Australian Credit Card Approval Project

Arnab Paul, Diellza Malazogu

MA 5790

December 2020

Abstract:

Credit Cards have been a perennial part of most of our lives. Determination of who should get credit cards based on various criterias is important for a bank as well as the bank customers. Banks and financial institutions use various criterias like Social Security records, driving record And previous financial information to determine if credit is approved to a customer.

Our goal in this project is to find a good our model based on the predictors (or criterias) provided to us and determine credit worthiness, and check how our model performs when compared to the actual credit approval data.

TABLE OF CONTENTS

Background

The data set that was used on this report is about Australian Credit Card Approval. This data set is obtained from "UCI Machine Learning Repository".
All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. This dataset has a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values.

Variable Introduction and Definitions

The sample size of this data set is 690. The target variable is categorical; 1 Accepted, 2 Rejected Application. There are 14 Predictors out of which 8 are categorical and 6 are continuous. A more detailed information about variables is presented below.

```
A1:     0,1    CATEGORICAL
    a,b
A2:     continuous.
A3:     continuous.
A4:     1,2,3          CATEGORICAL
    p,g,gg
A5:  1, 2,3,4,5, 6,7,8,9,10,11,12,13,14    CATEGORICAL
     ff,d,i,k,j,aa,m,c,w, e, q, r,cc, x

A6:      1, 2,3, 4,5,6,7,8,9    CATEGORICAL
    ff,dd,j,bb,v,n,o,h,z

A7:     continuous.
A8:     1, 0       CATEGORICAL
    t, f.
A9: 1, 0          CATEGORICAL
    t, f.
A10:    continuous.
A11:  1, 0        CATEGORICAL
      t, f.
A12:    1, 2, 3    CATEGORICAL
        s, g, p
A13:    continuous.
A14:    continuous.
A15:   1,2
       +,-          (class attribute)
```

Preprocessing of the predictors

Before building models to predict, we initially pre-process the data using different pre-processing tools In order to prepare the data for model building. Below are shown the approaches that we used, the reasons behind these approaches as well as the end results.

a) Adding Predictors (Dummy Variables)

Our data set contains categorical variables, more specifically A1, A4, A5, A6, A8, A9, A11, A12 are categorical variables. Since A1, A8, A9, A11 already had two levels [0,1] so we leave them that way nonetheless the remaining predictors; A4, A5, A6, A12 are transformed into Dummy Variables. After this transformation and after discarding one of the dummy variables, at the end we get 38 predictors; 32 categorical and 6 continuous.

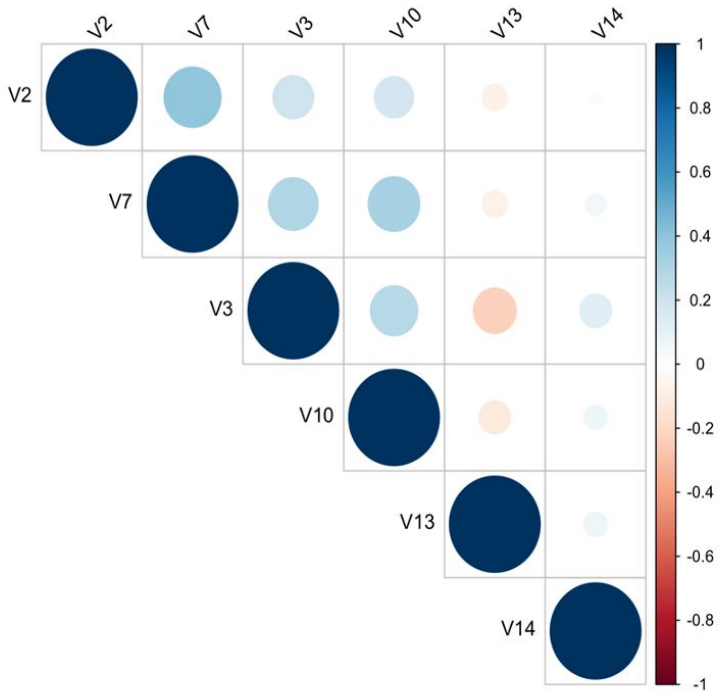
b) Deleting Predictors (Near Zero Variance)

We used Near Zero Variance approach to delete some categorical predictors. From our obtained results, we saw that no predictor had zero variance while 10 predictors had near zero variance. As a result, we decided to delete 10 categorical predictors with near zero variance, consequently we will be having 28 remaining predictors in total.

	freqRatio	percentUnique	zeroVar	nzv
V1	2.108108	0.2898551	FALSE	FALSE
V2	1.333333	50.7246377	FALSE	FALSE
V3	1.105263	31.1594203	FALSE	FALSE
V7	2.000000	19.1304348	FALSE	FALSE
V8	1.097264	0.2898551	FALSE	FALSE
V9	1.338983	0.2898551	FALSE	FALSE
V10	5.563380	3.3333333	FALSE	FALSE
V11	1.183544	0.2898551	FALSE	FALSE
V13	3.771429	24.7826087	FALSE	FALSE
V14	10.172414	34.7826087	FALSE	FALSE
V15	1.247557	0.2898551	FALSE	FALSE
V4_1	3.233129	0.2898551	FALSE	FALSE
V4_2	3.181818	0.2898551	FALSE	FALSE
V4_3	344.000000	0.2898551	FALSE	TRUE
V5_1	12.018868	0.2898551	FALSE	FALSE
V5_2	22.000000	0.2898551	FALSE	TRUE
V5_3	10.694915	0.2898551	FALSE	FALSE
V5_4	12.529412	0.2898551	FALSE	FALSE
V5_5	68.000000	0.2898551	FALSE	TRUE
V5_6	11.777778	0.2898551	FALSE	FALSE
V5_7	17.157895	0.2898551	FALSE	FALSE
V5_8	3.726027	0.2898551	FALSE	FALSE
V5_9	9.781250	0.2898551	FALSE	FALSE
V5_10	26.600000	0.2898551	FALSE	TRUE
V5_11	7.846154	0.2898551	FALSE	FALSE
V5_12	229.000000	0.2898551	FALSE	TRUE
V5_13	15.829268	0.2898551	FALSE	FALSE
V5_14	17.157895	0.2898551	FALSE	FALSE
V6_1	11.105263	0.2898551	FALSE	FALSE
V6_2	114.000000	0.2898551	FALSE	TRUE
V6_3	85.250000	0.2898551	FALSE	TRUE
V6_4	1.446809	0.2898551	FALSE	FALSE
V6_5	10.694915	0.2898551	FALSE	FALSE
V6_7	114.000000	0.2898551	FALSE	TRUE
V6_8	4.000000	0.2898551	FALSE	FALSE
V6_9	85.250000	0.2898551	FALSE	TRUE
V12_1	11.105263	0.2898551	FALSE	FALSE
V12_2	9.615385	0.2898551	FALSE	FALSE
V12_3	85.250000	0.2898551	FALSE	TRUE

c) Deleting Predictors (Highly Correlated Predictors)

For continuous predictors we checked their correlation by setting a 0.8 cutoff. As a result, we saw that predictors are not highly correlated and none of the predictors met the 0.8 cutoff, consequently we decided to keep all of the predictors.



