Homework 6

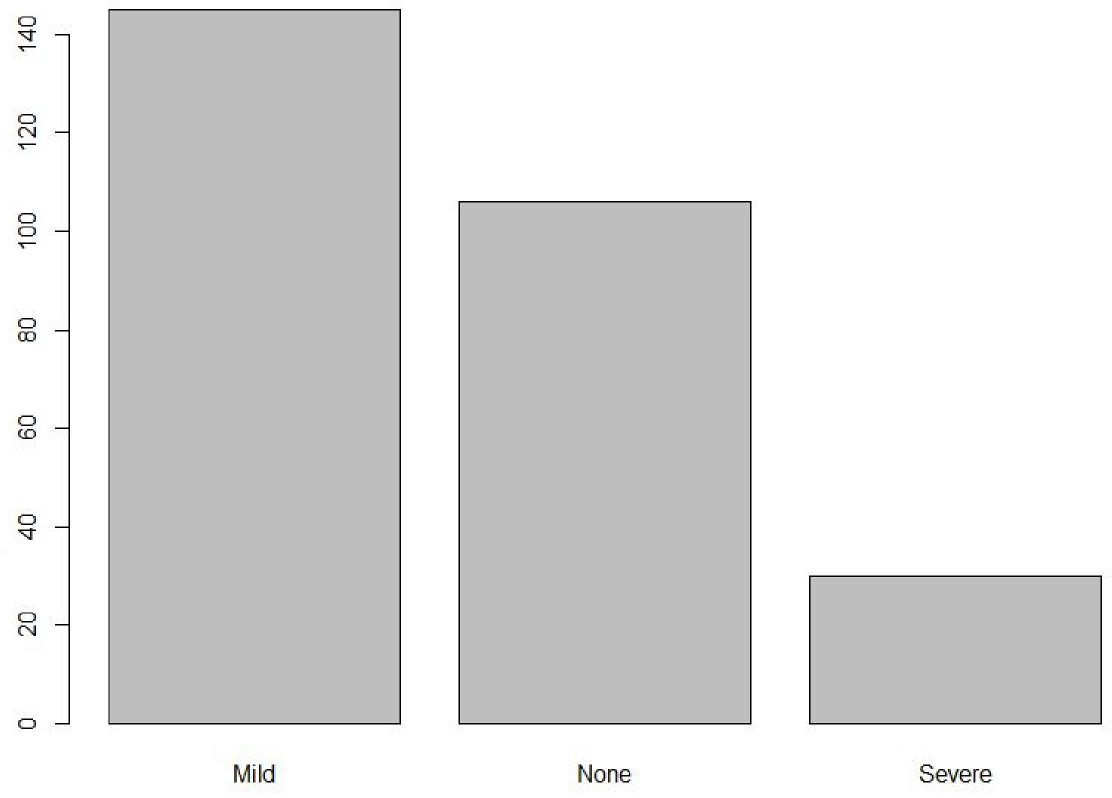
Supriya Bachal

Use the hepatic injury data from the previous exercise set (Exercise 12.1). Recall that the matrices bio and chem contain the biological assay and chemical fingerprint predictors for the 281 compounds, while the vector injury contains the liver damage classification for each compound

(a) Work with the same training and testing sets as well as pre-processing steps as you did in your previous work on these data. Using the same classification statistic as before, build models described in this chapter for the biological predictors and separately for the chemical fingerprint predictors. Which model has the best predictive ability for the biological predictors and what is the optimal performance? Which model has the best predictive ability for the chemical predictors and what is the optimal performance? Does the nonlinear structure of these models help to improve the classification performance?

Using the same preprocessing steps as before first we observe:

The injury status shows very high imbalance in classes as seen in the barplot below.



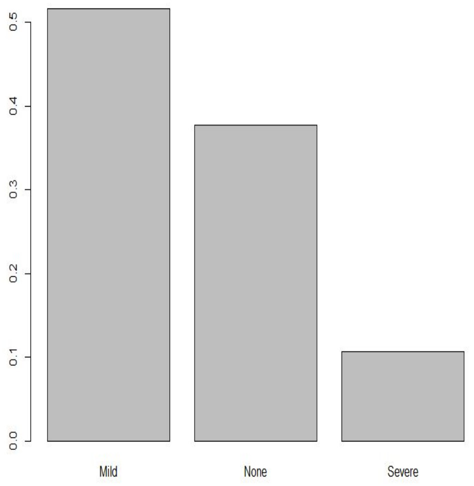
Statistically the class wise count of each of the injuries is as below:

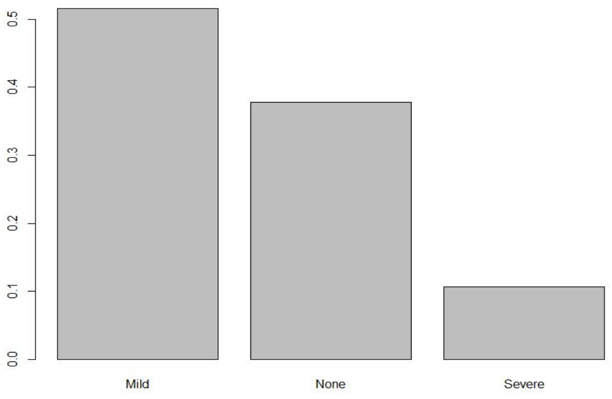
```
injury
  Mild   None Severe
   145    106     30
```

Since the instances are not well distributed among the classes We use stratified sampling :

We creates one partition with 80 percent for training and 20 percent for testing data.

From this we conclude the class distribution: for the original and partitioned data,





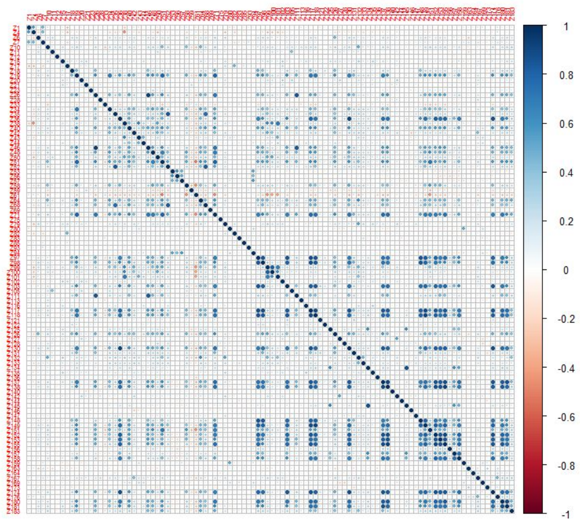
Preprocessing the data:

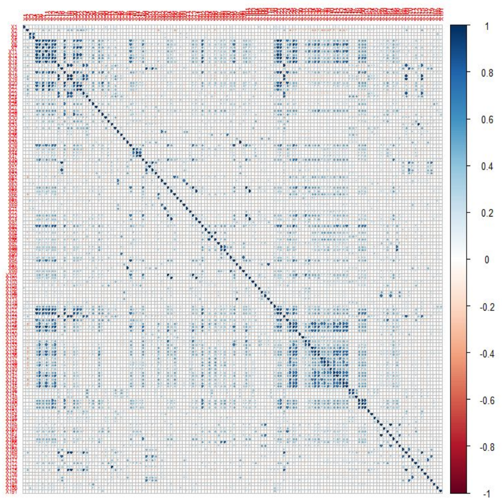
First we drop the near zero variance and zero variance predictors from the bio data.Dropping 82 zero variance columns from 184 (fraction= 0.445652).

Then we drop near zero and zero variance predictors from the chem data.Dropping 58 zero variance columns from 192 (fraction= 0.302083).

The next step is to find any correlation between the data;

Further we found the correlation between the predictors to check for linearly dependent columns.The left figure is depicting the correlation in bio predictors and the right one in chem predictors.





We see that in both the datasets there are some very evident correlations. We have the cut off to 75 percent meaning that the correlations greater than 75 percent have been dropped.

This is the list of predictors for the bio predictors that has been dropped:

68 88 15 75 61 76

6 predictors have been dropped.

This is the list of predictors for chem which has been dropped

10 56 62 82 83 84 87 123 127   5   6   7   8 12 15 14   9 18 16 13 40 47 34 55 48

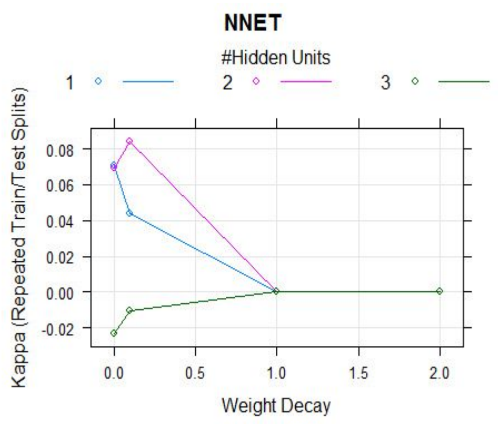
108 115

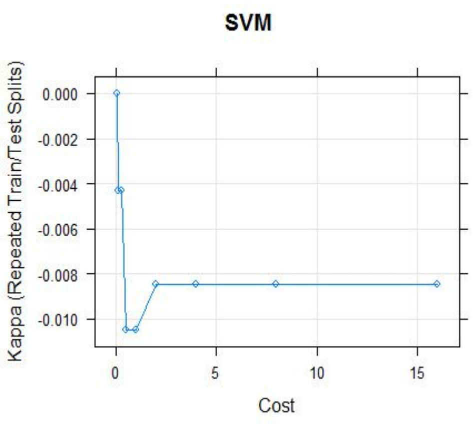
27 predictors were dropped.

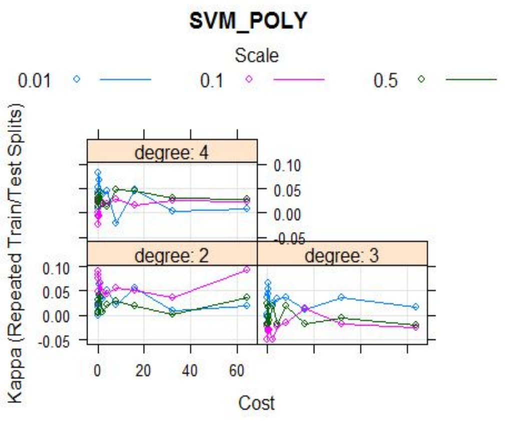
For all the models Leave one group our method for 5 folds of cross validation from the caret library is used after training and testing sets are created using stratified sampling.

