Deanna Springgay

Ch 12 Hw

12.1

a) To account for the classification imbalance in the response variable, I decided to split the data using stratified splitting using the response as the metric for splitting. This way, the training and testing sets both have roughly the same amount of each outcome: None, Mild, and Severe.

b) To optimize the models, I chose kappa optimization since the outcome has three levels. If the outcome had two levels instead, then I would have chosen ROC instead. The best model will have the highest kappa values.

c)

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Tuning Parameter** | **Training Kappa** | **Testing Kappa** |
| Logistic Regression | decay = 0.1 | 0.04114121 | 0.01282051 |
| Linear Discriminant Analysis | N/A | 0.02941645 | 0.0661803 |
| Partial Least Squares | ncomp = 6 | 0.062481748 | 0.0804077 |
| Penalized Models | alpha = 1 and lambda = 0.05222222 | 0.0694473883 | -0.02083333 |
| Nearest Shrunken Centroids | threshold = 0.8 | 1.738690e-02 | -0.03157895 |

**Logistic Regression**

Penalized Multinomial Regression

225 samples

184 predictors

3 classes: 'Mild', 'None', 'Severe'

Pre-processing: nearest neighbor imputation (103), centered (103), scaled (103), remove (81)

Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)

Summary of sample sizes: 169, 169, 169, 169, 169, 169, ...

Resampling results across tuning parameters:

decay Accuracy Kappa

0e+00 0.3942857 0.03258549

1e-04 0.4007143 0.03457717

1e-01 0.4321429 0.04114121

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

The final value used for the model was decay = 0.1.

Confusion Matrix and Statistics

Reference

Prediction Mild None Severe

Mild 957 658 187

None 842 641 143

Severe 376 276 120

Overall Statistics

Accuracy : 0.409

95% CI : (0.3941, 0.4241)

No Information Rate : 0.5179

P-Value [Acc > NIR] : 1

Kappa : 0.0359

Mcnemar's Test P-Value : <2e-16

Statistics by Class:

Class: Mild Class: None Class: Severe

Sensitivity 0.4400 0.4070 0.26667

Specificity 0.5827 0.6248 0.82613

Pos Pred Value 0.5311 0.3942 0.15544

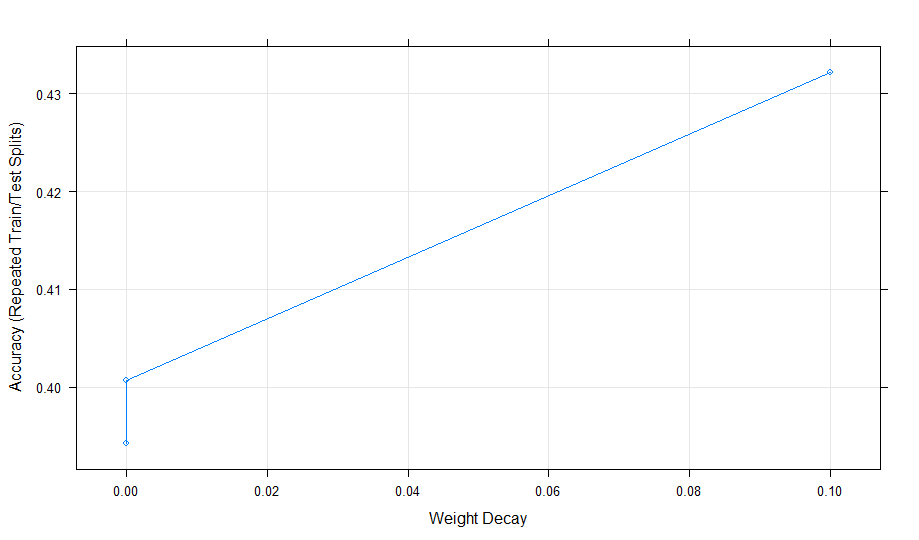
Neg Pred Value 0.4921 0.6371 0.90373

Prevalence 0.5179 0.3750 0.10714

Detection Rate 0.2279 0.1526 0.02857

Detection Prevalence 0.4290 0.3871 0.18381

Balanced Accuracy 0.5114 0.5159 0.54640



**Linear Discriminant Analysis**

Linear Discriminant Analysis

225 samples

184 predictors

3 classes: 'Mild', 'None', 'Severe'

Pre-processing: centered (103), scaled (103), remove (81)

Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)

Summary of sample sizes: 169, 169, 169, 169, 169, 169, ...

Resampling results:

Accuracy Kappa

0.4042857 0.02941645

Confusion Matrix and Statistics

Reference

Prediction Mild None Severe

Mild 334 245 60

None 255 183 41

Severe 136 97 49

Overall Statistics

Accuracy : 0.4043

95% CI : (0.3784, 0.4305)

No Information Rate : 0.5179

P-Value [Acc > NIR] : 1

Kappa : 0.0294

Mcnemar's Test P-Value : 2.468e-11

Statistics by Class:

Class: Mild Class: None Class: Severe

Sensitivity 0.4607 0.3486 0.3267

Specificity 0.5481 0.6617 0.8136

Pos Pred Value 0.5227 0.3820 0.1738

Neg Pred Value 0.4862 0.6287 0.9097

Prevalence 0.5179 0.3750 0.1071

Detection Rate 0.2386 0.1307 0.0350

Detection Prevalence 0.4564 0.3421 0.2014

Balanced Accuracy 0.5044 0.5051 0.5701

**Partial Least Squares**

Partial Least Squares

225 samples

184 predictors

3 classes: 'Mild', 'None', 'Severe'

Pre-processing: centered (103), scaled (103), remove (81)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 225, 225, 225, 225, 225, 225, ...

Resampling results across tuning parameters:

ncomp Accuracy Kappa

1 0.4680020 -0.005727374

2 0.4710967 0.009402382

3 0.4836082 0.040974221

4 0.4763682 0.037091151

5 0.4801500 0.052370659

6 0.4838229 0.062481748

7 0.4779034 0.053590590

8 0.4688525 0.041922948

9 0.4676697 0.044545229

10 0.4604698 0.036783891

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

The final value used for the model was ncomp = 6.

Confusion Matrix and Statistics

Reference

Prediction Mild None Severe

Mild 6867 5137 1298

None 3237 2598 900

Severe 146 165 42

Overall Statistics

Accuracy : 0.4663

95% CI : (0.4594, 0.4731)

No Information Rate : 0.5027

P-Value [Acc > NIR] : 1

Kappa : 0.0155

Mcnemar's Test P-Value : <2e-16

Statistics by Class:

Class: Mild Class: None Class: Severe

Sensitivity 0.6700 0.3289 0.01875

Specificity 0.3654 0.6688 0.98287

Pos Pred Value 0.5162 0.3857 0.11898

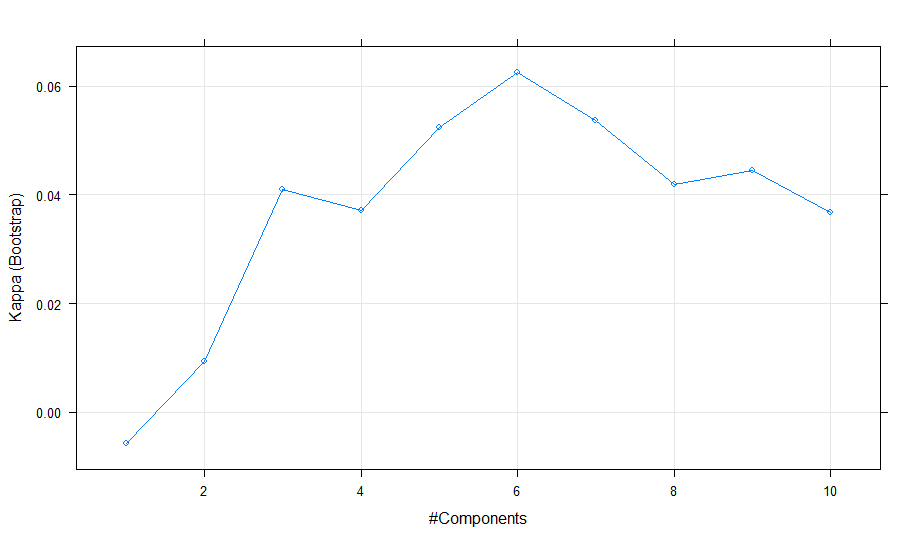
Neg Pred Value 0.5227 0.6117 0.89030

Prevalence 0.5027 0.3874 0.10986

Detection Rate 0.3368 0.1274 0.00206

Detection Prevalence 0.6524 0.3303 0.01731

Balanced Accuracy 0.5177 0.4988 0.50081



**Penalized Models**

glmnet

225 samples

184 predictors

3 classes: 'Mild', 'None', 'Severe'

Pre-processing: centered (103), scaled (103), remove (81)

Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)

Summary of sample sizes: 169, 169, 169, 169, 169, 169, ...

Resampling results across tuning parameters:

alpha lambda Accuracy Kappa

0.0 0.01000000 0.4550000 0.0372774740

0.0 0.03111111 0.4600000 0.0269327541

0.0 0.05222222 0.4728571 0.0384805581

0.0 0.07333333 0.4807143 0.0453979731

0.0 0.09444444 0.4785714 0.0357633178

0.0 0.11555556 0.4764286 0.0269356525

0.0 0.13666667 0.4757143 0.0204405121

0.0 0.15777778 0.4750000 0.0153134505

0.0 0.17888889 0.4757143 0.0127327294

0.0 0.20000000 0.4742857 0.0065672329

0.1 0.01000000 0.4592857 0.0436415622

0.1 0.03111111 0.4728571 0.0389887240

0.1 0.05222222 0.4807143 0.0375897853

0.1 0.07333333 0.4742857 0.0148856897

0.1 0.09444444 0.4785714 0.0136493863

0.1 0.11555556 0.4735714 -0.0033136888

0.1 0.13666667 0.4785714 -0.0010996126

0.1 0.15777778 0.4785714 -0.0068942486

0.1 0.17888889 0.4828571 -0.0032650911

0.1 0.20000000 0.4864286 0.0003810371

0.2 0.01000000 0.4635714 0.0448779521

0.2 0.03111111 0.4721429 0.0300414134

0.2 0.05222222 0.4792857 0.0240736724

0.2 0.07333333 0.4807143 0.0126360454

0.2 0.09444444 0.4828571 0.0054982905

0.2 0.11555556 0.4857143 0.0038681373

0.2 0.13666667 0.4957143 0.0162770525

0.2 0.15777778 0.5057143 0.0285521820

0.2 0.17888889 0.5092857 0.0304389092

0.2 0.20000000 0.5100000 0.0277160527

0.4 0.01000000 0.4678571 0.0515835566

0.4 0.03111111 0.4800000 0.0314360842

0.4 0.05222222 0.4878571 0.0229225226

0.4 0.07333333 0.4950000 0.0207458548

0.4 0.09444444 0.5121429 0.0419595203

0.4 0.11555556 0.5185714 0.0496591828

0.4 0.13666667 0.5200000 0.0389705229

0.4 0.15777778 0.5185714 0.0260268770

0.4 0.17888889 0.5235714 0.0301647973

0.4 0.20000000 0.5292857 0.0349430460

0.6 0.01000000 0.4728571 0.0564752673

0.6 0.03111111 0.4921429 0.0391936024

0.6 0.05222222 0.4942857 0.0200888134

0.6 0.07333333 0.5178571 0.0514859197

0.6 0.09444444 0.5235714 0.0497781152

0.6 0.11555556 0.5214286 0.0316071892

0.6 0.13666667 0.5307143 0.0398648657

0.6 0.15777778 0.5264286 0.0243914983

0.6 0.17888889 0.5221429 0.0106819257

0.6 0.20000000 0.5185714 0.0016842105

0.8 0.01000000 0.4700000 0.0461432548

0.8 0.03111111 0.4964286 0.0412768395

0.8 0.05222222 0.5171429 0.0528767505

0.8 0.07333333 0.5285714 0.0600318436

0.8 0.09444444 0.5250000 0.0360026895

0.8 0.11555556 0.5285714 0.0319252410

0.8 0.13666667 0.5221429 0.0106819257

0.8 0.15777778 0.5185714 0.0016842105

0.8 0.17888889 0.5178571 0.0000000000

0.8 0.20000000 0.5178571 0.0000000000

1.0 0.01000000 0.4742857 0.0484065446

1.0 0.03111111 0.4978571 0.0323704171

1.0 0.05222222 0.5278571 0.0694473883

1.0 0.07333333 0.5292857 0.0501477959

1.0 0.09444444 0.5271429 0.0291839899

1.0 0.11555556 0.5207143 0.0067192064

1.0 0.13666667 0.5185714 0.0016842105

1.0 0.15777778 0.5178571 0.0000000000

1.0 0.17888889 0.5178571 0.0000000000

1.0 0.20000000 0.5178571 0.0000000000

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

The final values used for the model were alpha = 1 and lambda = 0.05222222.

Confusion Matrix and Statistics

Reference

Prediction Mild None Severe

Mild 40942 28646 7541

None 9372 7586 2712

Severe 436 518 247

Overall Statistics

Accuracy : 0.4977

95% CI : (0.4946, 0.5008)

No Information Rate : 0.5179

P-Value [Acc > NIR] : 1

Kappa : 0.0263

Mcnemar's Test P-Value : <2e-16

Statistics by Class:

Class: Mild Class: None Class: Severe

Sensitivity 0.8067 0.20642 0.02352

Specificity 0.2341 0.80271 0.98910

Pos Pred Value 0.5308 0.38566 0.20566

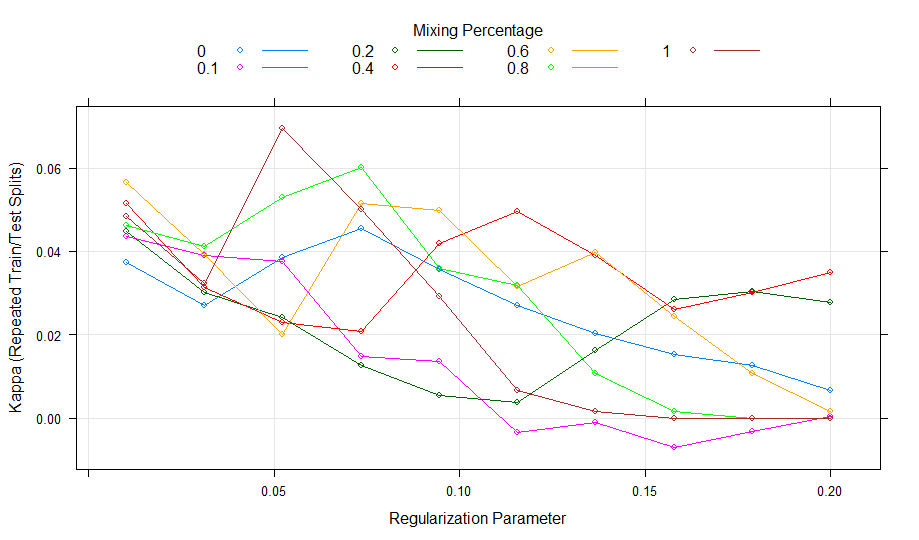
Neg Pred Value 0.5301 0.62768 0.89408

Prevalence 0.5179 0.37500 0.10714

Detection Rate 0.4178 0.07741 0.00252

Detection Prevalence 0.7870 0.20071 0.01226

Balanced Accuracy 0.5204 0.50457 0.50631



**Nearest Shrunken Centroids**

Nearest Shrunken Centroids

225 samples

184 predictors

3 classes: 'Mild', 'None', 'Severe'

Pre-processing: centered (103), scaled (103), remove (81)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 225, 225, 225, 225, 225, 225, ...