	V2	V3	V7	V10	V13	V14
V2	1.00000000	0.2013152	0.39278787	0.18557383	-0.07715894	0.01853870
V3	0.20131524	1.0000000	0.29890156	0.27120674	-0.22234629	0.12312115
V7	0.39278787	0.2989016	1.00000000	0.32232967	-0.07638852	0.05134493
V10	0.18557383	0.2712067	0.32232967	1.00000000	-0.11980806	0.06369244
V13	-0.07715894	-0.2223463	-0.07638852	-0.11980806	1.00000000	0.06560887
V14	0.01853870	0.1231212	0.05134493	0.06369244	0.06560887	1.00000000

d) Missing Data

According to data set description, prior there were missing values. 37 cases(5%) had one or more missing values. The missing values from particular attributes were:

A1: 12
A2: 12
A4: 6
A5: 6
A6: 9
A7: 9
A14: 13

The missing data were handled as following:

- For categorical attributes it was used mode imputation.
- For continuous attributes it was used mean imputation.

e) Transformations

Before any models may be fit to the data, the distributions of the continuous variables must be analyzed. Below are histograms and box plots of the continuous predictors before any sort of transformations have been applied.

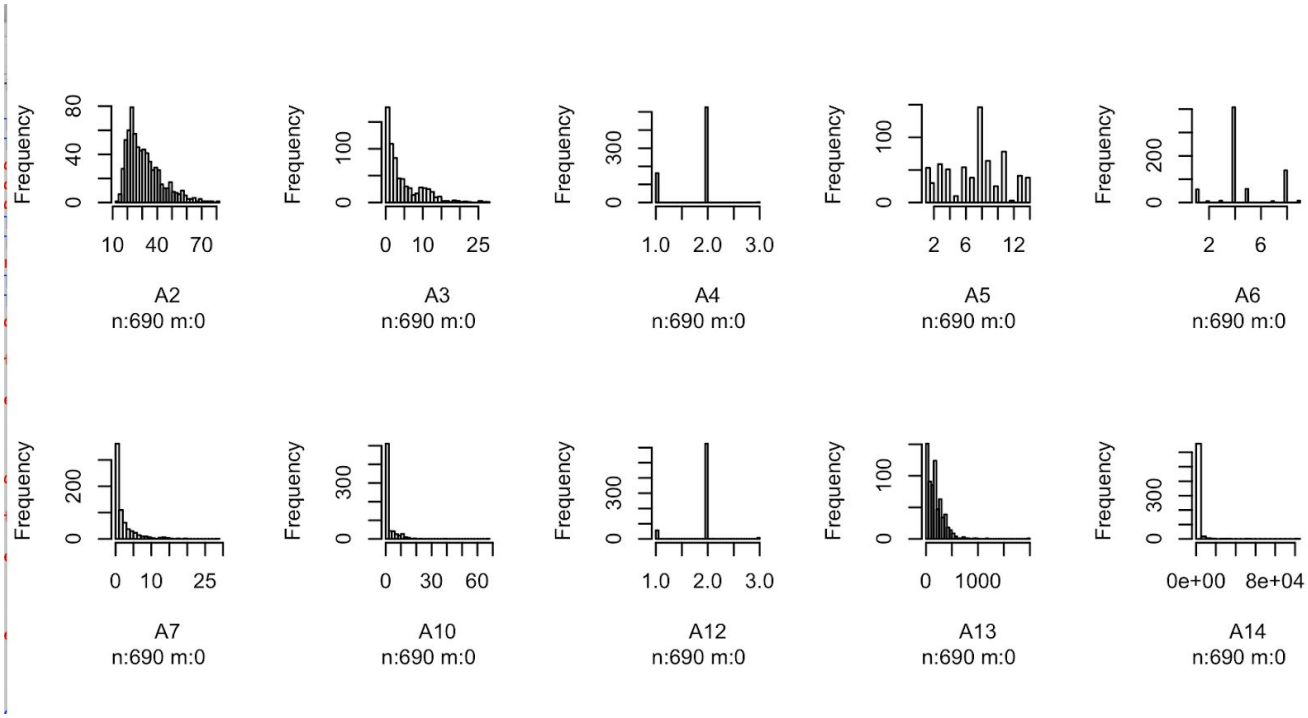


Fig 1.1 Predictor Variables Distributions

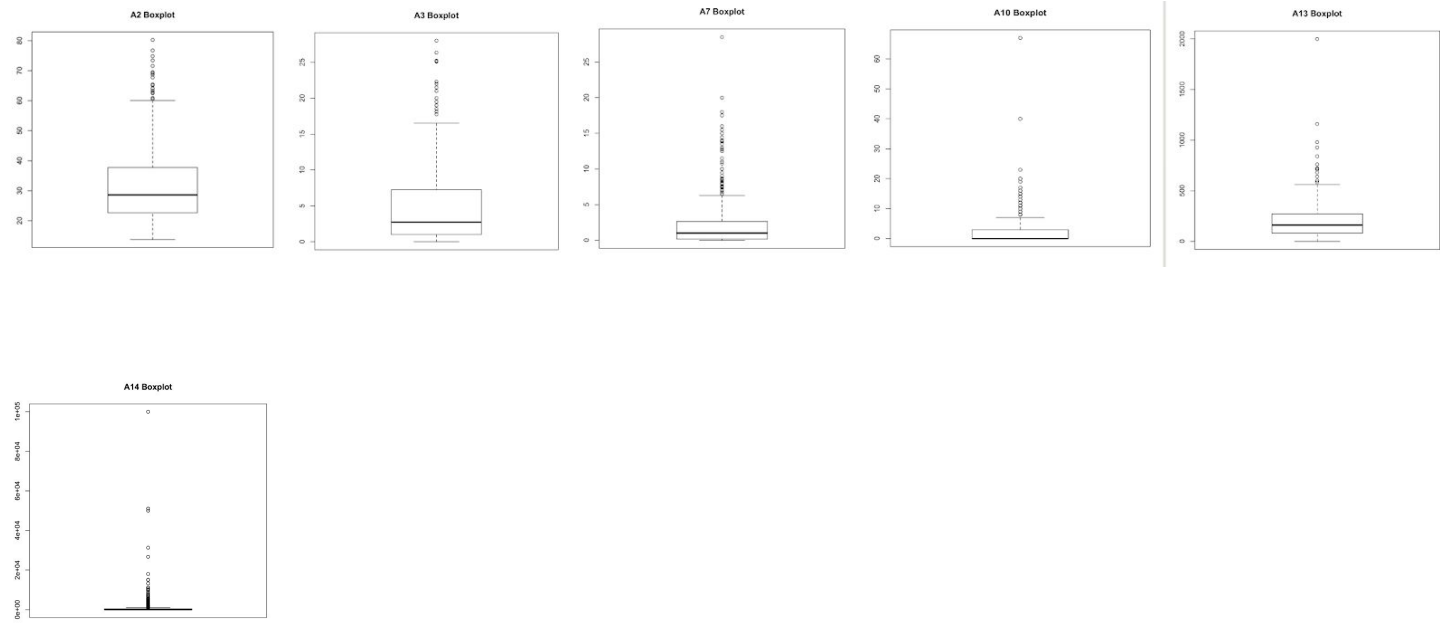


Fig 1.2 Predictor Variables Boxplot

From Fig 1 and 2 for some predictors, we can clearly see that mostly the data are skewed and have outliers. More precisely we obtained the skewness of predictors below:

```
        A1         A2         A3         A4         A5         A6         A7         A8
-0.76319531  1.15342059  1.48557462 -1.15094599 -0.06903997  0.46739292  2.88504112 -0.09285353
      A5_7       A5_8       A5_9      A5_11      A5_13      A5_14       A6_4       A6_5
 3.90079184  1.41223573  2.80775469  2.44409591  3.72725669  3.90079184 -0.37146330  2.96452654
```

```
        A9        A10        A11        A12        A13
 0.29294794  5.14131196  0.16871304 -1.94049525  2.74393007
       A6_8       A12_2
