STAT 602_FINAL_GROUP 11

GROUP 11

4/28/2021

Introduction

Dry bean- Phaseolus vulgaris L. is a major cultivated grain species in the genus Phaseolus that is widely consumed worldwide for its edible legume and pea pods (Heuze et al., 2015). Nevertheless, selecting the best seed species is one of the main concerns for both bean producers and the market. Since different genotypes are cultivated worldwide, it is important to separate the best seed variety from the mixed dry bean population, otherwise the market value of these mixed species of beans could drop enormously (Varankaya & Ceyhan, 2012). The aim of our project is to develop an automated method to multiclass classification of dry beans that could predict the net worth of a given bean species harvested from a 'population cultivation' from a single farm when presented in the market.

Table 1: OLS Regression Model

lfdi_chem	Coef.	St.Err.	t-value	p-value	[95% Conf	Interval]	Sig
lrac	.574	.522	1.10	.279	-.485	1.633	
lav_wages	2.335	1.697	1.38	.177	-1.104	5.774	
lpop	-.831	.796	-1.04	.304	-2.444	.783	
lenergy	-2.193	.97	-2.26	.03	-4.158	-.228	**
lproximity	1.703	.558	3.05	.004	.573	2.833	***
lunion	-1.089	.54	-2.01	.051	-2.183	.006	*
lunemp	1.239	.707	1.75	.088	-.192	2.671	*
lmileage	.027	.455	0.06	.953	-.894	.948	
Constant	3.007	6.288	0.48	.635	-9.733	15.748	
Mean dep var	6.681	SD dep var	1.587				
R-squared	0.665	No of obs	46				
F-test	16.553	Prob > F	0.000				
AIC	139.687	BIC	156.144				
*** *p<.01*,	** *p<.05*,	* *p<.1*					

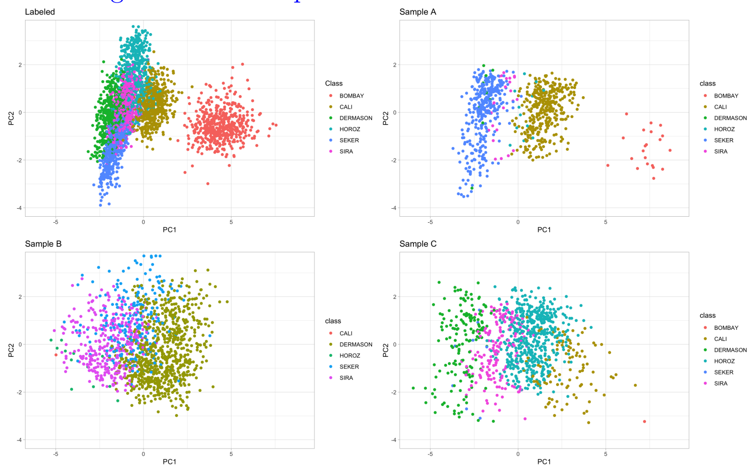
Methodology

This is a multiclass classification problem. Therefore, we tried five supervised classifiers including Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors (KNN), Random Forest (RF), and Support Vector Machines (SVM). In this project, we first use the labeled dataset to fit these models with different feature selections, and compare their leave-one-out cross validation (LOOCV) performance, then choose the model with the best performance and do prediction on sample A, B, and C datasets. The last step is to use bootstrap technique to do a simulation and measure the prediction accuracy.

Feature Selection: We have tried four different feature combinations. The first one uses all the 8 features. The second one uses only four variables: Area, Eccentricity, Extent, and Roundness in order to get rid of the effect of high correlation among other features. The third one uses only three variables: Area, Eccentricity, and Extend in order to see the effect of the newly added Roundness variable. The last one uses the first three principal components to check whether PCA helps denoise the dataset.

Model description: Both LDA and QDA classifiers are based on Bayes' theorem, with the assumption that every class is normally distributed. However, LDA has constant variance assumption among all classes, while QDA relaxes the assumption of constant variance among all classes. The LDA produces a linear decision boundary, while QDA produces a non-linear decision boundary. The QDA also requires more training data due to its non-constant variance assumption compared to LDA. The Random Forest classifier contains a large number of individual decision trees, where each individual tree in the random forest produces a class prediction and the class with the most votes becomes our model's prediction. LOOCV is used to select the optimal number of features, 'mtry' and optimal number of trees, 'ntree'. The KNN classifier predicts the observation class by finding the majority of the classes of the k-nearest training data points. Where, 'nearest' implies minimum Euclidean distance. LOOCV is used to find the optimal k (Figure 9). The SVM classifier

Figure 1: Scatter plot of abatement costs and FDI



source: US state-level data for the years 1977 – 1994 on FDI

identifies the best hyperplane that acts as a decision boundary among the different classes. We use a radial kernel for the SVM model in this project.

Results & Discussion

Our results (Figure 10) indicate that there is not much of a difference in performance for each of the five models while using all variables, four selected variables, and three selected variables. All these models underperform when three principal components are used. For each type of feature combination, QDA, Random Forest, and SVM consistently outperform other models. We selected QDA as our final model with all variables for predictions on sample A, B, and C. From Table 13, we see that all samples have a small number of predictions for BOMBAY. The classes with the highest prediction are CALI and SEKER for sample A, DERMASON for sample B, and HOROZ for sample C. We also visualize this comparison in Figure 12. Then we calculated the predicted price for each sample, the result is shown in the 'Predicted.Net.Worth' column of Table 17. Sample A is predicted to have higher price than Sample B and C. Finally, we construct the probability of count of each class given that the predicted class is one of the six classes (Table 16), and use it to do a bootstrap simulation and get the prediction interval for each sample (Table 17). Sample A has a narrower 2.5% to 97.5% price prediction interval, compared to Sample B and C.

Conclusion & Recommendation

QDA, Random Forest, and SVM did a good job of predicting the beans classes, their LOOCV accuracy rates are all 90%. Extent and Eccentricity are good predictors, Area, Perimeter, MajorAxisLength, MinorAxisLength, and ConvexArea are highly correlated, either one of them can be a good predictor. The newly added variable Roundness does not add much prediction power to our models. With the final model we used, QDA, it has a better prediction accuracy for Sample A compared to Sample B and C. We recommend using any one of these three models (QDA, Random Forest, and SVM), and including Extent, Eccentricity and at least one of the five highly correlated features as predictors to automate the classification of dry beans.

Appendix

```
labeled <- read.csv('labeled.csv') %>% dplyr::select(-X)
sampA <- read.csv('samp.A.csv')%>% dplyr::select(-X)
sampB <- read.csv('samp.B.csv')%>% dplyr::select(-X)
sampC <- read.csv('samp.C.csv')%>% dplyr::select(-X)
#convert Class into factor
labeled$Class <- as.factor(labeled$Class)

#set up a new variable 'Roundness'
#Roundness = 4*Area*pi/(perimeter)^2 (refer to the dry bean paper)
Roundess <- 4*pi*labeled$Area/(labeled$Perimeter)^2
labeled <- add_column(labeled, Roundness = Roundess, .after = 7)
sampA$Roundness <- 4*pi*sampA$Area/(sampA$Perimeter)^2
sampB$Roundness <- 4*pi*sampB$Area/(sampB$Perimeter)^2
sampC$Roundness <- 4*pi*sampC$Area/(sampC$Perimeter)^2

#check for duplicate rows
dup.rows = sum(labeled%>%duplicated(), sampA%>%duplicated(),
               sampB%>%duplicated(),sampC%>%duplicated())
```

Data Exploration

Summary Statistic

For this project, we used two datasets namely 'labeled' and 'unlabeled' sets. The labeled (training) dataset contains 3000 observations and 8 variables. The dependent variable has 6 levels (Classes): BOMBAY, CALI, DERMASON, HOROZ, SEKER, and SIRA. Each class has 500 observations. The unlabeled dataset is drawn from the three samples namely Sample A, B, and C. The total observations for sample A, B, and C are 777, 1373, and 982 respectively. Roundness, which is the measure of how closely the shape of beans approaches a perfect circle, was calculated and added as an additional predictor variable (Koklu & Ozkan, 2020). Tables 1 through 4 show the summary statistics of the variables in the labeled data, Sample A, B, and C, respectively .

```
summary.stats <- round(as.data.frame((labeled[,-9])%>%psych::describe())%>%dplyr::select(n,mean, sd, me
kable(summary.stats, caption="Statistical distribution of features of dry beans varieties (in pixels) -
```

Table 2: Statistical distribution of features of dry beans varieties (in pixels) - Label

	n	mean	sd	median	min	max	range	se
Area	3000	69874.978	49578.516	48714.500	20645.000	251320.000	230675.000	905.176
Perimeter	3000	1012.238	347.749	941.897	384.169	2164.100	1779.931	6.349
MajorAxisLength	3000	362.048	124.520	332.901	161.517	740.969	579.452	2.273
MinorAxisLength	3000	225.193	73.350	202.735	106.003	473.395	367.391	1.339
Eccentricity	3000	0.756	0.102	0.773	0.301	0.945	0.644	0.002
ConvexArea	3000	70944.115	50382.269	50807.500	8912.000	259965.000	251053.000	919.850
Extent	3000	0.753	0.052	0.766	0.571	0.850	0.279	0.001
Roundness	3000	0.840	0.294	0.771	0.391	2.056	1.664	0.005

The variables, Area and Convex Area, had the largest range for all four datasets. There are large differences in the range of variables, the variables with larger ranges will dominate over those with small ranges which may lead to biased results, therefore it is necessary to transform/scale these variables before fitting our distance-based models (i.e., KNN and SVM).

labeled data: variance check for each classes

```
var.tab1 <- labeled%>%group_by(Class)%>%summarize(Var.Area=var(Area),Var.Perimeter=var(Perimeter), var.M


aa <- kable(var.tab1, caption = "Variance of distribution")%>%kable_styling(latex_option=c("hold_positio
```

The variance of each variable by class shows evidence of non-constant variance (Tables 5 & 6). Based on the normality distribution and non-constant variance, we expect the QDA model to perform well.

