LDA, PCA, & Boosting with Engineered Features

Mikhail Lara

```
options(warn = -1)
suppressMessages(library(data.table))
suppressMessages(library(ggplot2))
suppressMessages(library(gridExtra))
suppressMessages(library(e1071))
suppressMessages(library(MASS))
suppressMessages(library(caret))
suppressMessages(library(gbm))

filename<-'/Users/Mikey/Documents/ML-Case-Studies/Glass/glass.csv'
DT<-fread(filename)
```

The data consists of 214 observations of 6 different types of glass. The number of observations in the dataset vary significantly across the glass types, which could pose a problem if ML techniques are used without data preprocessing.

```
invisible(DT[,Type:=as.factor(Type)])
print('Number of Occurrences for Each Glass Type')
```

```
## [1] "Number of Occurrences for Each Glass Type"
```

```
print(DT[,.N,by=Type])
```

```
##    Type  N
## 1:    1 70
## 2:    2 76
## 3:    3 17
## 4:    5 13
## 5:    6  9
## 6:    7 29
```

Brute Force Stochastic Gradient Boosting without Pre-Processing

A stochastic gradient boosting model is fit to the raw data without pre-processing. The out-of-sample accuracy is estimated using leave-one-out cross-validation(CV).

The final model consists of **150 trees** with **interaction depth of 2**.

The CV **Accuracy** is **0.7850467**.

The CV **Concordance(kappa)** is **0.7028226**.

```
DT.brute<-data.table(DT)
invisible(DT.brute[,Type:=as.factor(as.character(Type))])

train_controlA<- trainControl(method="LOOCV")

set.seed(123)

suppressMessages(gbm.fit.brute1<- train(Type~., data=DT.brute, trControl=train_controlA, method="gbm",v

gbm.fit.brute1
```

```
## Stochastic Gradient Boosting
##
## 214 samples
##   9 predictor
##   6 classes: '1', '2', '3', '5', '6', '7'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 213, 213, 213, 213, 213, 213, ...
## Resampling results across tuning parameters:
##
##   n.trees  interaction.depth  Accuracy   Kappa
##    50      1                  0.7102804  0.5910618
##    50      2                  0.7570093  0.6620095
##    50      3                  0.7943925  0.7146234
##   100      1                  0.7336449  0.6285401
##   100      2                  0.7803738  0.6957837
##   100      3                  0.7897196  0.7079783
##   150      1                  0.7523364  0.6557606
##   150      2                  0.7850467  0.7028226
##   150      3                  0.7850467  0.7020220
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 50, interaction.depth
##  = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
#confusionMatrix(predict(gbm.fit.brute1,DT.brute),DT.brute$Type)
remove(DT.brute)
```

Exploratory Analysis & Modeling

Discrimination Criterion: A unique characteristic of type 6 glass is that none of the recorded observations have any levels of potassium(K), iron(Fe), or barium(Ba). The absense of these three elements in the glass can be used to uniquely distinguish type 6 glass from the other 5 types. This helps with modeling because the glass type with the fewest occurrences can be omitted.

The **Accuracy** is **0.9766355**.

The **Sensitivity(recall)** is **1**.

The **True Negative Rate(TNR)** is **0.9756098**.

The **Positive Predictive Value(PPV)** is **0.6428571**.

The **Negative Predictive Value(NPV)** is **1**.

```
print('Summary of Type 6 Glass(Tableware)')
```

```
## [1] "Summary of Type 6 Glass(Tableware)"
```

```
print(summary(DT[Type=='6']))
```

```
##        RI               Na               Mg               Al
```

```
##  Min.   :1.511   Min.   :13.79   Min.   :0.000   Min.   :0.340
##  1st Qu.:1.518   1st Qu.:14.09   1st Qu.:0.000   1st Qu.:1.190
##  Median :1.519   Median :14.40   Median :1.740   Median :1.560
##  Mean   :1.517   Mean   :14.65   Mean   :1.306   Mean   :1.367
##  3rd Qu.:1.519   3rd Qu.:14.56   3rd Qu.:2.240   3rd Qu.:1.660
##  Max.   :1.520   Max.   :17.38   Max.   :2.410   Max.   :2.090
##        Si              K           Ca               Ba          Fe
##  Min.   :72.37   Min.   :0   Min.   : 6.650   Min.   :0   Min.   :0
##  1st Qu.:72.50   1st Qu.:0   1st Qu.: 9.260   1st Qu.:0   1st Qu.:0
##  Median :72.74   Median :0   Median : 9.570   Median :0   Median :0
##  Mean   :73.21   Mean   :0   Mean   : 9.357   Mean   :0   Mean   :0
##  3rd Qu.:73.48   3rd Qu.:0   3rd Qu.: 9.950   3rd Qu.:0   3rd Qu.:0
##  Max.   :75.41   Max.   :0   Max.   :11.220   Max.   :0   Max.   :0
##  Type
##  1:0
##  2:0
##  3:0
##  5:0
##  6:9
##  7:0
```

```
#Cleaning
#Processing: OMIT ANY Type 6 Glass (Minor Cross-Over with Groups 1[1],2[3],3[1])
accuracy<-(9+(nrow(DT)-nrow(DT[((K==0)&(Fe==0)&(Ba==0))])))/nrow(DT)
recall<-9/nrow(DT[Type=='6'])
TNR<-(nrow(DT)-nrow(DT[((K==0)&(Fe==0)&(Ba==0))]))/(nrow(DT[Type!='6'])) #specificity
PPV<-nrow(DT[Type=='6'])/nrow(DT[((K==0)&(Fe==0)&(Ba==0))])
NPV<-(nrow(DT)-nrow(DT[((K==0)&(Fe==0)&(Ba==0))]))   /   ((nrow(DT)-nrow(DT[((K==0)&(Fe==0)&(Ba==0))]))
```

Observations with Zero Potassium, Iron, and Barium

```
DT[((K==0)&(Fe==0)&(Ba==0)),.N,by=Type]
```

```
##    Type N
## 1:    1 1
## 2:    2 3
## 3:    3 1
## 4:    6 9
```

```
DT.omit6 <- DT[!((K==0)&(Fe==0)&(Ba==0))]
```

Distribution of Remaining Glass Types Data

```
DT.omit6[,.N,by=Type]
```

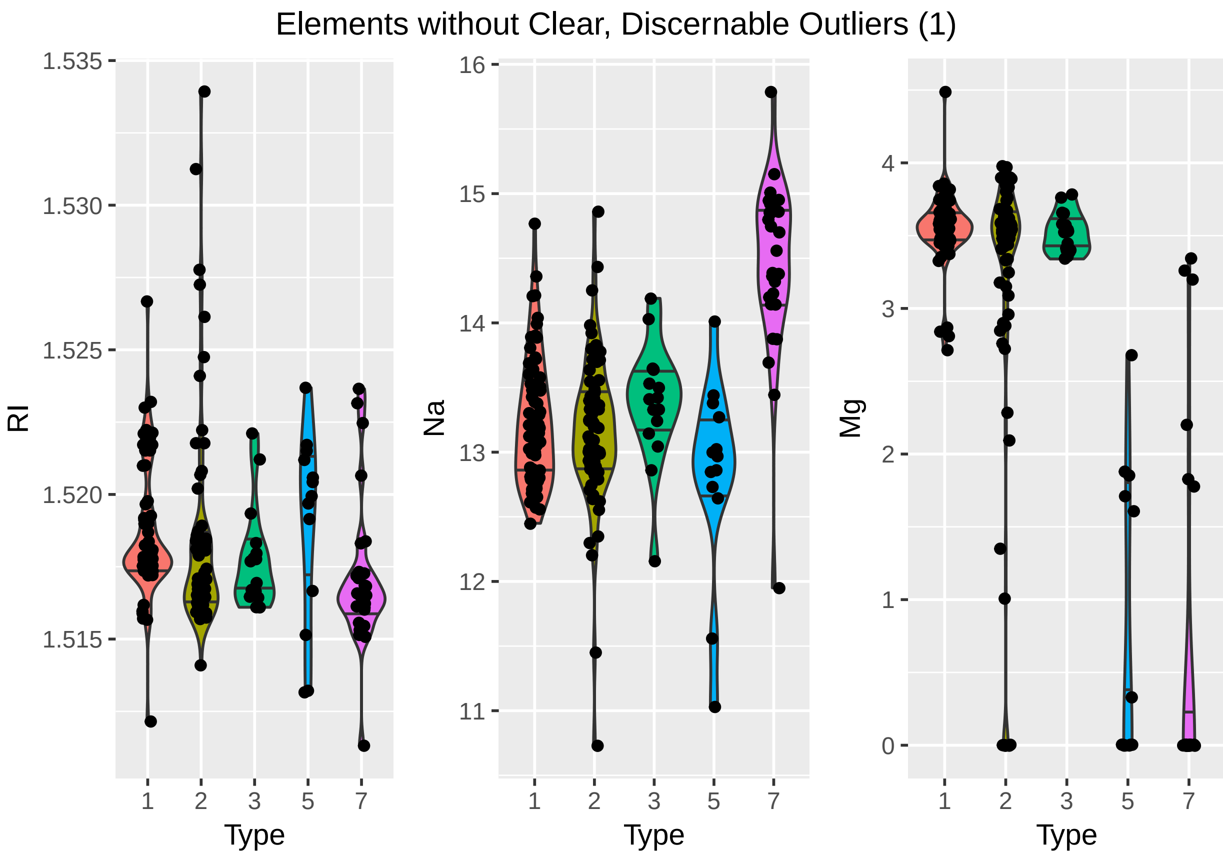
```
##    Type  N
## 1:    1 69
## 2:    2 73
## 3:    3 16
## 4:    5 13
## 5:    7 29
```

