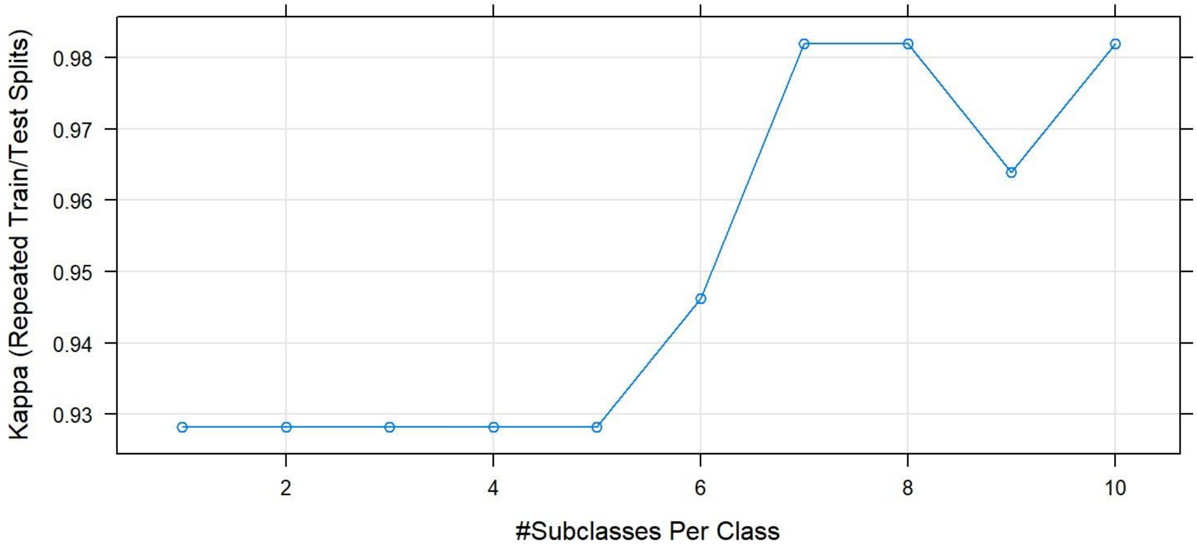
Resampling results across tuning parameters:

subclasses	Accuracy	Kappa
1	0.9500	0.9281846
2	0.9500	0.9281846
3	0.9500	0.9281846
4	0.9500	0.9281846
5	0.9500	0.9281846
6	0.9625	0.9462352
7	0.9875	0.9819495
8	0.9875	0.9819495
9	0.9750	0.9638989
10	0.9875	0.9819495

Kappa was used to select the optimal model using the largest value.

The final value used for the model was subclasses = 7.

Plot:



Regularized Discriminant Analysis

Regularized Discriminant Analysis

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

Pre-processing: centered (6), scaled (6)

Resampling: Repeated Train/Test Splits Estimated (4 reps, 75%)

Summary of sample sizes: 76, 76, 76, 76

Resampling results across tuning parameters:

gamma	lambda	Accuracy	Kappa
0.1	0.0000000	0.9500	0.9281846
0.1	0.1111111	0.9500	0.9281846
0.1	0.2222222	0.9500	0.9281846
0.1	0.3333333	0.9500	0.9281846
0.1	0.4444444	0.9500	0.9281846

0.1	0.5555556	0.9500	0.9281846
0.1	0.6666667	0.9500	0.9281846
0.1	0.7777778	0.9500	0.9281846
0.1	0.8888889	0.9500	0.9281846
0.1	1.0000000	0.9500	0.9281846
0.2	0.0000000	0.9375	0.9108954
0.2	0.1111111	0.9500	0.9281846
0.2	0.2222222	0.9500	0.9281846
0.2	0.3333333	0.9500	0.9281846
0.2	0.4444444	0.9500	0.9281846
0.2	0.5555556	0.9500	0.9281846
0.2	0.6666667	0.9500	0.9281846
0.2	0.7777778	0.9500	0.9281846
0.2	0.8888889	0.9500	0.9281846
0.2	1.0000000	0.9500	0.9281846
0.3	0.0000000	0.9375	0.9108954
0.3	0.1111111	0.9500	0.9281846
0.3	0.2222222	0.9500	0.9281846
0.3	0.3333333	0.9500	0.9281846
0.3	0.4444444	0.9500	0.9281846
0.3	0.5555556	0.9500	0.9281846
0.3	0.6666667	0.9500	0.9281846
0.3	0.7777778	0.9500	0.9281846
0.3	0.8888889	0.9500	0.9281846
0.3	1.0000000	0.9500	0.9281846
0.4	0.0000000	0.9375	0.9108954
0.4	0.1111111	0.9375	0.9108954
0.4	0.2222222	0.9500	0.9281846
0.4	0.3333333	0.9500	0.9281846
0.4	0.4444444	0.9500	0.9281846