Resampling results across tuning parameters:

threshold Accuracy Kappa

0.0 0.4648082 7.728634e-05

0.1 0.4638998 -3.681036e-03

0.2 0.4659905 -3.561967e-03

0.3 0.4695771 -1.175407e-04

0.4 0.4769335 9.382166e-03

0.5 0.4815659 1.417196e-02

0.6 0.4783845 3.843685e-03

0.7 0.4844042 1.110648e-02

0.8 0.4905210 1.738690e-02

0.9 0.4898652 1.160332e-02

1.0 0.4873859 3.263438e-03

1.1 0.4878010 1.715461e-03

1.2 0.4853646 -5.017835e-03

1.3 0.4853838 -6.452964e-03

1.4 0.4872393 -4.024197e-03

1.5 0.4897824 3.387077e-04

1.6 0.4896806 -1.228219e-03

1.7 0.4907068 -1.091829e-04

1.8 0.4902573 -1.595228e-03

1.9 0.4907573 -1.061967e-03

2.0 0.4907068 -1.643836e-03

2.1 0.4912068 -8.275862e-04

2.2 0.4912068 -8.275862e-04

2.3 0.4917068 0.000000e+00

2.4 0.4917068 0.000000e+00

2.5 0.4917068 0.000000e+00

2.6 0.4917068 0.000000e+00

2.7 0.4917068 0.000000e+00

2.8 0.4917068 0.000000e+00

2.9 0.4917068 0.000000e+00

3.0 0.4917068 0.000000e+00

3.1 0.4917068 0.000000e+00

3.2 0.4917068 0.000000e+00

3.3 0.4917068 0.000000e+00

3.4 0.4917068 0.000000e+00

3.5 0.4917068 0.000000e+00

3.6 0.4917068 0.000000e+00

3.7 0.4917068 0.000000e+00

3.8 0.4917068 0.000000e+00

3.9 0.4917068 0.000000e+00

4.0 0.4917068 0.000000e+00

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

The final value used for the model was threshold = 0.8.

Confusion Matrix and Statistics

Reference

Prediction Mild None Severe

Mild 40044 32759 8937

None 1762 1465 198

Severe 96 11 8

Overall Statistics

Accuracy : 0.4868

95% CI : (0.4835, 0.4902)

No Information Rate : 0.4913

P-Value [Acc > NIR] : 0.9959

Kappa : -8e-04

Mcnemar's Test P-Value : <2e-16

Statistics by Class:

Class: Mild Class: None Class: Severe

Sensitivity 0.95566 0.04279 8.750e-04

Specificity 0.03878 0.96160 9.986e-01

Pos Pred Value 0.48989 0.42774 6.957e-02

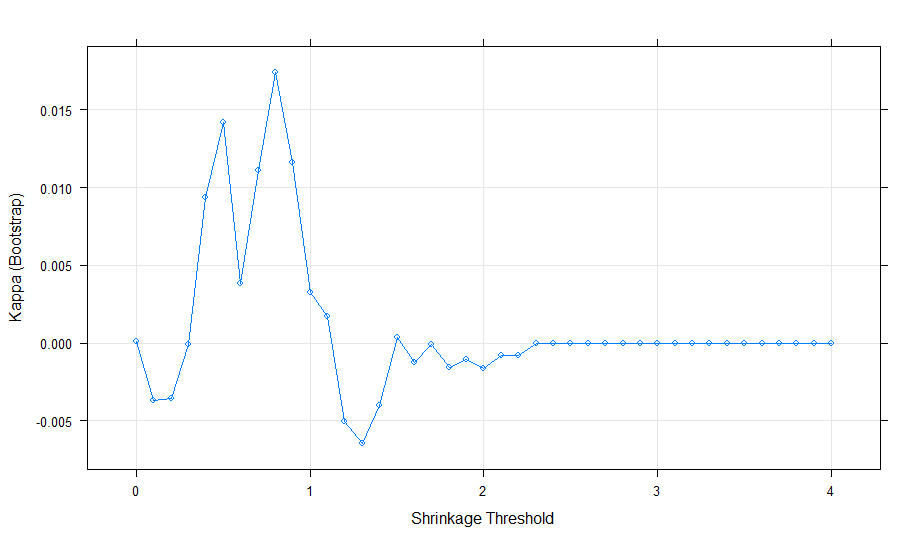
Neg Pred Value 0.47514 0.59966 8.927e-01

Prevalence 0.49135 0.40144 1.072e-01

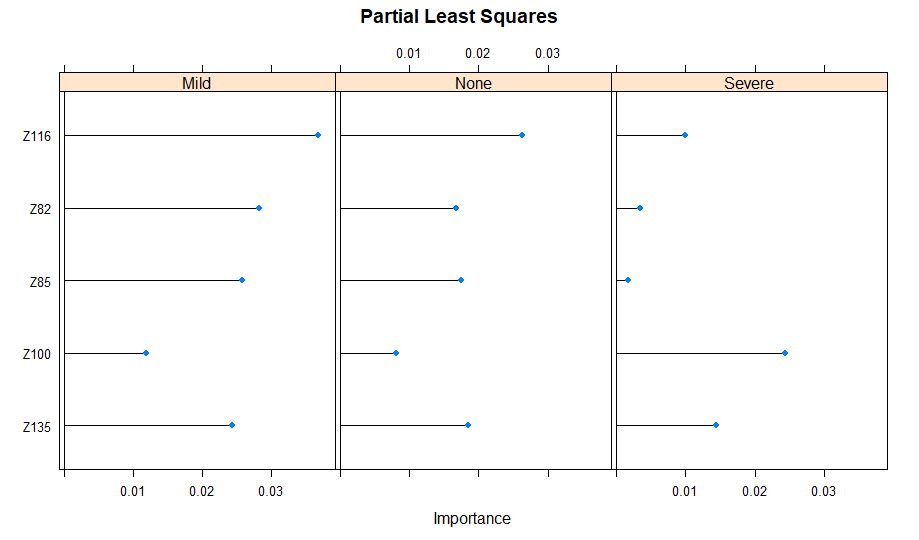
Detection Rate 0.46956 0.01718 9.381e-05

Detection Prevalence 0.95849 0.04016 1.348e-03

Balanced Accuracy 0.49722 0.50220 4.997e-01



d) Based on the kappa values, the optimal model is partial least squares. The top 5 predictors for partial least squares are Z116, Z82, Z85, Z100, and Z135.