Table 3: Variance of distribution

Class	Var.Area	Var.Perimeter	var.Maj.Axis.	var.Min.Axis.	var.Eccentricity	var.ConvexArea	var.Ex
BOMBAY	552697974	32015.80	3177.8992	826.6840	0.5472634	259965	0.8502
CALI	91528410	26272.79	1188.1780	491.4581	0.6183656	117510	0.8427
DERMASON	24651963	22913.81	696.7121	498.7684	0.5494947	56174	0.8471
HOROZ	56765885	24960.58	1252.4894	456.5644	0.7227374	82462	0.8420
SEKER	22567179	24170.41	736.8507	419.4165	0.3006355	65674	0.8183
SIRA	22641401	23977.06	782.4715	401.8085	0.6098838	73945	0.8418

Price per seed

Table 7 shows that Bombay has the highest grams and price per seed. The price per seed is the product of the price per pound and price per seed divided by the total weight of 453.592 grams.

```
classes <- c("BOMBAY", "CALI", "DERMASON", "HOROZ", "SEKER", "SIRA")
price.per.1b <- c("$5.56", "$6.02", "$1.98", "$2.43", "$2.72", "$5.40")
price.per.pound <- c(5.56, 6.02, 1.98, 2.43, 2.72, 5.40)
names(price.per.pound) <- classes
grams.per.seed <- c(1.92, 0.61, 0.28, 0.52, 0.49, 0.38)
names(grams.per.seed) <- classes
grams.per.pound <- 453.592
price.per.seed <- round(((price.per.pound*grams.per.seed)/grams.per.pound),6)
price.weight.data <-  cbind(price.per.1b, grams.per.seed, price.per.seed)
kable(price.weight.data, col.names=c("price per pound", "grams per seed", "price per seed"), caption="di
```

Table 4: distribution of types of dry beans and prices per seed

	price per pound	grams per seed	price per seed
BOMBAY	$5.56	1.92	0.023535
CALI	$6.02	0.61	0.008096
DERMASON	$1.98	0.28	0.001222
HOROZ	$2.43	0.52	0.002786
SEKER	$2.72	0.49	0.002938
SIRA	$5.40	0.38	0.004524

Histogram of each feature

Histogram of each feature - Labeled Data

The histograms from the labeled data (Figure 1) show evidence of multimodality behavior in the variables. This means that at least one of the classes of beans is very distinct from the others. The multimodality behavior is also shown in the histograms from Sample A (Figure 2), but not from Sample B or C (Figures 3 & 4). We expect to see very low predictions of BOMBAY for Sample B and C, because there is no multimodality behavior in their histograms.

```
grid.arrange(
labeled%>%ggplot() + geom_histogram(aes(x=Area), fill="light blue", col="brown")+ labs(title = "Area"),
labeled%>%ggplot() + geom_histogram(aes(x=Perimeter), fill="light blue", col="brown")+ labs(title = "Pe
labeled%>%ggplot() + geom_histogram(aes(x=MajorAxisLength), fill="light blue", col="brown")+ labs(title
labeled%>%ggplot() + geom_histogram(aes(x=MinorAxisLength), fill="light blue", col="brown")+ labs(title
labeled%>%ggplot() + geom_histogram(aes(x=ConvexArea), fill="light blue", col="brown")+ labs(title = "C
labeled%>%ggplot() + geom_histogram(aes(x=Roundness), fill="light blue", col="brown")+ labs(title = "Ro
labeled%>%ggplot() + geom_histogram(aes(x=Extent), fill="light blue", col="brown")+ labs(title = "Exten
labeled%>%ggplot() + geom_histogram(aes(x=Eccentricity), fill="light blue", col="brown")+ labs(title =
```

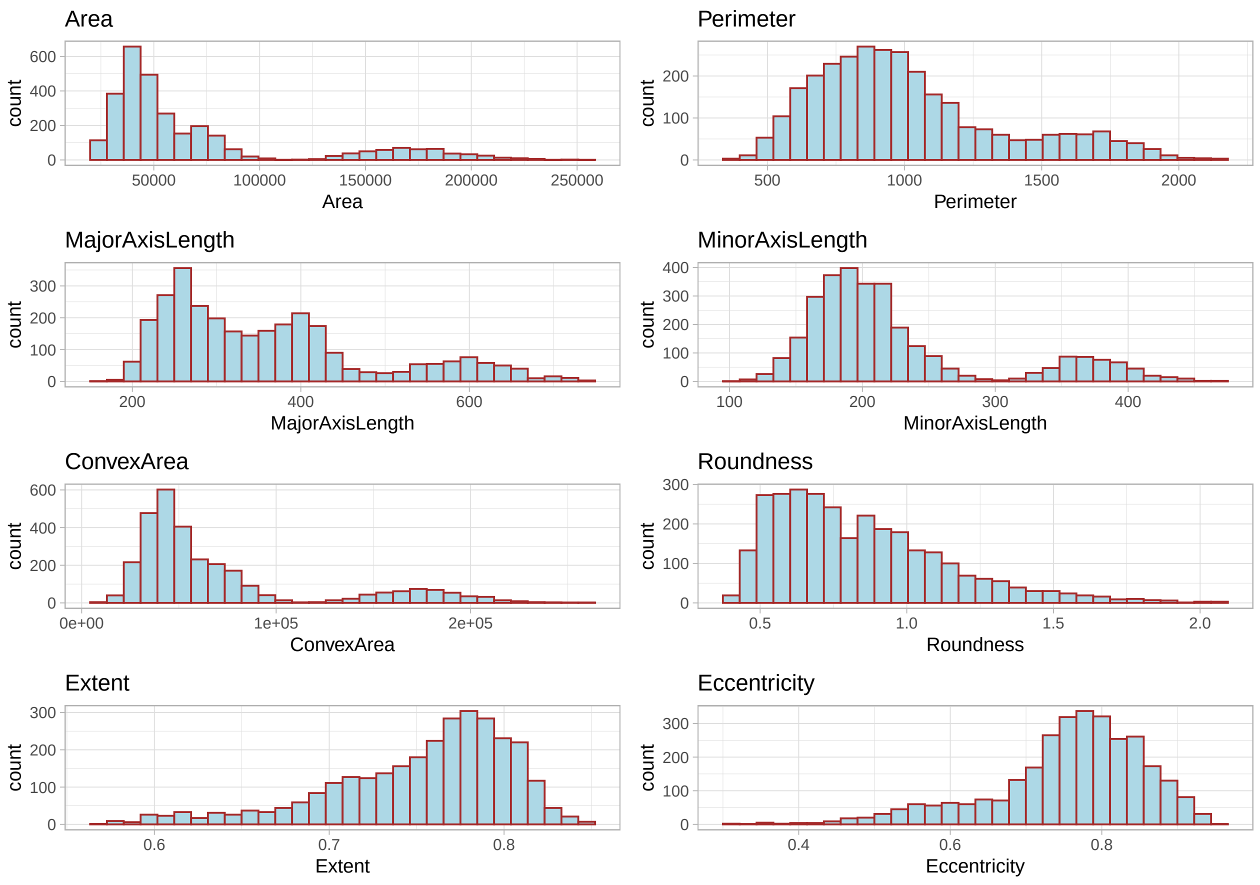


Figure 2: Histograms

Boxplot and Violin plots for each class

The boxplots from the labeled data (Figure 5) show that BOMBAY and CALI beans are very distinct from the other beans. It can be seen from the boxplots that Roundness and Extent seems to be a strong predictor for the SEKER. Eccentricity seems to be a good predictor to HOROZ. The violin plots for each class (Figure 5) shows that most of the class distributions are approximately normal except for the distributions for Roundness and Extent. From these distributions, we expect BOMBAY and CALI to be easily predicted by our models.

```
library(gridExtra)
grid.arrange(
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=Area), col="brown")  + labs(title =
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=Perimeter), col="brown") + labs(titl
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=MajorAxisLength), col="brown") + lab
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=MinorAxisLength), col="brown") + lab
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=ConvexArea), col="brown") + labs(tit
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=Roundness), col="brown") + labs(titl
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=Extent), col="brown") + labs(title =
labeled%>%group_by(Class)%>%ggplot() + geom_boxplot(aes(x=Class, y=Eccentricity), col="brown") + labs(t
```

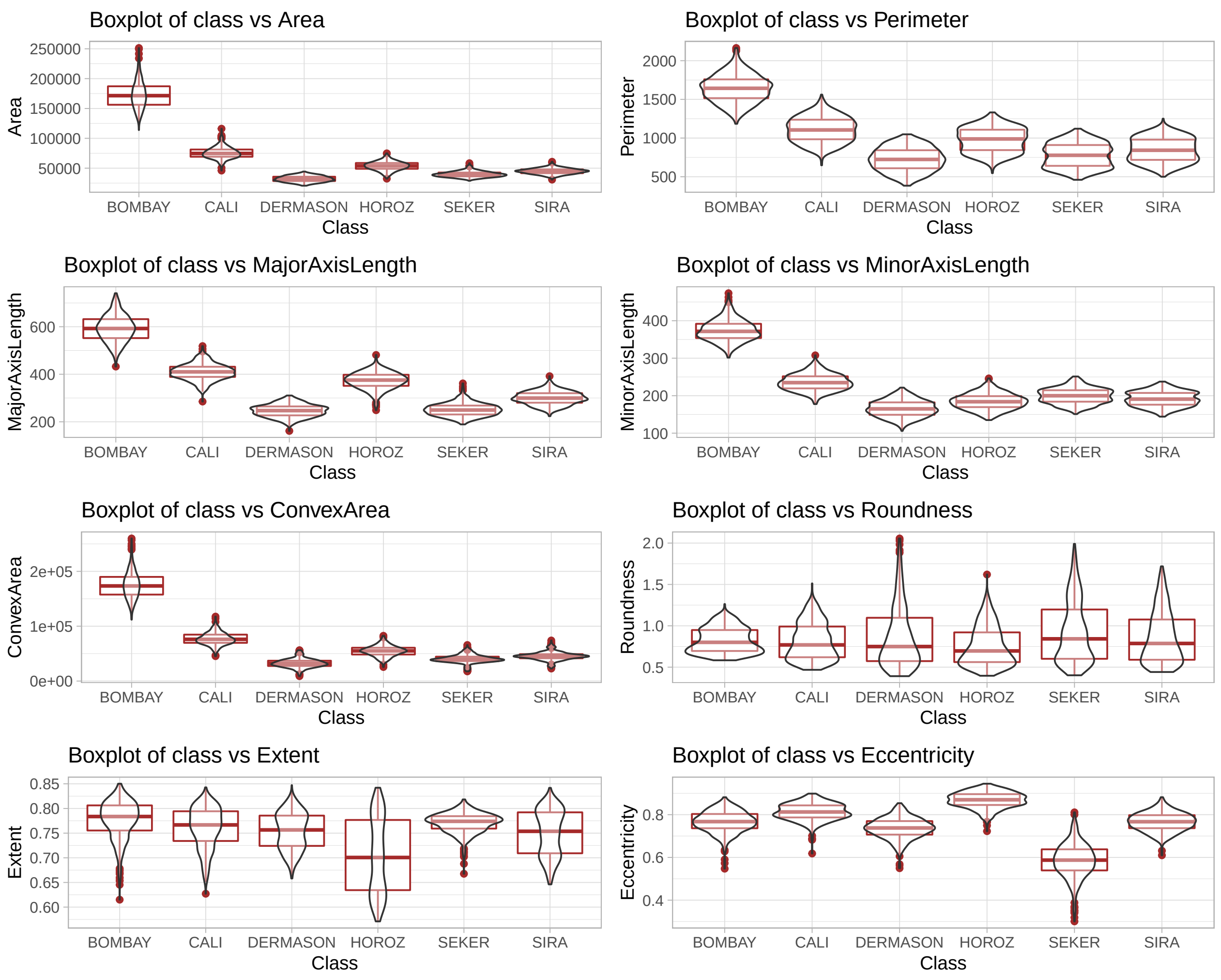


Figure 3: Boxplots and Voilin Polts of Variables by Classes

Correlation Plot

Most of the variables except for Eccentricity, Extent, and Roundness, are highly correlated (Figure 6) in each dataset. This behavior is also seen in the correlation of the variables by classes (Figure 7). The principal component analysis (Figure 8) indicates that the first 3 principal components, which are new variables that are constructed as linear combinations or mixtures of the initial variables, explained more than 90% of all variance in the dataset.

```r
par(mfrow =c(2,2))

corrplot(cor(labeled%>% dplyr::select(-Class)), method = 'ellipse', type = "lower")

corrplot(cor(sampA), method = 'ellipse', type = "lower")

corrplot(cor(sampB), method = 'ellipse', type = "lower")

corrplot(cor(sampC), method = 'ellipse', type = "lower")
```

