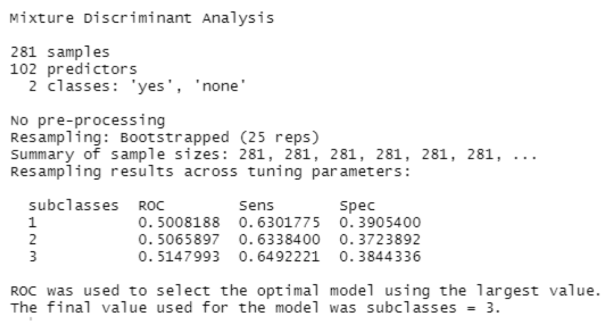
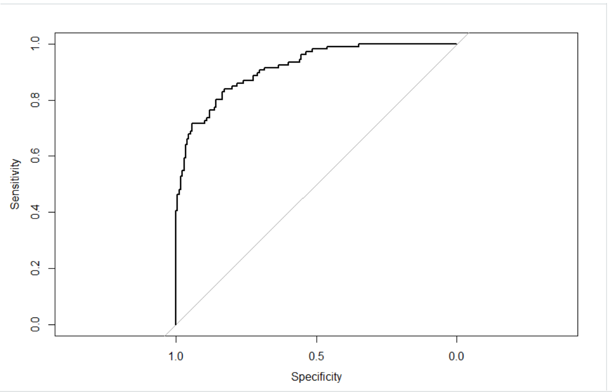
MA4790 Homework 6

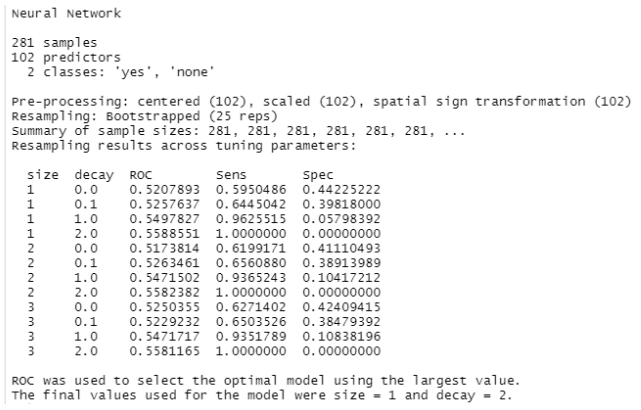
Ian Boulis

13.1 Which model has the best predictability for the biological predictors and what is the optimal performance?

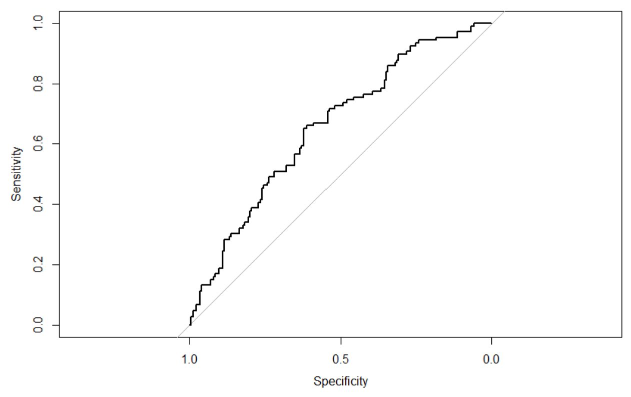


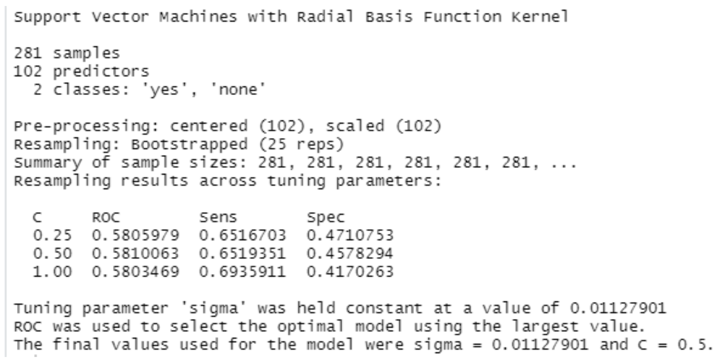
```
Mixture Discriminant Analysis

281 samples
102 predictors
  2 classes: 'yes', 'none'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 281, 281, 281, 281, 281, 281, ...
Resampling results across tuning parameters:

  subclasses  ROC        Sens       Spec
  1           0.5008188  0.6301775  0.3905400
  2           0.5065897  0.6338400  0.3723892
  3           0.5147993  0.6492221  0.3844336

ROC was used to select the optimal model using the largest value.
The final value used for the model was subclasses = 3.
```