Pre-Processing: Outlier Identification

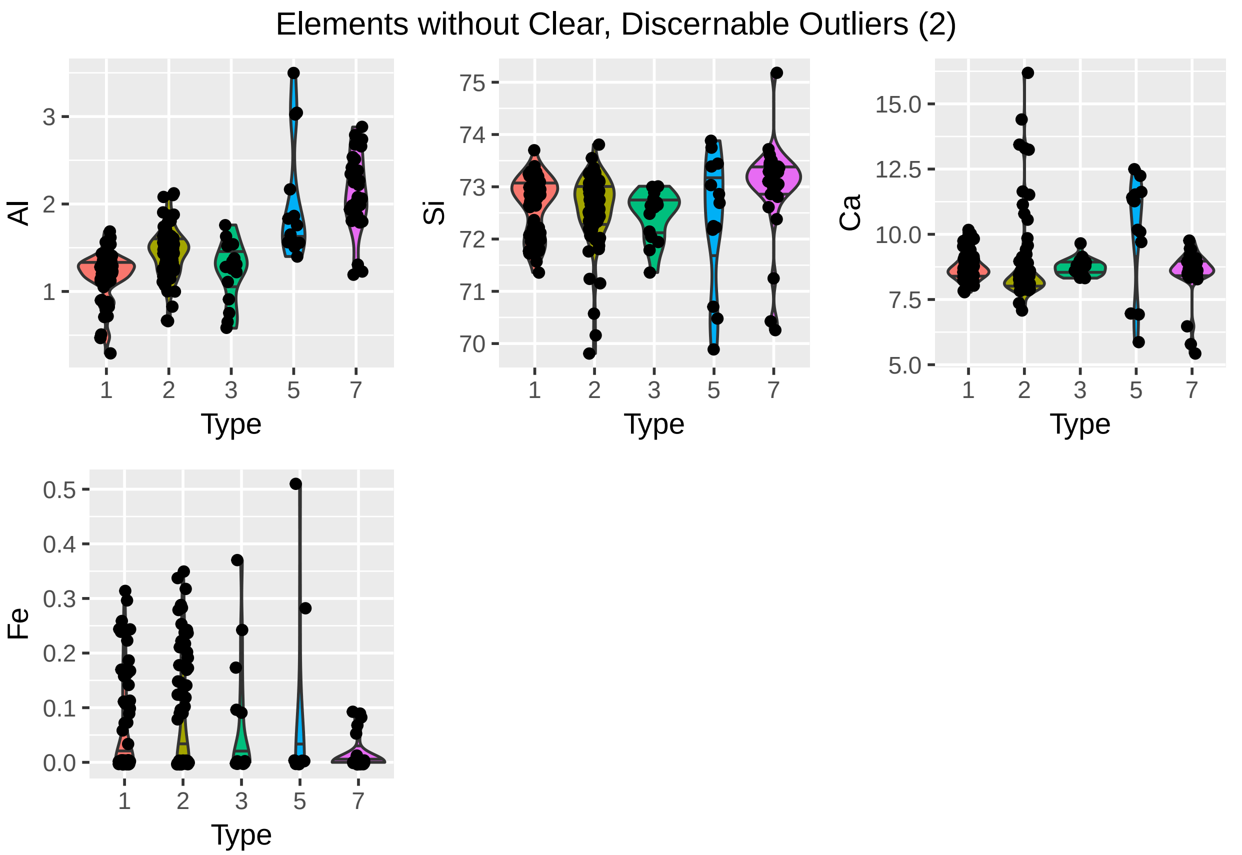
Outliers of the in-group attribute distributions needs to be removed before doing feature engineering. This step is particularly important for principle component analysis(PCA) since that algorithm is highly sensitive to spurrious data.

Since the in-group attribute distributions are **not normally distributed**, outliers are identified as points that are 'significantly far' outside of the 25% - 75% quantile limits.

```r
g_RI<-ggplot(data=DT.omit6,aes(y=RI,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+ geom_jitter(width=.1)

g_Na<-ggplot(data=DT.omit6,aes(y=Na,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+
    geom_jitter(width=.1) # Group 7 has higher Na than other groups & is Normal (p=0.001)

g_Mg<-ggplot(data=DT.omit6,aes(y=Mg,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+
    geom_jitter(width=.1) # Groups 1,2,3 have much higher Mg than 5&7

g_Al<-ggplot(data=DT.omit6,aes(y=Al,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+
    geom_jitter(width=.1) # Groups 5 & 7 have higher and distinct Al levels than other groups

g_Si<-ggplot(data=DT.omit6,aes(y=Si,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+ geom_jitter(width=.1)

g_Ca<-ggplot(data=DT.omit6,aes(y=Ca,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+ geom_jitter(width=.1)

g_Fe<-ggplot(data=DT.omit6,aes(y=Fe,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+
    geom_jitter(width=.1) # Group 7 has lower Fe than other groups

grid.arrange(g_RI,g_Na,g_Mg,ncol=3,
             top='Elements without Clear, Discernable Outliers (1)')
```