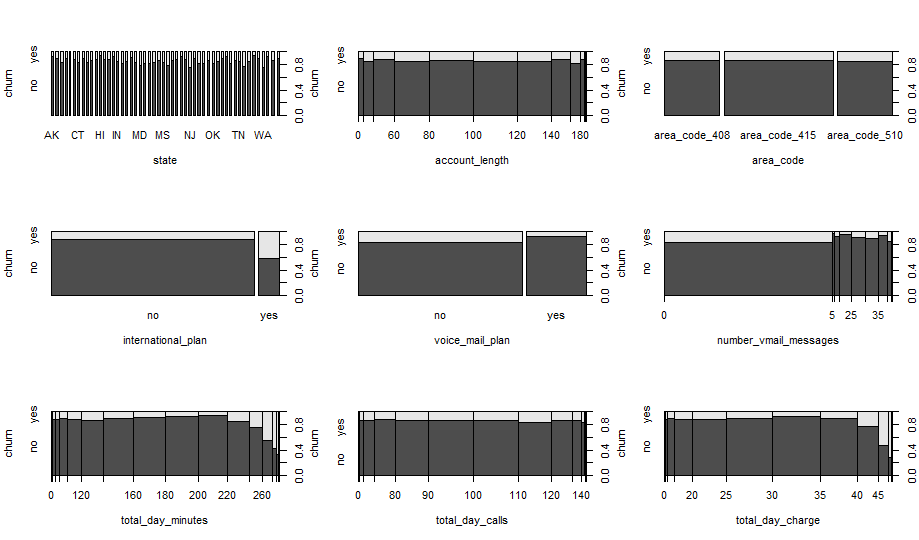


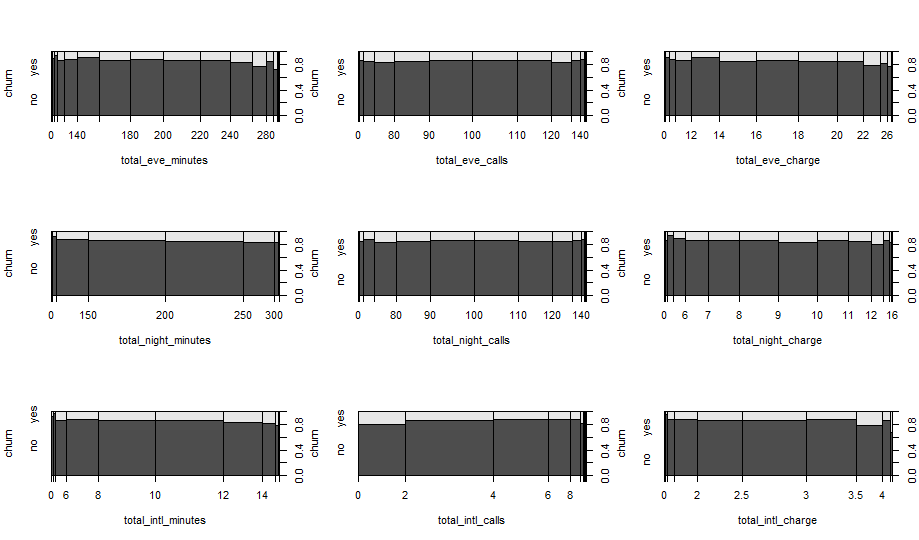
12.3

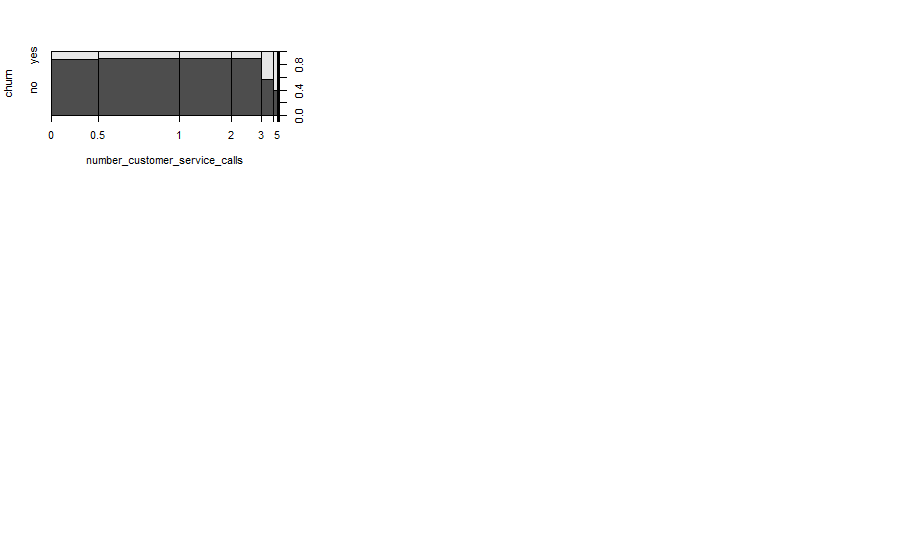
a) The first plot shown is a scatterplot matrix of all the data provided. Unfortunately, it is too large to really understand what is happening. The following plots of predictors against churn show that not only is churn unbalanced, but that there are multiple degenerate predictors: international\_plan, voice\_mail\_plan, and number\_vmail\_messages. While the large matrix is difficult to see, it’s apparent that there is strong correlation between the call length and the call charge for each time of day.

Background pattern

Description automatically generated







b) For this dataset, since the response variable has two levels, I chose ROC to optimize the models. The best model will have the highest area under the ROC curve and the highest accuracy.

c) Based on the following table, the best model for this dataset would be Partial Least Squares Discriminant Analysis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Tuning Parameter** | **Training AOC** | **Training Accuracy** | **Testing Accuracy** |
| Logistic Regression | N/A | 0.8092 | 0.864 | 0.8708709 |
| Linear Discriminant Analysis | N/A | 0.8142 | 0.8569 | 0.8648649 |
| Partial Least Squares | ncomp = 5 | 0.8104 | 0.8621 | 0.8708709 |
| Penalized Models | alpha = 0.6 and lambda = 0.01 | 0.764 | 0.861 | 0.8728729 |
| Nearest Shrunken Centroids | threshold = 0.5 | 0.7812 | 0.8569 | 0.8588589 |

**Logistic Regression**

Generalized Linear Model

4001 samples

67 predictor

2 classes: 'yes', 'no'

No pre-processing

Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)

Summary of sample sizes: 3002, 3002, 3002, 3002, 3002, 3002, ...