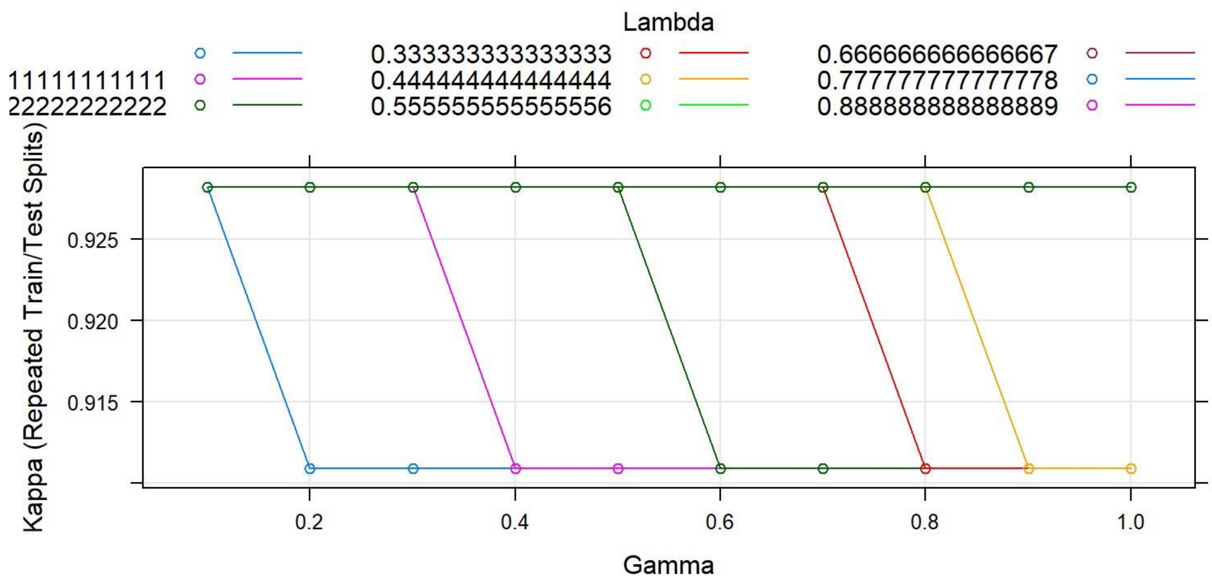
0.4	0.5555556	0.9500	0.9281846
0.4	0.6666667	0.9500	0.9281846
0.4	0.7777778	0.9500	0.9281846
0.4	0.8888889	0.9500	0.9281846
0.4	1.0000000	0.9500	0.9281846
0.5	0.0000000	0.9375	0.9108954
0.5	0.1111111	0.9375	0.9108954
0.5	0.2222222	0.9500	0.9281846
0.5	0.3333333	0.9500	0.9281846
0.5	0.4444444	0.9500	0.9281846
0.5	0.5555556	0.9500	0.9281846
0.5	0.6666667	0.9500	0.9281846
0.5	0.7777778	0.9500	0.9281846
0.5	0.8888889	0.9500	0.9281846
0.5	1.0000000	0.9500	0.9281846
0.6	0.0000000	0.9375	0.9108954
0.6	0.1111111	0.9375	0.9108954
0.6	0.2222222	0.9375	0.9108954
0.6	0.3333333	0.9500	0.9281846
0.6	0.4444444	0.9500	0.9281846
0.6	0.5555556	0.9500	0.9281846
0.6	0.6666667	0.9500	0.9281846
0.6	0.7777778	0.9500	0.9281846
0.6	0.8888889	0.9500	0.9281846
0.6	1.0000000	0.9500	0.9281846
0.7	0.0000000	0.9375	0.9108954
0.7	0.1111111	0.9375	0.9108954
0.7	0.2222222	0.9375	0.9108954
0.7	0.3333333	0.9500	0.9281846
0.7	0.4444444	0.9500	0.9281846

0.7	0.5555556	0.9500	0.9281846
0.7	0.6666667	0.9500	0.9281846
0.7	0.7777778	0.9500	0.9281846
0.7	0.8888889	0.9500	0.9281846
0.7	1.0000000	0.9500	0.9281846
0.8	0.0000000	0.9375	0.9108954
0.8	0.1111111	0.9375	0.9108954
0.8	0.2222222	0.9375	0.9108954
0.8	0.3333333	0.9375	0.9108954
0.8	0.4444444	0.9500	0.9281846
0.8	0.5555556	0.9500	0.9281846
0.8	0.6666667	0.9500	0.9281846
0.8	0.7777778	0.9500	0.9281846
0.8	0.8888889	0.9500	0.9281846
0.8	1.0000000	0.9500	0.9281846
0.9	0.0000000	0.9375	0.9108954
0.9	0.1111111	0.9375	0.9108954
0.9	0.2222222	0.9375	0.9108954
0.9	0.3333333	0.9375	0.9108954
0.9	0.4444444	0.9375	0.9108954
0.9	0.5555556	0.9500	0.9281846
0.9	0.6666667	0.9500	0.9281846
0.9	0.7777778	0.9500	0.9281846
0.9	0.8888889	0.9500	0.9281846
0.9	1.0000000	0.9500	0.9281846
1.0	0.0000000	0.9375	0.9108954
1.0	0.1111111	0.9375	0.9108954
1.0	0.2222222	0.9375	0.9108954
1.0	0.3333333	0.9375	0.9108954
1.0	0.4444444	0.9375	0.9108954

1.0	0.5555556	0.9500	0.9281846
1.0	0.6666667	0.9500	0.9281846
1.0	0.7777778	0.9500	0.9281846
1.0	0.8888889	0.9500	0.9281846
1.0	1.0000000	0.9500	0.9281846

Kappa was used to select the optimal model using the largest value.

The final values used for the model were gamma = 0.1 and lambda = 1.



Flexible Discriminant Analysis

Flexible Discriminant Analysis

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

Pre-processing: centered (6), scaled (6)

Resampling: Repeated Train/Test Splits Estimated (4 reps, 75%)

Summary of sample sizes: 76, 76, 76, 76

Resampling results across tuning parameters:

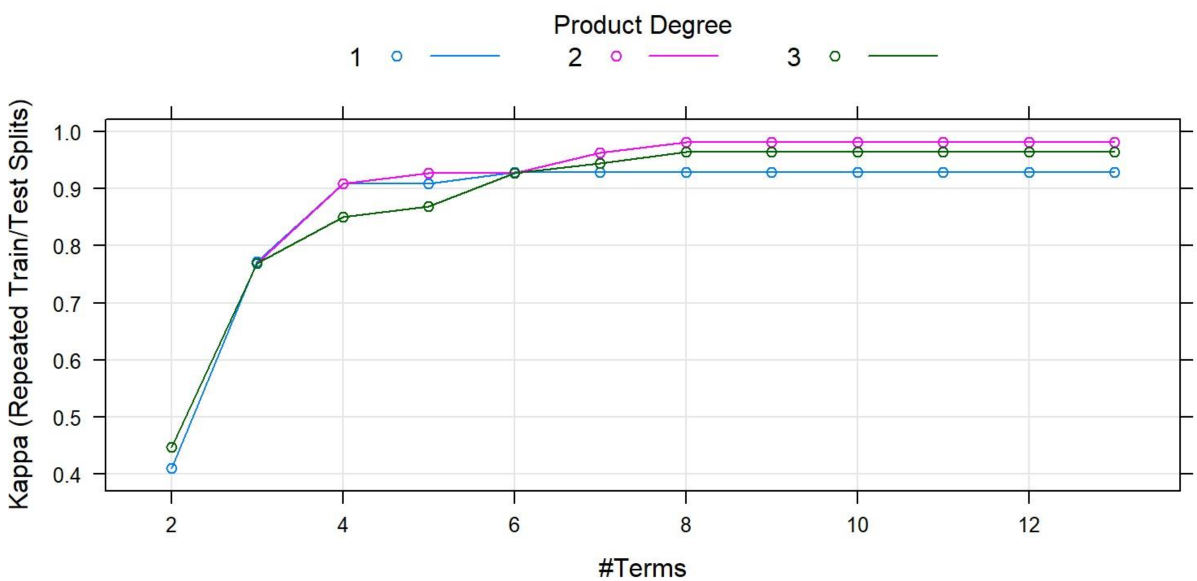
degree	nprune	Accuracy	Kappa
1	2	0.6250	0.4100043
1	3	0.8500	0.7722821
1	4	0.9375	0.9091981
1	5	0.9375	0.9091981
1	6	0.9500	0.9281846
1	7	0.9500	0.9281846
1	8	0.9500	0.9281846
1	9	0.9500	0.9281846
1	10	0.9500	0.9281846
1	11	0.9500	0.9281846
1	12	0.9500	0.9281846
1	13	0.9500	0.9281846
2	2	0.6625	0.4466514
2	3	0.8500	0.7697925
2	4	0.9375	0.9084517
2	5	0.9500	0.9272487
2	6	0.9500	0.9272487
2	7	0.9750	0.9632229
2	8	0.9875	0.9819495
2	9	0.9875	0.9819495
2	10	0.9875	0.9819495
2	11	0.9875	0.9819495
2	12	0.9875	0.9819495
2	13	0.9875	0.9819495
3	2	0.6625	0.4466514
3	3	0.8500	0.7697925

3	4	0.9000	0.8499930
3	5	0.9125	0.8687900
3	6	0.9500	0.9272487
3	7	0.9625	0.9449124
3	8	0.9750	0.9638989
3	9	0.9750	0.9638989
3	10	0.9750	0.9638989
3	11	0.9750	0.9638989
3	12	0.9750	0.9638989
3	13	0.9750	0.9638989

Kappa was used to select the optimal model using the largest value.

The final values used for the model were degree = 2 and nprune = 8.

PLOT:



SVM Radial

Support Vector Machines with Radial Basis Function Kernel

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

Pre-processing: centered (6), scaled (6)

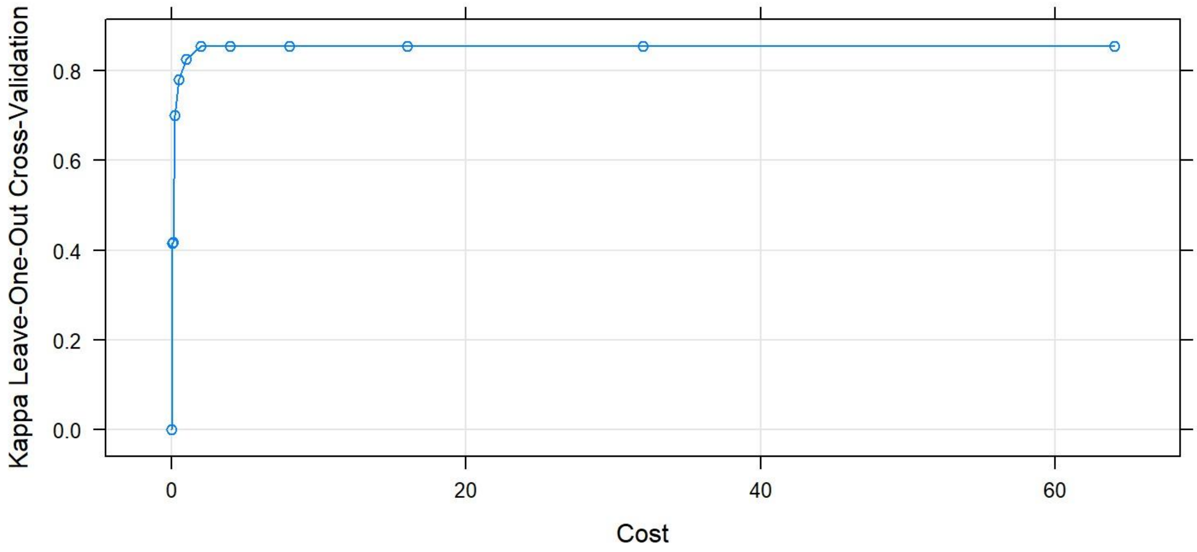
Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 95, 95, 95, 95, 95, 95, ...

Resampling results across tuning parameters:

C	Accuracy	Kappa
0.015625	0.3854167	0.0000000
0.031250	0.3854167	0.0000000
0.062500	0.6250000	0.4159202
0.125000	0.6250000	0.4180839
0.250000	0.7916667	0.7007015
0.500000	0.8437500	0.7801191
1.000000	0.8750000	0.8266105
2.000000	0.8958333	0.8550287
4.000000	0.8958333	0.8550287
8.000000	0.8958333	0.8550287
16.000000	0.8958333	0.8550287
32.000000	0.8958333	0.8550287
64.000000	0.8958333	0.8550287

Tuning parameter 'sigma' was held constant at a value of 0.04707239 Kappa
was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.04707239 and C = 2.



SVM Polynomial

Support Vector Machines with Polynomial Kernel

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

Pre-processing: centered (6), scaled (6)

Resampling: Leave-One-Out Cross-Validation

Summary of sample sizes: 95, 95, 95, 95, 95, 95, ...