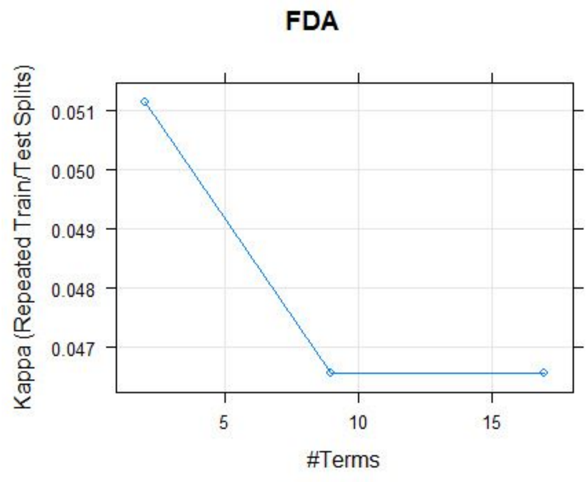
```
ctrl <- trainControl(method="LGOCV",classProbs = TRUE,summaryFunction =
multiClassSummary,savePredictions = TRUE,number = 5)
```

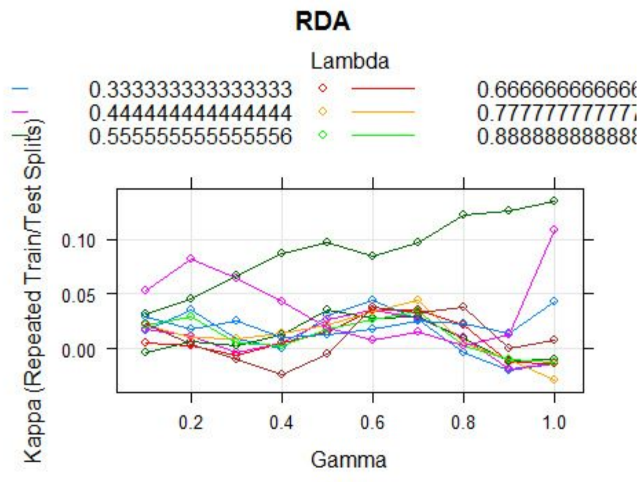
MODEL COMPARISON

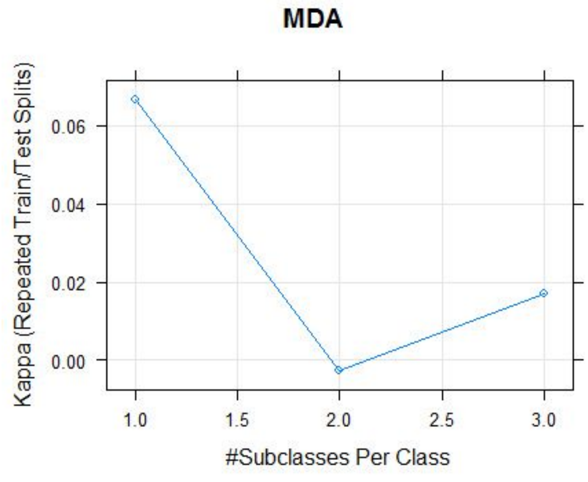

NNET
#Hidden Units
1
2
3
Kappa (Repeated Train/Test Splits)
0.08
0.06
0.04
0.02
0.00
-0.02
0.0
0.5
1.0
1.5
2.0
Weight Decay

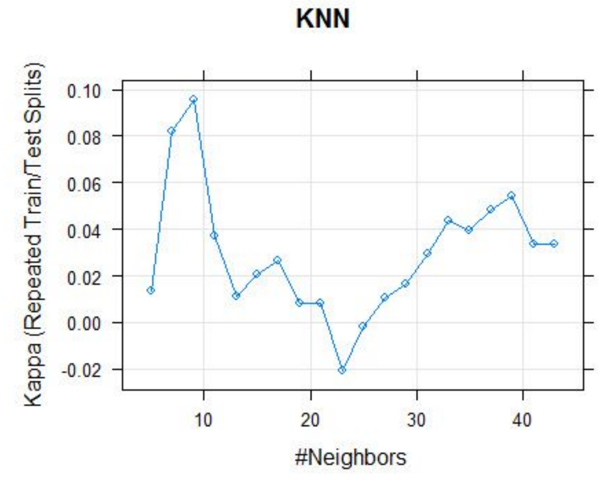

SVM
Kappa (Repeated Train/Test Splits)
0.000
-0.002
-0.004
-0.006
-0.008
-0.010
0
5
10
15
Cost


SVM_POLY
Scale
0.01
0.1
0.5
degree: 4
degree: 2
degree: 3
Kappa (Repeated Train/Test Splits)
0.10
0.05
0.00
-0.05
0
20
40
60
Cost









Name	Tuning Parameters	Pre-Processing	Kappa	Accuracy	AUC
Neural Network	size = 3 and decay = 0.1	Centered Scaled & Spatial sign transformation	0.9069	0.9467	0.8496436
MDA	Subclasses=1	None	0.5794	0.7644	0.7957601

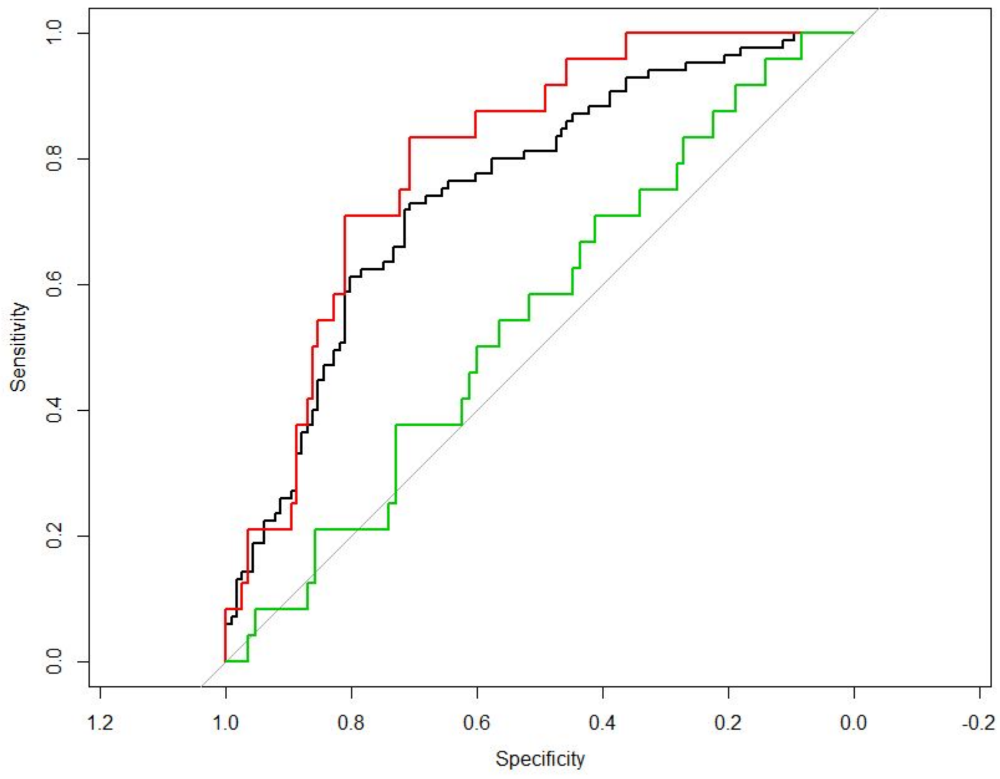
FDA	degree = 1 and nprune = 9	Centered Scaled	0.1141	0.56	0.7916061
RDA	gamma = 1 and lambda = 1	Centered and Scaled	0.3725	0.6444	0.7916061
SVM Radial	sigma = 0.002816175 and C = 0.0625	Centered, Scaled	-0.4911	0.1067	0.9971264
k-Nearest Neighbors	k = 9	Centered Scaled	0.1886	0.5733	0.5317987
Naive Bayes	fL = 0, usekernel = TRUE adjust = 1	None	0.1817	0.3556	0.7916061
SVM Polynomial	degree = 2, scale = 0.1 and C = 64	Centered Scaled	0.1299	0.5689	0.7916061

The three models that can be shortlisted for the Biological predictors are NNET,MDA and RDA. We compare them on basis of their Kappa values since there is a major class imbalance in this dataset. Let us now see the results of the testing sets for these three models.

For Test Sets the results are:

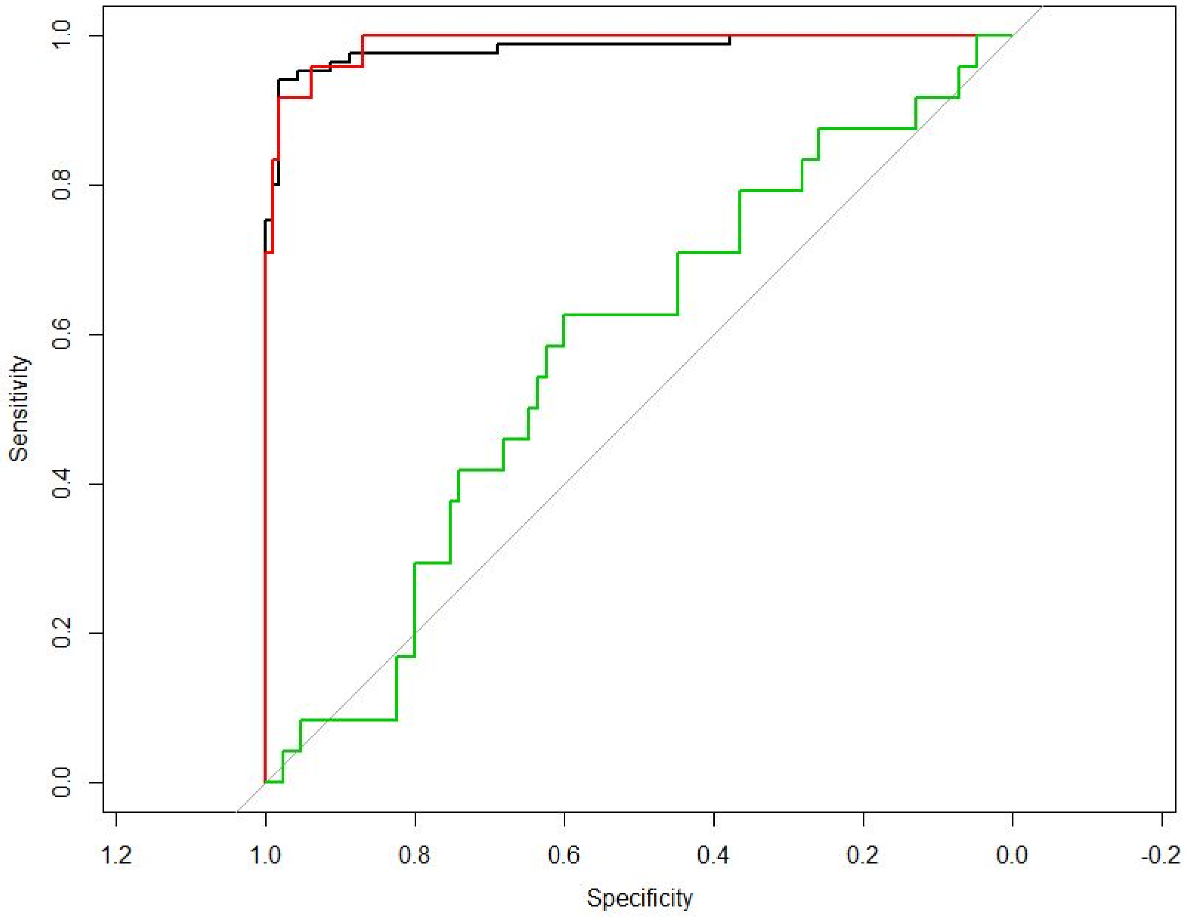
RDA: ACCURACY:0.4286 KAPPA:-0.0217

```
 Reference
Prediction Mild None Severe
    Mild     27   21     5
    None      1    0     1
    Severe    1    0     0
```