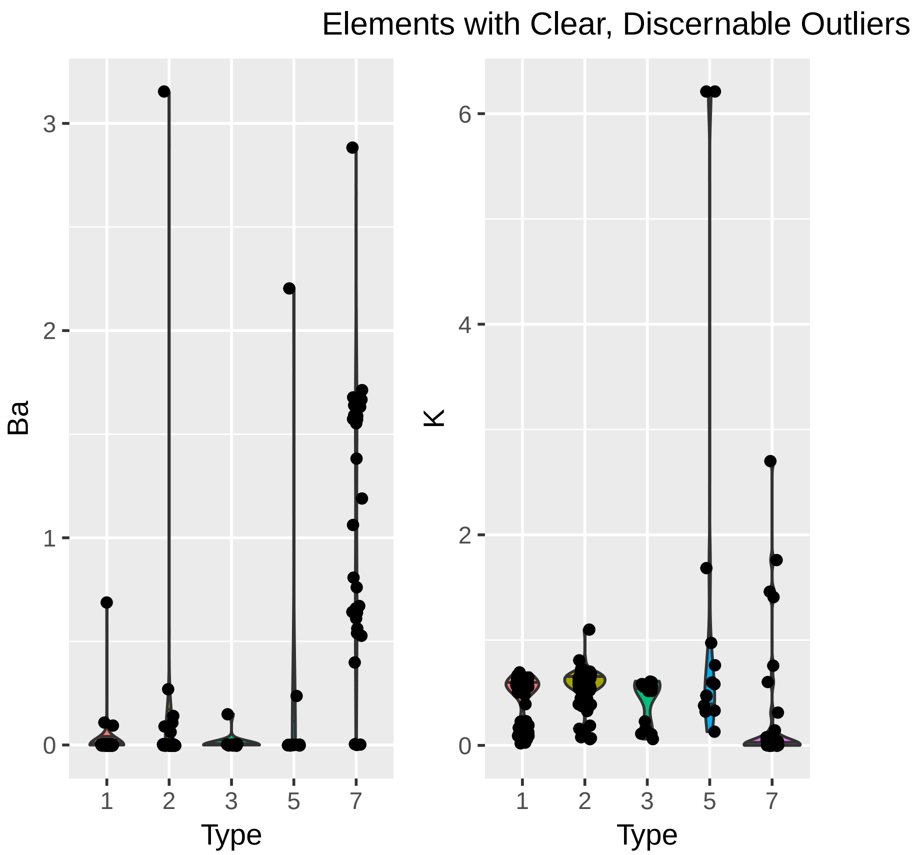



```r
grid.arrange(g_Al, g_Si,g_Ca,g_Fe,ncol=3,
             top='Elements without Clear, Discernable Outliers (2)')
```



The glass attributes that have ‘clear, discernable’ outliers according to the quantile criterion are Ba and K. A total of **6 outliers** are removed based on this criterion.

```r
g_Ba<-ggplot(data=DT.omit6,aes(y=Ba,x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+
    geom_jitter(width=.1) # For Group 5 & 7, 7 is somewhat normal (p=0.001)

g_K<-ggplot(data=DT.omit6,aes(y=K, x=as.factor(Type),fill=as.factor(Type)))+
    xlab('Type')+scale_fill_discrete(guide=FALSE)+
    geom_violin(draw_quantiles=c(0.25,0.75))+
    geom_jitter(width=.1) # Groups 7 skew higher for K than other groups

grid.arrange(g_Ba,g_K,ncol=3, top='Elements with Clear, Discernable Outliers')
```




```
#Remove Barium Outliers
#(Ba>2)
#(Type=='1')&(Ba>0.5)

#Remove K Outliers
type5.avg.K<-mean(DT.omit6[Type=='5']$K)
type5.sd.K<-sd(DT.omit6[Type=='5']$K)
thresh.type5.sd.K<-type5.avg.K+2*sd(DT.omit6[Type=='5']$K)
#(K>thresh.type5.sd.K)&(Type==5)

DT.clean<-DT.omit6[!((Ba>2)|(Type=='1')&(Ba>0.5)|(K>thresh.type5.sd.K)&(Type==5) )]
```

Glass Type Distribution with Outliers Removed

```
DT.clean[,.N,by=Type]
```

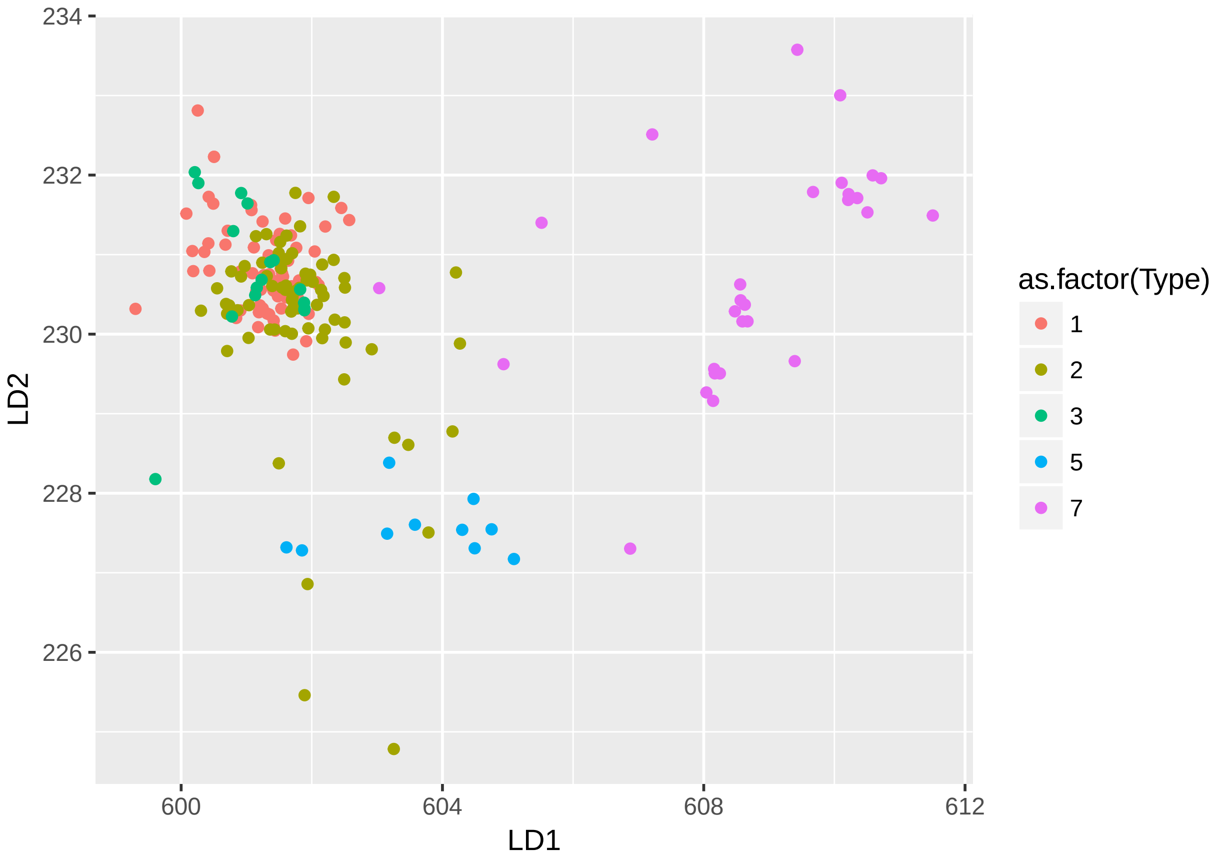
```
##    Type  N
## 1:    1 68
## 2:    2 72
## 3:    3 16
## 4:    5 10
## 5:    7 28
```