Resampling results across tuning parameters:

degree scale C Accuracy Kappa 2

0.01 0.015625 0.3854167 0.0000000

2 0.01 0.031250 0.3854167 0.0000000

2 0.01 0.062500 0.3854167 0.0000000

2	0.01	0.125000	0.3854167	0.0000000
2	0.01	0.250000	0.3854167	0.0000000
2	0.01	0.500000	0.7187500	0.5818006
2	0.01	1.000000	0.7916667	0.7031081
2	0.01	2.000000	0.9375000	0.9163884
2	0.01	4.000000	0.9375000	0.9167028
2	0.01	8.000000	0.9270833	0.9027637
2	0.01	16.000000	0.9479167	0.9307259
2	0.01	32.000000	0.9583333	0.9443720
2	0.01	64.000000	0.9583333	0.9446526
2	0.10	0.015625	0.4375000	0.1060528
2	0.10	0.031250	0.4687500	0.1721339
2	0.10	0.062500	0.7812500	0.6864209
2	0.10	0.125000	0.9479167	0.9301717
2	0.10	0.250000	0.9270833	0.9020265
2	0.10	0.500000	0.9270833	0.9027637
2	0.10	1.000000	0.9270833	0.9027637
2	0.10	2.000000	0.9583333	0.9443720
2	0.10	4.000000	0.9583333	0.9443720
2	0.10	8.000000	0.9583333	0.9443720
2	0.10	16.000000	0.9687500	0.9582124
2	0.10	32.000000	0.9791667	0.9720971
2	0.10	64.000000	0.9791667	0.9720971
2	0.50	0.015625	0.8958333	0.8573339
2	0.50	0.031250	0.9375000	0.9157032
2	0.50	0.062500	0.9375000	0.9157032
2	0.50	0.125000	0.9479167	0.9296394
2	0.50	0.250000	0.9687500	0.9582124
2	0.50	0.500000	0.9687500	0.9582124
2	0.50	1.000000	0.9687500	0.9582124

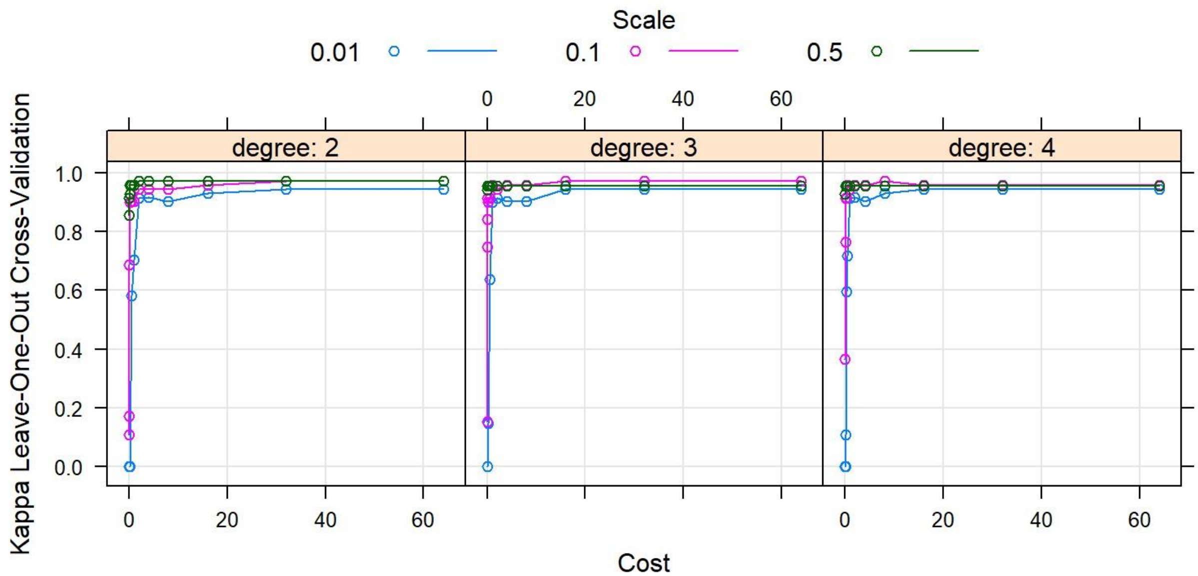
2	0.50	2.000000	0.9791667	0.9720971
2	0.50	4.000000	0.9791667	0.9720971
2	0.50	8.000000	0.9791667	0.9720971
2	0.50	16.000000	0.9791667	0.9720971
2	0.50	32.000000	0.9791667	0.9720971
2	0.50	64.000000	0.9791667	0.9720971
3	0.01	0.015625	0.3854167	0.0000000
3	0.01	0.031250	0.3854167	0.0000000
3	0.01	0.062500	0.3854167	0.0000000
3	0.01	0.125000	0.3854167	0.0000000
3	0.01	0.250000	0.4583333	0.1471040
3	0.01	0.500000	0.7500000	0.6369939
3	0.01	1.000000	0.9270833	0.9012636
3	0.01	2.000000	0.9375000	0.9163884
3	0.01	4.000000	0.9270833	0.9027637
3	0.01	8.000000	0.9270833	0.9027637
3	0.01	16.000000	0.9583333	0.9443720
3	0.01	32.000000	0.9583333	0.9446526
3	0.01	64.000000	0.9583333	0.9446526
3	0.10	0.015625	0.4583333	0.1520299
3	0.10	0.031250	0.8229167	0.7488458
3	0.10	0.062500	0.8854167	0.8424586
3	0.10	0.125000	0.9375000	0.9157032
3	0.10	0.250000	0.9270833	0.9020265
3	0.10	0.500000	0.9375000	0.9157032
3	0.10	1.000000	0.9687500	0.9582124
3	0.10	2.000000	0.9583333	0.9443720
3	0.10	4.000000	0.9687500	0.9582124
3	0.10	8.000000	0.9687500	0.9582124
3	0.10	16.000000	0.9791667	0.9720971