Resampling results:

ROC Sens Spec

0.809527 0.2357447 0.9672261

Confusion Matrix and Statistics

Reference

Prediction yes no

yes 831 703

no 2694 20747

Accuracy : 0.864

95% CI : (0.8597, 0.8682)

No Information Rate : 0.8589

P-Value [Acc > NIR] : 0.009985

Kappa : 0.2657

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.23574

Specificity : 0.96723

Pos Pred Value : 0.54172

Neg Pred Value : 0.88507

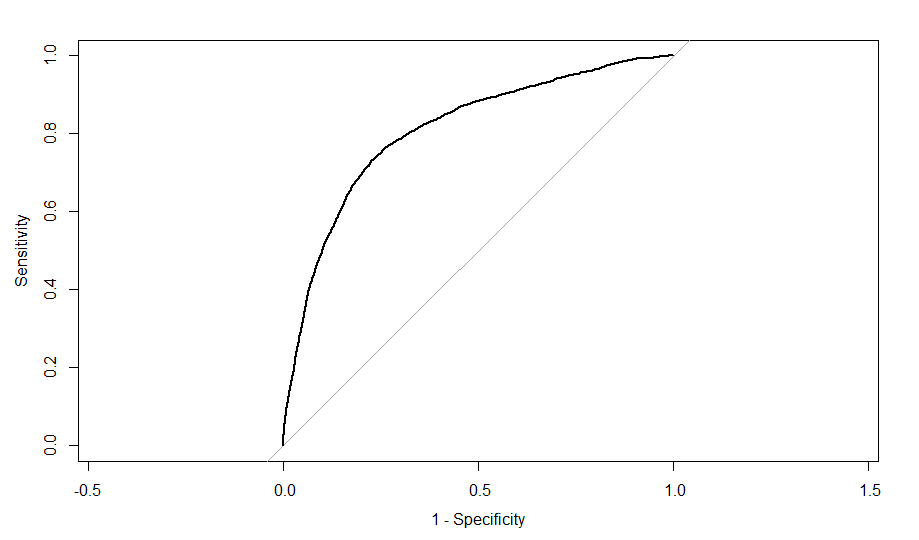
Prevalence : 0.14114

Detection Rate : 0.03327

Detection Prevalence : 0.06142

Balanced Accuracy : 0.60149

'Positive' Class : yes



**Linear Discriminant Analysis**

Linear Discriminant Analysis

4001 samples

67 predictor

2 classes: 'yes', 'no'

No pre-processing

Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)

Summary of sample sizes: 3002, 3002, 3002, 3002, 3002, 3002, ...

Resampling results:

ROC Sens Spec

0.8148725 0.2539007 0.9559907

Confusion Matrix and Statistics

Reference

Prediction yes no

yes 895 944

no 2630 20506

Accuracy : 0.8569

95% CI : (0.8525, 0.8612)

No Information Rate : 0.8589

P-Value [Acc > NIR] : 0.816

Kappa : 0.2623

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.25390

Specificity : 0.95599

Pos Pred Value : 0.48668

Neg Pred Value : 0.88632

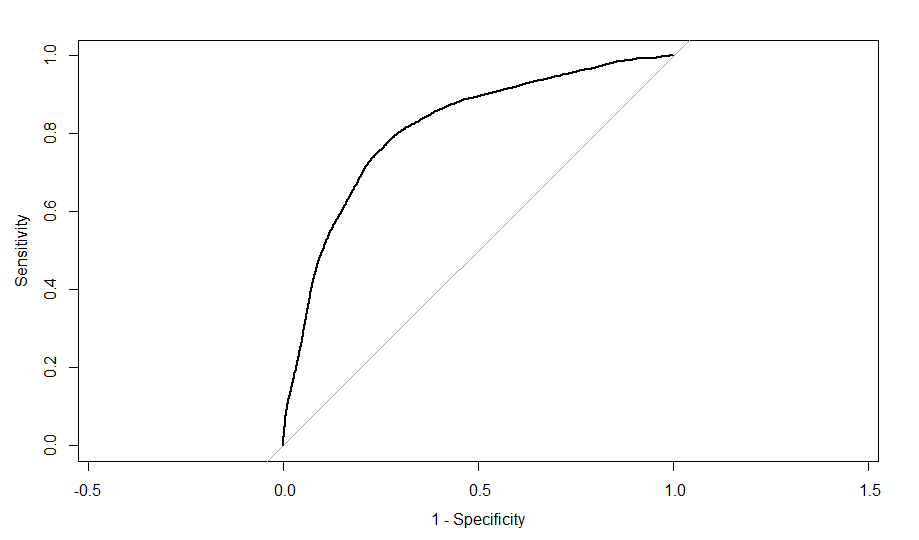
Prevalence : 0.14114

Detection Rate : 0.03584

Detection Prevalence : 0.07363

Balanced Accuracy : 0.60495

'Positive' Class : yes



**Partial Least Squares**

Partial Least Squares

4001 samples

67 predictor

2 classes: 'yes', 'no'

Pre-processing: centered (67), scaled (67)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 4001, 4001, 4001, 4001, 4001, 4001, ...

Resampling results across tuning parameters:

ncomp ROC Sens Spec

1 0.7926907 0.07176943 0.9971262

2 0.8117610 0.11153767 0.9870269

3 0.8126848 0.11718495 0.9863862

4 0.8129797 0.11833023 0.9862935

5 0.8131174 0.11832003 0.9861986

6 0.8130824 0.11775656 0.9861980

7 0.8126281 0.11776522 0.9863255

8 0.8127462 0.11793152 0.9863245

9 0.8129302 0.11809476 0.9862293

10 0.8128663 0.11791295 0.9862293

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

The final value used for the model was ncomp = 5.

Confusion Matrix and Statistics

Reference

Prediction yes no

yes 5885 3942

no 46585 310048

Accuracy : 0.8621

95% CI : (0.861, 0.8632)

No Information Rate : 0.8568

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

Kappa : 0.1506

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.11216

Specificity : 0.98745

Pos Pred Value : 0.59886

Neg Pred Value : 0.86938

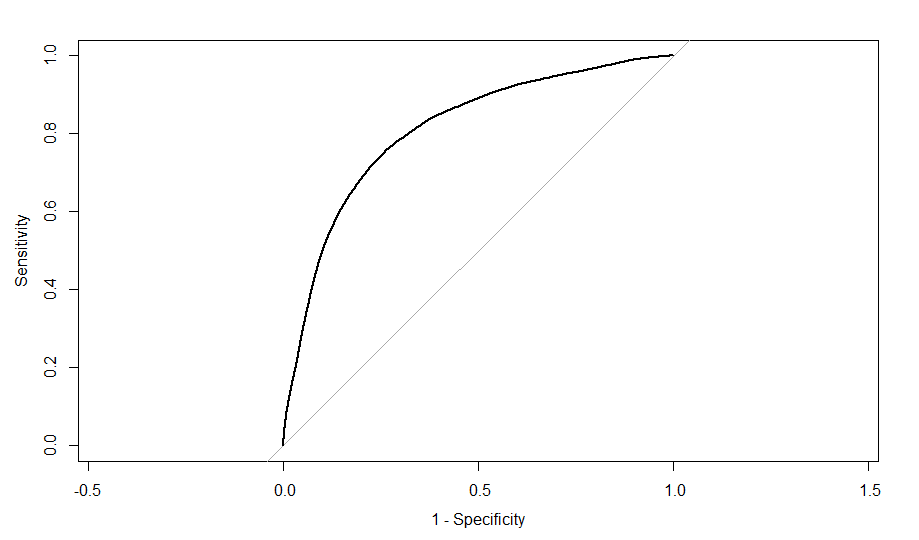
Prevalence : 0.14318

Detection Rate : 0.01606

Detection Prevalence : 0.02682

Balanced Accuracy : 0.54980

'Positive' Class : yes



**Penalized Models**

glmnet

4001 samples

67 predictor

2 classes: 'yes', 'no'

Pre-processing: centered (67), scaled (67)

Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)

Summary of sample sizes: 3002, 3002, 3002, 3002, 3002, 3002, ...