Linear Discriminant Analysis

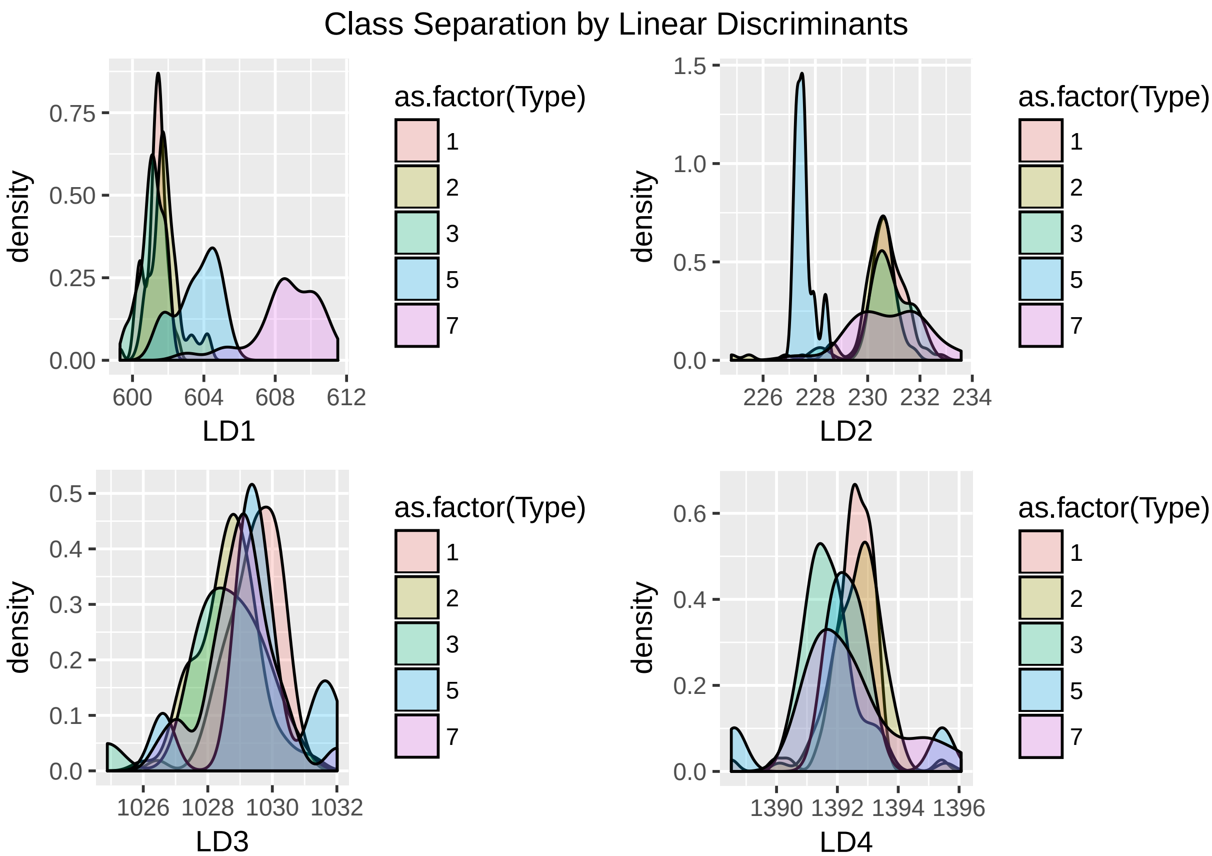
```r
lda.fit<-lda(Type~.,data = DT.clean)
lda.fit
```

```
## Call:
## lda(Type ~ ., data = DT.clean)
##
## Prior probabilities of groups:
##          1          2          3          5          7
## 0.35051546 0.37113402 0.08247423 0.05154639 0.14432990
##
## Group means:
##         RI       Na       Mg       Al       Si         K        Ca
## 1 1.518651 13.22029 3.548235 1.167206 72.65103 0.4588235  8.786029
## 2 1.518216 13.19153 3.168889 1.428333 72.59583 0.5419444  8.828750
## 3 1.517757 13.38188 3.521250 1.224375 72.46125 0.4318750  8.738750
## 5 1.520455 12.67300 0.738000 1.688000 72.96900 0.5010000 11.185000
## 7 1.517074 14.44393 0.557500 2.133571 72.96964 0.2864286  8.563571
##            Ba         Fe
## 1 0.002941176 0.05867647
## 2 0.009305556 0.08027778
## 3 0.009375000 0.06062500
## 5 0.024000000 0.07900000
## 7 0.974285714 0.01392857
##
## Coefficients of linear discriminants:
##            LD1        LD2         LD3         LD4
## RI 290.5916372 -8.3218913 266.1218302 854.1187960
## Na   1.9357382  2.9233697   5.1284692   0.1807660
## Mg   0.2412544  2.8235401   5.4398705   0.1954522
## Al   2.8083164  1.4169605   4.6135898   2.4526528
## Si   1.7066689  2.4159616   6.6513780   1.3726915
## K    2.2525525  1.6937208   5.0125436   0.7932657
## Ca   0.5957778  1.9147157   5.3827120  -1.2245384
## Ba   3.5467448  3.5148390   5.6159250  -1.2099323
## Fe  -0.5443469  0.6054221  -0.3787965   2.3798178
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4
## 0.8930 0.0753 0.0218 0.0099
```

```r
DT.lda<-as.matrix(DT.clean[,-c('Type'),with=FALSE])
DT.lda<- DT.lda %*% lda.fit$scaling
DF.lda<-data.frame(DT.lda,DT.clean$Type)
names(DF.lda)<-c( "LD1"     ,      "LD2"     ,      "LD3"     ,      "LD4"          ,  "Type")

ggplot(data=DF.lda, aes(x=LD1,y=LD2,colour=as.factor(Type)))+geom_point()
```




```
glda1<-ggplot(data=DF.lda, aes(x=LD1,fill=as.factor(Type)))+geom_density(alpha=0.25) # Type 5 & 7
glda2<-ggplot(data=DF.lda, aes(x=LD2,fill=as.factor(Type)))+geom_density(alpha=0.25) # Type 5
glda3<-ggplot(data=DF.lda, aes(x=LD3,fill=as.factor(Type)))+geom_density(alpha=0.25)
glda4<-ggplot(data=DF.lda, aes(x=LD4,fill=as.factor(Type)))+geom_density(alpha=0.25) # Type 3?

grid.arrange(glda1,glda2,glda3,glda4,ncol=2,top='Class Separation by Linear Discriminants')
```