3	0.10	32.000000	0.9791667	0.9720971
3	0.10	64.000000	0.9791667	0.9720971
3	0.50	0.015625	0.9583333	0.9436206
3	0.50	0.031250	0.9583333	0.9436206
3	0.50	0.062500	0.9583333	0.9439988
3	0.50	0.125000	0.9687500	0.9579316
3	0.50	0.250000	0.9687500	0.9579316
3	0.50	0.500000	0.9687500	0.9579316
3	0.50	1.000000	0.9687500	0.9579316
3	0.50	2.000000	0.9687500	0.9579316
3	0.50	4.000000	0.9687500	0.9579316
3	0.50	8.000000	0.9687500	0.9579316
3	0.50	16.000000	0.9687500	0.9579316
3	0.50	32.000000	0.9687500	0.9579316
3	0.50	64.000000	0.9687500	0.9579316
4	0.01	0.015625	0.3854167	0.0000000
4	0.01	0.031250	0.3854167	0.0000000
4	0.01	0.062500	0.3854167	0.0000000
4	0.01	0.125000	0.4375000	0.1060528
4	0.01	0.250000	0.7291667	0.5962472
4	0.01	0.500000	0.8020833	0.7190821
4	0.01	1.000000	0.9375000	0.9163884
4	0.01	2.000000	0.9375000	0.9167028
4	0.01	4.000000	0.9270833	0.9027637
4	0.01	8.000000	0.9479167	0.9307259
4	0.01	16.000000	0.9583333	0.9443720
4	0.01	32.000000	0.9583333	0.9446526
4	0.01	64.000000	0.9583333	0.9446526
4	0.10	0.015625	0.5729167	0.3658772
4	0.10	0.031250	0.8333333	0.7663168

4	0.10	0.062500	0.9375000	0.9157032
4	0.10	0.125000	0.9375000	0.9157032
4	0.10	0.250000	0.9375000	0.9157032
4	0.10	0.500000	0.9687500	0.9582124
4	0.10	1.000000	0.9583333	0.9443720
4	0.10	2.000000	0.9687500	0.9582124
4	0.10	4.000000	0.9687500	0.9582124
4	0.10	8.000000	0.9791667	0.9720971
4	0.10	16.000000	0.9687500	0.9580786
4	0.10	32.000000	0.9687500	0.9580786
4	0.10	64.000000	0.9687500	0.9580786
4	0.50	0.015625	0.9479167	0.9292557
4	0.50	0.031250	0.9687500	0.9579316
4	0.50	0.062500	0.9687500	0.9579316
4	0.50	0.125000	0.9687500	0.9579316
4	0.50	0.250000	0.9687500	0.9579316
4	0.50	0.500000	0.9687500	0.9579316
4	0.50	1.000000	0.9687500	0.9579316
4	0.50	2.000000	0.9687500	0.9579316
4	0.50	4.000000	0.9687500	0.9579316
4	0.50	8.000000	0.9687500	0.9579316
4	0.50	16.000000	0.9687500	0.9579316
4	0.50	32.000000	0.9687500	0.9579316
4	0.50	64.000000	0.9687500	0.9579316

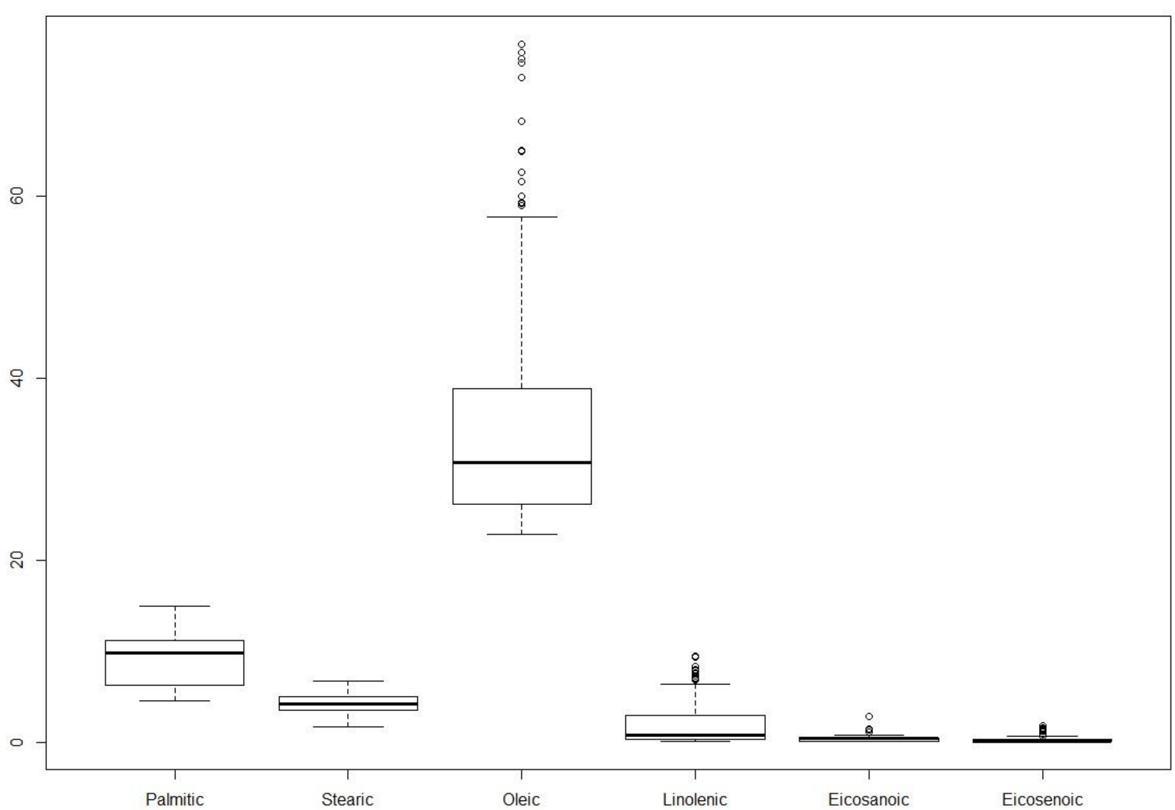
Kappa was used to select the optimal model using the largest value.

The final values used for the model were degree = 2, scale = 0.5 and C = 2.



Neural Network

Neural Network



Since the boxplot shows outliers we will preprocess using spatial sign transformation to get rid of the outliers, as NNet is sensitive to outliers.