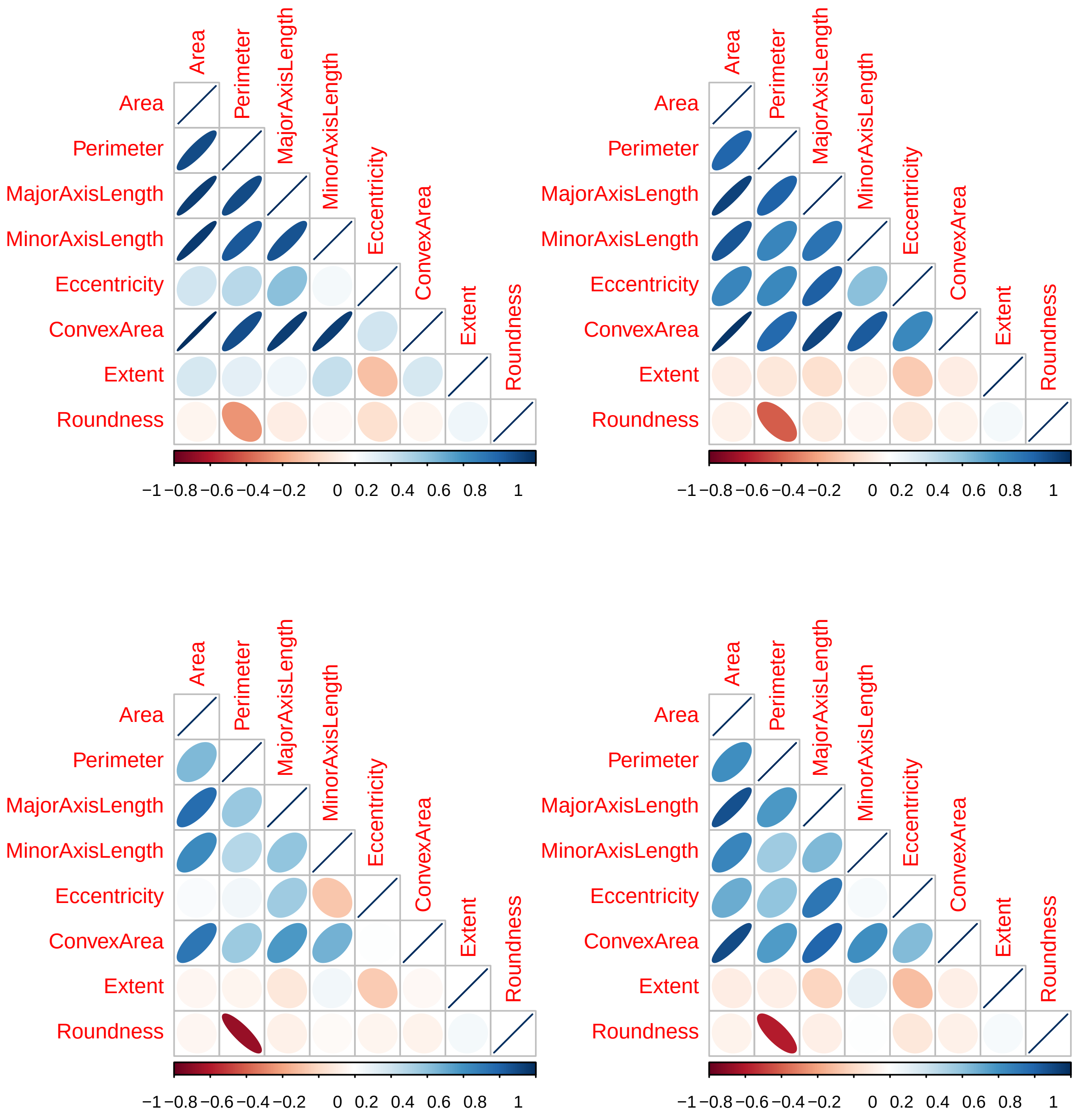


Figure 4: Correlation plot

Correlation plot by class for labeled dataset

```r
#labeled%>%filter(Class=="BOMBAY")%>%ggcorr()

par(mfrow=c(3,2))

test <- labeled%>%filter(Class=="BOMBAY")%>% dplyr::select(-Class)
corrplot(cor(test ), method = 'ellipse', type = "lower")

testb <- labeled%>%filter(Class=="CALI")%>% dplyr::select(-Class)
corrplot(cor(testb), method = 'ellipse', type = "lower")

test <- labeled%>%filter(Class=="DERMASON")%>% dplyr::select(-Class)
corrplot(cor(test), method = 'ellipse', type = "lower")

test <- labeled%>%filter(Class=="SEKER")%>% dplyr::select(-Class)
corrplot(cor(test), method = 'ellipse', type = "lower")

test <- labeled%>%filter(Class=="SIRA")%>% dplyr::select(-Class)
corrplot(cor(test), method = 'ellipse', type = "lower")

test <- labeled%>%filter(Class=="HOROZ")%>% dplyr::select(-Class)
corrplot(cor(test), method = 'ellipse', type = "lower")
```

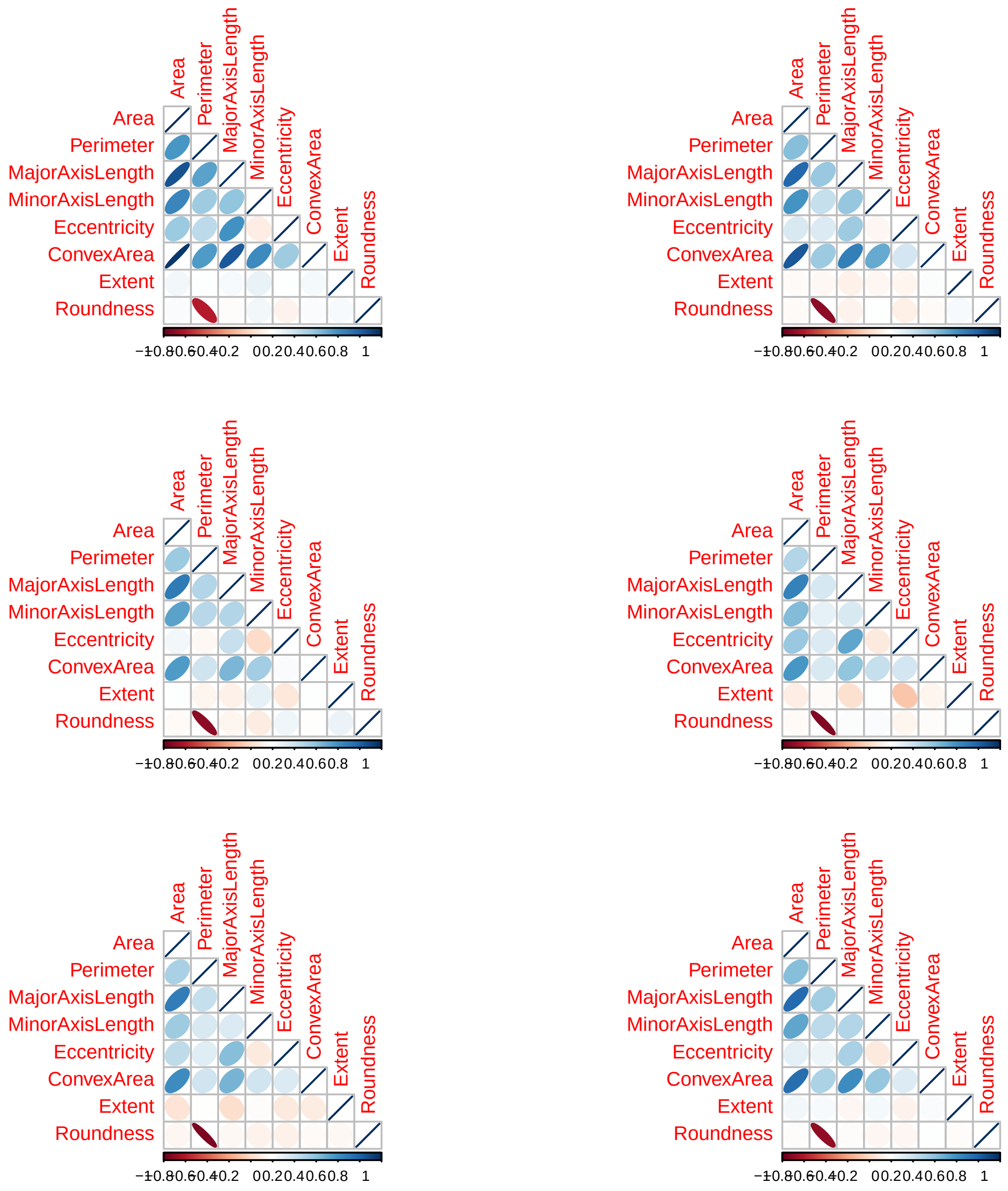


Figure 5: correlation plot by class for labeled dataset

Principle components analysis

```r
#####pca#####
pca.labeled <- prcomp(labeled %>% dplyr::select(-Class), scale = TRUE)
pca.sampA <- prcomp(sampA, scale = TRUE)
pca.sampB <- prcomp(sampB, scale = TRUE)
pca.sampC <- prcomp(sampC, scale = TRUE)
```

```r
#plot the variance explained by the first few principal components.
par(mfrow = c(4,2))
plot(pca.labeled, col="blue")
plot(pca.sampA, col="blue")
plot(pca.sampB, col="blue")
plot(pca.sampC, col="blue")
#plot the variance explained by the first few principal components.

plot(cumsum(pca.labeled$sdev^2 / sum(pca.labeled$sdev^2)),
     xlab = 'PC', ylab = 'Cumm Var Exp', main = 'pca.labeled', col="blue")
abline(h=0.9, col='red')
plot(cumsum(pca.sampA$sdev^2 / sum(pca.sampA$sdev^2)),
     xlab = 'PC', ylab = 'Cumm Var Exp', main = 'pca.sampA', col="blue")
abline(h=0.9, col='red')
plot(cumsum(pca.sampB$sdev^2 / sum(pca.sampB$sdev^2)),
     xlab = 'PC', ylab = 'Cumm Var Exp', main = 'pca.sampB', col="blue")
abline(h=0.9, col='red')
plot(cumsum(pca.sampC$sdev^2 / sum(pca.sampC$sdev^2)),
     xlab = 'PC', ylab = 'Cumm Var Exp', main = 'pca.sampC', col="blue")
abline(h=0.9, col='red')
```