 1.50000000 -2.77837805
```

```
       A14        A4_2       A5_3       A5_4       A5_6
13.11207111 -1.22315326  2.96452654  3.25718037  3.14049095
```

From above we can conclude that most predictors are skewed, some highly some moderately both in a right and left way.

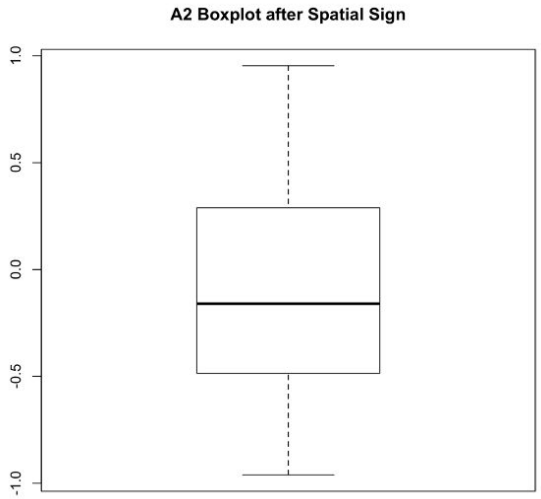
Consequently to handle the skewness of data we used Box-Cox transformation to transform data. As a result, we obtain the following results:

```
        A1         A2         A3         A4         A5         A6         A7         A8         A9        A10
-0.76319531  0.04679134  1.48557462 -0.93650009 -0.06903997  0.03755849  2.88504112 -0.09285353  0.29294794  5.14131196
      A5_7       A5_8       A5_9      A5_11      A5_13      A5_14       A6_4       A6_5       A6_8       A12_2
 3.90079184  1.41223573  2.80775469  2.44409591  3.72725669  3.90079184 -0.37146330  2.96452654  1.50000000 -2.77837805
>
```

```
        A11         A12         A13         A14        A4_2        A5_3        A5_4        A5_6
 0.16871304 -0.20713727  2.74393007  0.50928777 -1.22315326  2.96452654  3.25718037  3.14049095
```

From the above we see that after performing box-cox the skewness for some predictors has improved.

From Fig 2 we also saw that most of the predictors have outliers, as a result we performed Spatial Sign outliers to bring the outliers towards the majority of data.



From figure above we could see how for example for A2 if we compare to the previous boxplot before performing spatial sign, the outliers were brought towards the majority of data after applying spatial sign.

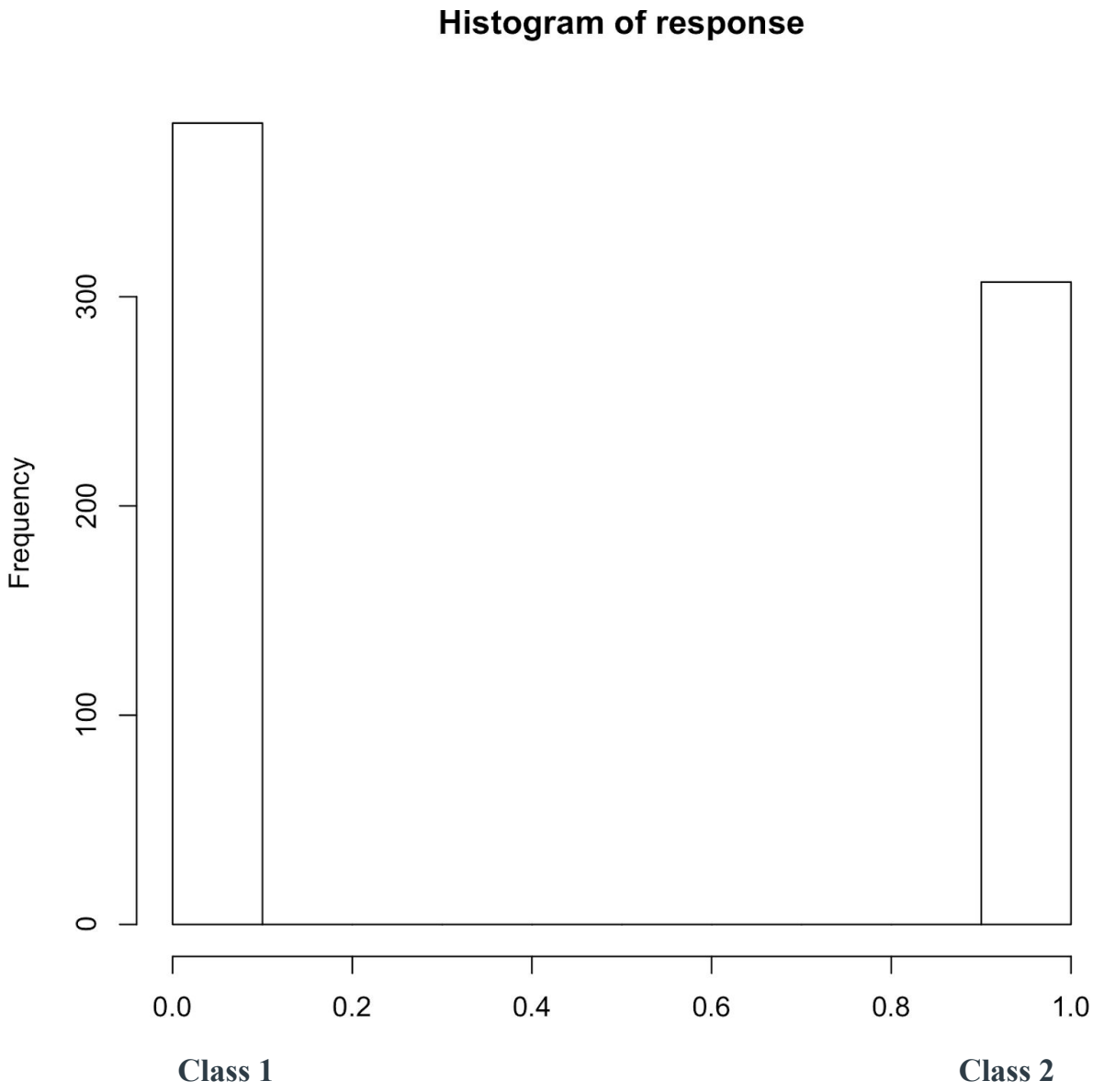
Considering that we have 28 predictors, we thought of also performing dimensionality reduction. When we perform PCA with 95% threshold, we would need 22 PCA components which is lower than the total 28 variables previously as a result we decide to keep the 22 PCA components which explain 95% of the variance.