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

Pre-processing: centered (6), scaled (6), spatial sign transformation (6)

Resampling: Repeated Train/Test Splits Estimated (4 reps, 75%)

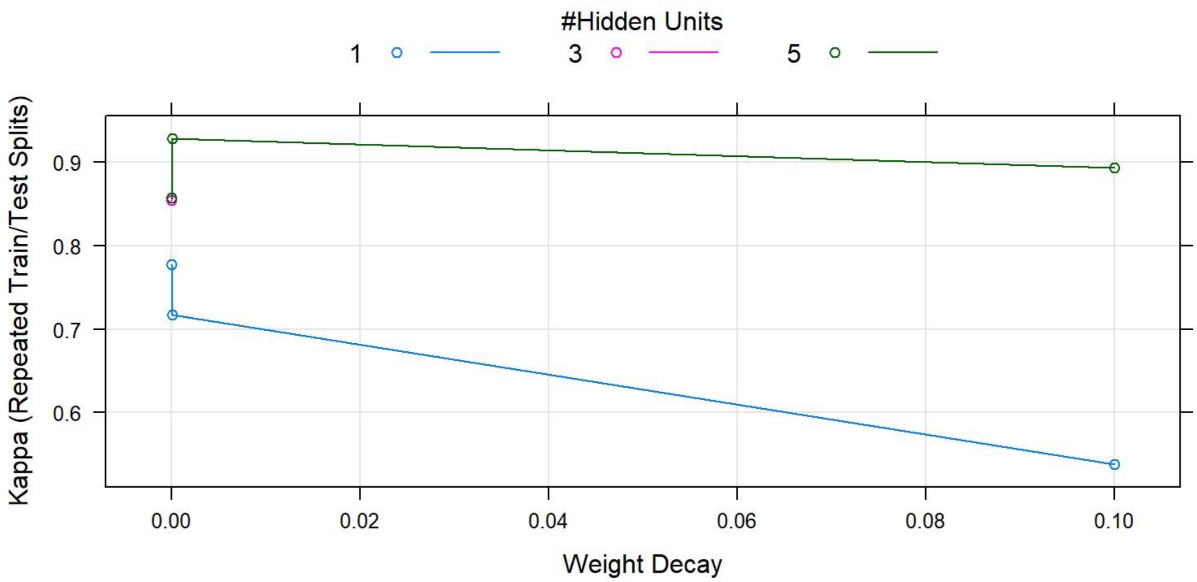
Summary of sample sizes: 76, 76, 76, 76

Resampling results across tuning parameters:

size	decay	Accuracy	Kappa
1	0e+00	0.850	0.7774847
1	1e-04	0.800	0.7169346
1	1e-01	0.700	0.5380597
3	0e+00	0.900	0.8545466
3	1e-04	0.950	0.9288247
3	1e-01	0.925	0.8934170
5	0e+00	0.900	0.8575515
5	1e-04	0.950	0.9283070
5	1e-01	0.925	0.8934170

Kappa was used to select the optimal model using the largest value.

The final values used for the model were size = 3 and decay = 1e-04.



KNN #### k-

Nearest Neighbors

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

Pre-processing: centered (6), scaled (6)

Resampling: Repeated Train/Test Splits Estimated (4 reps, 75%)

Summary of sample sizes: 76, 76, 76, 76

Resampling results across tuning parameters:

k	Accuracy	Kappa
1	0.9750	0.96428571
3	0.9500	0.92818463
5	0.9500	0.92818463
7	0.9500	0.92818463
9	0.9500	0.92818463
11	0.9375	0.91013409

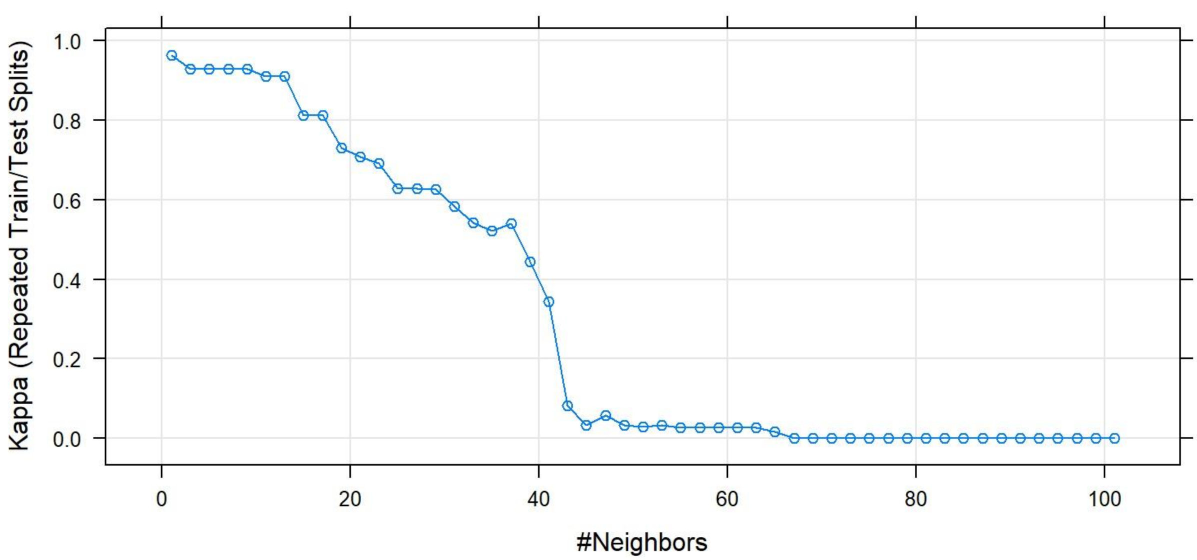
13	0.9375	0.91013409
15	0.8750	0.81304196
17	0.8750	0.81304196
19	0.8250	0.73090295
21	0.8125	0.70919961
23	0.8000	0.69171428
25	0.7625	0.62765726
27	0.7625	0.62789069
29	0.7625	0.62609489
31	0.7375	0.58341179
33	0.7125	0.54203949
35	0.7000	0.52235445
37	0.7125	0.54038692
39	0.6625	0.44388795
41	0.6125	0.34388262
43	0.4750	0.08260464
45	0.4500	0.03292499

47	0.4625	0.05758242
49	0.4500	0.03292499
51	0.4500	0.02973683
53	0.4500	0.03292499
55	0.4500	0.02654867
57	0.4500	0.02654867
59	0.4500	0.02654867
61	0.4500	0.02654867
63	0.4500	0.02654867
65	0.4500	0.01672685
67	0.4500	0.00000000
69	0.4500	0.00000000
71	0.4500	0.00000000

73	0.4500	0.00000000
75	0.4500	0.00000000
77	0.4500	0.00000000
79	0.4500	0.00000000
81	0.4500	0.00000000
83	0.4500	0.00000000
85	0.4500	0.00000000
87	0.4500	0.00000000
89	0.4500	0.00000000
91	0.4500	0.00000000
93	0.4500	0.00000000
95	0.4500	0.00000000
97	0.4500	0.00000000
99	0.4500	0.00000000
101	0.4500	0.00000000

Kappa was used to select the optimal model using the largest value.