NNET ACCURACY:Accuracy : 0.4107 KAPPA:Kappa : -0.1119

```
Prediction Mild None Severe

    Mild     19   18     4

    None      9    3     1

    Severe    1    0     1
```

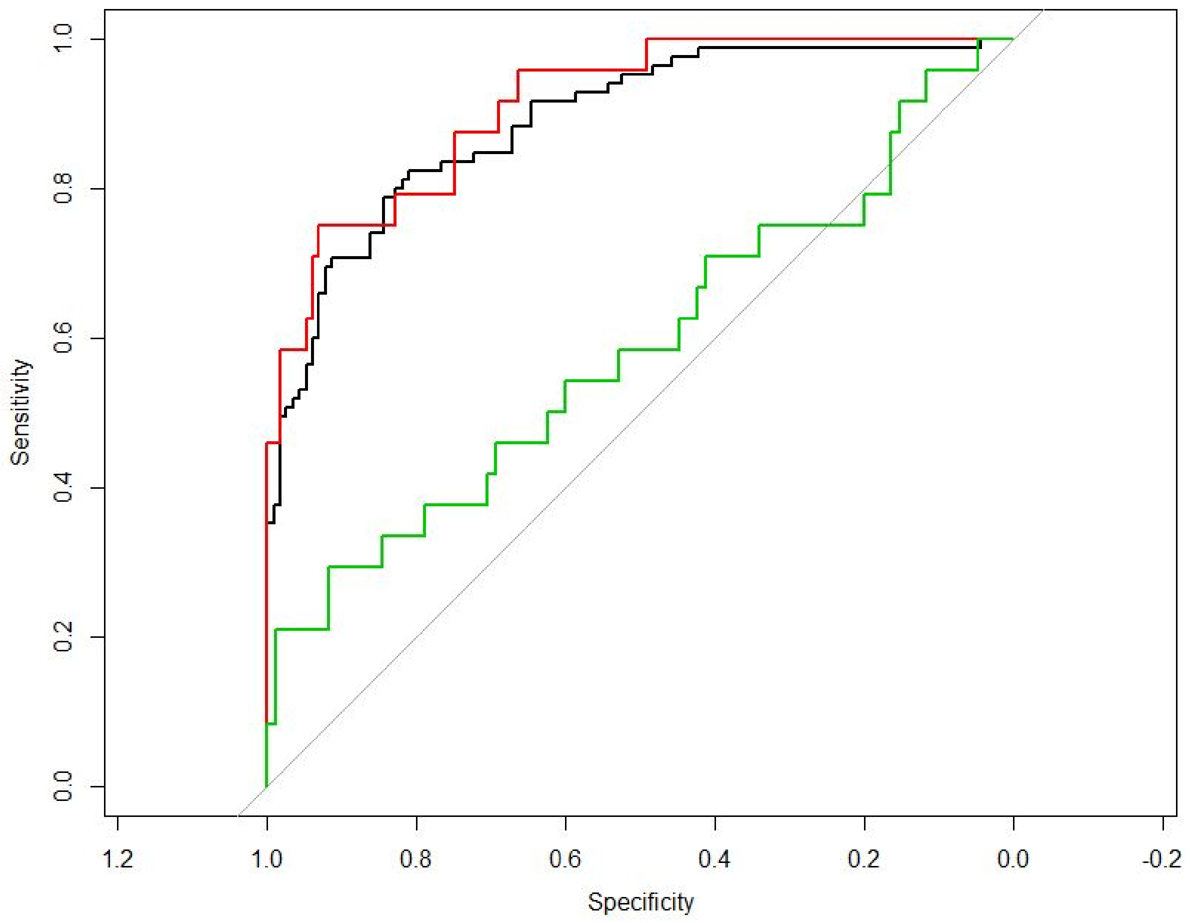


MDA ACCURACY: 0.4643 Kappa : 0.0077

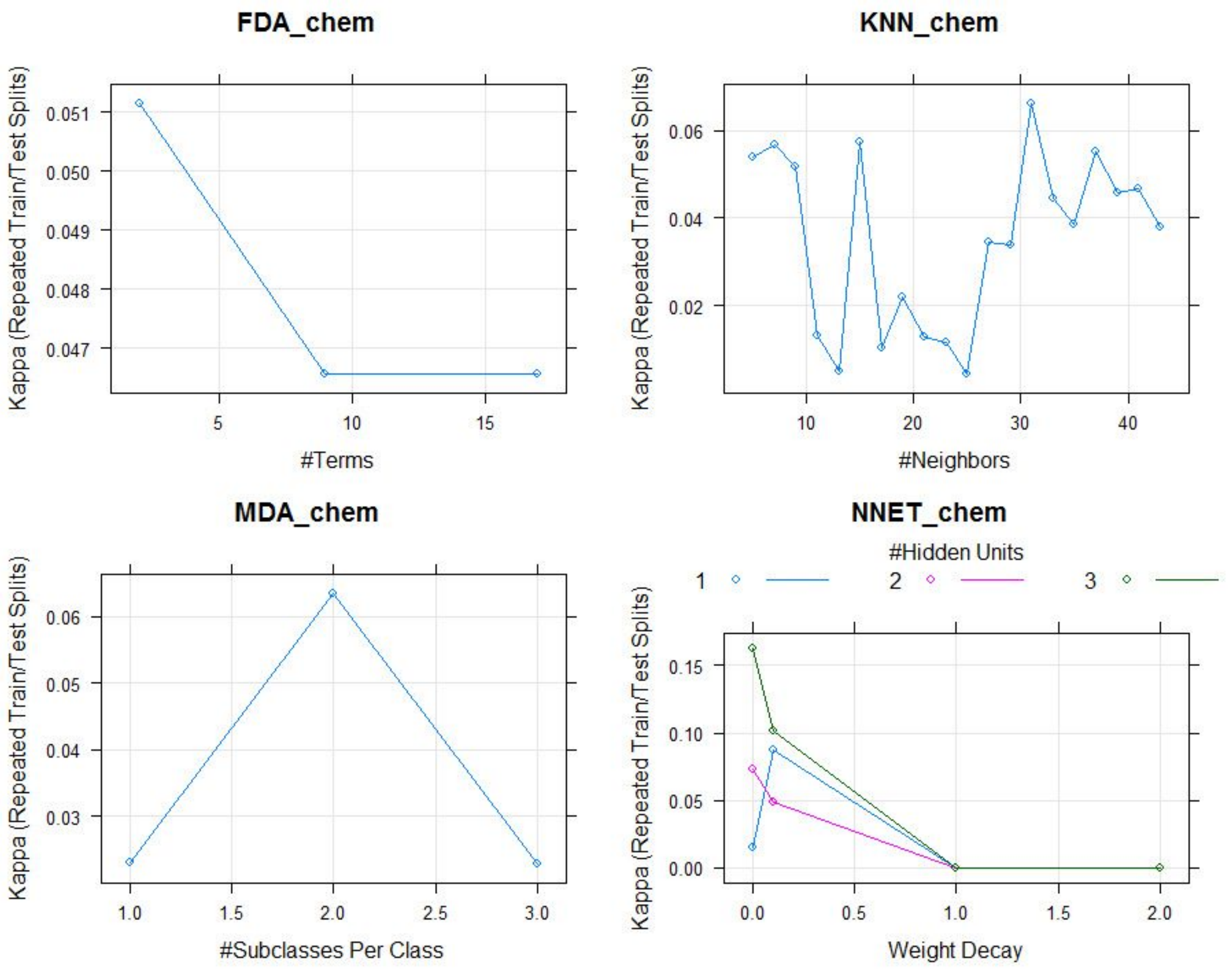
```
Prediction Mild None Severe

    Mild     20   15     4

    None      8    5     1
```

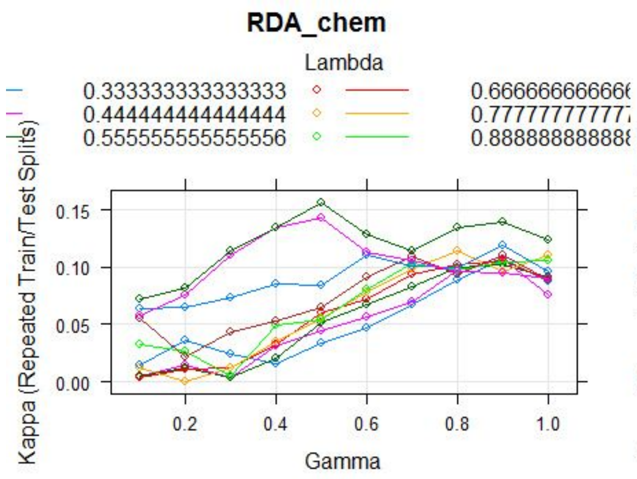
Severe 1 1 1

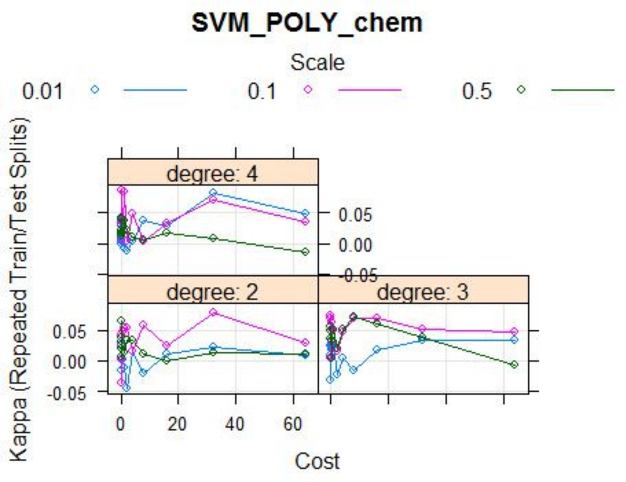
MDA model gives best Kappa and accuracy for Bio dataset.

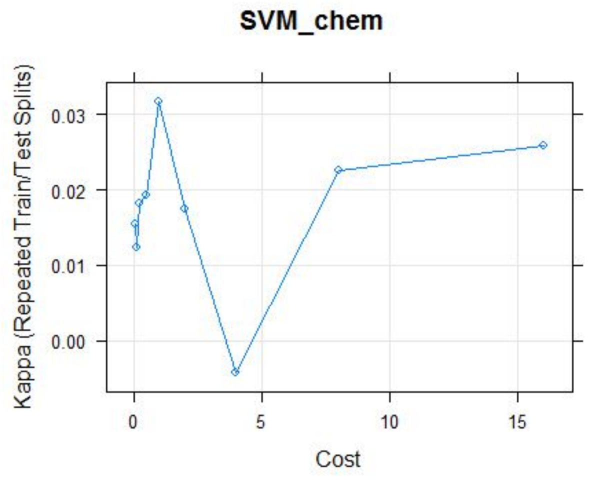


Chemical data:


FDA_chem
Kappa (Repeated Train/Test Splits)
#Terms
KNN_chem
#Neighbors
MDA_chem
#Subclasses Per Class
NNET_chem
#Hidden Units
1
2
3
Weight Decay







Name	Tuning Parameters	Pre-Processing	Kappa	Accuracy	AUC
Neural Network	size = 3 and decay = 0.1	Centered Scaled & Spatial sign transformation	0.7803	0.8667	0.8289892
MDA	Subclasses=2	None	0.4911	0.7644	0.7777989

FDA	degree = 1 and nprune = 9	Centered Scaled	0.1209	0.5511	0.5335193
RDA	gamma = 0.5 and lambda = 1.	Centered and Scaled	0.4876	0.6978	0.769705
SVM Radial	sigma = 0.006110133 and C = 1	Centered, Scaled	0.7927	0.1067	0.955819
k-Nearest Neighbors	k = 31	Centered Scaled	0.1755	0.5511	0.5317987
Naive Bayes	fL = 0, usekernel = TRUE adjust = 1	None	0.2575	0.6267	0.7471067
SVM Polynomial	degree = 4, scale = 0.1 and C = 0.03125	Centered Scaled	0.1299	0.5689	0.8906553

FROM THE ABOVE MODELS ,NNET,SVM_RADIAL AND MDA PERFORM WELL.