Resampling results across tuning parameters:

alpha lambda ROC Sens Spec

0.0 0.01000000 0.8126641 0.1934751773 0.9737529

0.0 0.03111111 0.8137331 0.1478014184 0.9830769

0.0 0.05222222 0.8138101 0.1160283688 0.9900699

0.0 0.07333333 0.8135109 0.0890780142 0.9946853

0.0 0.09444444 0.8130576 0.0649645390 0.9969697

0.0 0.11555556 0.8126059 0.0496453901 0.9988345

0.0 0.13666667 0.8122264 0.0365957447 0.9994406

0.0 0.15777778 0.8117446 0.0241134752 0.9998601

0.0 0.17888889 0.8113161 0.0164539007 1.0000000

0.0 0.20000000 0.8108621 0.0130496454 1.0000000

0.1 0.01000000 0.8153066 0.1841134752 0.9748718

0.1 0.03111111 0.8191499 0.1299290780 0.9861538

0.1 0.05222222 0.8196452 0.0896453901 0.9923077

0.1 0.07333333 0.8186202 0.0609929078 0.9968765

0.1 0.09444444 0.8169075 0.0337588652 0.9992541

0.1 0.11555556 0.8149617 0.0212765957 0.9998601

0.1 0.13666667 0.8126721 0.0136170213 1.0000000

0.1 0.15777778 0.8105484 0.0065248227 1.0000000

0.1 0.17888889 0.8086703 0.0028368794 1.0000000

0.1 0.20000000 0.8067831 0.0011347518 1.0000000

0.2 0.01000000 0.8175136 0.1767375887 0.9758508

0.2 0.03111111 0.8205961 0.1160283688 0.9879720

0.2 0.05222222 0.8185197 0.0731914894 0.9941725

0.2 0.07333333 0.8147994 0.0368794326 0.9985548

0.2 0.09444444 0.8113436 0.0184397163 0.9998135

0.2 0.11555556 0.8079439 0.0079432624 1.0000000

0.2 0.13666667 0.8035874 0.0025531915 1.0000000

0.2 0.15777778 0.7981322 0.0002836879 1.0000000

0.2 0.17888889 0.7940358 0.0000000000 1.0000000

0.2 0.20000000 0.7915474 0.0000000000 1.0000000

0.4 0.01000000 0.8203101 0.1676595745 0.9783217

0.4 0.03111111 0.8184143 0.0879432624 0.9904429

0.4 0.05222222 0.8123890 0.0397163121 0.9974825

0.4 0.07333333 0.8043902 0.0113475177 0.9998135

0.4 0.09444444 0.7968727 0.0017021277 1.0000000

0.4 0.11555556 0.7929281 0.0000000000 1.0000000

0.4 0.13666667 0.7845805 0.0000000000 1.0000000

0.4 0.15777778 0.7663197 0.0000000000 1.0000000

0.4 0.17888889 0.7405971 0.0000000000 1.0000000

0.4 0.20000000 0.6582314 0.0000000000 1.0000000

0.6 0.01000000 0.8213447 0.1560283688 0.9806993

0.6 0.03111111 0.8147551 0.0675177305 0.9931469

0.6 0.05222222 0.8034281 0.0178723404 0.9994406

0.6 0.07333333 0.7965308 0.0014184397 1.0000000

0.6 0.09444444 0.7842333 0.0000000000 1.0000000

0.6 0.11555556 0.7443606 0.0000000000 1.0000000

0.6 0.13666667 0.6432315 0.0000000000 1.0000000

0.6 0.15777778 0.5441105 0.0000000000 1.0000000

0.6 0.17888889 0.5000000 0.0000000000 1.0000000

0.6 0.20000000 0.5000000 0.0000000000 1.0000000

0.8 0.01000000 0.8212372 0.1475177305 0.9820979

0.8 0.03111111 0.8111643 0.0476595745 0.9957110

0.8 0.05222222 0.7986518 0.0042553191 1.0000000

0.8 0.07333333 0.7820680 0.0000000000 1.0000000

0.8 0.09444444 0.7156442 0.0000000000 1.0000000

0.8 0.11555556 0.5627521 0.0000000000 1.0000000

0.8 0.13666667 0.5000000 0.0000000000 1.0000000

0.8 0.15777778 0.5000000 0.0000000000 1.0000000

0.8 0.17888889 0.5000000 0.0000000000 1.0000000

0.8 0.20000000 0.5000000 0.0000000000 1.0000000

1.0 0.01000000 0.8206940 0.1404255319 0.9835897

1.0 0.03111111 0.8047741 0.0334751773 0.9974359

1.0 0.05222222 0.7935828 0.0002836879 1.0000000

1.0 0.07333333 0.7224697 0.0000000000 1.0000000

1.0 0.09444444 0.5536364 0.0000000000 1.0000000

1.0 0.11555556 0.5000000 0.0000000000 1.0000000

1.0 0.13666667 0.5000000 0.0000000000 1.0000000

1.0 0.15777778 0.5000000 0.0000000000 1.0000000

1.0 0.17888889 0.5000000 0.0000000000 1.0000000

1.0 0.20000000 0.5000000 0.0000000000 1.0000000

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

The final values used for the model were alpha = 0.6 and lambda = 0.01.

Confusion Matrix and Statistics

Reference

Prediction yes no

yes 9348 5565

no 237402 1495935

Accuracy : 0.861

95% CI : (0.8605, 0.8615)

No Information Rate : 0.8589

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

Kappa : 0.0563

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.037884

Specificity : 0.996294

Pos Pred Value : 0.626836

Neg Pred Value : 0.863038

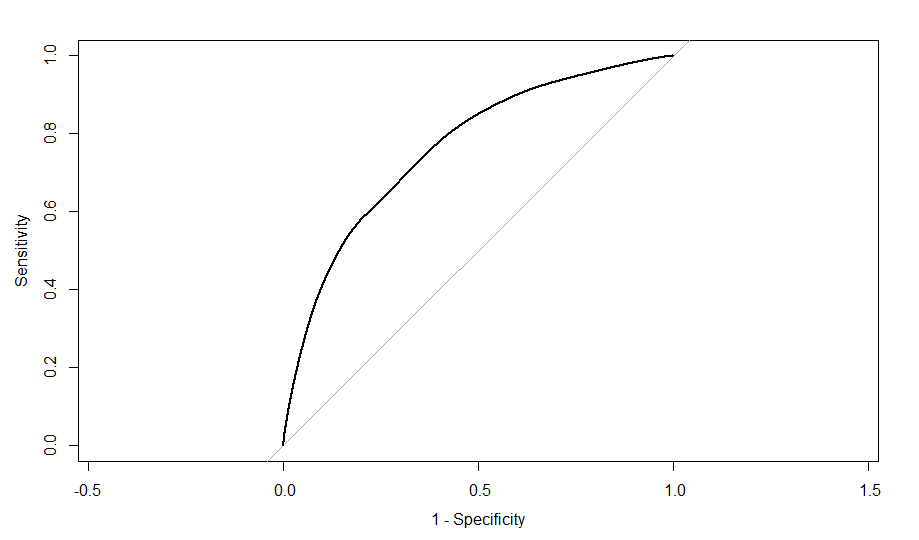
Prevalence : 0.141141

Detection Rate : 0.005347

Detection Prevalence : 0.008530

Balanced Accuracy : 0.517089

'Positive' Class : yes



**Nearest Shrunken Centroids**

Nearest Shrunken Centroids

4001 samples

67 predictor

2 classes: 'yes', 'no'

Pre-processing: centered (67), scaled (67)

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 4001, 4001, 4001, 4001, 4001, 4001, ...