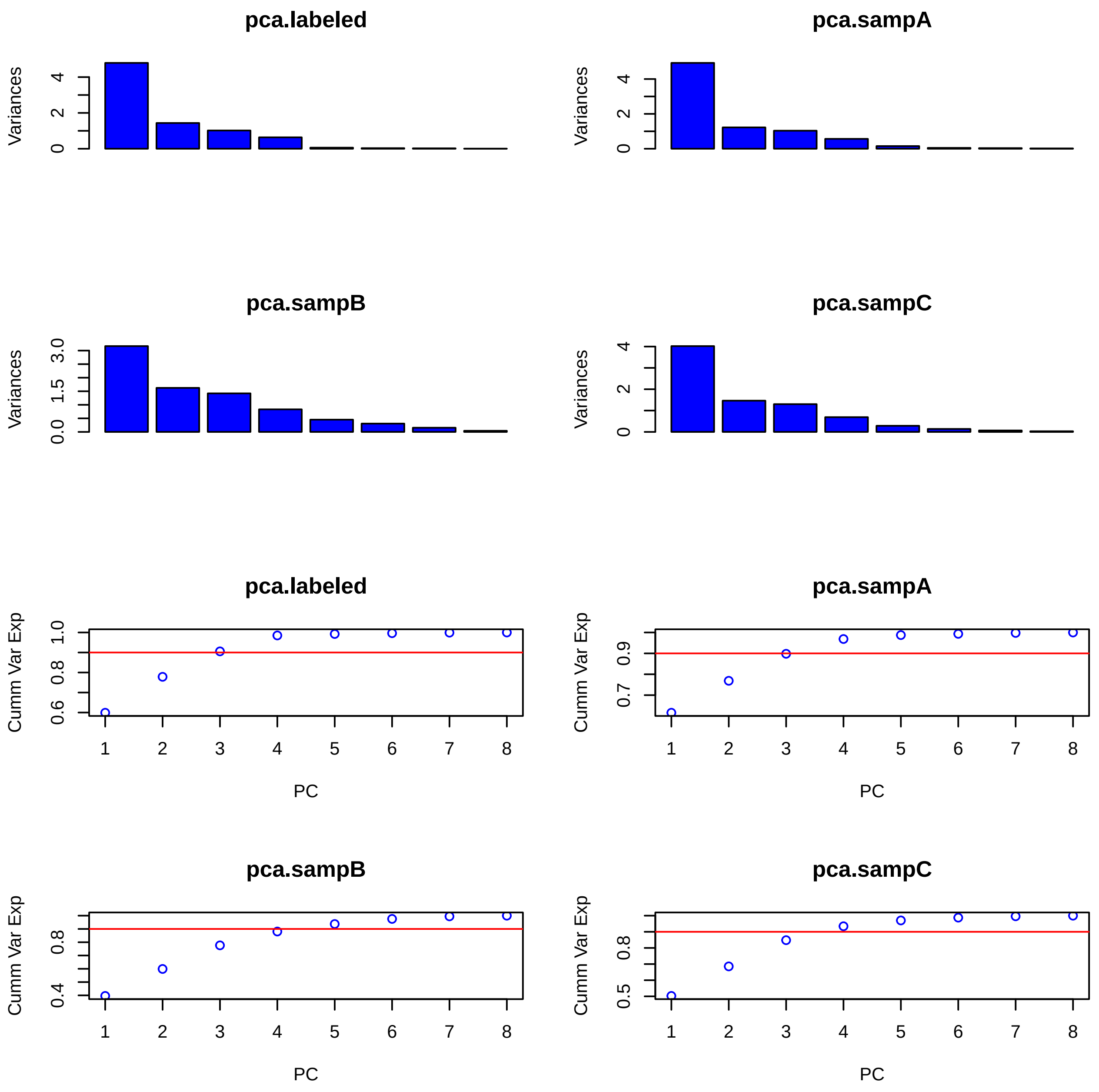


Figure 6: Variance explained by each components

```r
## Model performance function

perf.measure <- function(Preds, Truth){
  Preds <- as.character(Preds)
  Truth <- as.character(Truth)
  CV.tab.dat <- cbind(Preds, Truth)
  conf.tab <- xtabs(~Preds+Truth, CV.tab.dat)
  #accuracy rate
  accuracy.rate <- round(mean(Preds==Truth),2)
  #error rate
  error.rate <- round(1-accuracy.rate, 2)
  #each Class
  tp <- c(conf.tab[1,1], conf.tab[2,2], conf.tab[3,3],
        conf.tab[4,4], conf.tab[5,5], conf.tab[6,6])
  fp <- apply(conf.tab, 1, sum) - tp
  fn <- apply(conf.tab, 2, sum) - tp
  tn <- sum(conf.tab) - tp - fn - fp
  #precision (true positive among all predicted positive)
  precision.Class <- round(tp/(tp+fp),2)
  precision.Avg <- round(mean(tp/(tp+fp)),2)
  #recall (percent of all positives are corrected predicted)
  recall.Class <- round(tp/(tp+fn),2)
  recall.Avg <- round(mean(tp/(tp+fn)),2)
  #specificity (percent of all negatives are corrected predicted)
  specificity.Class <- round(tn/(tn+fp),2)
  specificity.Avg <- round(mean(tn/(tn+fp)),2)
  #F1.score = 2*precision*recall / (precision+recall)
  F1.score.Class <- round((2* tp/(tp+fp)* tp/(tp+fn))/(tp/(tp+fp) + tp/(tp+fn)),2)
  F1.score.Avg <- round(mean((2* tp/(tp+fp)* tp/(tp+fn))/(tp/(tp+fp) + tp/(tp+fn))),2)
  return(list(accuracy.rate = accuracy.rate, error.rate = error.rate,
              precision.Class = precision.Class, precision.Avg = precision.Avg,
              recall.Class = recall.Class, recall.Avg = recall.Avg,
              specificity.Class = specificity.Class, specificity.Avg = specificity.Avg,
              F1.score.Class = F1.score.Class, F1.score.Avg = F1.score.Avg,
              conf.tab <- conf.tab))
}
```

```r
## Construct labled.sc dataset and pca dataset

#construc scaled label data
labeled.sc <- as.data.frame(scale(labeled %>% dplyr::select(-Class)))
labeled.sc$Class <- labeled$Class

#construct pca label data
labeled.pca <- as.data.frame(pca.labeled$x)
labeled.pca$Class <- labeled$Class
```

```r
## Model validation (LOOCV)

## LDA


#fit lda and predict with CV (leave-one-out cross validation)
```

```r
lda.all <- lda(Class~., data = labeled, CV = TRUE)
lda.3var <- lda(Class ~ Area + Eccentricity + Extent,
                data = labeled, CV = TRUE)
lda.4var <- lda(Class ~ Area + Eccentricity + Extent + Roundness,
                data = labeled, CV = TRUE)
lda.3pca <- lda(Class ~ PC1 + PC2 + PC3,
                data = labeled.pca, CV = TRUE)

#lda CV performance
lda.all.perf <- perf.measure(Preds = lda.all$class, Truth = labeled$Class)
lda.3var.perf <- perf.measure(Preds = lda.3var$class, Truth = labeled$Class)
lda.4var.perf <- perf.measure(Preds = lda.4var$class, Truth = labeled$Class)
lda.3pca.perf <- perf.measure(Preds = lda.3pca$class, Truth = labeled$Class)
```

```r
## QDA

#fit qda and predict with CV (leave-one-out cross validation)
qda.all <- qda(Class~., data = labeled, CV = TRUE)
qda.3var <- qda(Class ~ Area + Eccentricity + Extent,
                data = labeled, CV = TRUE)
qda.4var <- qda(Class ~ Area + Eccentricity + Extent + Roundness,
                data = labeled, CV = TRUE)
qda.3pca <- qda(Class ~ PC1 + PC2 + PC3,
                data = labeled.pca, CV = TRUE)

#qda CV performance
qda.all.perf <- perf.measure(Preds = qda.all$class, Truth = labeled$Class)
qda.3var.perf <- perf.measure(Preds = qda.3var$class, Truth = labeled$Class)
qda.4var.perf <- perf.measure(Preds = qda.4var$class, Truth = labeled$Class)
qda.3pca.perf <- perf.measure(Preds = qda.3pca$class, Truth = labeled$Class)
```

```r
## Random Forest

### find optimal mtry (No. of variables tried at each split). Test mtry with 1 to n with minimum error

# all variables
set.seed(12345)
n <- ncol(labeled) -1
errRate <- c(1)
for (i in 1:n){
m <- randomForest(Class~.,data=labeled,mtry=i,CV=TRUE)
err<-mean(m$err.rate)
errRate[i] <- err
}
a= which.min(errRate)
# my result is 2


# three variables
labeled.3var<-labeled[,c("Area", "Eccentricity", "Extent", "Class")]
n <- ncol(labeled.3var) -1
errRate <- c(1)
for (i in 1:n){
```

```r
m <- randomForest(Class~.,data=labeled.3var,mtry=i,CV=TRUE)
err<-mean(m$err.rate)
errRate[i] <- err
}
b= which.min(errRate)
 # my result is 1


# four variables
labeled.4var<-labeled[,c("Area", "Eccentricity", "Extent", "Roundness", "Class")]
n <- ncol(labeled.4var) -1
errRate <- c(1)
for (i in 1:n){
m <- randomForest(Class~.,data=labeled.4var,mtry=i,CV=TRUE)
err<-mean(m$err.rate)
errRate[i] <- err
}
c= which.min(errRate)
 # my result is 2


# three pca
labeled.3pca<-labeled.pca[,c("PC1","PC2","PC3","Class")]
n <- ncol(labeled.3pca) -1
errRate <- c(1)
for (i in 1:n){
m <- randomForest(Class~.,data=labeled.3pca,mtry=i,CV=TRUE)
err<-mean(m$err.rate)
errRate[i] <- err
}
d= which.min(errRate)
# my result is 2

# find optimal ntree(number of decision tree we want to create in our random forest). Fit random forest
# The black solid line in the plot means Out-of-Bag error rate , the other dotted line means each class
# Acctually, we should use as many ntree as we can since lower ntree lead higher error rate for model,

par(mfrow=c(2,2))
set.seed(12345)
# all variables
forest.all<-randomForest(Class~., data = labeled, mtry=2)
plot(forest.all, main ="random forest for all variables") #ntree=500

# three variables
forest.3var<-randomForest(Class~., data = labeled.3var, mtry=1)
plot(forest.3var, main ="random forest for three variables") # ntree=500

# four variables
forest.4var<-randomForest(Class~., data = labeled.4var, mtry=2)
plot(forest.4var, main ="random forest for four variables") # ntree=500

# three pca
forest.3pca<-randomForest(Class~., data = labeled.3pca, mtry=2)
```

```
plot(forest.3pca, main ="random forest for three pca") #ntree=500

# Since we cannot get straight error rate line in each plot and error rate does not change a lot, we us

rf.opt <- cbind(rbind(a, b, c, d), rep("500", 5))
rownames(rf.opt) <- c("All variables", "3 variables", "4 variables", "5 variables")
kable(rf.opt, caption="Optimal parameters for Random forest model", col.names = c("Opt no of features",
```

```
## Random Forest

set.seed(12345)
#fit randomForest and predict with CV (leave-one-out cross validation)
forest.all<-randomForest(Class~., data = labeled,CV = TRUE, ntree=500,mtry=2)
forest.3var<-randomForest(Class~Area + Eccentricity + Extent, data = labeled, CV = TRUE, ntree=500,mtry=
forest.4var<-randomForest(Class~Area + Eccentricity + Extent + Roundness, data = labeled, CV = TRUE, nt
forest.3pca<-randomForest(Class~PC1 + PC2 + PC3, data = labeled.pca, CV = TRUE,ntree=500,mtry=2)

#randomForest CV performance
forest.all.perf <- perf.measure(Preds = forest.all$predicted, Truth = labeled$Class)
forest.3var.perf <- perf.measure(Preds = forest.3var$predicted, Truth = labeled$Class)
forest.4var.perf <- perf.measure(Preds = forest.4var$predicted, Truth = labeled$Class)
forest.3pca.perf <- perf.measure(Preds = forest.3pca$predicted, Truth = labeled$Class)
```

```
## KNN


set.seed(12345)
AR.all <- NULL
for (k in 1:100) {
test <- knn.cv(labeled.sc[,1:8],cl=labeled$Class, k)
AR.all[k] <- mean(test==labeled.sc$Class)
}
k.all <- which(AR.all==max(AR.all)) # my result is 15/17
knn.all.sc <- knn.cv(labeled.sc[,1:8],
                     cl=labeled.sc$Class, k=k.all[1])
###
set.seed(12345)
AR.3var <- NULL
for (k in 1:100) {
test <- knn.cv(labeled.sc[,c("Area", "Eccentricity", "Extent")],
               cl=labeled.sc$Class, k)
AR.3var[k] <- mean(test==labeled.sc$Class)
}
k.3var=which(AR.3var==max(AR.3var)) # my result is 17
knn.3var.sc <- knn.cv(labeled.sc[,c("Area", "Eccentricity", "Extent")],
                      cl=labeled.sc$Class, k=k.3var[1])
####
set.seed(12345)
AR.4var <- NULL
for (k in 1:100) {
test <- knn.cv(labeled.sc[,c("Area", "Eccentricity", "Extent", "Roundness")],
               cl=labeled.sc$Class, k)
AR.4var[k] <- mean(test==labeled.sc$Class)
```

```r
}
k.4var=which(AR.4var==max(AR.4var)) # my result is 15
knn.4var.sc <- knn.cv(labeled.sc[,c("Area", "Eccentricity", "Extent", "Roundness")],
                      cl=labeled.sc$Class, k=k.4var[1])
###
set.seed(12345)
AR.3pca <- NULL
for (k in 1:100) {
test <- knn.cv(labeled.pca[,1:3],cl=labeled.pca$Class, k)
AR.3pca[k] <- mean(test==labeled.pca$Class)
}
k.3pca=which(AR.3pca==max(AR.3pca)) # my result is 18/19
knn.3pca.sc <- knn.cv(labeled.pca[,1:3],cl=labeled.pca$Class, k=k.3pca[1])
```

```r
#knn optimual parameters plot
par(mfrow = c(2,2))
plot(AR.all, ylim=c(0.8,0.9), xlab = 'k value', ylab = 'LOOCV accuracy rate',
     main = 'knn model with all variables')
abline(v=k.all[1], col = 'red')
legend(x=k.all[1], y=0.85, legend = paste('optimal k=',k.all[1]), bty='n')

plot(AR.3var, ylim=c(0.8,0.9), xlab = 'k value', ylab = 'LOOCV accuracy rate',
     main = 'knn model with 3 variables')
abline(v=k.3var[1], col = 'red')
legend(x=k.3var[1], y=0.85, legend = paste('optimal k=',k.3var[1]), bty='n')

plot(AR.4var, ylim=c(0.8,0.9), xlab = 'k value', ylab = 'LOOCV accuracy rate',
     main = 'knn model with 4 variables')
abline(v=k.4var[1], col = 'red')
legend(x=k.4var[1], y=0.88, legend = paste('optimal k=',k.4var[1]), bty='n')

plot(AR.3pca, ylim=c(0.8,0.9), xlab = 'k value', ylab = 'LOOCV accuracy rate',
     main = 'knn model with first three pca variables')
abline(v=k.3pca[1], col = 'red')
legend(x=k.3pca[1], y=0.88, legend = paste('optimal k=',k.3pca[1]), bty='n')
```