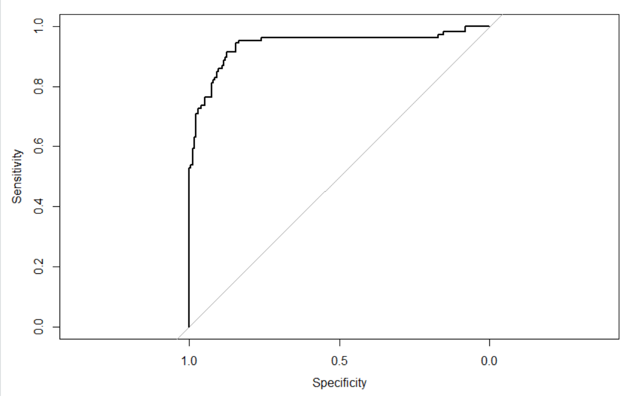


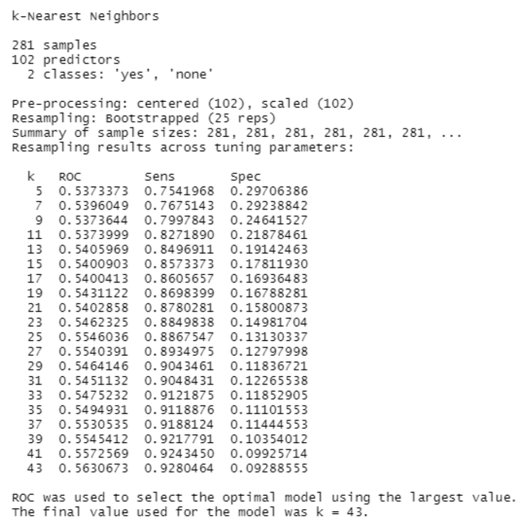
```
Neural Network

281 samples
102 predictors
  2 classes: 'yes', 'none'

Pre-processing: centered (102), scaled (102), spatial sign transformation (102)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 281, 281, 281, 281, 281, 281, ...
Resampling results across tuning parameters:

  size  decay  ROC        Sens       Spec
  1     0.0    0.5207893  0.5950486  0.44225222
  1     0.1    0.5257637  0.6445042  0.39818000
  1     1.0    0.5497827  0.9625515  0.05798392
  1     2.0    0.5588551  1.0000000  0.00000000
  2     0.0    0.5173814  0.6199171  0.41110493
  2     0.1    0.5263461  0.6560880  0.38913989
  2     1.0    0.5471502  0.9365243  0.10417212
  2     2.0    0.5582382  1.0000000  0.00000000
  3     0.0    0.5250355  0.6271402  0.42409415
  3     0.1    0.5229232  0.6503526  0.38479392
  3     1.0    0.5471717  0.9351789  0.10838196
  3     2.0    0.5581165  1.0000000  0.00000000

ROC was used to select the optimal model using the largest value.
The final values used for the model were size = 1 and decay = 2.
```