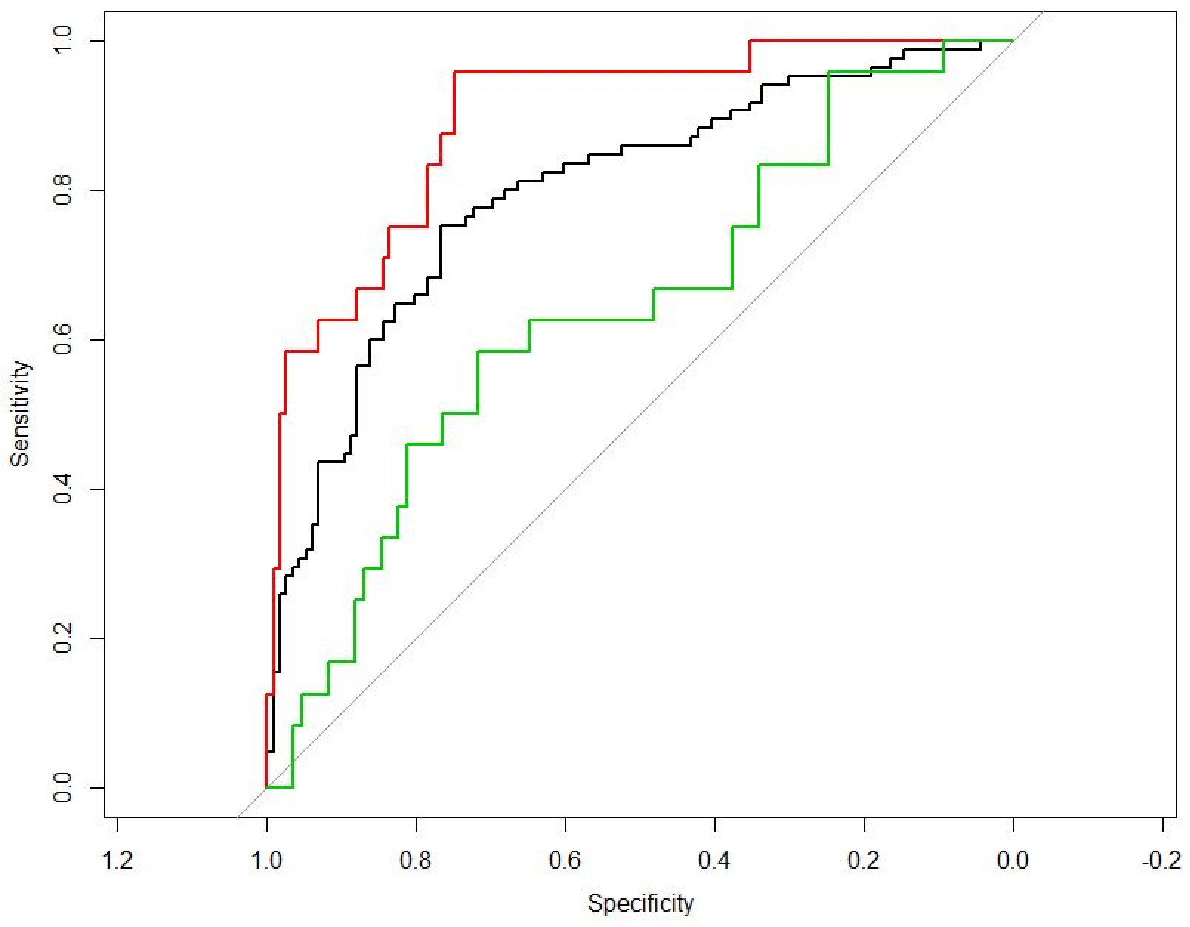
We take kappa metric into consideration as our data set is highly imbalanced. So these three models are chosen based on their kappa values now we test the test set on these models to see if they perform well with the test set.

MDA:

```
Reference
Prediction Mild None Severe
    Mild     20   12      4
    None      6    7      1
    Severe    3    2      1
```

Accuracy : 0.5

 Kappa : 0.1101

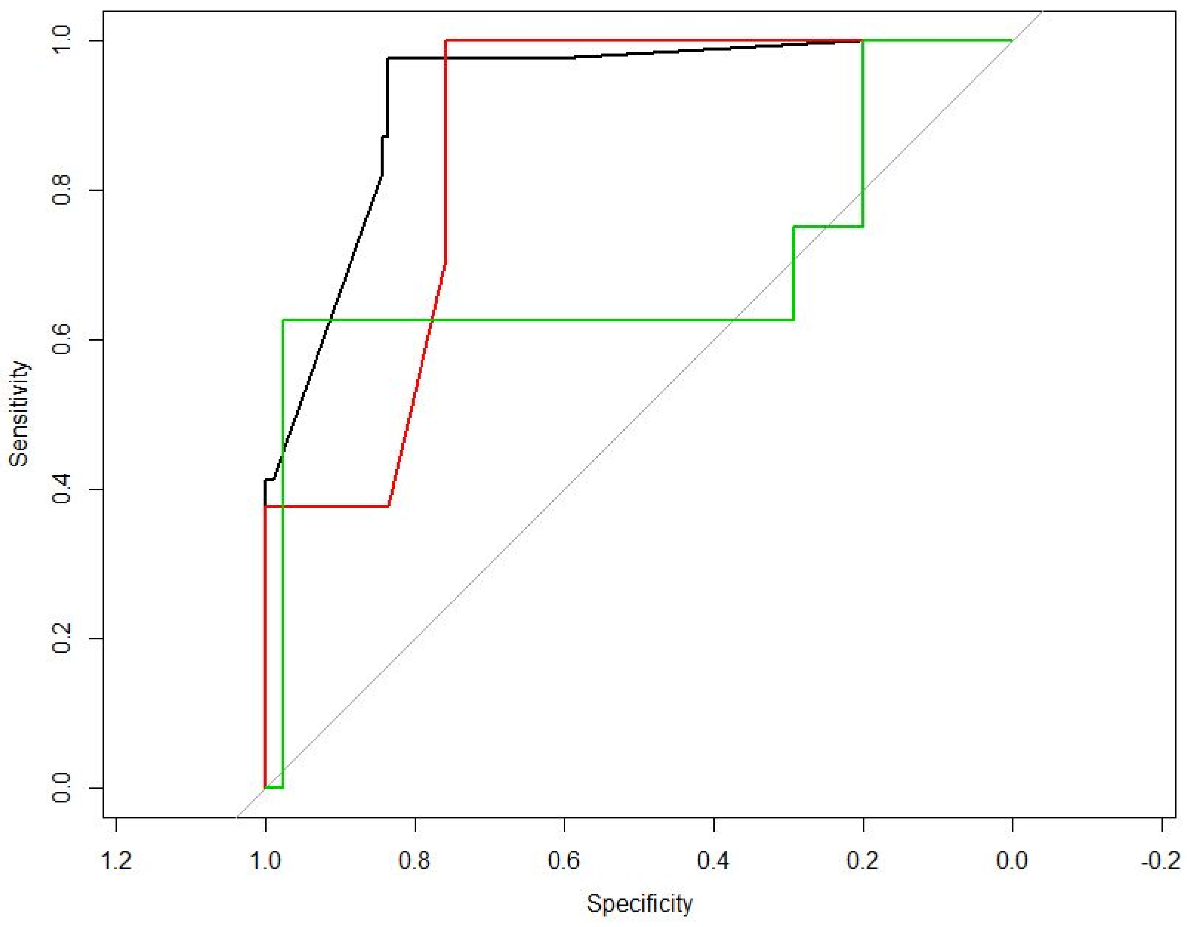


NNET:

```
Prediction Mild None Severe
   Mild     15    5      3
   None      7   13      2
   Severe    7    3      1
```

Accuracy : 0.5179

Kappa : 0.221



SVM-RADIAL

```
Reference

Prediction Mild None Severe

    Mild     28   21     5

    None      1    0     1

    Severe    0    0      0
```

Accuracy : 0.5

Kappa : -0.0262

For Chemical Predictors Neural Nets gives better kappa and accuracy as compared to other models. Hence this is the chosen model for Chemical predictors.

Does the nonlinear structure of these models help to improve the classification performance?

Yes the nonlinear structure certainly helps improve the performance for the bio database since the linear models gave us a best kappa value of **0.191 on the training set however the values for most of the models are higher than that value. The chemical dataset had highest kappa of 0.16 for the linear model however for the non linear models it has a max kappa of 0.79 which is definitely a better performance than the linear model.For the combined model, 0.0913 was the highest Kappa and now for nonlinear models it is** 0.9846.

(b) For the optimal models for both the biological and chemical predictors, what are the top five important predictors?