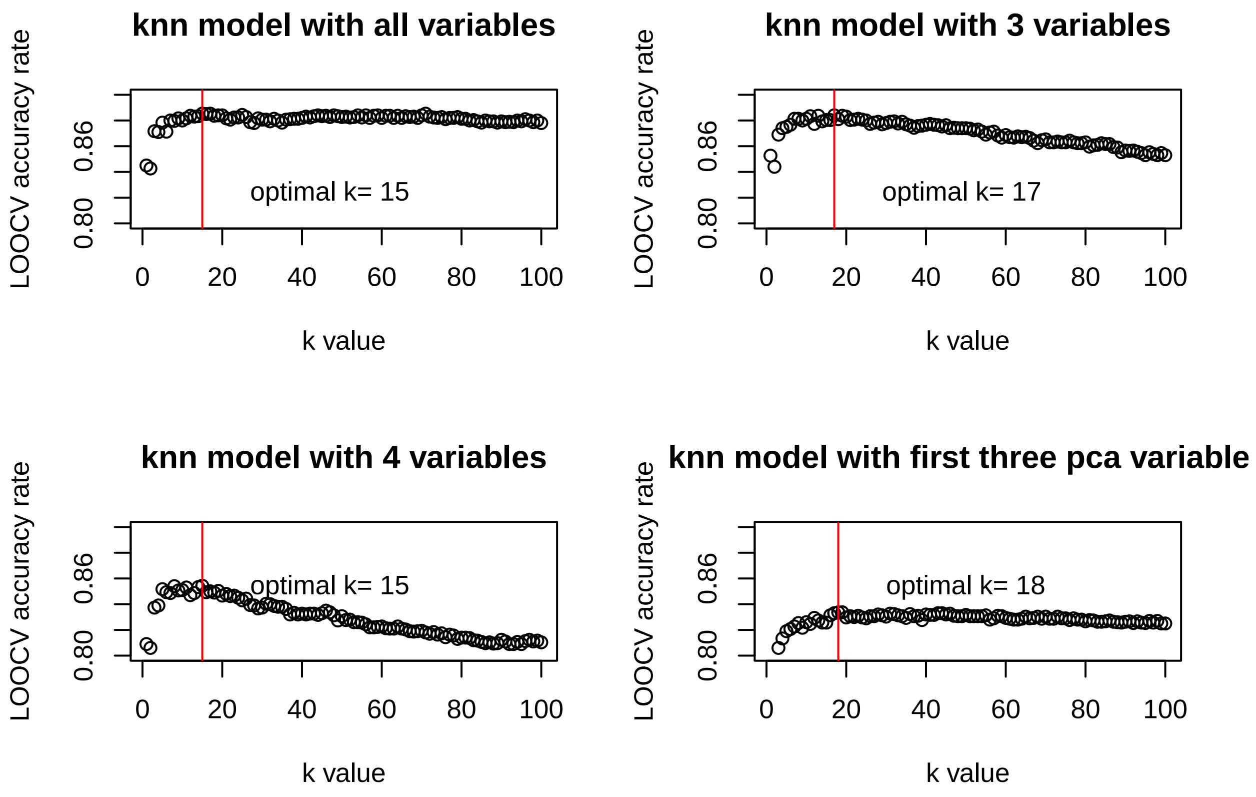


Figure 7: optimal k value choices plots for knn model

```r
#knn CV performance
knn.all.sc.perf <- perf.measure(Preds = knn.all.sc, Truth = labeled$Class)
knn.3var.sc.perf <- perf.measure(Preds = knn.3var.sc, Truth = labeled$Class)
knn.4var.sc.perf <- perf.measure(Preds = knn.4var.sc, Truth = labeled$Class)
knn.3pca.perf <- perf.measure(Preds = knn.3pca.sc, Truth = labeled$Class)
```

```r
## SVM
## all variables
set.seed(12345)
svm_radial.all <-  as.factor(NULL)
levels(svm_radial.all) <- levels(labeled$Class)
for(i in 1:3000){
  train <- labeled[-i,]
  test <- labeled[i,]
  mol <- svm(Class ~., data = train, scale = TRUE, kernel = 'radial')
  svm_radial.all[i] <- predict(mol, newdata = test)
}

all<-as.matrix(svm_radial.all)
write.csv(all,file="svm_radial.all.csv")

## three variables
set.seed(12345)
svm_radial.3var <-  as.factor(NULL)
levels(svm_radial.3var) <- levels(labeled$Class)
for(i in 1:3000){
  train <- labeled[-i,]
  test <- labeled[i,]
  mol <- svm(Class ~ Area + Eccentricity + Extent, data = train, scale = TRUE, kernel = 'radial')
  svm_radial.3var[i] <- predict(mol, newdata = test[,c("Area", "Eccentricity", "Extent")])
}

var3<-as.matrix(svm_radial.3var)
write.csv(var3,file="svm_radial.3var.csv")

## four variables
set.seed(12345)
svm_radial.4var <-  as.factor(NULL)
levels(svm_radial.4var) <- levels(labeled$Class)
for(i in 1:3000){
  train <- labeled[-i,]
  test <- labeled[i,]
  mol <- svm(Class ~ Area + Eccentricity + Extent + Roundness, data = train, scale = TRUE, kernel = 'ra
  svm_radial.4var[i] <- predict(mol, newdata = test[,c("Area", "Eccentricity", "Extent", "Roundness")])
}

var4<-as.matrix(svm_radial.4var)
write.csv(var4,file="svm_radial.4var.csv")

##3pca
set.seed(12345)
svm_radial.3pca <-  as.factor(NULL)
levels(svm_radial.3pca) <- levels(labeled.pca$Class)
for(i in 1:3000){
  train <- labeled.pca[-i,]
  test <- labeled.pca[i,]
  mol <- svm(Class ~ PC1 + PC2 + PC3, data = train, scale = FALSE, kernel = 'radial')
  svm_radial.3pca[i] <- predict(mol, newdata = test[,c("PC1", "PC2", "PC3")])
}

pca3<-as.matrix(svm_radial.3pca)
write.csv(pca3,file="svm_radial.3pca.csv")
# Use scale = FALSE since pca data already be scaled data.
```

```
## Because we do loocv svm mannually (write a loop), it takes a long time to run.
## We decided to save loocv prediction result and reload here to save knitting time.
svm_radial.all.sc <- read.csv('svm_radial.all.csv')
svm_radial.3var.sc <- read.csv('svm_radial.3var.csv')
svm_radial.4var.sc <- read.csv('svm_radial.4var.csv')
svm_radial.3pca.sc <- read.csv('svm_radial.3pca.csv')

svm.all.sc.perf <- perf.measure(Preds = svm_radial.all.sc$V1, Truth = labeled$Class)
svm.3var.sc.perf <- perf.measure(Preds = svm_radial.3var.sc$V1, Truth = labeled$Class)
svm.4var.sc.perf <- perf.measure(Preds = svm_radial.4var.sc$V1, Truth = labeled$Class)
svm.3pca.perf <- perf.measure(Preds = svm_radial.3pca.sc$V1, Truth = labeled$Class)
```

```r
## construct performance table and plot

COL.NAME <- c('lda', 'qda', 'RandomForest', 'knn.sc','svm.sc')
Row.NAME <- c('all.var', '3var', '4var', '3pca')

accuracy.rate <- as.data.frame(rbind(c(lda.all.perf$accuracy.rate, qda.all.perf$accuracy.rate,
                                       forest.all.perf$accuracy.rate,
                                       knn.all.sc.perf$accuracy.rate,svm.all.sc.perf$accuracy.rate),
                                     c(lda.3var.perf$accuracy.rate, qda.3var.perf$accuracy.rate,
                                       forest.3var.perf$accuracy.rate,
                                       knn.3var.sc.perf$accuracy.rate,svm.3var.sc.perf$accuracy.rate),
                                     c(lda.4var.perf$accuracy.rate, qda.4var.perf$accuracy.rate,
                                       forest.4var.perf$accuracy.rate,
                                       knn.4var.sc.perf$accuracy.rate,svm.4var.sc.perf$accuracy.rate),
                                     c(lda.3pca.perf$accuracy.rate, qda.3pca.perf$accuracy.rate,
                                       forest.3pca.perf$accuracy.rate,
                                       knn.3pca.perf$accuracy.rate,svm.3pca.perf$accuracy.rate)))

colnames(accuracy.rate) <- COL.NAME
rownames(accuracy.rate) <- Row.NAME


#precision (true positive among all predicted positive)
precision.Avg <- as.data.frame(rbind(c(lda.all.perf$precision.Avg, qda.all.perf$precision.Avg,
                                       forest.all.perf$precision.Avg,
                                       knn.all.sc.perf$precision.Avg,svm.all.sc.perf$precision.Avg),
                                     c(lda.3var.perf$precision.Avg, qda.3var.perf$precision.Avg,
                                       forest.3var.perf$precision.Avg,
                                       knn.3var.sc.perf$precision.Avg,svm.3var.sc.perf$precision.Avg),
                                     c(lda.4var.perf$precision.Avg, qda.4var.perf$precision.Avg,
                                       forest.4var.perf$precision.Avg,
                                       knn.4var.sc.perf$precision.Avg,svm.4var.sc.perf$precision.Avg),
                                     c(lda.3pca.perf$precision.Avg, qda.3pca.perf$precision.Avg,
                                       forest.3pca.perf$precision.Avg,
                                       knn.3pca.perf$precision.Avg,svm.all.sc.perf$precision.Avg)))