Resampling results across tuning parameters:

threshold ROC Sens Spec

0.0 0.7930876 0.0036514759 1

0.1 0.7939855 0.0029123814 1

0.2 0.7947584 0.0018787548 1

0.3 0.7952939 0.0009744157 1

0.4 0.7956403 0.0007534213 1

0.5 0.7957997 0.0005647420 1

0.6 0.7957562 0.0003686636 1

0.7 0.7955860 0.0001843318 1

0.8 0.7953635 0.0000000000 1

0.9 0.7950832 0.0000000000 1

1.0 0.7947427 0.0000000000 1

1.1 0.7943223 0.0000000000 1

1.2 0.7938835 0.0000000000 1

1.3 0.7934197 0.0000000000 1

1.4 0.7929147 0.0000000000 1

1.5 0.7924038 0.0000000000 1

1.6 0.7918392 0.0000000000 1

1.7 0.7912577 0.0000000000 1

1.8 0.7907178 0.0000000000 1

1.9 0.7901230 0.0000000000 1

2.0 0.7894549 0.0000000000 1

2.1 0.7887387 0.0000000000 1

2.2 0.7880189 0.0000000000 1

2.3 0.7872197 0.0000000000 1

2.4 0.7864150 0.0000000000 1

2.5 0.7855754 0.0000000000 1

2.6 0.7846788 0.0000000000 1

2.7 0.7837970 0.0000000000 1

2.8 0.7828587 0.0000000000 1

2.9 0.7818391 0.0000000000 1

3.0 0.7808672 0.0000000000 1

3.1 0.7799055 0.0000000000 1

3.2 0.7790051 0.0000000000 1

3.3 0.7781043 0.0000000000 1

3.4 0.7772942 0.0000000000 1

3.5 0.7765178 0.0000000000 1

3.6 0.7756896 0.0000000000 1

3.7 0.7750193 0.0000000000 1

3.8 0.7744074 0.0000000000 1

3.9 0.7737600 0.0000000000 1

4.0 0.7730422 0.0000000000 1

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

The final value used for the model was threshold = 0.5.

Confusion Matrix and Statistics

Reference

Prediction yes no

yes 59 0

no 215068 1287359

Accuracy : 0.8569

95% CI : (0.8563, 0.8574)

No Information Rate : 0.8568

P-Value [Acc > NIR] : 0.4459

Kappa : 5e-04

Mcnemar's Test P-Value : <2e-16

Sensitivity : 2.743e-04

Specificity : 1.000e+00

Pos Pred Value : 1.000e+00

Neg Pred Value : 8.569e-01

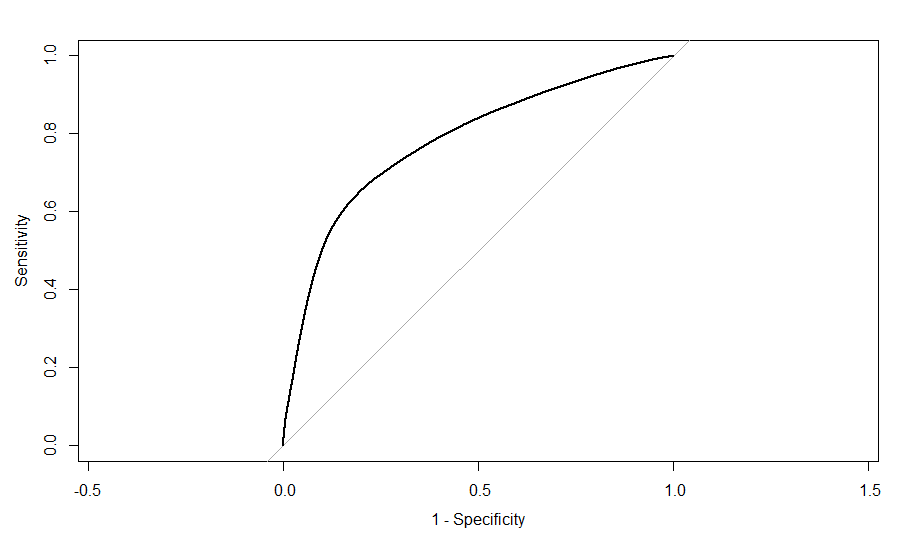
Prevalence : 1.432e-01

Detection Rate : 3.927e-05

Detection Prevalence : 3.927e-05

Balanced Accuracy : 5.001e-01

'Positive' Class : yes



**R Code**

library(AppliedPredictiveModeling)

library(caret)

library(glmnet)

library(MASS)

library(modeldata)

library(pROC)

library(tidyverse)

####12.1####

set.seed(80)

data(hepatic)

#a) I would create a training and testing set by splitting the data using

#stratified random sampling and using injury as the factor

#b) I would optimize using kappa since we have more than two levels of classes,

#which makes using ROC ineffective.

####12.1c####

fullSet <- bio

fullSet[, 185] <- injury

#injury is named V378

trainingRows <- createDataPartition(fullSet[,185], p=0.80, list=FALSE)

fullTraining <- fullSet[trainingRows,]

fullTesting <- fullSet[-trainingRows,]

#Logistic Regression

ctrl <- trainControl(method = "LGOCV",

summaryFunction = defaultSummary,

classProbs = TRUE,

savePredictions = TRUE)

lrFit <- train(fullTraining[,1:184],

y = fullTraining$V185,

method = "multinom",

metric = "Accuracy",

preProc = c("knnImpute", "nzv"),

trControl = ctrl)

lrFit

confusionMatrix(data = lrFit$pred$pred,

reference = lrFit$pred$obs)

plot(lrFit)

predicted <- predict(lrFit, fullTesting[,1:184])

lrValues <- data.frame(obs = fullTesting[,185], pred = predicted)

defaultSummary(lrValues)

#Linear Discriminant Analysis

ldaFit <- train(fullTraining[,1:184],

y = fullTraining$V185,

method = "lda",

metric = "Kappa",

preProc = c("center", "scale", "nzv", "corr"),

trControl = ctrl)

ldaFit

confusionMatrix(data = ldaFit$pred$pred,

reference = ldaFit$pred$obs)

plot(ldaFit) #no tuning parameters

predicted <- predict(ldaFit, fullTesting[,1:184])

ldaValues <- data.frame(obs = fullTesting[,185], pred = predicted)

defaultSummary(ldaValues)

#Partial Least Squares Discriminant Analysis

ctrl <- trainControl(summaryFunction = defaultSummary,

classProbs = TRUE,

savePredictions = TRUE)

plsFit <- train(fullTraining[,1:184],

y = fullTraining$V185,

method = "pls",

tuneGrid = expand.grid(.ncomp = 1:10),

preProc = c("center","scale", "nzv", "corr"),

metric = "Kappa",

maxit = 2000,

trControl = ctrl)

plsFit

confusionMatrix(data = plsFit$pred$pred,

reference = plsFit$pred$obs)

plot(plsFit)

predicted <- predict(plsFit, fullTesting[,1:184])

plsValues <- data.frame(obs = fullTesting[,185], pred = predicted)

defaultSummary(plsValues)

#Penalized Models

ctrl <- trainControl(method = "LGOCV",

summaryFunction = defaultSummary,

classProbs = TRUE,

savePredictions = TRUE)

glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),

.lambda = seq(.01, .2, length = 10))

glmnFit <- train(fullTraining[,1:184],

y = fullTraining$V185,

method = "glmnet",

tuneGrid = glmnGrid,

preProc = c("center", "scale", "nzv"),

metric = "Kappa",

trControl = ctrl)

glmnFit

confusionMatrix(data = glmnFit$pred$pred,

reference = glmnFit$pred$obs)

plot(glmnFit)

predicted <- predict(glmnFit, fullTesting[,1:184])

glmnValues <- data.frame(obs = fullTesting[,185], pred = predicted)

defaultSummary(glmnValues)