CHEM:

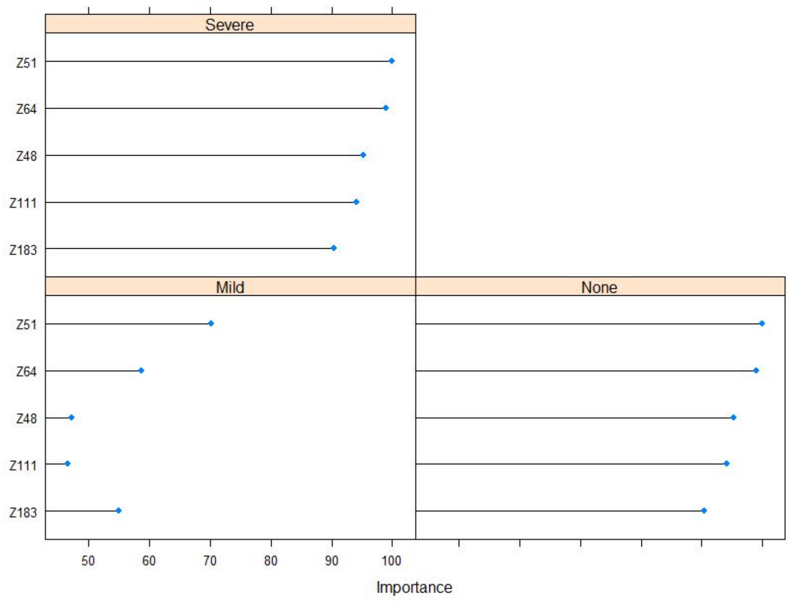
NNETS

	Overall	Mild	None	Severe
X68	100.00	100.00	100.00	100.00
X155	78.27	78.27	78.27	78.27
X143	77.06	77.06	77.06	77.06
X67	76.95	76.95	76.95	76.95
X53	70.47	70.47	70.47	70.47

BIO:

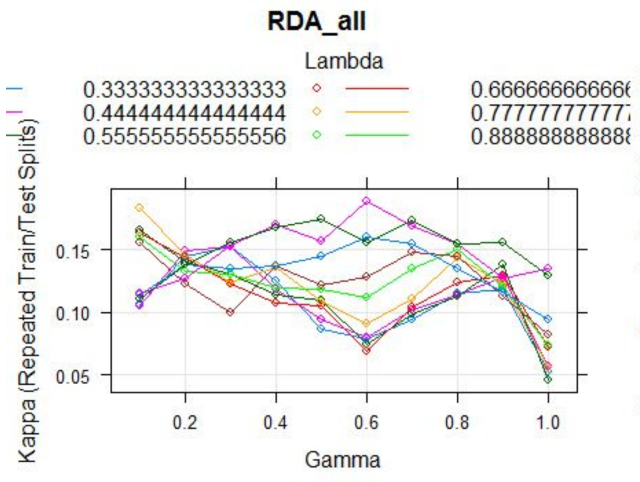
	Mild	None	Severe

```
Z51  70.14 100.00 100.00

Z64  58.74  98.93  98.93

Z48  47.23  95.19  95.19

Z111 46.50  94.12  94.12

Z183 54.99  90.37  90.37
```

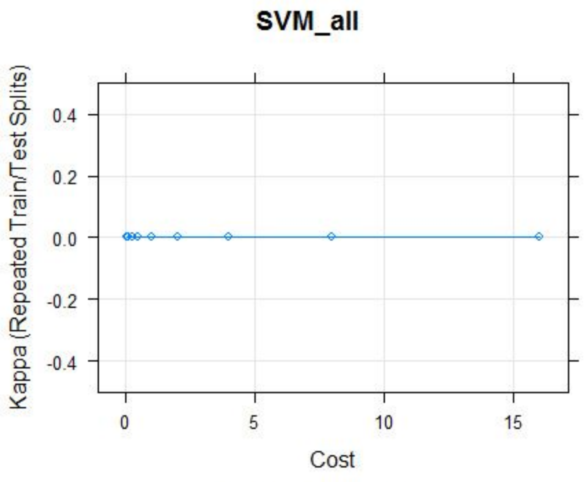


\

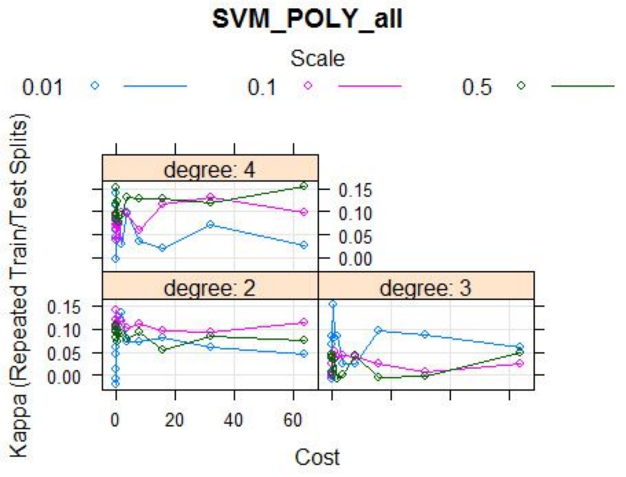
(c) Now combine the biological and chemical fingerprint predictors into one predictor set. Re-train the same set of predictive models you built from part (a).



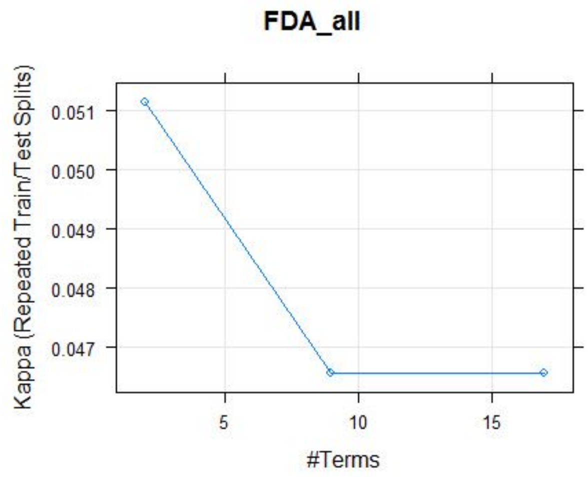
RDA_all
Lambda
0.333333333333333
0.444444444444444
0.555555555555556
0.66666666666
0.77777777777
0.88888888888
Kappa (Repeated Train/Test Splits)
Gamma

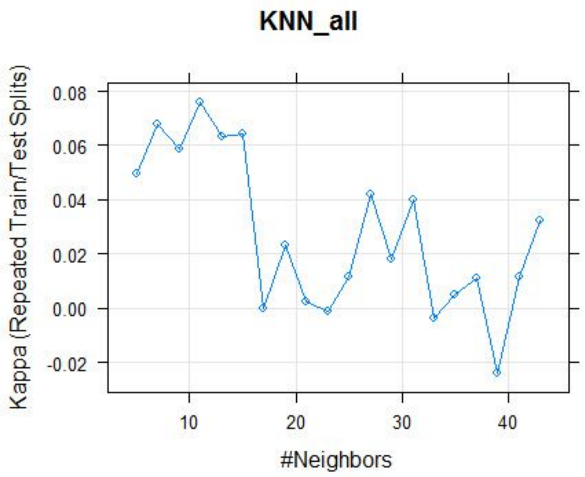


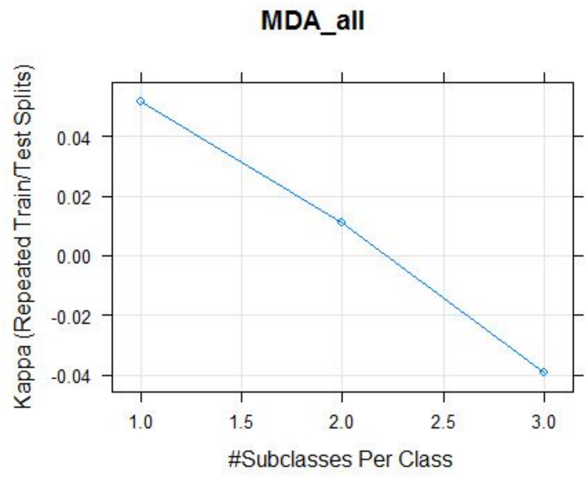
SVM_all
Kappa (Repeated Train/Test Splits)
Cost

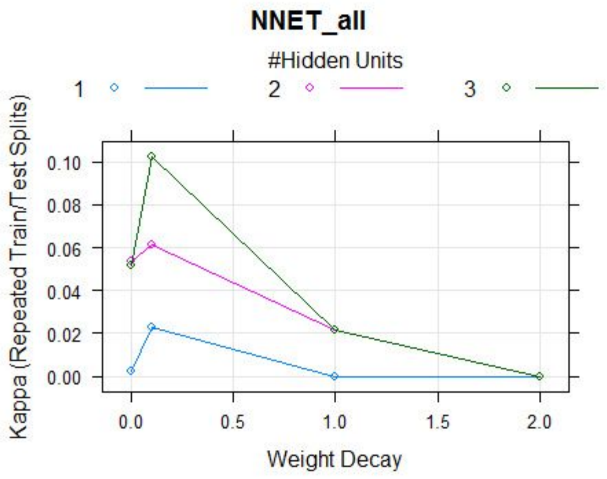


SVM_POLY_all
Scale
0.01
0.1
0.5
degree: 4
degree: 2
degree: 3
Kappa (Repeated Train/Test Splits)
Cost









Name	Tuning Parameters	Pre-Processing	Kappa	Accuracy	AUC
Neural Network	size = 3 and decay = 0.1	Centered Scaled & Spatial sign transformation	0.9846	0.9911	0.8817191
MDA	Subclasses=1	None	0.8282	0.9022	0.8574938

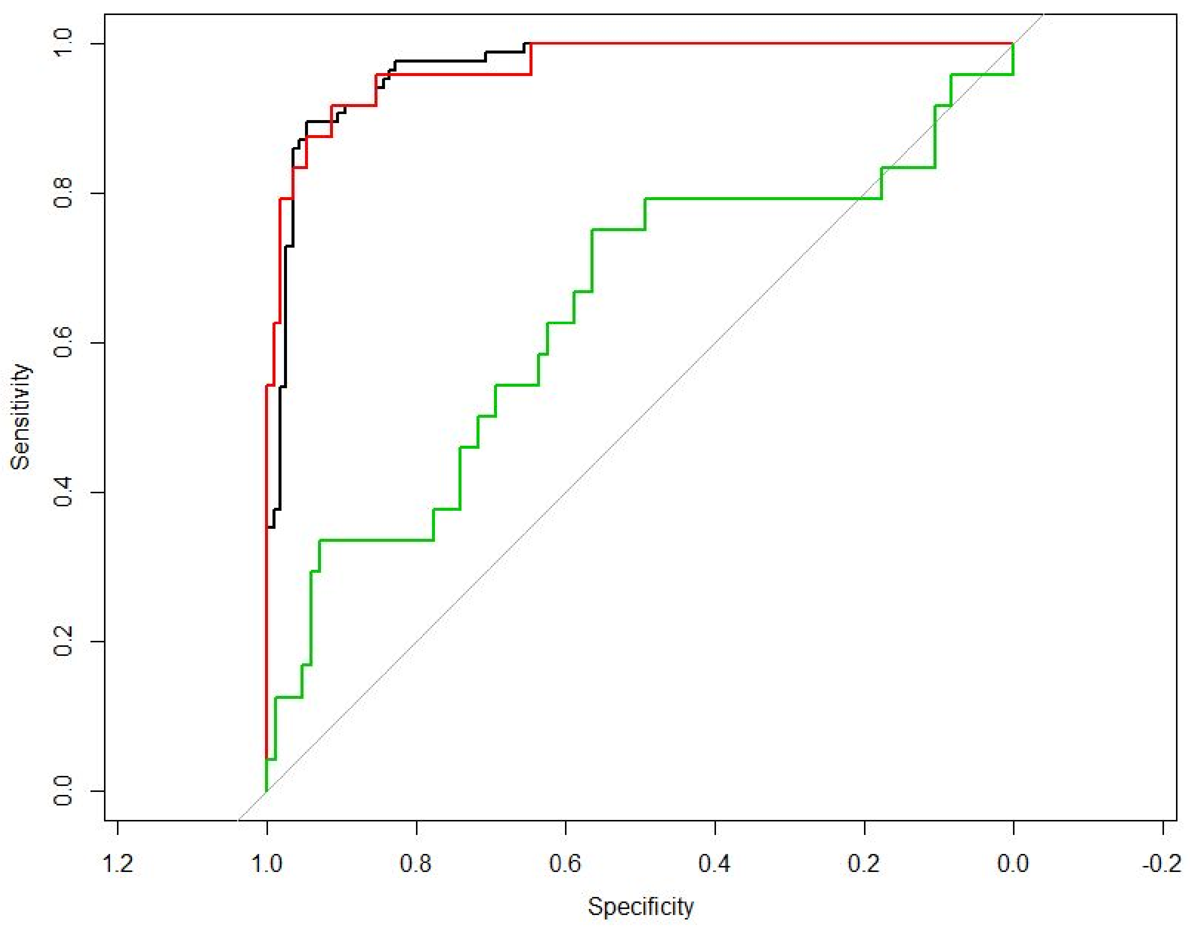
FDA	degree = 1 and nprune = 2	Centered Scaled	0	0.5156	0.4018199
RDA	gamma = 0.6 and lambda = 0.8888889	Centered and Scaled	0.8147	0.8933	0.887724
SVM Radial	sigma = 0.002248074 and C = 0.0625.	Centered, Scaled	Kappa : -0.8003	0	1
k-Nearest Neighbors	k = 11	Centered Scaled	0.2172	0.5911	0.5317987
Naive Bayes	fL = 0, usekernel = TRUE adjust = 1	None	0.5675	0.7644	
SVM Polynomial	degree = 4, scale = 0.5 and C = 64	Centered Scaled	0.8012	0.8933	1

THE BEST MODELS FOR THE COMBINED DATA SETS ARE MDA,NNET,RDA and SVM_polynomial.