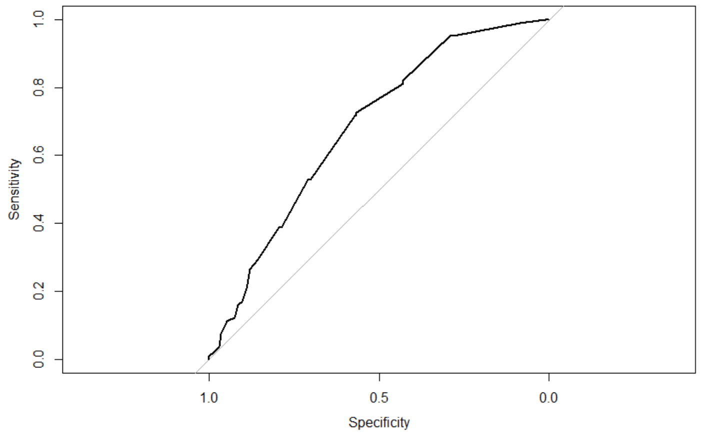


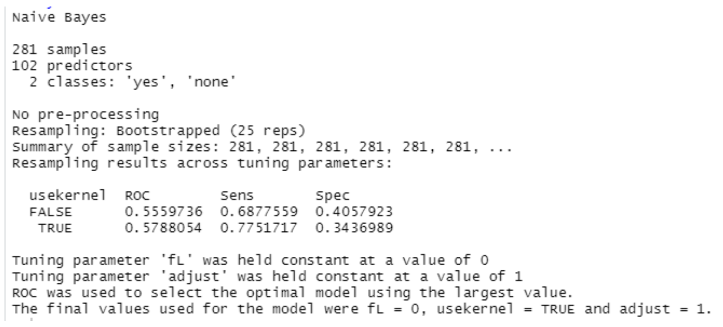
```
Support Vector Machines with Radial Basis Function Kernel

281 samples
102 predictors
  2 classes: 'yes', 'none'

Pre-processing: centered (102), scaled (102)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 281, 281, 281, 281, 281, 281, ...
Resampling results across tuning parameters:

  C     ROC        Sens       Spec
  0.25  0.5805979  0.6516703  0.4710753
  0.50  0.5810063  0.6519351  0.4578294
  1.00  0.5803469  0.6935911  0.4170263

Tuning parameter 'sigma' was held constant at a value of 0.01127901
ROC was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.01127901 and C = 0.5.
```



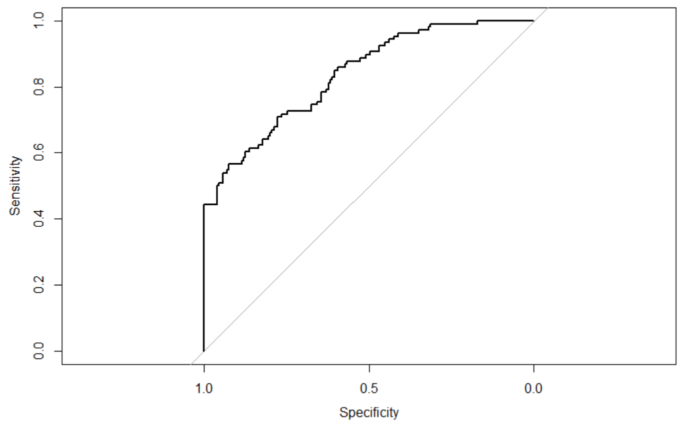


```
k-Nearest Neighbors

281 samples
102 predictors
  2 classes: 'yes', 'none'

Pre-processing: centered (102), scaled (102)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 281, 281, 281, 281, 281, 281, ...
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   5  0.5373373  0.7541968  0.29706386
   7  0.5396049  0.7675143  0.29238842
   9  0.5373644  0.7997843  0.24641527
  11  0.5373999  0.8271890  0.21878461
  13  0.5405969  0.8496911  0.19142463
  15  0.5400903  0.8573373  0.17811930
  17  0.5400413  0.8605657  0.16936483
  19  0.5431122  0.8698399  0.16788281
  21  0.5402858  0.8780281  0.15800873
  23  0.5462325  0.8849838  0.14981704
  25  0.5546036  0.8867547  0.13130337
  27  0.5540391  0.8934975  0.12797998
  29  0.5464146  0.9043461  0.11836721
  31  0.5451132  0.9048431  0.12265538
  33  0.5475232  0.9121875  0.11852905
  35  0.5494931  0.9118876  0.11101553
  37  0.5530535  0.9188124  0.11444553
  39  0.5545412  0.9217791  0.10354012
  41  0.5572569  0.9243450  0.09925714
  43  0.5630673  0.9280464  0.09288555

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 43.
```





```
Naive Bayes

281 samples
102 predictors
  2 classes: 'yes', 'none'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 281, 281, 281, 281, 281, 281, ...
Resampling results across tuning parameters:

  usekernel  ROC        Sens       Spec
  FALSE      0.5559736  0.6877559  0.4057923
   TRUE      0.5788054  0.7751717  0.3436989

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
ROC 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.
```