f) **After pre-processing**

After performing all the pre-processing techniques, we are left with 22 Predictors, 1 Target Variable and a sample size of 690.

Splitting of the Data

Class 1 and Class 2 have a distribution of almost 55% and 45% so as a result we used stratified random sampling method by splitting 70% of data into a Training set and 30% of data into a Testing set.



As regard resampling, to estimate our model performance we used Bootstrapping since our sample size is not big.

Model Fitting

Linear Classification Models

We used several linear models to find the model that works best for our dataset. ROC was used as the criteria to find the best classification model. Here are the details below-

Model	**Best Tuning Parameter**	**Train ROC**
GLM	NA	0.9000939
PLS	ncomp = 2	0.9101840
LDA	NA	0.9124481
GLMNET	alpha = 0 lambda = 0.052	0.9114264
NSC	threshold = 0.2	0.9111206

Non-Linear Classification Models

We are working on a binary classification where we predict if a credit card application is accepted or not. We use ROC as our performance measure. We try to use the best tuning parameters for each model to find the best ROC. Here are the details (ROC & Tuning Parameters) used for the model-

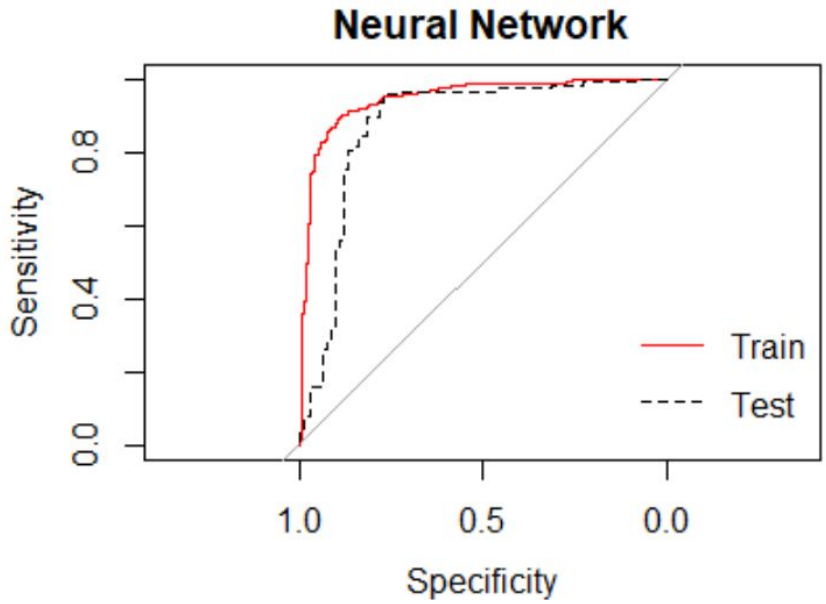
Model	**Best Tuning Parameter**	**Train ROC**
SVM	sigma = 0.02, C = 1	0.9139706
kNN	N=97	0.8978270

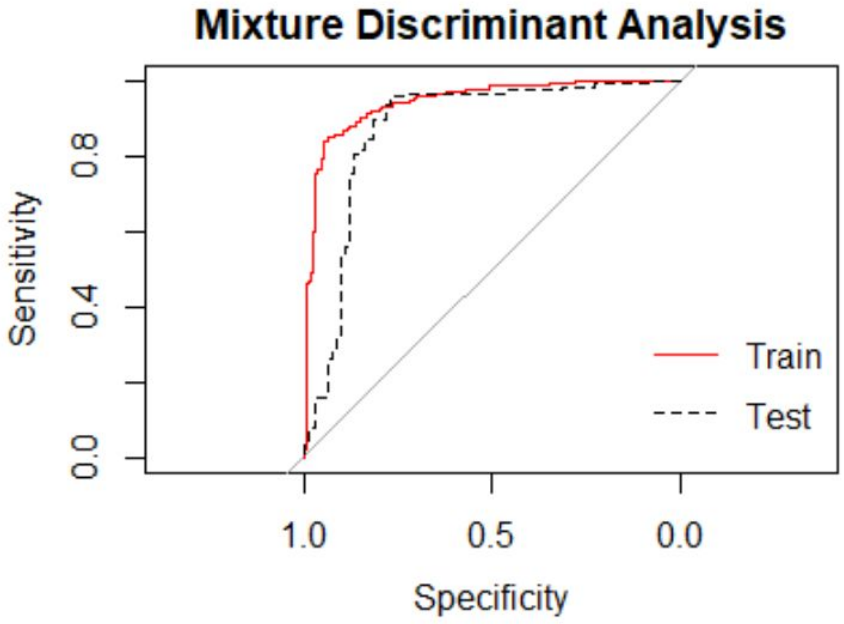
Naive Bayes	fL =1, adjust = 4, usekernel=TRUE	0.9047352
QDA	NA	0.860887
MDA	Subclasses=1	0.9228119
Neural Network	size = 1, decay = 0.1	0.9265594

Model Selection & Outcome

Below are the table of summaries for the two best models selected based on the best ROC outputs from the train function. We used bootstrapping due to the size of the dataset. We have shown the various metrics-Train & test AUC, Accuracy & Kappa Values. We have also shown the sensitivity and specificity plot.

Model	Best T. Parameter	Train AUC	Test AUC	Train Accuracy	Test Accuracy	Train Kappa	Test Kappa
MDA	Subclasses=1	0.9433	0.8794	0.8801653	0.8398058	0.7595546	0.6762549
Neural Network	size = 1, decay = 0.1	0.9456	0.8824	0.892562	0.8543689	0.782821	0.7041363





Though we got very similar Train ROC for the two best models, we have decided to go for **Neural Network** as our Final Model because of higher AUC & Kappa Values. Our dataset size was also quite small and so the neural network execution time did not impact our decision.