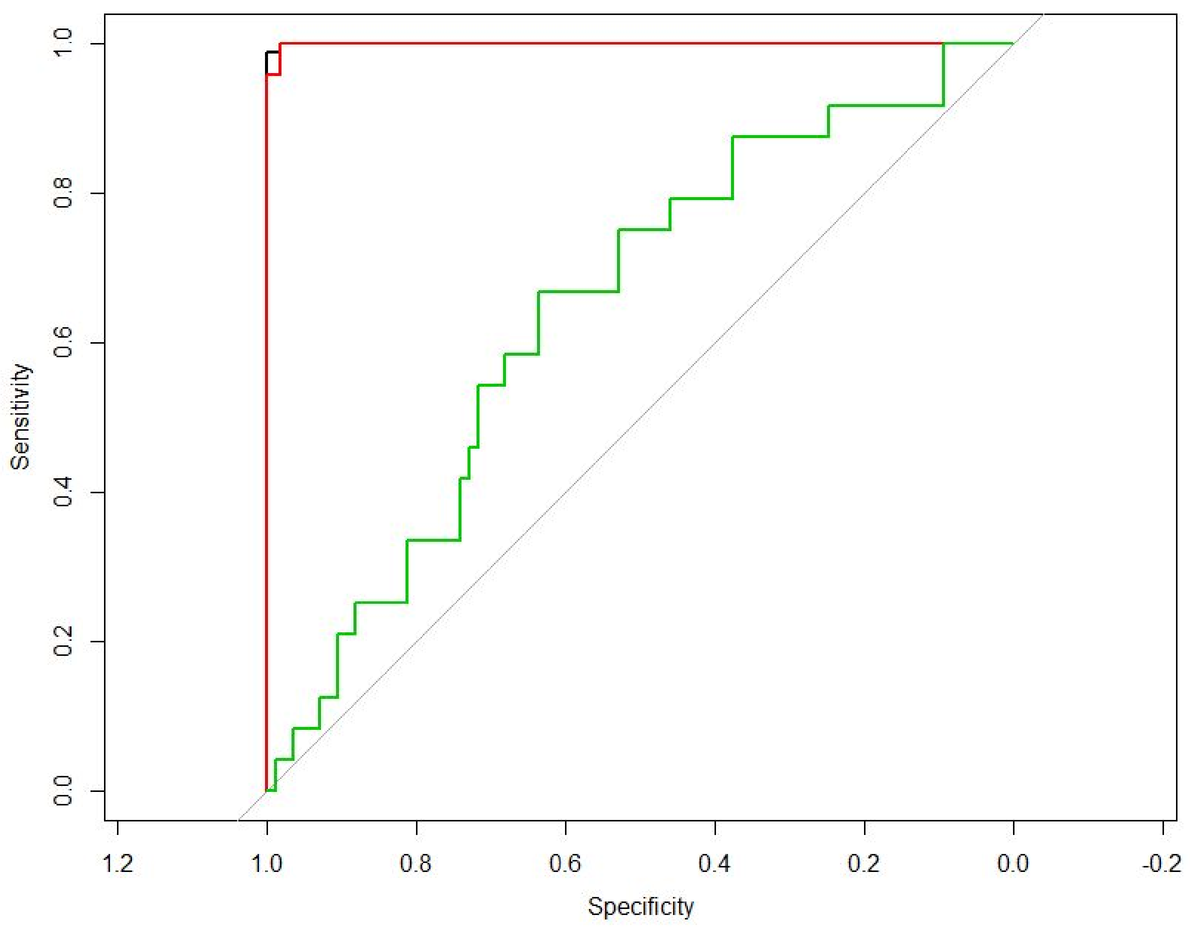
TEST SET:

MDA:

```
Confusion Matrix and Statistics


          Reference

Prediction Mild None Severe

    Mild     17    8      4

    None      8   11      1

    Severe    4    2      1

Accuracy : 0.5179

Kappa : 0.1751
```



```
NNET:

Prediction Mild None Severe

    Mild      18   10      3

    None       6   10      1

    Severe     5    1      2

Accuracy : 0.5357

Kappa : 0.2052
```



SVM

Confusion Matrix and Statistics

```
          Reference
Prediction Mild None Severe
    Mild     27   17      6
    None      2    3      0
    Severe    0    1      0
```

Accuracy : 0.5357

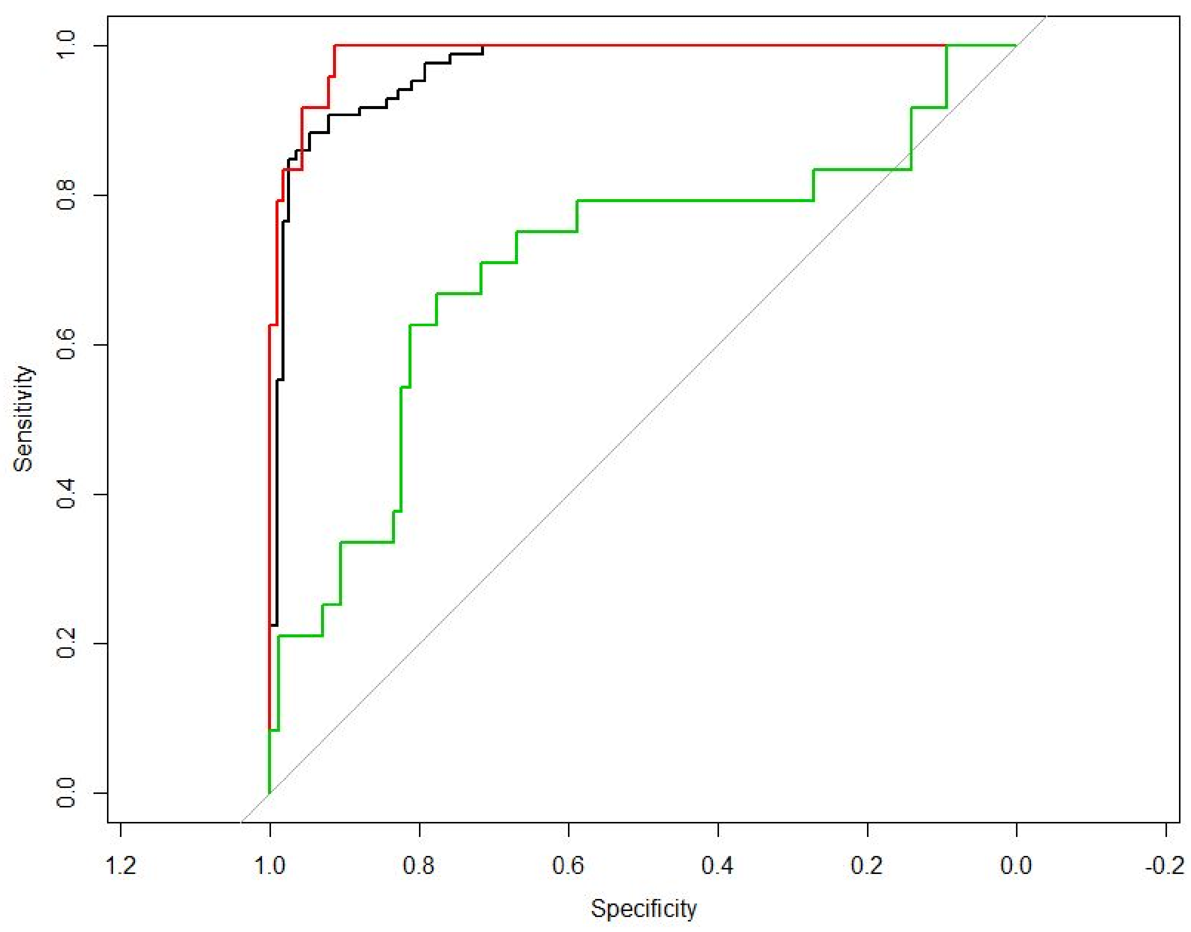
 Kappa : 0.0756

RDA:

```
Reference
Prediction Mild None Severe
    Mild     19    8      3
    None      7    9      1
    Severe    3    4      2
```

Accuracy : 0.5357

Kappa : 0.2251



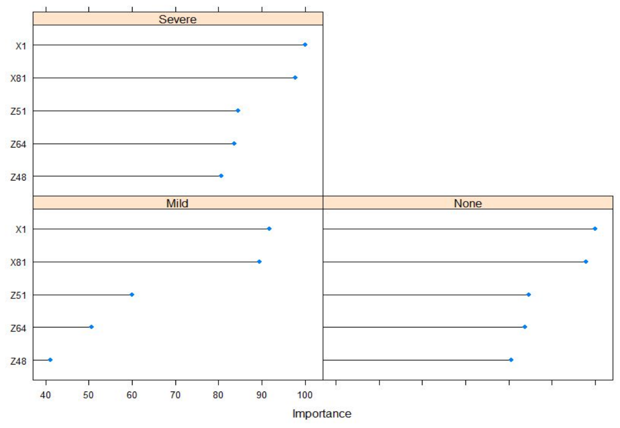
Which model yields best predictive performance?

In this dataset which is the combined data set RDA shows the best performance.Hence RDA can be chosen for this dataset.

Is the model performance better than either of the best models from part (a)?

Yes the performance of the combined model(acc=0.5357, kappa=0.225) is slightly better than the performance of the best model which was NNETS(acc=0.5179,kappa=0.221) for CHEM predictor and is much better than the performance of the best model for bio predictors which was MDA(acc=0.4642,kappa=0.0077).

What are the top 5 important predictors for the optimal model? How do these compare with the optimal predictors from each individual predictor set? How do these important predictors compare the predictors from the linear models?



	Mild	None	Severe
X1	91.70	100.00	100.00
X81	89.46	97.80	97.80
Z51	59.98	84.58	84.58
Z64	50.59	83.70	83.70
Z48	41.11	80.62	80.62

Comparing with the important predictors of the CHEM and BIO models we see that predictors Z48,Z64 and Z51 were an important part of the Bio model however predictors X81 and X1 were not a part of the list for CHEM predictors.Surprisingly the two most important predictors for the combined model were not a part of the important predictors in the CHEM model.