The final value used for the model was k = 1.



Naive Bayes

Naive Bayes

96 samples

6 predictor

7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'

No pre-processing

Resampling: Repeated Train/Test Splits Estimated (4 reps, 75%)

Summary of sample sizes: 76, 76, 76, 76

Resampling results across tuning parameters:

usekernel	Accuracy	Kappa
FALSE	NaN	NaN
TRUE	0.975	0.9642857

Tuning parameter 'fL' was held constant at a value of 0

Tuning

parameter 'adjust' was held constant at a value of 1

Kappa was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE
and adjust = 1.

Name	Tuning Parameters	Pre- Processing	Kappa	Accuracy
Neural Networ k	size = 3 and decay = 1e-04.	Centered Scaled & Spatial sign transformation	0.8232	0.8778
MDA	subclasses = 7	None	0.9819495	0.9875
FDA	degree = 2 and nprune = 8.	Centered Scaled	0.9819495	0.9875

RDA	gamma = 0.1 and lambda = 1	Centered and Scaled	0.9288247	0.950
SVM Radial:	sigma = 0.03 034568 and C = 2.	Centered, Scaled	0.6545	0.762

k-Nearest Neighbors	k = 1	Centered Scaled	0.3366	0.6039
Naive Bayes	fL = 0, usekernel = TRUE adjust = 1	None	0.9639	0.975
SVM Polynomial	degree = 2, scale = 0.5 and C = 2 .	Centered Scaled	0.8084	0.8615

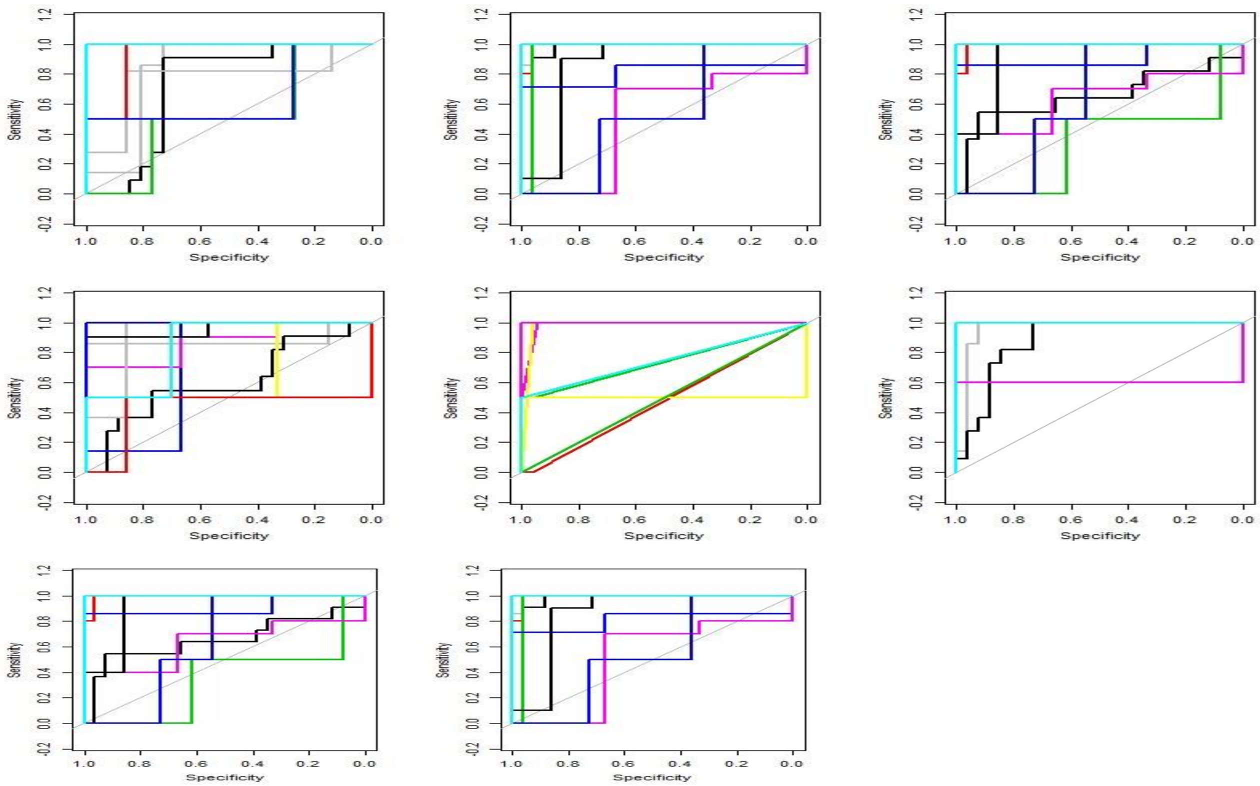
The Model Plots:

Since the Models were run for 6 classes we used multi.class.ROC to get the values for ROC PLOT using the following code: ***rda.predictions_all = predict(my_model, train,type='prob')***

rda.rocCurve.all = pROC::multiclass.roc(response=oilType, predictor=rda.predictions_all[,1])

rda.auc.all = rda.rocCurve.all$auc[1]

The plots derived are as follows (in order of above table from left to right)



FDA, MDA, RDA and naïve bayes are the top best models choosen through the model predictions over different groups using LGOCV.

These values are obtained by using the code:

confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

After shortlisting the model we will use the predict function to find the confusion matrices for all the models.