We used **avNNet** for our Neural Network model. The parameters for our best model are- size = 1, decay = 0.1

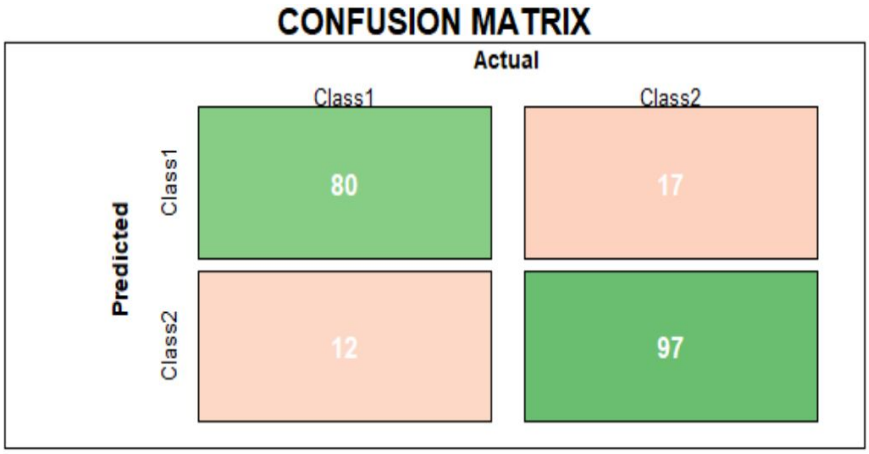
Pros

- Pretty straightforward to implement
- Ability to detect all possible interactions between predictor variables

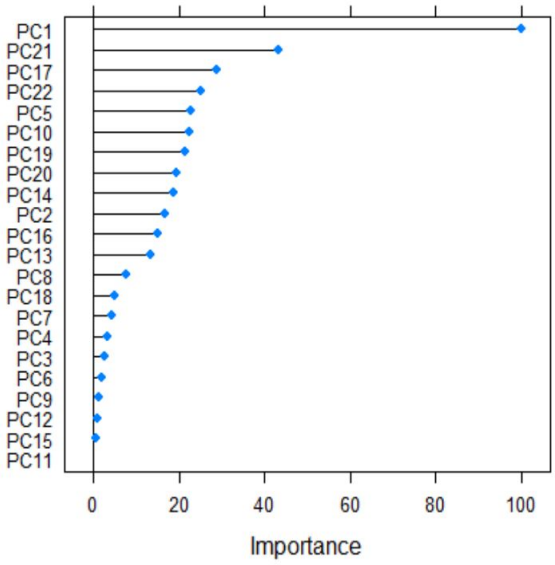
Cons

- Black Box
- Resource Intensive

Confusion Matrix



Most Important Predictors



Since we are using Principal Components, in order to know from original predictors which ones are the most important predictors we can calculate the correlation between the original data and the component. In our case the data and PC1 since PC1 is shown to have the highest importance.

	PC1
A2	-0.2659737
A3	-0.3870417
A4	-0.3196816
A5	-0.5583178
A6	-0.5665419
A7	-0.5085990

Summary

We have decided to go for **Neural Network** as the final model with parameters size = 1, decay = 0.1. The resulting Area under the curve & Kappa for the testing data is 0.8824 & 0.676 respectively. These results were satisfactory, and we believe that the model can be used for prediction with good accuracy.

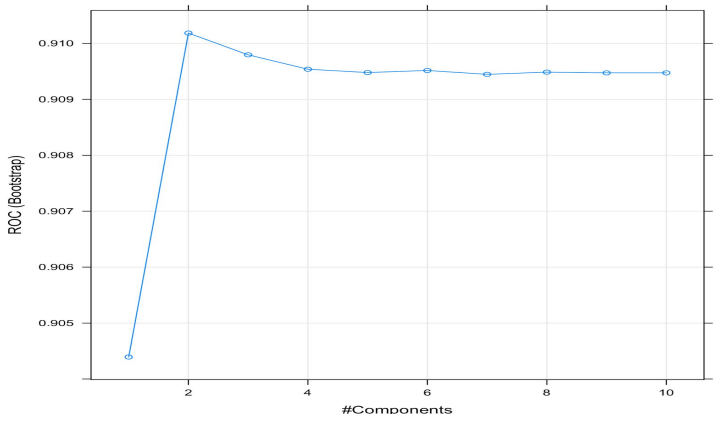
Appendix

Appendix 1: Supplemental Material for Linear Models

Linear Discriminant Analysis did not have any tuning parameters.

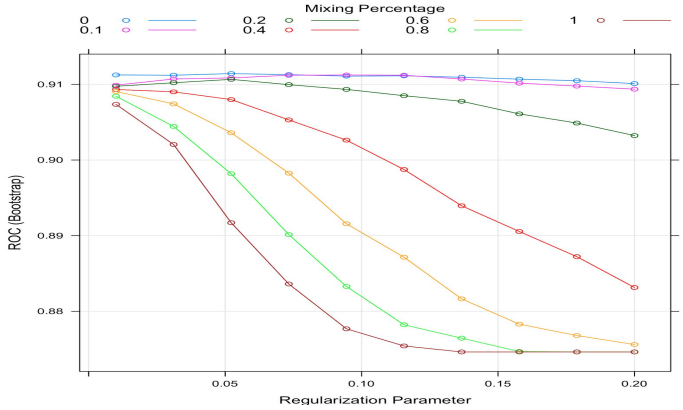
Partial Least Squares Discriminant Analysis

The tuning parameter used was the number of components=2.