The linear models

X98 ,Z164,X120, X118,Z44 were important in the combined set however there is no overlap between the important predictors for the linear and non linear models.

(d) Which model (either model of individual biology or chemical fingerprints or the combined predictor model), if any, would you recommend using to predict compounds' hepatic toxicity? Explain

I would not recommend any model for using hepatic toxicity. Because many kappa values can be interpreted as follows :

Cohen suggested the Kappa result be interpreted as follows: values ≤ 0 as indicating no agreement and 0.01–0.20 as none to slight, 0.21–0.40 as fair, 0.41– 0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1.00 as almost perfect agreement.

Since this is health related data we cannot recommend a model based on slight agreement on tests sets as it can cause life threatening damage .(https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/)

However if i had to recommend a model from the above three,I would recommend the combined model. As it performs better than the other two models.

Rcode:

```
barplot(table(injury))

injury=as.character(injury)

table( injury) / sum( table( injury ) )

t = createDataPartition( injury, p=0.8)[[1]]

table( injury[t] ) / sum( table( injury[t] ) )


zv_cols = nearZeroVar(bio)

print( sprintf("Dropping %d zero variance columns from %d (fraction=%10.6f)",
length(zv_cols), dim(bio)[2], length(zv_cols)/dim(bio)[2]) );

X = bio[,-zv_cols]
```

```
zv_cols1 = nearZeroVar(chem)

print( sprintf("Dropping %d zero variance columns from %d (fraction=%10.6f)",
length(zv_cols1), dim(chem)[2], length(zv_cols1)/dim(chem)[2]) );

Y = chem[,-zv_cols1]


corr_bio <- cor(X)

corrplot(corr_bio, tl.cex = 0.8)

remove <- findCorrelation(corr_bio,cutoff = 0.75)

X<-X[,-remove]



corr_chem <- cor(Y)

corrplot(corr_chem, tl.cex = 0.5)

remove1 <- findCorrelation(corr_chem,cutoff=0.75)

Y<-Y[,-remove1]


preProc_bio <- preProcess(X, method = c("center", "scale", "BoxCox"))

X <- predict(preProc_bio, X)


preProc_chem <- preProcess(Y, method = c("center", "scale", "BoxCox"))

Y<- predict(preProc_chem, Y)
```

```
training_bio <- X[t, ]

test_bio <- X[-t, ]



training_chem <- Y[t, ]

test_chem <- Y[-t, ]

#training_bio$Y<-as.factor(injury[t])

#training_chem$Y<-as.factor(injury[t])


set.seed(1)

ctrl = trainControl(method="LGOCV",classProbs = TRUE,summaryFunction =
multiClassSummary,savePredictions = TRUE,number = 5)




#

if( build_mda_model ){

  set.seed(1)

mda.classifier = train( training_bio,injury[t], method="mda",
tuneGrid=expand.grid(subclasses=1:3), metric="Kappa", trControl=ctrl )

  mda.predictions = predict( mda.classifier, training_bio,  )

  mda.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=mda.predictions[,1] )

  mda.auc = mda.rocCurve$auc[1]

  mda=list( classifier=mda.classifier, roc=mda.rocCurve, auc=mda.auc )
}


# Neural Networks:

#

set.seed(1)

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

nnet.classifier1 = train( training_bio,injury[t], method="nnet",
preProc=c("center","scale","spatialSign"), tuneGrid=nnetGrid, metric="Kappa", trace=FALSE,
maxit=2000, trControl=ctrl )

nnet.predictions1 = predict( nnet.classifier1, training_bio  ) # <- returns probability of "Yes"
(event of interest) & "No"

nnet.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=nnet.predictions[,1] )

nnet.auc = nnet.rocCurve$auc[1]

nnet=list( classifier=nnet.classifier, roc=nnet.rocCurve, auc=nnet.auc )


# Support Vector Machines:

#

set.seed(1)

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

svarid = expand.grid(sigma=sigmaEst[1], C=2^seq(-4,+4))

svm.classifier = train( training_bio,injury[t], method="svmRadial", tuneGrid=svarid,
preProc=c("center","scale"), metric="Kappa", fit=FALSE, trControl=ctrl )

svm.predictions = predict( svm.classifier, training_bio,  )

svm.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=svm.predictions[,1] )
svm.auc = svm.rocCurve$auc[1]

svm=list( classifier=svm.classifier, roc=svm.rocCurve, auc=svm.auc )


# K-Nearest Neighbors:

#

set.seed(1)

knn.classifier = train( training_bio,injury[t], method="knn", tuneLength=20,
preProc=c("center","scale"), metric="Kappa", trControl=ctrl )

knn.predictions = predict( knn.classifier, training_bio,  )

knn.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=knn.predictions[,1] )

knn.auc = knn.rocCurve$auc[1]

knn=list( classifier=knn.classifier, roc=knn.rocCurve, auc=knn.auc )


# Naive Bayes:

#

set.seed(1)

nb.classifier = train( training_bio,injury[t], method="nb", metric="Kappa", trControl=ctrl )

nb.predictions = predict( nb.classifier, training_bio,  )

nb.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=nb.predictions[,1] )

nb.auc = nb.rocCurve$auc[1]

nb=list( classifier=nb.classifier, roc=nb.rocCurve, auc=nb.auc )


##FDA
```

```
set.seed(1)

fda.classifier = train( training_bio,injury[t], method="fda",preProc=c("center","scale"),
metric="Kappa", trControl=ctrl )

fda.predictions = predict( fda.classifier, training_bio )

fda.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=fda.predictions[,1] )

fda.auc = nb.rocCurve$auc[1]

fda=list( classifier=fda.classifier, roc=nb.rocCurve, auc=nb.auc )



##RDA

set.seed(1)

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

rda.classifier = train( training_bio,injury[t],
method="rda",preProc=c("center","scale"),tuneGrid=grid, metric="Kappa", trControl=ctrl )

rda.predictions = predict( my_model_RDA_bio, training_bio)

rda.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=rda.predictions[,1] )

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

rda=list( classifier=my_model_RDA_bio, roc=nb.rocCurve, auc=nb.auc )



result = list( nnet=nnet, svm=svm, knn=knn, nb=nb )

if( build_mda_model ){ result = c(result, list(mda=mda)) }

return( result )
```

```
}


##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))

svm1.classifier <- train(training_bio, y=injury[t], method="svmPoly", metric="Kappa",
trControl=ctrl,

                preProc=c('center', 'scale'), fit=FALSE, tuneGrid = poly_grid)

svm1.predictions = predict( svm1.classifier, training_bio)

svm1.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=rda.predictions[,1] )

svm1.auc = nb.rocCurve$auc[1]

svm1=list( classifier=my_model_RDA_bio, roc=nb.rocCurve, auc=nb.auc )


par(mfrow=c(1,3))

plot(mda.classifier,main="MDA")

plot(rda.classifier,main="RDA")

plot(fda.classifier,main="FDA")

plot(nnet.classifier1,main="NNET")

plot(svm.classifier,main="SVM")

plot(svm1.classifier,main="SVM_POLY")

plot(knn.classifier,main="KNN")
plot(varImp(mda.classifier),top=5)




barplot(table(injury))

injury=as.character(injury)

table( injury) / sum( table( injury ) )

t = createDataPartition( injury, p=0.8)[[1]]

table( injury[t] ) / sum( table( injury[t] ) )


zv_cols = nearZeroVar(bio)

print( sprintf("Dropping %d zero variance columns from %d (fraction=%10.6f)",
length(zv_cols), dim(bio)[2], length(zv_cols)/dim(bio)[2]) );

X = bio[,-zv_cols]


zv_cols1 = nearZeroVar(chem)

print( sprintf("Dropping %d zero variance columns from %d (fraction=%10.6f)",
length(zv_cols1), dim(chem)[2], length(zv_cols1)/dim(chem)[2]) );

Y = chem[,-zv_cols1]


corr_bio <- cor(X)

corrplot(corr_bio, tl.cex = 0.8)

remove <- findCorrelation(corr_bio,cutoff = 0.75)

X<-X[,-remove]
```

```
corr_chem <- cor(Y)

corrplot(corr_chem, tl.cex = 0.5)

remove1 <- findCorrelation(corr_chem,cutoff=0.75)

Y<-Y[,-remove1]


preProc_bio <- preProcess(X, method = c("center", "scale", "BoxCox"))

X <- predict(preProc_bio, X)


preProc_chem <- preProcess(Y, method = c("center", "scale", "BoxCox"))

Y<- predict(preProc_chem, Y)


training_chem, <- X[t, ]

test_bio <- X[-t, ]




training_chem <- Y[t, ]

test_chem <- Y[-t, ]

#training_chem,$Y<-as.factor(injury[t])

#training_chem$Y<-as.factor(injury[t])


set.seed(1)
```

```
ctrl = trainControl(method="LGOCV",classProbs = TRUE,summaryFunction =
multiClassSummary,savePredictions = TRUE,number = 5)




#

if( build_mda_model ){

  set.seed(1)

  mda.classifier.chem = train( training_chem,injury[t], method="mda",
tuneGrid=expand.grid(subclasses=1:3), metric="Kappa", trControl=ctrl )

  mda.predictions.chem = predict( mda.classifier.chem, training_chem)

  mda.rocCurve.chem = pROC::multiclass.roc( response=injury[t],
predictor=mda.predictions.chem[,1] )

  mda.auc.chem = mda.rocCurve.chem$auc[1]

  mda=list( classifier=mda.classifier.chem, roc=mda.rocCurve, auc=mda.auc )

}


# Neural Networks:

#

set.seed(1)

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

nnet.classifier.chem = train( training_chem,injury[t], method="nnet",
preProc=c("center","scale","spatialSign"), tuneGrid=nnetGrid, metric="Kappa", trace=FALSE,
maxit=2000, trControl=ctrl )
```

```
nnet.predictions.chem = predict( nnet.classifier.chem, training_chem) # <- returns probability of
"Yes" (event of interest) & "No"

nnet.rocCurve.chem = pROC::multiclass.roc( response=injury[t],
predictor=nnet.predictions.chem[,1] )

nnet.auc.chem = nnet.rocCurve.chem$auc[1]

nnet=list( classifier=nnet.classifier.chem, roc=nnet.rocCurve.chem, auc=nnet.auc.chem )


# Support Vector Machines:

#

set.seed(1)

sigmaEst = kernlab::sigest( as.matrix(training_chem,) )

svarid = expand.grid(sigma=sigmaEst[1], C=2^seq(-4,+4))

svm.classifier.chem = train( training_chem,injury[t], method="svmRadial", tuneGrid=svarid,
preProc=c("center","scale"), metric="Kappa", fit=FALSE, trControl=ctrl )

svm.predictions.chem = predict( svm.classifier.chem, training_chem,type='prob')

svm.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=svm.predictions.chem[,1] )

svm.auc.chem= svm.rocCurve$auc[1]

svm=list( classifier=svm.classifier.chem, roc=svm.rocCurve, auc=svm.auc )


# K-Nearest Neighbors:

#

set.seed(1)

knn.classifier.chem = train( training_chem,injury[t], method="knn", tuneLength=20,
preProc=c("center","scale"), metric="Kappa", trControl=ctrl )