The predict function is used in the following way:

pred<-predict(my_model,train) confusionMatrix(pred,oilType)

MDA:

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
```

RDA:

```
Confusion Matrix and Statistics
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

FDA:

Confusion Matrix and Statistics

Reference

Prediction A B C D E F G

A 35 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 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 : 0.9792

95% CI : (0.9268, 0.9975)

No Information Rate : 0.3854

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9723

Naïve Bayes:

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

The above confusion matrices show that both MDA and Naïve Bayes have 1 Kappa,100 percent accuracy.

Which model has the best predictive ability?

The MDA model as well as the Naïve Bayes model has the best predictive ability among the linear models. With Kappa=1 and Accuracy=1.

How does this optimal model's performance compare to the best linear model's performance?

The optimal model of the nonlinear set performs similar to the best model of linear modelling set as achieve achieve 100 percent accuracy and build ideal models. In the linear set of models the nearest shrunken centroid model as well as the Logistic regression model performed to full Accuracy in the nonlinear set Naïve Bayes and MDA models performed equally well.

Would you infer that the data have nonlinear separation boundaries based on this comparison?

Since linear models had similar performance on this data as nonlinear models we cannot make a conclusive statement for data to have nonlinear separation boundaries. We investigate further;

LDA 0.9620253

GLMNET 0.9367089

NSC 1

PLSDA 0.9493671

Logistic regression 1

LDA 0.9500527

GLMNET 0.9143353

NSC 1

PLSDA 0.9327516

Logistic regression 1

Performances of linear models.

FDA:Accuracy : 0.9688	Kappa : 0.9585
RDA:Accuracy : 0.9792	Kappa : 0.9723
Naïve Bayes: 1	Kappa=1
MDA:1	Kappa=1

Performance of non-linear models.

By comparing the performances of the model that are the top 4 performing we see that the nonlinear models in general show better accuracy and Kappa values than the linear models. Hence we can conclude that the boundaries are nonlinear.

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

The optimal model predicts all oil types properly. Since it is an ideal model. In addition, none of the oil types wrongly.

#rcode: ```

{r data}

data(oil) data<-

fattyAcids

```

##my_model: ```{r

my_model_}

ctrl = trainControl(method="LGOCV",classProbs=TRUE,savePredictions=TRUE, number=4)

corr_bio <- cor(data) corrplot(corr_bio, tl.cex = 0.5)

remove <- findCorrelation(corr_bio,cutoff = 0.90) data<-data[,-remove]

train = data y_train

=oilType

```

```{r MDA}

cat('#### Mixture Discriminant Analysis ####') set.seed(1)

my_model <- train(train, y=y_train, method="mda", tuneGrid=expand.grid(subclasses=1:10),

metric="Kappa", trControl=ctrl)

my_model plot(my_model)

pred<-predict(my_model,train)

confusionMatrix(pred,oilType)

```

```{r RDA}

cat('#### Regularized Discriminant Analysis ####') set.seed(1)

grid <- expand.grid(.gamma = seq(0.1, 1, length = 10), .lambda = seq(0, 1, length = 10))

my_model <- train(train, y=y_train, method="rda",
```

```
            tuneGrid=grid, metric="Kappa", trControl=ctrl, preProc=c('center', 'scale'))
my_model



confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

plot(my_model) plot(my_model) pred<-predict(my_model,train)

confusionMatrix(pred,oilType)
```

```{r FDA}


cat('#### Flexible Discriminant Analysis ####')

set.seed(1)

marsGrid <- expand.grid(.degree = 1:3, .nprune = 2:13)

my_model <- train(train, y=y_train, method="fda",

            tuneGrid=marsGrid, metric="Kappa", trControl=ctrl,
preProc=c('center', 'scale'))

my_model

plot(my_model)

confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

pred<-predict(my_model,train)

confusionMatrix(pred,oilType)



```

```{r SVM_Radial}
```

````
cat('#### SVM Radial ####')

set.seed(1)

sigmaEst = kernlab::sigest(as.matrix(train))

svagrid = expand.grid(sigma=sigmaEst[1], C=2^seq(-6, +6))

my_model <- train(train, y=y_train, method="svmRadial",
metric="Kappa",trControl=trainControl(method = "LOOCV"),
preProc=c('center', 'scale'), fit=FALSE, tuneGrid = svagrid)

my_model

plot(my_model)

confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

pred<-predict(my_model,train)

confusionMatrix(pred,oilType)


```



```{r SVM_Polynomial}


cat('#### SVM Polynomial ####')

set.seed(1)

poly_grid = expand.grid(degree = c(2, 3, 4), C = 2^seq(-6, 6, length = 13),
scale = c(.5, .1, 0.01))

my_model <- train(train, y=y_train,
method="svmPoly",metric="Kappa",trControl=trainControl(method =
"LOOCV"),
            preProc=c('center', 'scale'), fit=FALSE, tuneGrid = poly_grid)

my_model
````

~~~~
plot(my_model)

confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

pred<-predict(my_model,train)

confusionMatrix(pred,oilType)


```


```{r NNET}


cat('#### Neural Network  ####')

set.seed(1)

nnetGrid = expand.grid(size=1:10, decay=c(0, 0.1, 1, 2))

my_model <- train(train, y=y_train, method="nnet",
preProc=c("center","scale","spatialSign"),

                 tunegrid=nnetGrid, metric="Kappa",trace=FALSE, maxit=3000,
trControl=ctrl)

my_model

plot(my_model)

confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

pred<-predict(my_model,train)

confusionMatrix(pred,oilType)


```


```{r KNN}
~~~~

````
cat('#### KNN ####')

set.seed(1)

knngrid = data.frame(.k = seq(1, 101, by = 2))

my_model <- train(train, y=y_train, method="knn", tuneGrid = knngrid,
preProc=c("center","scale"), metric="Kappa", trControl=ctrl)

my_model

plot(my_model)

confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

pred<-predict(my_model,train)

confusionMatrix(pred,oilType)


```


```{r NB}


cat('#### Naive Bayes ####')

set.seed(1)

my_model <- train(train, y=y_train, method="nb", metric="Kappa",
trControl=ctrl)

my_model

plot(my_model)

confusionMatrix(data=my_model$pred$pred, reference=my_model$pred$obs)

pred<-predict(my_model,train)

confusionMatrix(pred,oilType)
````

```
```