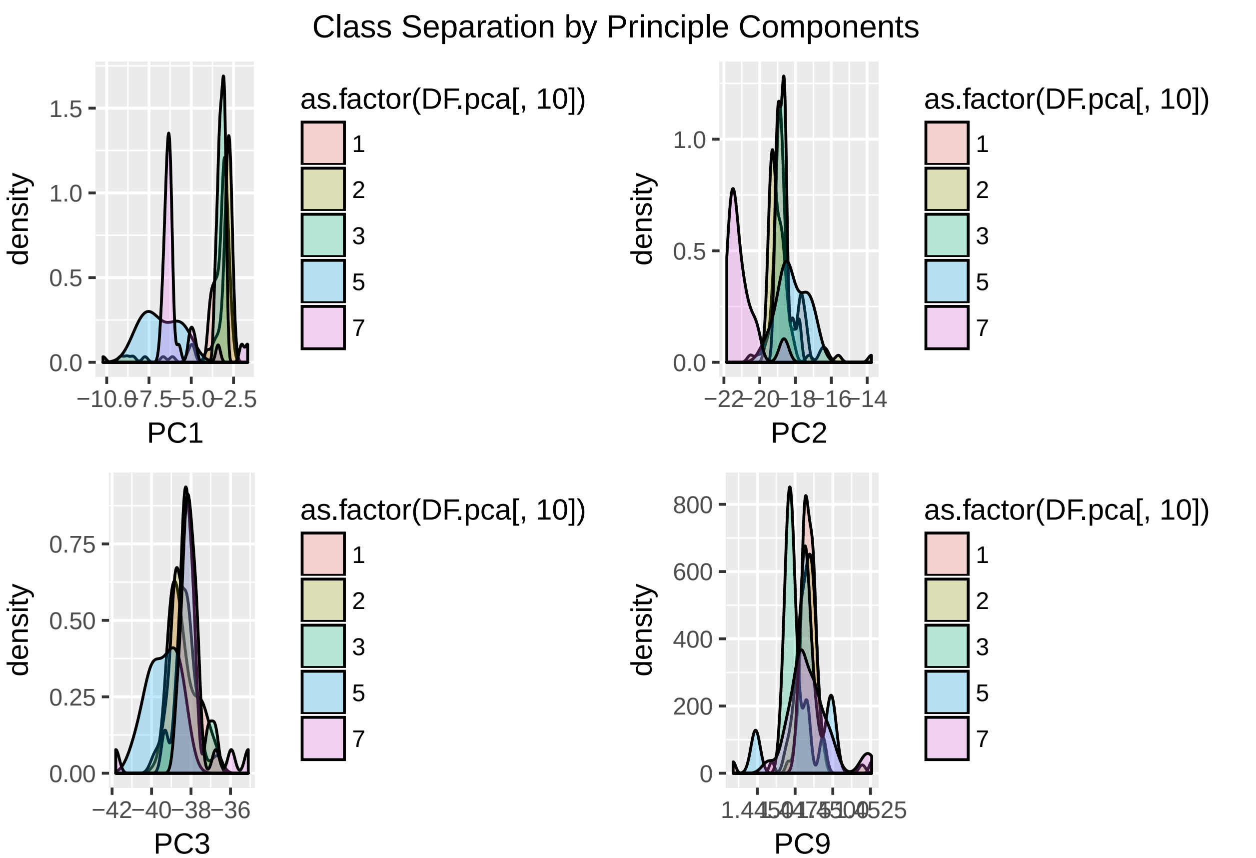
Principle Component Analysis

```r
PCA<-prcomp(x=DT.clean[,-c('Type'),with=FALSE])
PCA
```

```
## Standard deviations:
## [1] 1.6124839651 1.1441242559 0.7911784724 0.3664803591 0.2556042424
## [6] 0.2306272178 0.0968013232 0.0384479810 0.0009962787
##
## Rotation:
##             PC1          PC2          PC3           PC4          PC5
## RI -0.0007210930  0.001758772  0.001217291  9.151046e-05  0.001985627
## Na -0.1130827653 -0.326758805  0.702146747  3.496832e-01 -0.179858182
## Mg  0.7900985566  0.410108853  0.127527665  9.480438e-02 -0.093182290
## Al -0.1016043298 -0.270655758 -0.070674310 -5.744879e-01 -0.579576120
## Si  0.0091211999 -0.298701803 -0.658830844  5.700948e-01 -0.115233068
## K   0.0815081519 -0.001453638 -0.186149242 -4.454828e-01  0.366398979
## Ca -0.5792048837  0.708867522 -0.010138749  1.145344e-01 -0.061660636
## Ba -0.1021147585 -0.244360534  0.129586175 -3.631503e-02  0.686734288
## Fe -0.0003207267  0.018752751 -0.011194136 -2.823972e-02  0.011080860
##             PC6          PC7         PC8           PC9
## RI -0.0005119401  0.0005662999 0.004587502  0.9999846628
## Na  0.3222892447  0.0185282853 0.358591149 -0.0015269953
## Mg -0.1632330584  0.0690438121 0.377100613 -0.0019830876
## Al -0.2948633016  0.0795845618 0.390133508 -0.0002935833
```

```
## Si  0.0418136123  0.0412741401 0.367283662 -0.0001763299
## K   0.6892308801  0.0534586073 0.384836556 -0.0018417316
## Ca -0.0391923406  0.0708557641 0.372072666 -0.0033072261
## Ba -0.5502014184  0.0820612336 0.361496342 -0.0031484339
## Fe -0.0386621019 -0.9860221442 0.157696476 -0.0002238496
```

```
DT.pca<-as.matrix(DT.clean[,-c('Type'),with=FALSE])
DT.pca<- DT.pca %*% PCA$rotation
DF.pca<-data.frame(DT.pca,DT.clean$Type)

gpc1<-ggplot(data=DF.pca, aes(x=PC1,fill=as.factor(DF.pca[,10])))+geom_density(alpha=0.25)
gpc2<-ggplot(data=DF.pca, aes(x=PC2,fill=as.factor(DF.pca[,10])))+geom_density(alpha=0.25)
gpc3<-ggplot(data=DF.pca, aes(x=PC3,fill=as.factor(DF.pca[,10])))+geom_density(alpha=0.25)
gpc9<-ggplot(data=DF.pca, aes(x=PC9,fill=as.factor(DF.pca[,10])))+geom_density(alpha=0.25)

grid.arrange(gpc1,gpc2,gpc3,gpc9,ncol=2,top='Class Separation by Principle Components')
```



Feature Engineering for (123), 5, & 7

- F1
  - PC2 Separates Type 7 from Rest
- F2
  - Ba 25% Quantile, is higher that all other class maxima [Higher Ba ~ Higher Prob(Type 7)]
  - Can use Non-Parametric Bootstrap Simulations to Determine Quantile Threshold
- F3
  - PC3 Mildly Separates Type 5 from Rest

- F4
  - LD2 clearly separates Type 5
- F5
  - PC9 Mildly Separates Type 3 from Rest
- F6
  - Has Zero Potassium, Barium, or Iron (Unique Characteristic for that type of glass)
- F7
  - PC1 Separates 5 & 7 from Rest
- F8
  - LD1 May Be Used To Identity Groups 5 & 7
  - Making the Distributions More Peaked (kurtosis) May Help
- F9
  - Mean Mg of Combined Groups are Less than Mean Mg of Combined Groups 1,2,3
  - Use Non-Parametric Bootstrap Simulations to Determine Threshold

```
DT.feat<-data.table(DT.clean)

pca.2<-PCA$rotation[,2]
invisible(DT.feat[,F1:=RI*pca.2[1] +Na*pca.2[2] +Mg*pca.2[3] +Al*pca.2[4] +Si*pca.2[5] +K*pca.2[6] +Ca*p

sims<-matrix(nrow=10000,ncol=50)
quant<-vector(mode='numeric',length=10000)
samps.sin7<- DT.clean[(Type!='7')]$Ba
set.seed(123)
for(i in 1:10000){
    sims[i,]<-sample(x=samps.sin7, size=50,replace=TRUE)
    quant[i]<-quantile(sims[i,],probs=0.95)
}
Ba.Thresh<-mean(quant)
Ba.type7.mean<-mean(DT.feat[Type=='7']$Ba)
invisible(DT.feat[,F2:=ifelse(Ba>Ba.Thresh,exp(3.25/(abs(Ba-Ba.type7.mean)^1.7+Ba.type7.mean)),Ba)])

pca.3<-PCA$rotation[,3]
invisible(DT.feat[,F3:=RI*pca.3[1] +Na*pca.3[2] +Mg*pca.3[3] +Al*pca.3[4] +Si*pca.3[5] +K*pca.3[6] +Ca*p

ld.2<-lda.fit$scaling[,2]
invisible(DT.feat[,F4:=RI*ld.2[1] +Na*ld.2[2] +Mg*ld.2[3] +Al*ld.2[4] +Si*ld.2[5] +K*ld.2[6] +Ca*ld.2[7

pca.9<-PCA$rotation[,9]
invisible(DT.feat[,F5:=RI*pca.9[1] +Na*pca.9[2] +Mg*pca.9[3] +Al*pca.9[4] +Si*pca.9[5] +K*pca.9[6] +Ca*p

pca.1<-PCA$rotation[,1]
invisible(DT.feat[,F7:=RI*pca.1[1] +Na*pca.1[2] +Mg*pca.1[3] +Al*pca.1[4] +Si*pca.1[5] +K*pca.1[6] +Ca*p

ld.1<-lda.fit$scaling[,1]
invisible(DT.feat[,F8:=RI*ld.1[1] +Na*ld.1[2] +Mg*ld.1[3] +Al*ld.1[4] +Si*ld.1[5] +K*ld.1[6] +Ca*ld.1[7

sims<-matrix(nrow=10000,ncol=50)
quant<-vector(mode='numeric',length=10000)
samps<- DT.clean[(Type!='7')&(Type!='5')]$Mg
for(i in 1:10000){
    sims[i,]<-sample(samps, size=50,replace=TRUE)
    quant[i]<-quantile(sims[i,],probs=0.25)
}
```