colnames(precision.Avg) <- COL.NAME
rownames(precision.Avg) <- Row.NAME


#recall (percent of all positives are corrected predicted)
recall.Avg <- as.data.frame(rbind(c(lda.all.perf$recall.Avg, qda.all.perf$recall.Avg,
                                       forest.all.perf$recall.Avg,
                                       knn.all.sc.perf$recall.Avg,svm.all.sc.perf$recall.Avg),
                                     c(lda.3var.perf$recall.Avg, qda.3var.perf$recall.Avg,
                                       forest.3var.perf$recall.Avg,
                                       knn.3var.sc.perf$recall.Avg,svm.3var.sc.perf$recall.Avg),
                                     c(lda.4var.perf$recall.Avg, qda.4var.perf$recall.Avg,
                                       forest.4var.perf$recall.Avg,
                                       knn.4var.sc.perf$recall.Avg,svm.4var.sc.perf$recall.Avg),
                                     c(lda.3pca.perf$recall.Avg, qda.3pca.perf$recall.Avg,
                                       forest.3pca.perf$recall.Avg,
                                       knn.3pca.perf$recall.Avg,svm.3pca.perf$recall.Avg)))
```

```
colnames(recall.Avg) <- COL.NAME
rownames(recall.Avg) <- Row.NAME


#specificity (percent of all negatives are corrected predicted)
specificity.Avg <- as.data.frame(rbind(c(lda.all.perf$specificity.Avg, qda.all.perf$specificity.Avg,
                                       forest.all.perf$specificity.Avg,
                                       knn.all.sc.perf$specificity.Avg,svm.all.sc.perf$specificity.Avg)
                                     c(lda.3var.perf$specificity.Avg, qda.3var.perf$specificity.Avg,
                                       forest.3var.perf$specificity.Avg,
                                       knn.3var.sc.perf$specificity.Avg,svm.3var.sc.perf$specificity.Avg
                                     c(lda.4var.perf$specificity.Avg, qda.4var.perf$specificity.Avg,
                                       forest.4var.perf$specificity.Avg,
                                       knn.4var.sc.perf$specificity.Avg,svm.4var.sc.perf$specificity.Avg
                                     c(lda.3pca.perf$specificity.Avg, qda.3pca.perf$specificity.Avg,
                                       forest.3pca.perf$specificity.Avg,
                                       knn.3pca.perf$specificity.Avg,svm.3pca.perf$specificity.Avg)))
colnames(specificity.Avg) <- COL.NAME
rownames(specificity.Avg) <- Row.NAME


#F1.score = 2*precision*recall / (precision+recall)
F1.score.Avg <- as.data.frame(rbind(c(lda.all.perf$F1.score.Avg, qda.all.perf$F1.score.Avg,
                                       forest.all.perf$F1.score.Avg,
                                       knn.all.sc.perf$F1.score.Avg,svm.all.sc.perf$F1.score.Avg),
                                     c(lda.3var.perf$F1.score.Avg, qda.3var.perf$F1.score.Avg,
                                       forest.3var.perf$F1.score.Avg,
                                       knn.3var.sc.perf$F1.score.Avg,svm.3var.sc.perf$F1.score.Avg),
                                     c(lda.4var.perf$F1.score.Avg, qda.4var.perf$F1.score.Avg,
                                       forest.4var.perf$F1.score.Avg,
                                       knn.4var.sc.perf$F1.score.Avg,svm.4var.sc.perf$F1.score.Avg),
                                     c(lda.3pca.perf$F1.score.Avg, qda.3pca.perf$F1.score.Avg,
                                       forest.3pca.perf$F1.score.Avg,
                                       knn.3pca.perf$F1.score.Avg,svm.3pca.perf$F1.score.Avg)))
colnames(F1.score.Avg) <- COL.NAME
rownames(F1.score.Avg) <- Row.NAME
```

Table of Performance Measures

```r
#performance summary table
kable(accuracy.rate, caption = 'Average LOOCV Accuracy Rate across Classes ', format = "pandoc")%>%kabl
```

Table 5: Average LOOCV Accuracy Rate across Classes

	lda	qda	RandomForest	knn.sc	svm.sc
all.var	0.86	0.90	0.90	0.88	0.90
3var	0.87	0.90	0.90	0.88	0.90
4var	0.87	0.90	0.90	0.85	0.89
3pca	0.81	0.83	0.82	0.83	0.83

```r
kable(precision.Avg, caption = 'Average LOOCV Precision across Classes ', format = "pandoc")%>%kable_sty
```

Table 6: Average LOOCV Precision across Classes

	lda	qda	RandomForest	knn.sc	svm.sc
all.var	0.87	0.90	0.90	0.89	0.90
3var	0.88	0.90	0.90	0.89	0.90
4var	0.87	0.90	0.90	0.86	0.89
3pca	0.82	0.83	0.82	0.84	0.90

```r
kable(recall.Avg, caption = 'Average LOOCV Recall across Classes ', format = "pandoc")%>%kable_styling(
```

Table 7: Average LOOCV Recall across Classes

	lda	qda	RandomForest	knn.sc	svm.sc
all.var	0.86	0.90	0.90	0.88	0.90
3var	0.87	0.90	0.90	0.88	0.90
4var	0.87	0.90	0.90	0.85	0.89
3pca	0.81	0.83	0.82	0.83	0.83

```r
kable(specificity.Avg, caption = 'Average LOOCV Specificity across Classes ', format = "pandoc")%>%kabl
```

Table 8: Average LOOCV Specificity across Classes

	lda	qda	RandomForest	knn.sc	svm.sc
all.var	0.97	0.98	0.98	0.98	0.98
3var	0.97	0.98	0.98	0.98	0.98
4var	0.97	0.98	0.98	0.97	0.98
3pca	0.96	0.97	0.96	0.97	0.97

```
kable(F1.score.Avg, caption = 'Average LOOCV F1.score across Classes ', format = "pandoc")%>%kable_styl
```

Table 9: Average LOOCV F1.score across Classes

	lda	qda	RandomForest	knn.sc	svm.sc
all.var	0.86	0.90	0.90	0.89	0.90
3var	0.88	0.90	0.90	0.88	0.90
4var	0.87	0.90	0.90	0.85	0.89
3pca	0.81	0.83	0.82	0.84	0.84