#Nearest Shrunken Centroids

ctrl <- trainControl(summaryFunction = defaultSummary,

classProbs = TRUE,

savePredictions = TRUE)

nscGrid <- data.frame(.threshold = seq(0,4, by=0.1))

nscFit <- train(fullTraining[,1:184],

y = fullTraining$V185,

method = "pam",

preProc = c("center", "scale", "nzv"),

tuneGrid = nscGrid,

metric = "Kappa",

trControl = ctrl)

nscFit

confusionMatrix(data = nscFit$pred$pred,

reference = nscFit$pred$obs)

plot(nscFit)

predicted <- predict(nscFit, fullTesting[,1:184])

nscValues <- data.frame(obs = fullTesting[,185], pred = predicted)

defaultSummary(nscValues)

####12.1d####

#highest kappa value

plsImp <- varImp(plsFit, scale = FALSE)

plsImp

plot(plsImp, top = 5, main = "Partial Least Squares")

####12.3####

set.seed(80)

data(mlc\_churn)

mlc\_churn <- as.data.frame(mlc\_churn)

####12.3a####

#outcome: churn, yes/no

pairs(mlc\_churn)

par(mfrow=c(3,3))

plot(churn~state, data = mlc\_churn)

plot(churn~account\_length, data = mlc\_churn)

plot(churn~area\_code, data = mlc\_churn)

plot(churn~international\_plan, data = mlc\_churn)

plot(churn~voice\_mail\_plan, data = mlc\_churn)

plot(churn~number\_vmail\_messages, data = mlc\_churn)

plot(churn~total\_day\_minutes, data = mlc\_churn)

plot(churn~total\_day\_calls, data = mlc\_churn)

plot(churn~total\_day\_charge, data = mlc\_churn)

plot(churn~total\_eve\_minutes, data = mlc\_churn)

plot(churn~total\_eve\_calls, data = mlc\_churn)

plot(churn~total\_eve\_charge, data =mlc\_churn)

plot(churn~total\_night\_minutes, data = mlc\_churn)

plot(churn~total\_night\_calls, data = mlc\_churn)

plot(churn~total\_night\_charge, data = mlc\_churn)

plot(churn~total\_intl\_minutes, data = mlc\_churn)

plot(churn~total\_intl\_calls, data = mlc\_churn)

plot(churn~total\_intl\_charge, data = mlc\_churn)

plot(churn~number\_customer\_service\_calls, data = mlc\_churn)

#12.3b: ROC should be used since the outcome has two classes.

####12.3c####

dummRes <-dummyVars("~state+area\_code+international\_plan+voice\_mail\_plan", data=mlc\_churn, fullRank=TRUE)

Add\_dumm <- data.frame(predict(dummRes, newdata=mlc\_churn))

full\_churn <- Add\_dumm

full\_churn[,54:68] <- mlc\_churn[,6:20]

trainingRows <- createDataPartition(full\_churn$churn, p=0.80, list=FALSE)

fullTraining <- full\_churn[trainingRows,]

fullTesting <- full\_churn[-trainingRows,]

#Logistic Regression

ctrl <- trainControl(method = "LGOCV",

summaryFunction = twoClassSummary,

classProbs = TRUE,

savePredictions = TRUE)

lrFit <- train(fullTraining[,1:67],

y = fullTraining$churn,

method = "glm",

metric = "ROC",

trControl = ctrl)

lrFit

confusionMatrix(data = lrFit$pred$pred,

reference = lrFit$pred$obs)

lrRoc <- roc(response = lrFit$pred$obs,

predictor = lrFit$pred$yes,

levels = rev(levels(lrFit$pred$obs)))

plot(lrRoc, legacy.axes = TRUE)

auc(lrRoc)

predicted <- predict(lrFit, fullTesting[,1:67])

lrValues <- data.frame(obs = fullTesting[,68], pred = predicted)

defaultSummary(lrValues)

#Linear Discriminant Analysis

ldaFit <- train(x = fullTraining[,1:67],

y = fullTraining$churn,

method = "lda",

metric = "ROC",

trControl = ctrl)

ldaFit

confusionMatrix(data = ldaFit$pred$pred,

reference = ldaFit$pred$obs)

ldaRoc <- roc(response = ldaFit$pred$obs,

predictor = ldaFit$pred$yes,

levels = rev(levels(ldaFit$pred$obs)))

plot(ldaRoc, legacy.axes = TRUE)

auc(ldaRoc)

predicted <- predict(ldaFit, fullTesting[,1:67])

ldaValues <- data.frame(obs = fullTesting[,68], pred = predicted)

defaultSummary(ldaValues)

#Partial Least Squares

ctrl <- trainControl(summaryFunction = twoClassSummary,

classProbs = TRUE,

savePredictions = TRUE)

plsFit <- train(fullTraining[,1:67],

y = fullTraining$churn,

method = "pls",

tuneGrid = expand.grid(.ncomp = 1:10),

preProc = c("center","scale"),

metric = "ROC",

trControl = ctrl)

plsFit

confusionMatrix(data = plsFit$pred$pred,

reference = plsFit$pred$obs)

plsRoc <- roc(response = plsFit$pred$obs,

predictor = plsFit$pred$yes,

levels = rev(levels(plsFit$pred$obs)))

plot(plsRoc, legacy.axes = TRUE)

auc(plsRoc)

predicted <- predict(plsFit, fullTesting[,1:67])

plsValues <- data.frame(obs = fullTesting[,68], pred = predicted)

defaultSummary(plsValues)

#Penalized Models

ctrl <- trainControl(method = "LGOCV",

summaryFunction = twoClassSummary,

classProbs = TRUE,

savePredictions = TRUE)

glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),

.lambda = seq(.01, .2, length = 10))

glmnFit <- train(fullTraining[,1:67],

y = fullTraining$churn,

method = "glmnet",

tuneGrid = glmnGrid,

preProc = c("center", "scale"),

metric = "ROC",

trControl = ctrl)

glmnFit

confusionMatrix(data = glmnFit$pred$pred,

reference = glmnFit$pred$obs)

glmnRoc <- roc(response = glmnFit$pred$obs,

predictor = glmnFit$pred$yes,

levels = rev(levels(glmnFit$pred$obs)))

plot(glmnRoc, legacy.axes = TRUE)

auc(glmnRoc)

predicted <- predict(glmnFit, fullTesting[,1:67])

glmnValues <- data.frame(obs = fullTesting[,68], pred = predicted)

defaultSummary(glmnValues)

#Nearest Shrunken Centroids

ctrl <- trainControl(summaryFunction = twoClassSummary,

classProbs = TRUE,

savePredictions = TRUE)

nscGrid <- data.frame(.threshold = seq(0,4, by=0.1))

nscFit <- train(x = fullTraining[,1:67],

y = fullTraining$churn,

method = "pam",

preProc = c("center", "scale"),

tuneGrid = nscGrid,

metric = "ROC",

trControl = ctrl)

nscFit

confusionMatrix(data = nscFit$pred$pred,

reference = nscFit$pred$obs)

nscRoc <- roc(response = nscFit$pred$obs,

predictor = nscFit$pred$yes,

levels = rev(levels(nscFit$pred$obs)))

plot(nscRoc, legacy.axes = TRUE)

auc(nscRoc)

predicted <- predict(nscFit, fullTesting[,1:67])

nscValues <- data.frame(obs = fullTesting[,68], pred = predicted)

defaultSummary(nscValues)