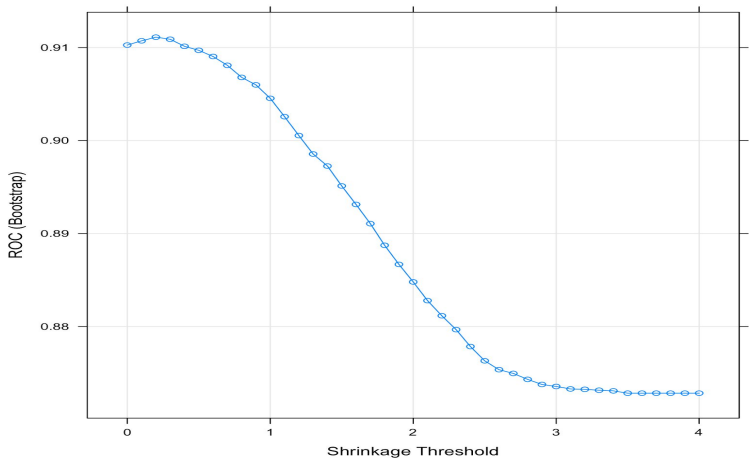
Penalized Model

The tuning parameters used were alpha = 0 and lambda = 0.052222



Nearest Shrunken Centroid

The best tuning parameter was threshold = 0.2

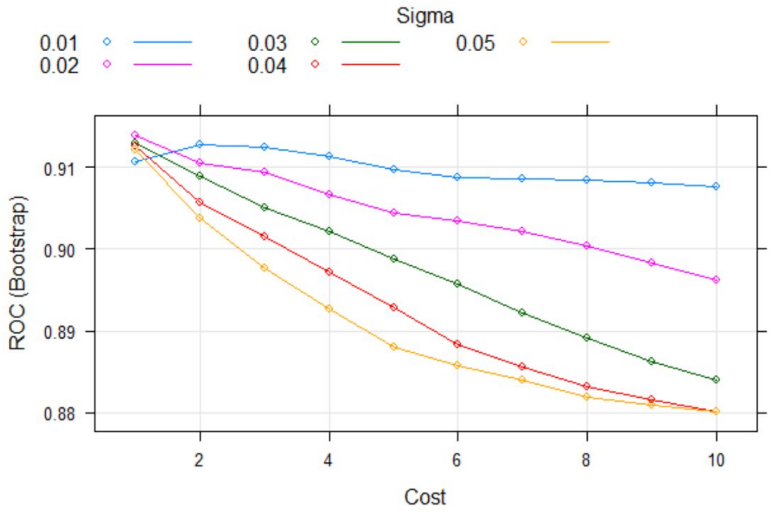


Appendix 2: Supplemental Material for Non-Linear Models

Quadratic Discriminant Analysis did not have any tuning parameters.

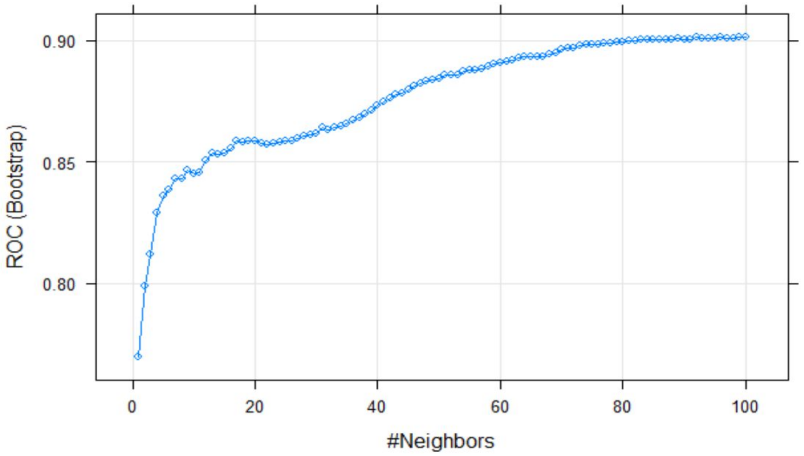
Support Vector Machine

The tuning parameters that worked best for ROC in the SVM model were sigma = 0.02 and C = 1.



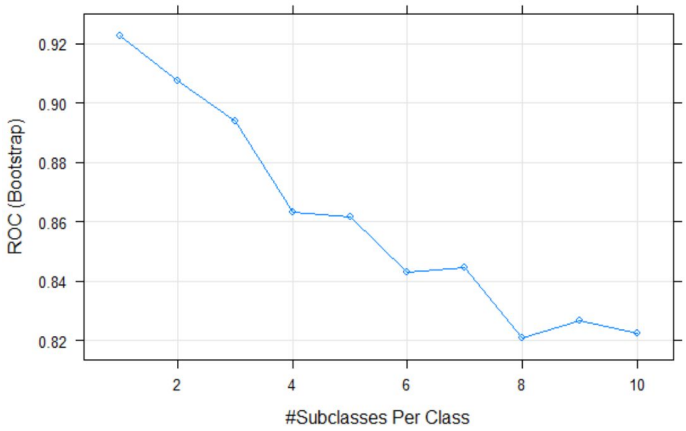
K-Nearest Neighbor

The number of neighbors that gave us the highest ROC was 97. Here is the ROC vs Neigbors plot to understand change in accuracy with respect to number of neighbors



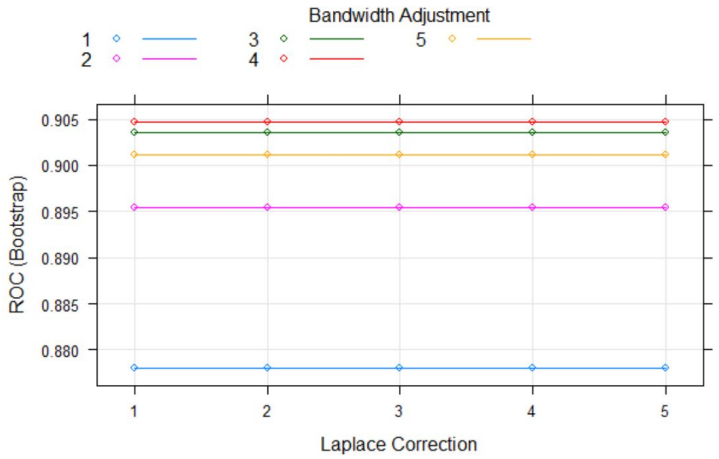
Mixture Discriminant Analysis

The number of subclasses that gave us the best ROC was 1.



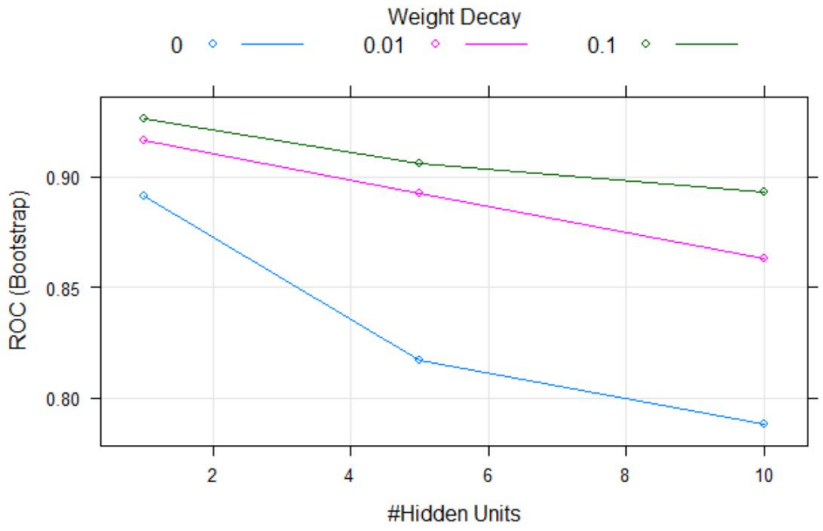
Naïve Bayes

We tried to find the best ROC n Naïve Bayes by using tuning parameters. The values used for the model were fL =1, usekernel = TRUE and adjust = 4 as



Neural Network

Neural Network used size = 1, decay = 0.1 tuning parameters to get the best ROC values.