Graph of Performance measures

```r
accuracy.rate$numb.var <- as.factor(c("all.var", "3var", "4var", "3pca"))
mdata <- melt(accuracy.rate, id="numb.var")%>%dplyr::rename(Model="variable")


a <- ggplot(mdata, aes(x=numb.var, y=value, group=Model)) +
  geom_line(aes(color=Model)) +
  geom_point(aes(color=Model)) +
  coord_cartesian(xlim = NULL, ylim = c(0.8,0.95),
                  expand = TRUE, default = FALSE,clip = "on") +
  theme(legend.position="top") + labs(title = "Accuracy rate") +
  xlab("Variable Selection") + ylab("LOOCV Accuracy rate") +
  guides(fill=guide_legend(title="Model"))

#precision.Avg

precision.Avg$number.var <- as.factor(c("all.var", "3var", "4var", "3pca"))

prec.data <- melt(precision.Avg, id="number.var")%>%dplyr::rename(Model="variable")


b <- ggplot(prec.data, aes(x=number.var, y=value, group=Model)) +
  geom_line(aes(color=Model))+
  geom_point(aes(color=Model)) + coord_cartesian(xlim = NULL, ylim = c(0.8,0.95),
                  expand = TRUE, default = FALSE,clip = "on") +
  theme(legend.position="top") + labs(title = "Precision rate") +
  xlab("Variable Selection") + ylab("LOOCV Precisioin rate") +
  guides(fill=guide_legend(title="Model"))


#Recall

recall.Avg$number.var <- as.factor(c("all.var", "3var", "4var", "3pca"))

rec.data <- melt(recall.Avg, id="number.var")%>%dplyr::rename(Model="variable")


c <- ggplot(rec.data, aes(x=number.var, y=value, group=Model)) +
  geom_line(aes(color=Model))+
  geom_point(aes(color=Model)) + coord_cartesian(xlim = NULL, ylim = c(0.8,0.95),
                  expand = TRUE, default = FALSE,clip = "on") +
  theme(legend.position="top") + labs(title = "Recall rate") +
  xlab("Variable Selection") + ylab("LOOCV Recall rate") +
  guides(fill=guide_legend(title="Model"))

#Specificity

specificity.Avg$number.var <- as.factor(c("all.var", "3var", "4var", "3pca"))

spec.data <- melt(specificity.Avg, id="number.var")%>%dplyr::rename(Model="variable")
```

```r
d <- ggplot(spec.data, aes(x=number.var, y=value, group=Model)) +
  geom_line(aes(color=Model))+
  geom_point(aes(color=Model)) + coord_cartesian(xlim = NULL, ylim = c(0.95, 1),
                  expand = TRUE, default = FALSE,clip = "on") +
  theme(legend.position="top") + labs(title = "Specificity rate") +
  xlab("Variable Selection") + ylab("LOOCV Specificity rate") +
  guides(fill=guide_legend(title="Model"))

#F1 scoore

F1.score.Avg$number.var <- as.factor(c("all.var", "3var", "4var", "3pca"))

f.data <- melt(F1.score.Avg, id="number.var")%>%dplyr::rename(Model="variable")


e <- ggplot(f.data, aes(x=number.var, y=value, group=Model)) +
  geom_line(aes(color=Model))+
  geom_point(aes(color=Model)) + coord_cartesian(xlim = NULL, ylim = c(0.8, 0.95),
                  expand = TRUE, default = FALSE,clip = "on") +
  theme(legend.position="top") + labs(title = "F1 Score") +
  xlab("Variable Selection") + ylab("LOOCV F1 Score rate") +
  guides(fill=guide_legend(title="Model"))

grid.arrange(a,b,c,d,e,ncol=2)
```

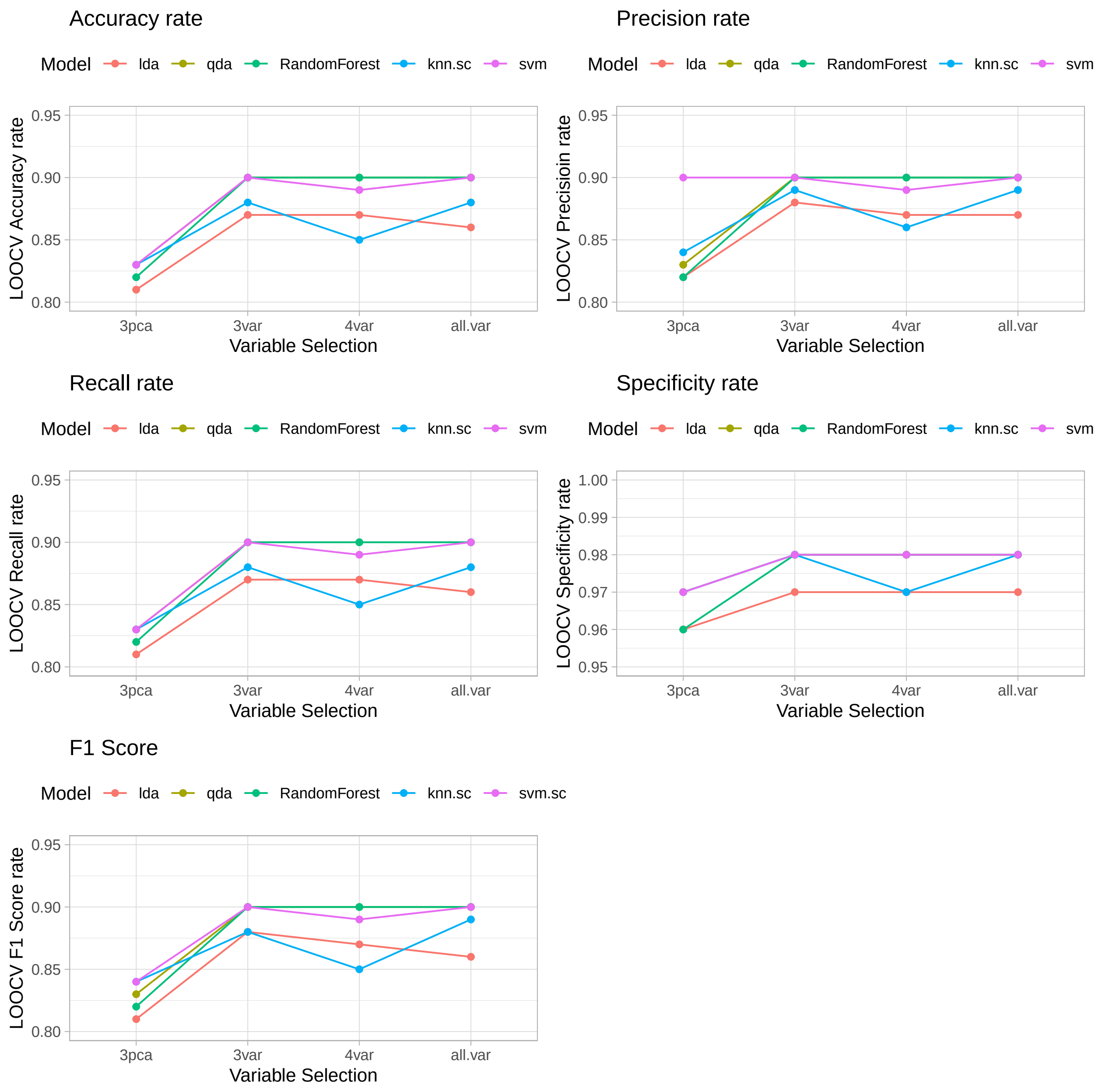


Figure 8: Model Performance

Visualize best-selected model:qda with all variables

```
pred.label.dat <- as.data.frame(cbind("Eccentricity"=labeled$Eccentricity, "Extent"=labeled$Extent, "cla
```

```
grid.arrange(
ggplot(labeled)+geom_point(aes(x=Extent, y=Eccentricity,col=Class))+ labs(title = "True Labeled")+
  theme(legend.position="bottom"),
ggplot(pred.label.dat)+geom_point(aes(x=Extent, y=Eccentricity,col=class))+ labs(title = "LOOCV qda Labe
```

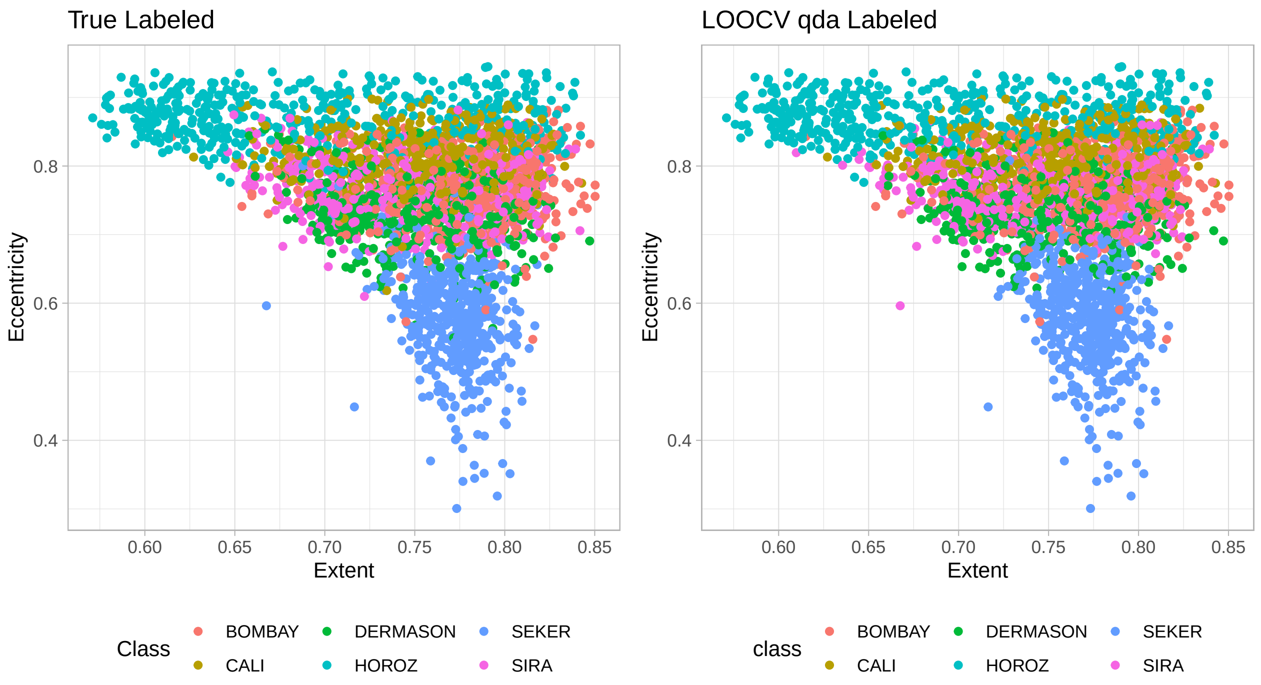


Figure 9: Final selected model (QDA)

Classes prediction result

```
#refit best-selected model
qda.mol <- qda(Class~., data = labeled, CV = FALSE)
#prediction
pred.A <- predict(qda.mol, newdata=sampA)
pred.B <- predict(qda.mol, newdata=sampB)
pred.C <- predict(qda.mol, newdata=sampC)

pred.dat <- as.data.frame(rbind(table(pred.A$class), table(pred.B$class), table(pred.C$class)),
                          row.names = c('sampleA', 'sampleB', 'sampleC'))
pred.dat$Num.obs. <- c(sum(table(pred.A$class)), sum(table(pred.B$class)), sum(table(pred.C$class)))
kable(pred.dat, caption = 'Prediction result for each sample', format = "pandoc")%>%kable_styling(latex_
```

Table 10: Prediction result for each sample

	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Num.obs.
sampleA	22	359	12	12	345	26	776
sampleB	0	1	779	15	238	340	1373
sampleC	1	102	161	540	8	170	982