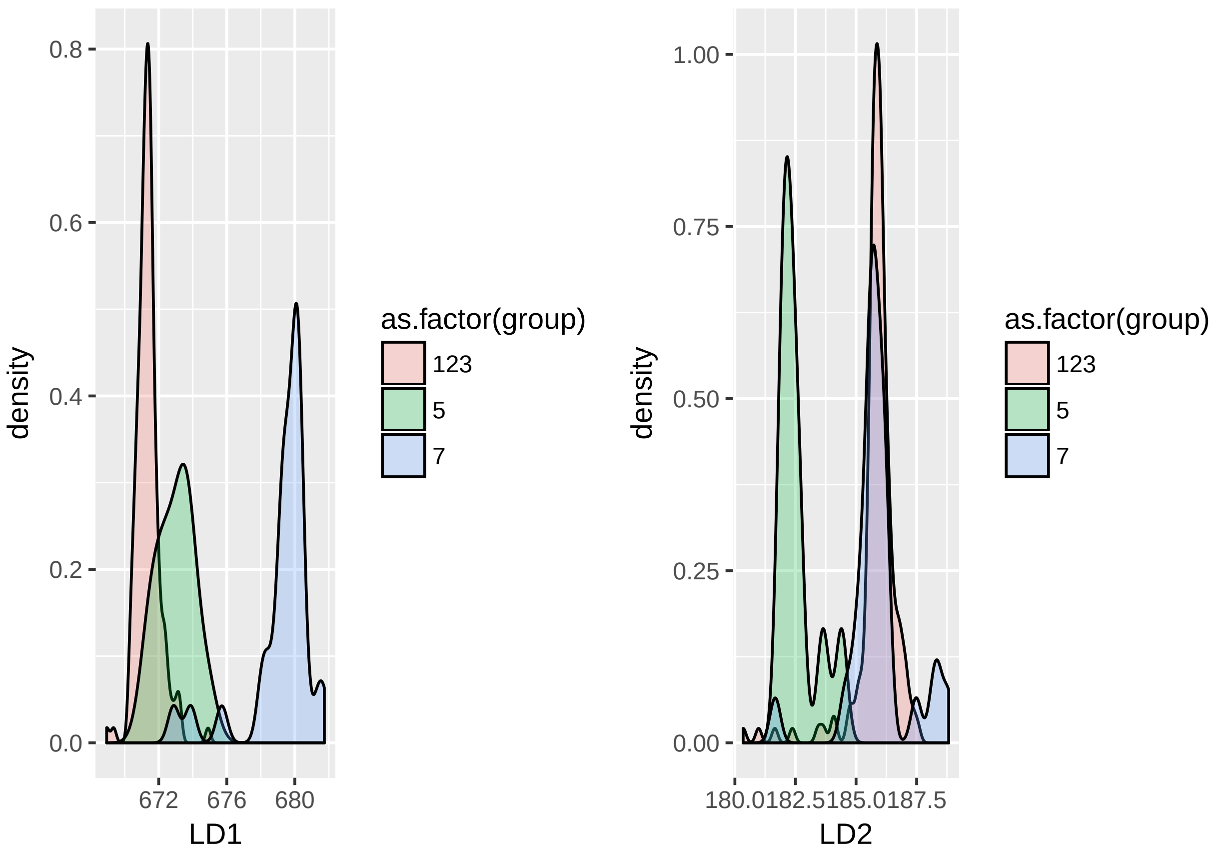
```r
F9_Thresh<-mean(quant)
invisible(DT.feat[,F9:=ifelse(Mg>=F9_Thresh,(Mg-F9_Thresh)^2,0)])

invisible(DT.feat[,group:=ifelse(Type=='7','7',ifelse(Type=='5','5','123'))])

invisible(DT.feat[,RI:=NULL])
invisible(DT.feat[,Na:=NULL])
invisible(DT.feat[,Mg:=NULL])
invisible(DT.feat[,Al:=NULL])
invisible(DT.feat[,Si:=NULL])
invisible(DT.feat[,K:=NULL])
invisible(DT.feat[,Ca:=NULL])
invisible(DT.feat[,Ba:=NULL])
invisible(DT.feat[,Fe:=NULL])
```

Linear Discriminant Analysis with Engineered Features

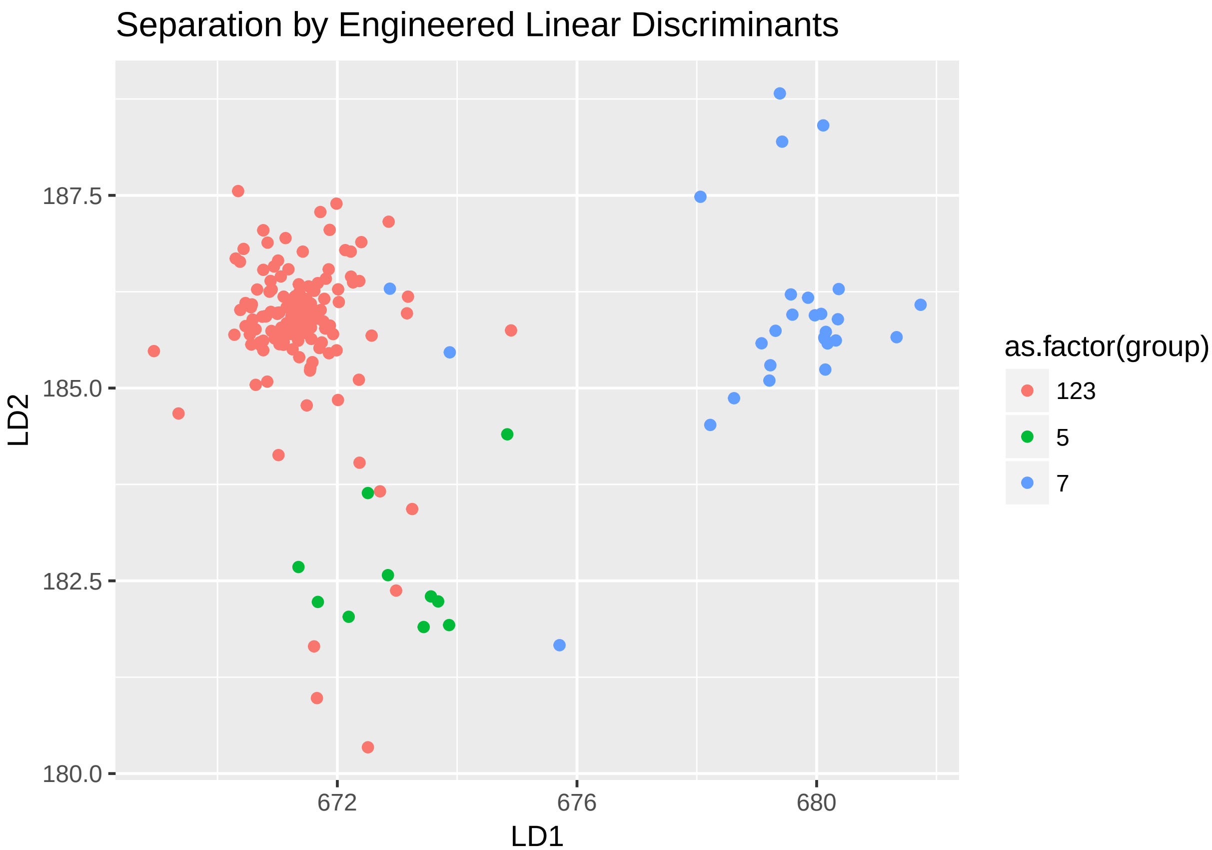
```r
lda.feat<-lda(group~.,data = DT.feat[,-c('Type'),with=FALSE])
lda.feat
```

```
## Call:
## lda(group ~ ., data = DT.feat[, -c("Type"), with = FALSE])
##
## Prior probabilities of groups:
##        123          5          7
## 0.80412371 0.05154639 0.14432990
##
## Group means:
##            F1         F2        F3       F4       F5        F7       F8
## 123 -18.73607  0.3391440 -38.39118 230.5380 1.448223 -3.360612 601.5229
## 5   -18.16495  0.7969379 -39.40369 227.5575 1.448247 -6.797092 603.6499
## 7   -21.02980 13.1053480 -38.02478 230.8040 1.448476 -5.781392 608.6650
##             F9
## 123 0.04584883
## 5   0.00000000
## 7   0.00000000
##
## Coefficients of linear discriminants:
##           LD1          LD2
## F1  0.2240685  -0.05818638
## F2  0.1581286   0.09777962
## F3 -0.1348076   0.41659577
## F4  0.1610714   0.50210871
## F5 64.5988374 112.05648516
## F7  0.0278715   0.33330453
## F8  0.8972250  -0.12657530
## F9  0.8870018  -1.91376967
##
## Proportion of trace:
##    LD1    LD2
## 0.9357 0.0643
```

```
#lda.feat$svd

mat.lda.feat<-as.matrix(DT.feat[,-c('group','Type'),with=FALSE])
mat.lda.feat<- mat.lda.feat %*% lda.feat$scaling
DF.lda.feat<-data.frame(mat.lda.feat,DT.feat$group)
names(DF.lda.feat)<-c('LD1','LD2','group')

g11<-ggplot(data=DF.lda.feat, aes(x=LD1,fill=as.factor(group)))+geom_density(alpha=0.25)
g12<-ggplot(data=DF.lda.feat, aes(x=LD2,fill=as.factor(group)))+geom_density(alpha=0.25)

grid.arrange(g11,g12,ncol=2)
```




```
ggplot(data=DF.lda.feat, aes(x=LD1,y=LD2,colour=as.factor(group)))+geom_point()+ggtitle('Separation by 
```