R Code

```
install.packages(c("glmnet", "pamr", "rms", "sparseLDA", "subselect"))

library(AppliedPredictiveModeling)
library(caret)
library(nnet)
library(corrplot)
library(plyr)
library(Hmisc)
library(MASS)

 data <- read.table('australian.dat' )

#Changing Column Names
data <- setNames(data, c("A1","A2","A3","A4","A5", "A6","A7","A8", "A9", "A10",
   "A11","A12","A13","A14","A15"))

#Creating Dummy Variables
data <- fastDummies::dummy_cols(data,select_columns = c('A4','A5','A6','A12'),remove_first_dummy =
   TRUE)
dim(data)

#Setting Predictor & Response Variable
predictor<-subset(data, select = -A15)
response<-data$A15

#Removing Columns that does not affect the model
predictor <- predictor[, -nearZeroVar(predictor)]
dim(predictor)

#Handling High Correlation in Continuous Variables
myvars <- c('A2', 'A3',  'A7', 'A10', 'A13', 'A14' )
cont_predictor <- predictor[myvars]
correlations <- cor(cont_predictor)
corrplot(correlations, method = "circle", tl.pos='n',type='upper')

#Setting 0.8 Cutoff

tooHigh <- findCorrelation(cor(cont_predictor), cutoff = .80)

#Box Cox
trans <- preProcess(predictor, method =  c("BoxCox","center","scale"))
predictor <- predict(trans, predictor)
```

```
#PCA

#trans <- preProcess(segDataCont, method = c("BoxCox", "center", "scale", "pca"))
trans <- preProcess(predictor, method =  "pca")
predictor <- predict(trans, predictor)

#Spatial Sign
predictor <- spatialSign(predictor)
predictor <- as.data.frame(predictor)

predictor$class <- factor(ifelse(runif(length(response)) <=
                     response, "Class1", "Class2"))

response <- predictor$class
predictor<-subset(predictor, select=-c(class))

#Splitting Train & Test
#Training Data
trainingRows <- createDataPartition(response, p = .70, list= FALSE)
trainPredictor <- predictor[trainingRows,]
trainResponse <- response[trainingRows]

#Testing Data
testPredictor <- predictor[-trainingRows,]
testResponse <- response[-trainingRows]

ctrl <- trainControl(summaryFunction = twoClassSummary,method = "boot",number = 10, classProbs = TRUE,
 savePredictions = TRUE)
```

```
> #Multinomial-Logistics Regression
> set.seed(1000)
>lr <- train(x=x_train, y = y_train, method = "multinom", metric = "ROC", tuneLength=10)
> predictionlr<-predict(lr,x_test)
> confusionMatrix(data =predictionlr, reference = y_test)
> plot(lrBio)
```

```
> #Linear Descriptive Analysis

>lda <- train(x = x_train, y = y_train, method = "lda", metric = "ROC", tuneLength=10)
```

```
> predictionlda<-predict(lda,x_test)
>  confusionMatrix(data =predictionlda, reference = y_test)

> #Partial Least Square Descriminant Analysis

> pls <- train(x = x_train, y = y_train, method = "pls", tuneGrid = expand.grid(.ncomp = 1:1), preProc =
    c("center","scale"), metric = "ROC", tuneLength=10)
> predictionPLSBio <-predict(pls,x_test)
> confusionMatrix(data =predictionPLS, reference = y_test)

> #Penalized Model (Penalized Logistic Regression)
> glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4),.lambda = seq(.01, .2, length = 10))
> set.seed(1000)
> glmnTunedLR <- train(x=x_train, y =y_train, method = "glmnet", tuneGrid = glmnGrid, preProc = c("center",
    "scale"), metric = "ROC", tuneLength=10)
> predictionGlmnet <-  predict(glmnTunedLR,x_test)
> confusionMatrix(data =predictionGlmnet, reference = y_test)
>plot(glmnTunedLR)

> #Nearest Shrunken Centroids
> library(pamr)
> nscGridBio <- data.frame(.threshold = seq(0,4, by=0.1))
> set.seed(1000)
> nscTuned <- train(x = x_train, y = y_train, method = "pam", preProc = c("center", "scale"),tuneGrid =
    nscGridBio, metric = "Accuracy", tuneLength=10)
> predictionNSCBio <-predict(nscTuned,x_test)
> confusionMatrix(data =predictionNSC,reference = y_test)
> plot(nscTuned)



install.packages(c("glmnet", "pamr", "rms", "sparseLDA", "subselect"))



library(AppliedPredictiveModeling)

library(caret)

library(nnet)

library(corrplot)

library(plyr)

library(Hmisc)

library(MASS)
```

```
setwd("C:/Users/arnab/OneDrive/Documents/Predictive Modelling/Presentation")

data <- read.table('australian.dat' )


#Changing Column Names

data <- setNames(data, c("A1","A2","A3","A4","A5", "A6","A7","A8", "A9", "A10",
"A11","A12","A13","A14","A15"))


#Creating Dummy Variables

data <- fastDummies::dummy_cols(data,select_columns = c('A4','A5','A6','A12'),remove_first_dummy = TRUE)

dim(data)


#Setting Predictor & Response Variable

predictor<-subset(data, select = -A15)

response<-data$A15


#Removing Columns that does not affect the model

predictor <- predictor[, -nearZeroVar(predictor)]

dim(predictor)


#Handling High Correlation in Continuous Variables

myvars <- c('A2', 'A3',  'A7', 'A10', 'A13', 'A14' )

cont_predictor <- predictor[myvars]

correlations <- cor(cont_predictor)

corrplot(correlations, method = "circle", tl.pos='n',type='upper')


#Setting 0.8 Cutoff
```

```
tooHigh <- findCorrelation(cor(cont_predictor), cutoff = .80)



#Box Cox

trans <- preProcess(predictor, method =  c("BoxCox","center","scale"))

predictor <- predict(trans, predictor)


#PCA


#trans <- preProcess(segDataCont, method = c("BoxCox", "center", "scale", "pca"))

trans <- preProcess(predictor, method =  "pca")

predictor <- predict(trans, predictor)


#Spatial Sign

predictor <- spatialSign(predictor)

predictor <- as.data.frame(predictor)


predictor$class <- factor(ifelse(runif(length(response)) <=

                        response, "Class1", "Class2"))


response <- predictor$class

predictor<-subset(predictor, select=-c(class))


#Splitting Train & Test

#Training Data

trainingRows <- createDataPartition(response, p = .70, list= FALSE)
```

```
trainPredictor <- predictor[trainingRows,]

trainResponse <- response[trainingRows]


#Testing Data

testPredictor <- predictor[-trainingRows,]

testResponse <- response[-trainingRows]


#-------------------------------------------------------------------------------------------------