Model	Tuning Parameters	ROC Area
MDA	Subclasses = 3	0.9147
Neural Network	Size = 1, Decay = 2	0.6553
Support Vector Machine	Sigma = 0.01127901, c =0.5	0.9388
K-Nearest Neighbors	K = 43	0.6812
Naïve Bayes	Fl=0, Kernel=true, Adjust =1	0.8354

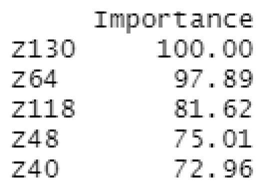
The Support Vector Machine seems to be the best model overall for this data.

B. Does the non-linear structure of these models help to improve the classification performance?

The non-linear models give a better prediction on the top end but have a lower area under the curve value for the worst performing model. The best ROC value from the linear models was 0.769, while the best for non-linear was 0.939, so it gives a better value for the best performing model. However, the worst linear model had an ROC value of 0.748 while the worst non-linear model had a value of 0.6553. I would say that the non-linear models are better for this data since the difference in their best-case scenarios is almost +0.3, and the difference in the worst case only be -0.1 meaning it was able to give a much better “best” performance.

C. For the optimal model, what is the top five important predictors?

Using the varImp function on the Support Vector Machine, we see that the top 5 most important predictors for the data are:



```
     Importance
Z130     100.00
Z64       97.89
Z118      81.62
Z48       75.01
Z40       72.96
```

```
##13.1##

library(caret)

library(glmnet)

library(pamr)

library(AppliedPredictiveModeling)

library(mda)

data(hepatic)

nzv = nearZeroVar(bio)

bio = bio[,-nzv]

damage = as.character( injury )

damage[ damage=="Mild" ] = "yes"

damage[ damage=="Severe" ] = "yes"

damage[ damage=="None" ] = "none"

damage = factor(damage, levels=c("yes","none"))

ctrl = trainControl(summaryFunction=twoClassSummary, classProbs=TRUE)

##MDA##

mda=train(bio, damage, method="mda", tuneGrid=expand.grid(subclasses=1:3), metric="ROC",
trControl=ctrl)

mdapred = predict(mda, bio, type="prob")

mdaroc = pROC::roc( response=damage, predictor=mdapred[,1] )

##Neural Network##

nngrid = expand.grid(size=1:3, decay =c(0,0.1,1,2))

nnet = train(bio, damage, method="nnet", preProcess=c("center","scale","spatialSign"),
tuneGrid=nngrid, metric="ROC", trControl=ctrl)

nnetpred = predict(nnet, bio, type="prob")

nnetroc = pROC::roc(damage, nnetpred[,1])

##SVM##

svm = train(bio, damage, method="svmRadial", preProc=c("center","scale"), metric="ROC",
trControl=ctrl )

svmpred = predict( svm, bio, type="prob" )

svmroc = pROC::roc(damage, svmpred[,1] )
```

```
##KNN##

knn = train(bio, damage, method="knn", tuneLength=20, preProc=c("center","scale"),
metric="ROC", trControl=ctrl )

knnpred = predict(knn, bio, type="prob" )

knnroc = pROC::roc(damage, knnpred[,1] )

##Naive Bayes##

bayes = train(bio, damage, method="nb", metric="ROC", trControl=ctrl )

bayespred = predict(bayes, bio, type="prob" )

bayesroc = pROC::roc(damage, bayespred[,1] )

##C##

varImp(svm)
```