```

```
knn.predictions.chem = predict( knn.classifier.chem, training_chem,,  )

knn.rocCurve.chem = pROC::multiclass.roc( response=injury[t],
predictor=knn.predictions.chem[,1] )

knn.auc.chem = knn.rocCurve$auc[1]

knn=list( classifier=knn.classifier.chem, roc=knn.rocCurve, auc=knn.auc )


# Naive Bayes:

#

set.seed(1)

nb.classifier.chem = train( training_chem,injury[t], method="nb", metric="Kappa",
trControl=ctrl )

nb.predictions.chem = predict( nb.classifier.chem, training_chem,type='prob')

nb.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=nb.predictions.chem[,1] )

nb.auc.chem = nb.rocCurve$auc[1]

nb=list( classifier=nb.classifier.chem, roc=nb.rocCurve, auc=nb.auc )


##FDA

set.seed(1)

fda.classifier.chem = train( training_chem,injury[t], method="fda",preProc=c("center","scale"),
metric="Kappa", trControl=ctrl )

fda.predictions.chem = predict( fda.classifier.chem, training_chem,type="prob" )

fda.rocCurve.chem = pROC::multiclass.roc( response=injury[t],
predictor=fda.predictions.chem[,1] )

fda.auc.chem = fda.rocCurve.chem$auc[1]
```

```
fda=list( classifier=fda.classifier.chem, roc=nb.rocCurve, auc=nb.auc )



##RDA

set.seed(1)

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

rda.classifier.chem = train( training_chem,injury[t],
method="rda",preProc=c("center","scale"),tuneGrid=grid, metric="Kappa", trControl=ctrl )

rda.predictions.chem = predict( rda.classifier.chem, training_chem,type='prob')

rda.rocCurve.chem = pROC::multiclass.roc( response=injury[t],
predictor=rda.predictions.chem[,1] )

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

rda=list( classifier=my_model_RDA_bio, roc=nb.rocCurve, auc=nb.auc )



result = list( nnet=nnet, svm=svm, knn=knn, nb=nb )

if( build_mda_model ){ result = c(result, list(mda=mda)) }

return( result )


}


##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))
```

```
svm1.classifier.chem <- train(training_chem, y=injury[t], method="svmPoly", metric="Kappa",
trControl=ctrl,

                    preProc=c('center', 'scale'), fit=FALSE, tuneGrid = poly_grid)

svm1.predictions_chem = predict( svm1.classifier.chem, training_chem,type='prob')

svm1.rocCurve = pROC::multiclass.roc( response=injury[t],
predictor=svm1.predictions_chem[,1] )

svm1.auc.chem = svm1.rocCurve$auc[1]

svm1=list( classifier=my_model_RDA_bio, roc=nb.rocCurve, auc=nb.auc )





nnet.chem=confusionMatrix(data=nnet.predictions.chem,reference=injury[t])


mda.chem=confusionMatrix(data=mda.predictions.chem,reference=injury[t])


fda.chem=confusionMatrix(data=fda.predictions.chem,reference=injury[t])


rda.chem=confusionMatrix(data=rda.predictions.chem,reference=injury[t])


knn.chem=confusionMatrix(data=knn.predictions.chem,reference=injury[t])


nb.chem=confusionMatrix(data=nb.predictions.chem,reference=injury[t])
```

```
svm.chem=confusionMatrix(data=svm.predictions.chem,reference=injury[t])


svm1.chem=confusionMatrix(data=svm1.predictions.chem,reference=injury[t])



##TEST
mda.predictions.chem.test = predict( mda.classifier.chem, test_chem)
mda.chem.test=confusionMatrix(data=mda.predictions.chem.test,reference=injury[-t])
mda.chem.test

nnet.predictions.chem.test = predict( nnet.classifier.chem, test_chem)
nnet.chem.test=confusionMatrix(data=nnet.predictions.chem.test,reference=injury[-t])
nnet.chem.test

svm.predictions.chem.test = predict( svm.classifier.chem, test_chem)
svm.chem.test=confusionMatrix(data=svm.predictions.chem.test,reference=injury[-t])
svm.chem.test



plot(mda.classifier.chem,main="MDA_chem")
plot(rda.classifier.chem,main="RDA_chem")
plot(fda.classifier,main="FDA_chem")
plot(nnet.classifier.chem,main="NNET_chem")
plot(svm.classifier.chem,main="SVM_chem")

plot(svm1.classifier.chem,main="SVM_POLY_chem")

plot(knn.classifier.chem,main="KNN_chem")



plot(varImp(nnet.classifier.chem),top=10)


train_all<-cbind(training_bio,train_all)

test_all<-cbind(test_bio,test_all)




#

if( build_mda_model ){

  set.seed(1)

  mda.classifier.all= train( train_all,injury[t], method="mda",
tuneGrid=expand.grid(subclasses=1:3), metric="Kappa", trControl=ctrl )

  mda.predictions_all = predict( mda.classifier.all, train_all)

  mda.rocCurve.all = pROC::multiclass.roc( response=injury[t],
predictor=mda.predictions_all[,1] )

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

  mda=list( classifier=mda.classifier.all, roc=mda.rocCurve, auc=mda.auc )

}
```

```
# Neural Networks:

#

set.seed(1)

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

nnet.classifier.all= train( train_all,injury[t], method="nnet",
preProc=c("center","scale","spatialSign"), tuneGrid=nnetGrid, metric="Kappa", trace=FALSE,
maxit=2000, trControl=ctrl )

nnet.predictions_all = predict( nnet.classifier.all, train_all) # <- returns probability of "Yes"
(event of interest) & "No"

nnet.rocCurve.all = pROC::multiclass.roc( response=injury[t], predictor=nnet.predictions_all[,1]
)

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

nnet=list( classifier=nnet.classifier.all, roc=nnet.rocCurve.all, auc=nnet.auc.all )


# Support Vector Machines:

#

set.seed(1)

sigmaEst = kernlab::sigest( as.matrix(train_all,) )

svarid = expand.grid(sigma=sigmaEst[1], C=2^seq(-4,+4))

svm.classifier.all= train( train_all,injury[t], method="svmRadial", tuneGrid=svarid,
preProc=c("center","scale"), metric="Kappa", fit=FALSE, trControl=ctrl )

svm.predictions_all = predict( svm.classifier.all, train_all,type='prob')

svm.rocCurve.all = pROC::multiclass.roc( response=injury[t], predictor=svm.predictions_all[,1]
)

svm.auc.all= svm.rocCurve$auc[1]
```

```
svm=list( classifier=svm.classifier.all, roc=svm.rocCurve, auc=svm.auc )


# K-Nearest Neighbors:

#

set.seed(1)

knn.classifier.all= train( train_all,injury[t], method="knn", tuneLength=20,
preProc=c("center","scale"), metric="Kappa", trControl=ctrl )

knn.predictions_all = predict( knn.classifier.all, train_all, )

knn.rocCurve.all = pROC::multiclass.roc( response=injury[t], predictor=knn.predictions_all[,1] )

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

knn=list( classifier=knn.classifier.all, roc=knn.rocCurve, auc=knn.auc )


# Naive Bayes:

#

set.seed(1)

nb.classifier.all= train( train_all,injury[t], method="nb", metric="Kappa", trControl=ctrl )

nb.predictions_all = predict( nb.classifier.all, train_all)

nb.rocCurve = pROC::multiclass.roc( response=injury[t], predictor=nb.predictions_all[,1] )

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

nb=list( classifier=nb.classifier.all, roc=nb.rocCurve, auc=nb.auc )


##FDA

set.seed(1)
```

```
fda.classifier.all= train( train_all,injury[t], method="fda",preProc=c("center","scale"),
metric="Kappa", trControl=ctrl )

fda.predictions_all = predict( fda.classifier.all, train_all)

fda.rocCurve.all = pROC::multiclass.roc( response=injury[t], predictor=fda.predictions_all[,1] )

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

fda=list( classifier=fda.classifier.all, roc=nb.rocCurve, auc=nb.auc )



##RDA

set.seed(1)

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

rda.classifier.all= train( train_all,injury[t],
method="rda",preProc=c("center","scale"),tuneGrid=grid, metric="Kappa", trControl=ctrl )

rda.predictions_all = predict( rda.classifier.all, train_all,type='prob')

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

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

rda=list( classifier=my_model_RDA_bio, roc=nb.rocCurve, auc=nb.auc )



result = list( nnet=nnet, svm=svm, knn=knn, nb=nb )

if( build_mda_model ){ result = c(result, list(mda=mda)) }

return( result )


}
```

```
##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))

svm1.classifier.all<- train(train_all, y=injury[t], method="svmPoly", metric="Kappa",
trControl=ctrl,

                        preProc=c('center', 'scale'), fit=FALSE, tuneGrid = poly_grid)

svm1.predictions_all = predict( svm1.classifier.all, train_all,type="prob")

svm1.rocCurve.all = pROC::multiclass.roc( response=injury[t],
predictor=svm1.predictions_all[,1] )

svm1.auc.all = svm1.rocCurve$auc[1]

svm1=list( classifier=my_model_RDA_bio, roc=nb.rocCurve, auc=nb.auc )




nnet.all=confusionMatrix(data=nnet.predictions_all,reference=injury[t])


mda.all=confusionMatrix(data=mda.predictions_all,reference=injury[t])


fda.all=confusionMatrix(data=fda.predictions_all,reference=injury[t])


rda.all=confusionMatrix(data=rda.predictions_all,reference=injury[t])
```

```
knn.all=confusionMatrix(data=knn.predictions_all,reference=injury[t])

nb.all=confusionMatrix(data=nb.predictions_all,reference=injury[t])

svm.all=confusionMatrix(data=svm.predictions_all,reference=injury[t])

svm1.all=confusionMatrix(data=svm1.predictions_all,reference=injury[t])


##TEST
mda.predictions_all.test = predict( mda.classifier.all, test_all)
mda.all.test=confusionMatrix(data=mda.predictions_all.test,reference=injury[-t])
mda.all.test

nnet.predictions_all.test = predict( nnet.classifier.all, test_all)
nnet.all.test=confusionMatrix(data=nnet.predictions_all.test,reference=injury[-t])
nnet.all.test

svm1.predictions_all.test = predict( svm1.classifier.all, test_all)
svm1.all.test=confusionMatrix(data=svm1.predictions_all.test,reference=injury[-t])
svm1.all.test

rda.predictions_all.test = predict( rda.classifier.all, test_all)
rda.all.test=confusionMatrix(data=rda.predictions_all.test,reference=injury[-t])

rda.all.test



nb.predictions_all.test = predict( nb.classifier.all, test_all)

nb.all.test=confusionMatrix(data=nb.predictions_all.test,reference=injury[-t])

nb.all.test



plot(mda.classifier.all,main="MDA_all")

plot(rda.classifier.all,main="RDA_all")

plot(fda.classifier,main="FDA_all")

plot(nnet.classifier.all,main="NNET_all")

plot(svm.classifier.all,main="SVM_all")

plot(svm1.classifier.all,main="SVM_POLY_all")

plot(knn.classifier.all,main="KNN_all")



Var.Imp <- varImp(rda.classifier.all, scale = T)

plot(varImp(rda.classifier.all), top = 5)
```