#NON-LINEAR MODEL

#-------------------------------------------------------------------------------------------------


#QDA

set.seed(476)

install.packages(c("glmnet", "pamr", "rms", "sparseLDA", "subselect"))

library(mda)

library(AppliedPredictiveModeling)

library(MASS)

library(rda)


ctrl <- trainControl(summaryFunction = twoClassSummary,method = "boot",number = 10,

                     classProbs = TRUE, savePredictions = TRUE)



qdaFit <- train(x = trainPredictor, y = trainResponse, method = "qda", metric = "ROC",

                trControl = ctrl)

qdaFit
```

```
plot(qdaFit)



#RDA

set.seed(476)

ctrl <- trainControl(summaryFunction = twoClassSummary,method = "boot",number = 10,
                     classProbs = TRUE, savePredictions = TRUE)



rdaFit <- train(x = trainPredictor, y = trainResponse, method = "rda", metric = "ROC",
                tuneGrid = expand.grid(gamma=seq(0,0.2,by=0.1),lambda=seq(0,1,by=0.111111111111)),
                trControl = ctrl)
rdaFit
plot(rdaFit)

#MDA

set.seed(476)

ctrl <- trainControl(summaryFunction = twoClassSummary,method = "boot",number = 10,
                     classProbs = TRUE, savePredictions = TRUE)

mdaFit <- train(x = trainPredictor, y = trainResponse, method = "mda", metric = "ROC",
                tuneGrid = expand.grid(subclasses=seq(1,10,by=1)),
```

```
        trControl = ctrl)


mdaFit

plot(mdaFit)


#Accuracy & Kappa


#Training Accuracy

mdaPred_train <- predict(mdaFit, newdata = trainPredictor);

postResample(mdaPred_train,trainResponse)

#Testing Accuracy

mdaPred_test <- predict(mdaFit, newdata = testPredictor);

postResample(mdaPred_test,testResponse)


roc_train <- roc(response=trainResponse,predictor=mdaPred_train[,1])

plot(roc_train)

roc_train$auc


roc_test <- roc(response=testResponse,predictor=mdaPred_train[,1])

plot(roc_test)

roc_test$auc


plot(roc_train,col="red",lty=1,ylab="Sensitivity",lwd=1,xlab="Specificity",main="Mixture Discriminant
Analysis")

lines(roc_test,col="black",lwd=1,lty=2)

legend(x = "bottomright", legend=c("Train","Test"),col=c("red","black"), lty=1:2,

    box.lty=0)
```

```
#FDA

set.seed(476)

ctrl <- trainControl(summaryFunction = twoClassSummary,method = "boot",number = 10,
                     classProbs = TRUE, savePredictions = TRUE)

fdaFit <- train(x = trainPredictor, y = trainResponse, method = "fda", metric = "ROC",
                tuneGrid = expand.grid(degree=seq(1,10,by=1),nprune=seq(2,20,by=1)),
                trControl = ctrl)

fdaFit
plot(fdaFit)



#Neural Network
library(nnet)
library(caret)



nnetGrid <- expand.grid(.decay=c(0, 0.01, 0.1),.size=c(1, 5, 10),.bag=FALSE)
netModel <- train(trainPredictor, trainResponse,
                  method = "avNNet",
                  metric = "ROC",
                  tuneGrid = nnetGrid,
                  trControl=ctrl,
```

```
            trace=FALSE,
            linout=TRUE,
            maxit=500)


#Training Accuracy
netPred_train <- predict(netModel, newdata = trainPredictor,type="prob");
postResample(netPred_train,trainResponse)
#Testing Accuracy
netPred_test <- predict(netModel, newdata = testPredictor,type="prob");
postResample(netPred_test,testResponse)

roc_train <- roc(response=trainResponse,predictor=netPred_train[,1])
plot(roc_train)
roc_train$auc

roc_test <- roc(response=testResponse,predictor=netPred_test[,1])
plot(roc_test)
roc_test$auc

plot(roc_train,col="red",lty=1,ylab="Sensitivity",lwd=1,xlab="Specificity",main="Neural Network")
lines(roc_test,col="black",lwd=1,lty=2)
 legend(x = "bottomright", legend=c("Train","Test"),col=c("red","black"), lty=1:2,
     box.lty=0)

#SVM
svmModel <- train(trainPredictor,
```

```
        trainResponse,
        method = "svmRadial",
        metric="ROC",
        tuneGrid = expand.grid(sigma=seq(0.01,0.05,by=0.01),C=seq(1,10,by=1)),
        tuneLength=10, trControl = ctrl)
svmModel
plot(svmModel)



#Training Accuracy
svmPred <- predict(svmModel, newdata = trainPredictor)
postResample(svmPred,trainResponse);


#Testing Accuracy
svmPred <- predict(svmModel, newdata = testPredictor)
postResample(svmPred,testResponse)


#KNN
#KNN MODEL
knnModel <- train(x = trainPredictor,
            y = trainResponse,
            method = "knn",
            metric="ROC",
            tuneGrid = expand.grid(k=seq(1,100,by=1)),
            tuneLength = 10,trControl = ctrl)
knnModel
plot(knnModel)
```

```
#Training Accuracy
knnPred <- predict(knnModel, newdata = trainPredictor,)
postResample(knnPred,trainResponse);

#Testing Accuracy
knnPred <- predict(knnModel, newdata = testPredictor)
postResample(knnPred,testResponse)

rfTestPred <- predict(knnModel, testPredictor, type = "prob")
testPredictor$RFprob <- rfTestPred[,"Class1"]
testPredictor$knnclass <- predict(knnModel, testPredictor)

library(pROC)
rocCurve <- roc(response = testPredictor$knnclass,
        predictor = testPredictor$RFprob,
        ## This function assumes that the second
        ## class is the event of interest, so we
        ## reverse the labels.
        levels = rev(levels(testPredictor$knnclass)))
auc(knnPred)

levels = rev(levels(testPredictor$class))

#Naive Bayes

naiveModel <- train(x = trainPredictor,
```

```
        y = trainResponse,
        method = "nb",
        metric="ROC",
        tuneGrid = expand.grid(fL = 1:5,adjust = seq(1, 5, by = 1),usekernel=TRUE),
        tuneLength = 10,trControl = ctrl)
naiveModel
plot(naiveModel)

#Training Accuracy
naivePred_train <- predict(naiveModel, newdata = trainPredictor,type="prob");

#Testing Accuracy
naivePred_test <- predict(naiveModel, newdata = testPredictor,type="prob");

roc_train <- roc(response=trainResponse,predictor=naivePred_train[,1])
plot(roc_train)
roc_train$auc[,1]

roc_test <- roc(response=testResponse,predictor=naivePred_test[,1])
plot(roc_test)
roc_test$sensitivities
```