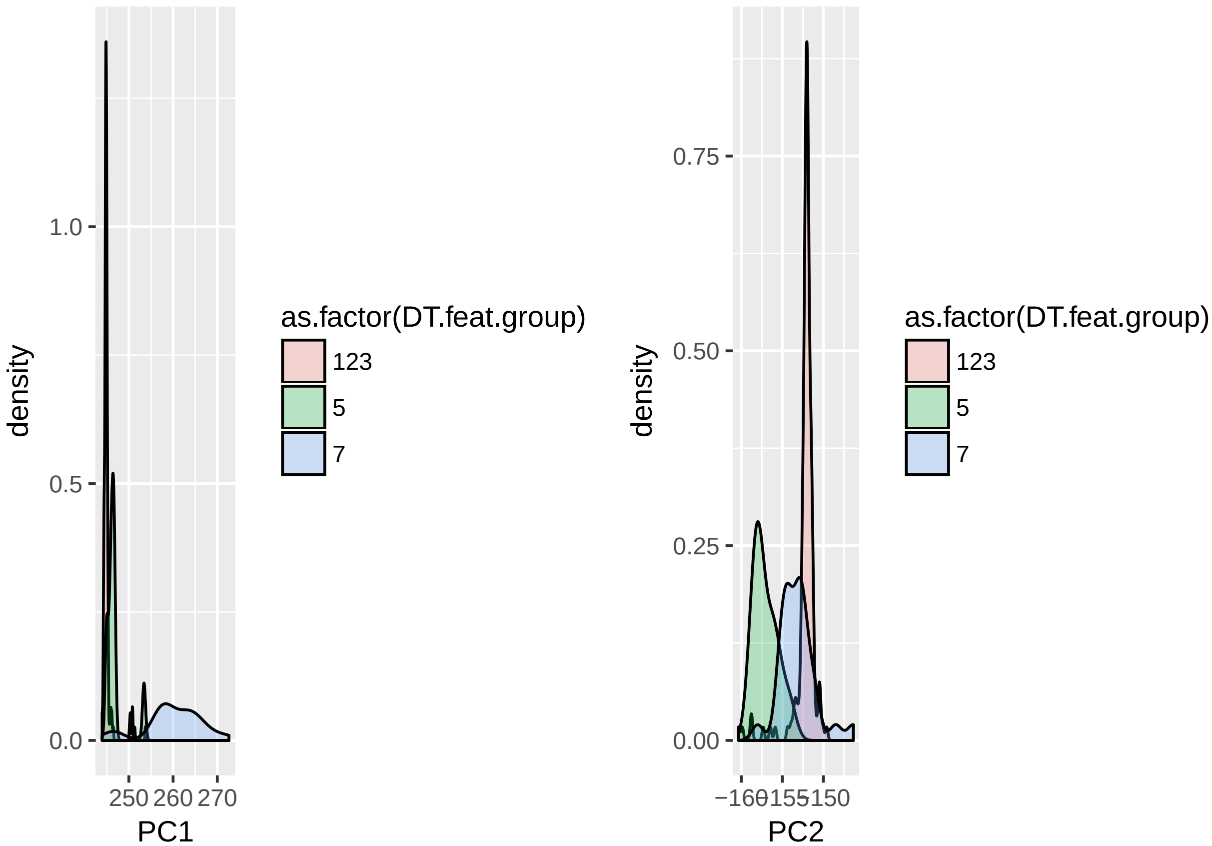


Principle Component Analysis with Engineered Features

```
PCA.feat<-prcomp(x=DT.feat[,-c('group','Type'),with=FALSE])
summary(PCA.feat)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     5.9307 1.92900 1.49599 0.95844 0.39081 0.28770
## Proportion of Variance 0.8316 0.08798 0.05291 0.02172 0.00361 0.00196
## Cumulative Proportion  0.8316 0.91961 0.97252 0.99424 0.99785 0.99981
##                            PC7       PC8
## Standard deviation     0.08963 0.0008772
## Proportion of Variance 0.00019 0.0000000
## Cumulative Proportion  1.00000 1.0000000
```

```
mat.pca.feat<-as.matrix(DT.feat[,-c('group','Type'),with=FALSE])
mat.pca.feat<- mat.pca.feat %*% PCA.feat$rotation
DF.pca.feat<-data.frame(mat.pca.feat,DT.feat$group)

g21<-ggplot(data=DF.pca.feat, aes(x=PC1,fill=as.factor(DT.feat.group)))+geom_density(alpha=0.25)
g22<-ggplot(data=DF.pca.feat, aes(x=PC2,fill=as.factor(DT.feat.group)))+geom_density(alpha=0.25)

grid.arrange(g21,g22,ncol=2)
```



Gradient Boosting for Types (123),5,and 7

PCA & LDA with the engineered features can distinguish between the glass types using the **2 linear discriminants** and the **first 2 principal components**.

- A single stochastic gradient boosting model can be built to classify the glass as:
  - Type 7 Glass
- Type 5 Glass
- Type 1, 2, or 3 Glass (Will Use a Second Model to Sub-Classify)