Visualize classes prediction

```
pred.sampA.dat <- as.data.frame(cbind("Eccentricity"=sampA$Eccentricity, "Extent"=sampA$Extent, "class"=

pred.sampB.dat <- as.data.frame(cbind("Eccentricity"=sampB$Eccentricity, "Extent"=sampB$Extent, "class"=

pred.sampC.dat <- as.data.frame(cbind("Eccentricity"=sampC$Eccentricity, "Extent"=sampC$Extent, "class"=
```

```
grid.arrange(
ggplot(labeled)+geom_point(aes(x=Extent, y=Eccentricity,col=Class))+ labs(title = "True Labeled")+ theme

ggplot(pred.sampA.dat)+geom_point(aes(x=Extent, y=Eccentricity,col=class))+ labs(title = "Labeled.True")

ggplot(pred.sampB.dat)+geom_point(aes(x=Extent, y=Eccentricity,col=class))+ labs(title = "Sample B.Preds

ggplot(pred.sampC.dat)+geom_point(aes(x=Extent, y=Eccentricity,col=class))+ labs(title = "Sample C.Preds
```

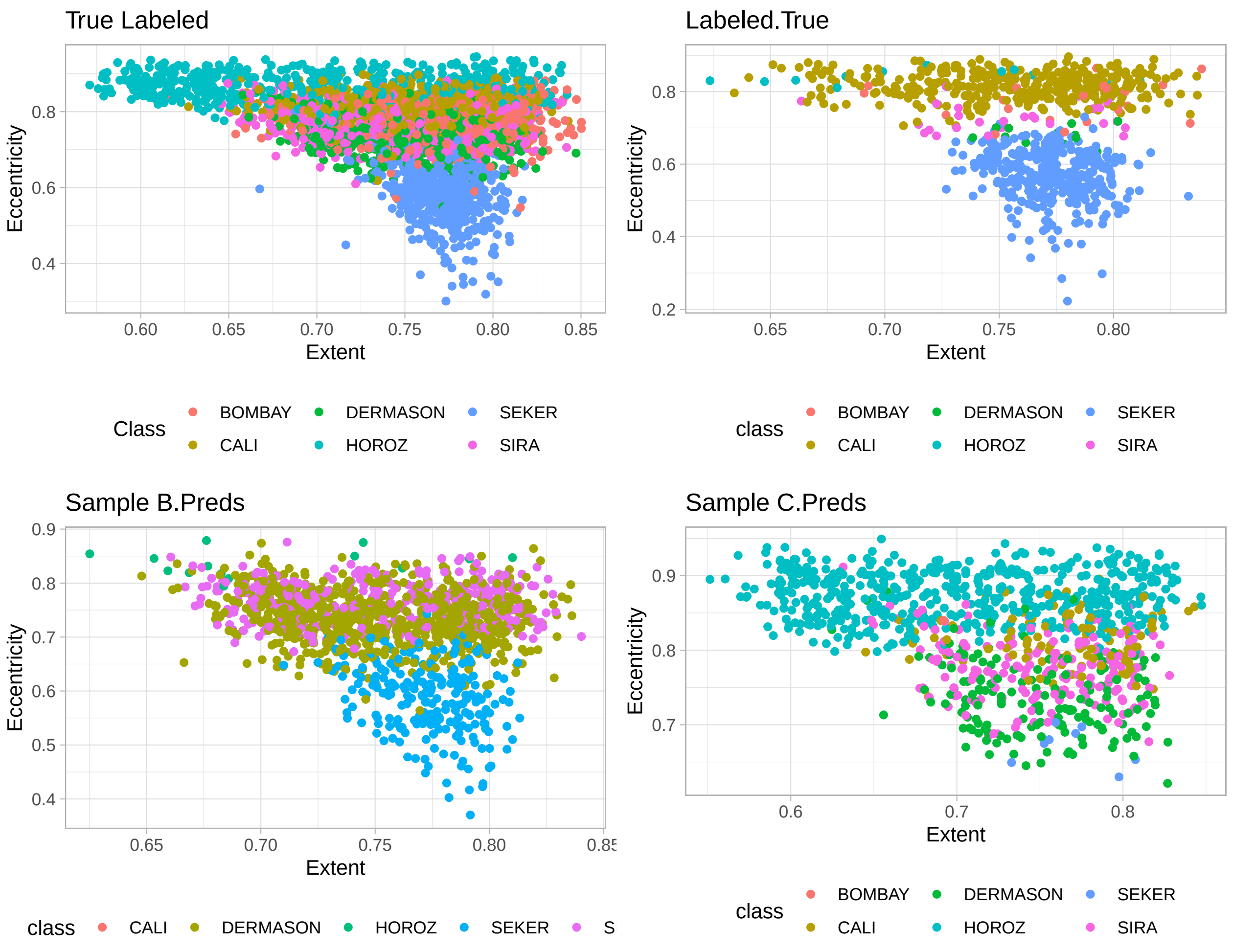


Figure 10: Prediction Visualization

```
## Price Prediction Result and Accuracy

p.lbs.sampA <- as.numeric(t(table(pred.A$class))%*%price.per.seed)
p.lbs.sampB <- as.numeric(t(table(pred.B$class))%*%price.per.seed)
p.lbs.sampC <- as.numeric(t(table(pred.C$class))%*%price.per.seed)
```

Confusion matrix of label data with LOOCV QDA

```
#second method
CV.tab.dat=cbind(Preds=qda.all$class,
                 Truth=labeled$Class)

conf.tab=xtabs(~Preds+Truth, CV.tab.dat)

#matrix(rowSums(conf.tab), nrow = 6, ncol = 6)

pred.tab.A=conf.probs=conf.tab/
             matrix(rowSums(conf.tab), nrow = 6, ncol = 6)

rownames(conf.tab) <- paste('Pred', classes, sep = '.')
colnames(conf.tab) <- paste('True', classes, sep = '.')
kable(conf.tab, caption = 'Confusion matrix of label data with LOOCV QDA (all variables)', format = "pa
```

Table 11: Confusion matrix of label data with LOOCV QDA (all variables)

	True.BOMBAY	True.CALI	True.DERMASON	True.HOROZ	True.SEKER	True.SIRA
Pred.BOMBAY	500	0	0	0	0	0
Pred.CALI	0	479	0	19	1	2
Pred.DERMASON	0	0	416	6	13	38
Pred.HOROZ	0	16	3	449	0	36
Pred.SEKER	0	2	16	0	454	24
Pred.SIRA	0	3	65	26	32	400

```
kable(rowSums(conf.tab), col.names = 'Num.Preds', caption = 'rowsums of confusion matrix', format = "pa
```

Table 12: rowsums of confusion matrix

	Num.Preds
Pred.BOMBAY	500
Pred.CALI	501
Pred.DERMASON	473
Pred.HOROZ	504
Pred.SEKER	496
Pred.SIRA	526

```r
rownames(pred.tab.A) <- paste('Pred', classes, sep = '.')
colnames(pred.tab.A) <- paste('True', classes, sep = '.')
kable(pred.tab.A, caption = 'multinominal distribution estimation', format = "pandoc")%>%kable_styling(
```

Table 13: multinominal distribution estimation

	True.BOMBAY	True.CALI	True.DERMASON	True.HOROZ	True.SEKER	True.SIRA
Pred.BOMBAY	1	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
Pred.CALI	0	0.9560878	0.0000000	0.0379242	0.0019960	0.0039920
Pred.DERMASON	0	0.0000000	0.8794926	0.0126850	0.0274841	0.0803383
Pred.HOROZ	0	0.0317460	0.0059524	0.8908730	0.0000000	0.0714286
Pred.SEKER	0	0.0040323	0.0322581	0.0000000	0.9153226	0.0483871
Pred.SIRA	0	0.0057034	0.1235741	0.0494297	0.0608365	0.7604563

Prediction result and accuracy

```r
set.seed(12345)
pred.accuracy <- function(pred.tab.A, size.tab){
condit.Par.BS=NULL
for (i in 1:1000){
  p.lbs <- NULL
  for (j in 1:6) {
    seed <- t(rmultinom(1, size = size.tab[j], prob = pred.tab.A[j,]))
    p.lbs <- c(p.lbs, seed %*% price.per.seed)
  }
condit.Par.BS=c(condit.Par.BS, sum(p.lbs))
}
return(condit.Par.BS)
}

pred.ar.A <- pred.accuracy(pred.tab.A = pred.tab.A, size.tab = table(pred.A$class))
pred.ar.B <- pred.accuracy(pred.tab.A = pred.tab.A, size.tab = table(pred.B$class))
pred.ar.C <- pred.accuracy(pred.tab.A = pred.tab.A, size.tab = table(pred.C$class))


pred.ar.dat <- rbind(quantile(pred.ar.A, c(0, 0.025, 0.975, 1)),
                     quantile(pred.ar.B, c(0, 0.025, 0.975, 1)),
                     quantile(pred.ar.C, c(0, 0.025, 0.975, 1)))
pred.ar.dat <- as.data.frame(pred.ar.dat)
rownames(pred.ar.dat) <- c('samp.A', 'samp.B', 'samp.C')

pred.ar.dat$Predicted.Net.Worth <- c(p.lbs.sampA, p.lbs.sampB, p.lbs.sampC)
pred.ar.dat$Range <- pred.ar.dat$`97.5%` - pred.ar.dat$`2.5%`


kable(round(pred.ar.dat,2), caption = 'prediction result and accuracy (in dollars)', format = "pandoc")
```

Table 14: prediction result and accuracy (in dollars)

	0%	2.5%	97.5%	100%	Predicted.Net.Worth	Range
samp.A	4.45	4.48	4.57	4.59	4.60	0.09
samp.B	3.22	3.25	3.39	3.44	3.24	0.14
samp.C	3.35	3.37	3.49	3.55	3.34	0.13

```r
library(MASS)
qda.mod=qda(Class~., labeled)
preds.C= predict(qda.mod, newdata = sampC)
pred.tabs=table(pred.C$class)
for(i in 1:10){
  qda.mod.iter=qda(Class~., prior = as.vector(pred.tabs/sum(pred.tabs)), labeled)
  preds.C= predict(qda.mod.iter, newdata = sampC)
  pred.tabs=table(pred.C$class)
  print(pred.tabs)
  flush.console()
}
```

```
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
##
##   BOMBAY     CALI DERMASON    HOROZ    SEKER     SIRA
##        1      102      161      540        8      170
```

References

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

2. Heuzé V., Tran G., Nozière P., & Lebas F. (2015). Common Bean (Phaseolus vulgaris), Feedipedia.org – Animal Feed Resources Information System – A programme by INRA, CIRAD, AFZ and FAO, http://www.feedipedia.org/node/266 (accessed on 29 April 2021).

3. Koklu, M., & Ozkan, I. A. (2020). Multiclass classification of dry beans using computer vision and ma-chine learning techniques. Computers and Electronics in Agriculture, 174, 105507. doi:10.1016/j.compag.2020.105507

4. Varankaya, S., & Ceyhan, E. (2012). Problems Encountered in Bean Farming in the Central Anatolia Region and Solution Suggestions. Selçuk Tarım Bilim. Journal. 26, 15–26.

5. https://en.m.wikipedia.org/wiki/Sensitivity_and_specificity

6. https://www.geeksforgeeks.org/loocvleave-one-out-cross-validation-in-r-programming/

7. https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710

8. https://alekhyo.medium.com/interview-questions-on-svm-bf13e5fbcca8://alekhyo.medium.com/interview-questions-on-svm-bf13e5fbcca8