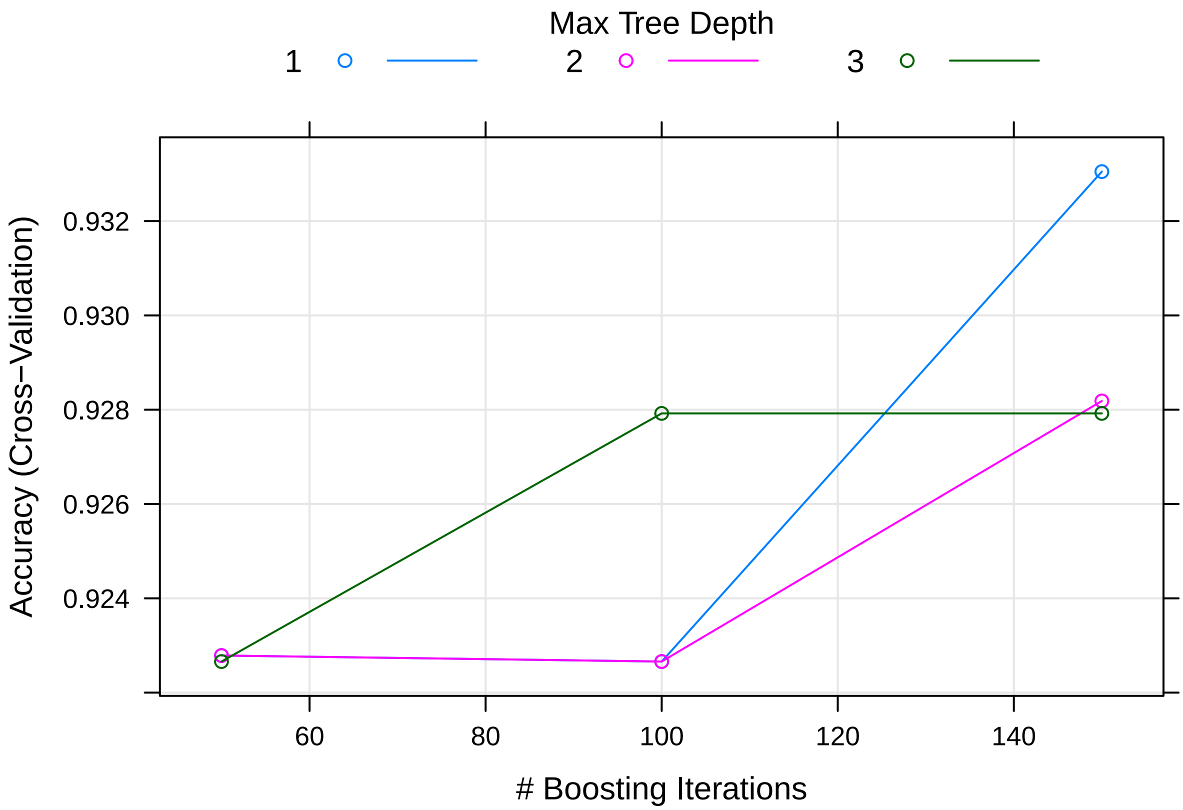
The final model consists of **150 trees** with **interaction depth of 1**. The CV **accuracy** is **0.9330499**. The CV **concordance(kappa)** is **0.8038155**.

```
DT.gbm<-data.table(DF.lda.feat,DF.pca.feat$PC1,DF.pca.feat$PC2)
setnames(DT.gbm,old=names(DT.gbm),new=c('LD1','LD2','group','PC1','PC2'))

# define training control
train_controlA<- trainControl(method="LOOCV")
train_controlB<- trainControl(method="cv", number=(5))

# train the model
set.seed(123)
gbm.fit<- train(group~., data=DT.gbm, trControl=train_controlB, method="gbm",verbose=FALSE)

gbm.fit
```

```
## Stochastic Gradient Boosting
```

```
##
## 194 samples
##   4 predictor
##   3 classes: '123', '5', '7'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 155, 154, 156, 156, 155
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9227868  0.7660125
##   1                  100      0.9226586  0.7760995
##   1                  150      0.9330499  0.8038155
##   2                   50      0.9227868  0.7693929
##   2                  100      0.9226586  0.7727191
##   2                  150      0.9281849  0.7910091
##   3                   50      0.9226586  0.7727191
##   3                  100      0.9279217  0.7878948
##   3                  150      0.9279217  0.7845379
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
plot(gbm.fit)
```




```
#summary(gbm.fit)
#confusionMatrix(predict(gbm.fit,DT.gbm),DT.gbm$group)
```

Gradient Boosting for Types 1, 2, & 3

- A second stochastic gradient boosting model can be built to classify the glass as:
  - Type 1, 2, or 3 Glass (Will Use a Second Model to Sub-Classify)

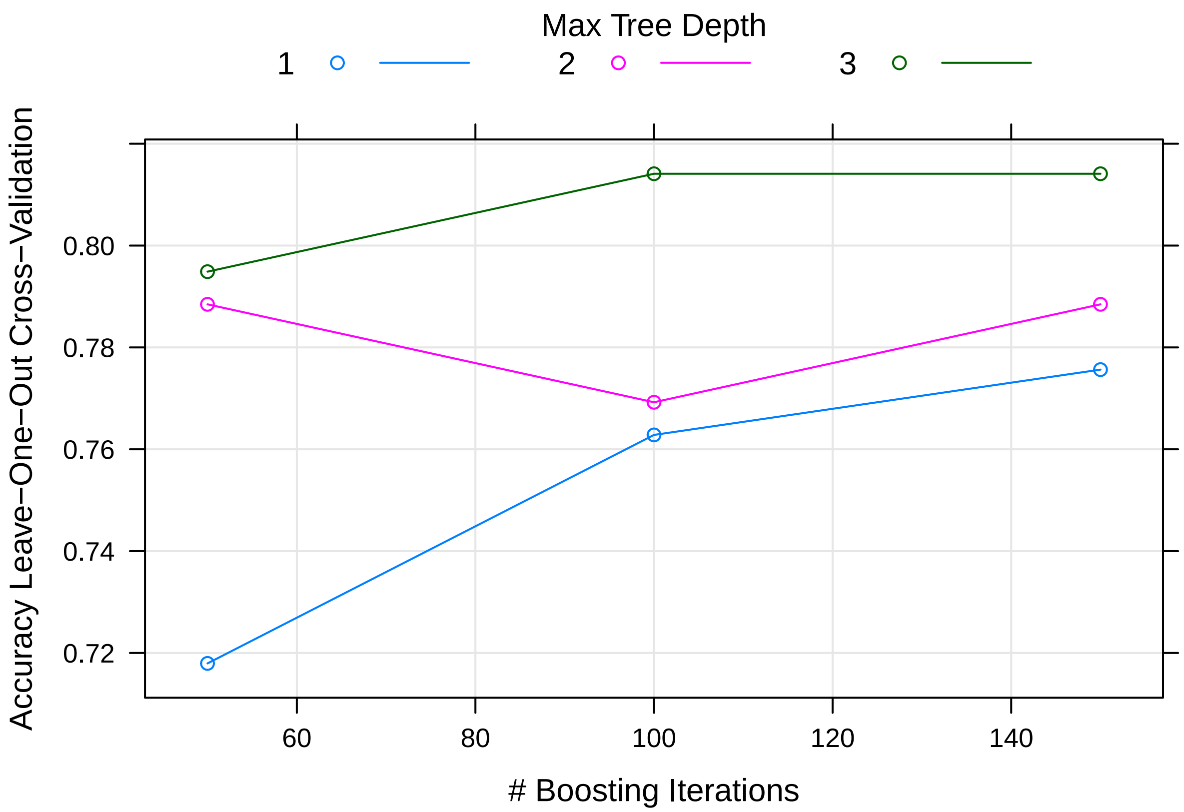
The final model consists of **100 trees** with **interaction depth of 3**. The CV **accuracy** is **0.8141026*. The CV **concordance(kappa)** is **0.6771339**.

```
DT123.clean<-data.table(DT.clean[(Type=='1')|(Type=='2')|(Type=='3')])
setattr(DT123.clean$Type,"levels",c('1','2','3'))

# define training control
train_controlA<- trainControl(method="LOOCV")

# train the model
set.seed(123)
gbm.fit123<- train(Type~.-Ba, data=DT123.clean, trControl=train_controlA, method="gbm",verbose=FALSE)

gbm.fit123
```

```
## Stochastic Gradient Boosting
##
## 156 samples
##   9 predictor
##   3 classes: '1', '2', '3'
```

```
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 155, 155, 155, 155, 155, 155, ...
## Resampling results across tuning parameters:
##
##   n.trees  interaction.depth  Accuracy   Kappa
##    50      1                  0.7179487  0.4930576
##    50      2                  0.7884615  0.6281422
##    50      3                  0.7948718  0.6422018
##   100      1                  0.7628205  0.5814965
##   100      2                  0.7692308  0.5924528
##   100      3                  0.8141026  0.6771339
##   150      1                  0.7756410  0.6038886
##   150      2                  0.7884615  0.6279271
##   150      3                  0.8141026  0.6756524
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 100,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
plot(gbm.fit123)
```




```r
#summary(gbm.fit123)
#confusionMatrix(predict(gbm.fit123,DT123.clean),DT123.clean$Type)
```

Summary

- Brute Force Gradient Boosting (Control)
  - CV **Accuracy** is **0.7850467**.
  - CV **Concordance(kappa)** is **0.7028226**.
- Omit 6
  - **Accuracy** is **0.9766355**.
  - **Sensitivity(recall)** is **1**.
  - **True Negative Rate(TNR)** is **0.9756098**.
  - **Positive predictive value(PPV)** is **0.6428571**.
  - **Negative predictive value(NPV)** is **1**.
- Gradient Boosting for Types (123), 5, & 7
  - CV **Accuracy** is **0.9330499**.
  - CV **Concordance(kappa)** is **0.8038155**.
- Gradient Boosting for Types 1,2, and 3
  - CV **Accuracy** is **0.8141026**.
  - CV **Concordance(kappa)** is **0.6771339**.