UNIVERSITY OF YORK

STATISTICAL MODELLING AND PRACTICAL DATA ANALYSIS WITH R

The Classification of Ancient Beads

Authors
Kate Clasper
Georgina Youlden
Sedinam Azanu
Barney Franklin

August 19, 2022

1 Introduction

In this report, we will look into four different methods of classifying information: principle component analysis, linear discriminant analysis, naive Bayes, and decision trees with a random forest. We will apply these methods to a data set consisting of information pertaining to beads found in archaeological dig sites. This dataset contains proportions of different amino acid levels as well as taxonomic information (Class, Order, Genus and Species) and a note on where the bead was found (henceforth referred to as Location). We aim to find appropriate ways to classify these beads according to Location, Class, Order, Genus and Species.

2 Principle Component Analysis

In this section we will be analysing the data set using principle component analysis (PCA). we will be performing PCA on the data using R then we am going to plot each of the PCA scores against each other and colour them by each of the beads class, genius, order, species and location. Using these we will be able to see for each bead variable which PCA scores give the best classification and separation of the different variable types. After plotting and comparing each PCA scores for each variable we have compared each of the graphs and determined which provides the best classification and which is the worst.

2.1 PCA by Class

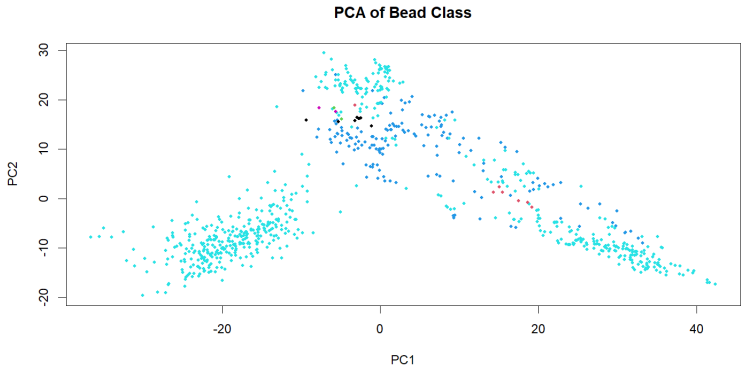

Figure 2.1

None of PCA scores plotted provided a clear separation of each bead type, the best can be seen in figure 2.1. In this figure there are three main groups of the light blue

(Gastropoda) with the left group being the most distinct group with very little overlap with any of the other colours. Despite the fact there are groups of light blue, they are not all in together meaning if we use this PCA data to class the data it wold be difficult as there is not a clear section of values of PC1 and PC2 that distinguish this bead. The dark blue (Bivalvia) have a group in the middle of the graph but there is cross over between this and the light blue. As well as this, there are small groups of black (BeadDen), pink (ScaphopodaA) and red (BeadGC) within the dark blue and (despite these all having their own group when classifying them based on PC1 and PC2) it will be difficult to distinguish between these bead types and the dark blue bead.

2.2 PCA by Location

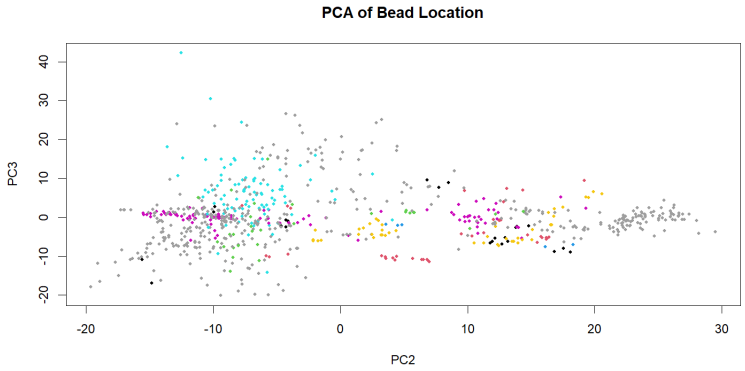

Figure 2.2

The most complicated bead variable to classify using PCA is location. In figure 2.2, PC2 and PC3 are plotted against each other; in this graph you can see small groups of each colour but the majority of the groups overlap and have beads distributed though out the graph. Using PCA to classify location becomes difficult becuase for the majority of the graph it is hard to distinguish between each colour, with the only exception being the grey bead on right of the graph. Therefore using PCA to classify Location is not a good choice.

3 LDA

This section will focus on using linear discriminant analysis (LDA) to classify the beads data. Linear discriminant analysis can be used to categorise variables into classification tables or confusion matrices which visualise the LDA model and explore the accuracy of said model. To begin with the data was split in to two sets:

the training set containing 70% of the data and the test set containing the remaining 30%. The LDA model was then trained using the training set and then the LDA model was used to classify the test set.

		Real					
	Class	BeadDen	BeadGC	BeadJak	Bivalia	Gastropoda	Scaphopoda
Predicted	BeadDen	1	0	0	5	1	0
	BeadGC	0	0	0	3	1	0
	BeadJak	0	0	0	2	0	0
	Bivalia	5	1	1	28	39	1
	Gastropoda	0	4	0	70	396	0
	Scaphopoda	0	0	0	0	0	0

Table 3.1

First the model was created to see how well we could classify the data using the response variable “Class". Table 3.1 (above) depicts the results of LDA model when classifying the training set for this response variable. This is a good classification with a classification rate of 76.16% but it could be improve on. This is to be expected as this is a model classifying the set that was used to train it.

Below is table 3.2, it shows the classification table for test data when classifying using the LDA model for the response variable Class. There is a classification rate of 75.31%. Despite being lower than the classification rate for training set, this is very good for the test set which indicates that the LDA model has been trained well and is classifying to a relevantly suitable level of accuracy.

		Real					
	Class	BeadDen	BeadGC	BeadJak	Bivalia	Gastropoda	Scaphopoda
Predicted	BeadDen	0	0	0	5	0	1
	BeadGC	0	0	0	1	0	0
	BeadJak	0	0	0	0	0	0
	Bivalia	2	0	1	18	24	0
	Gastropoda	0	2	0	23	162	0
	Scaphopoda	0	0	0	0	0	0

Table 3.2

The process was then repeated for all remaining response variables. Table 3.3 (below) shows both the training and test classification rates for every response variable. Interestingly, the variable Order has a higher test classification rate than train, which could be worrying as the training classification has been optimised for this model. We are argue however, that as it is not significantly higher, concern is not needed. In every other case, the model classifies the training data much better than the test data. Class and Order have encouraging training and test classification rates. For Genus, Species and location, the test classification rates are far lower than the training rates, with each of the classification rates being less than 5%. This is a clear sign

of over-fitting indicating that these models shouldn't be used to classify this data set for those response variables. This might be due to LDA requiring a normal distribution assumption on features/predictors but further exploration would confirm this. LDA could definitely be a viable option for classifying response variables Class and Order.

Response variable	Training classification rate (%)	Test classification rate (%)
Class	76.16	75.31
Order	76.70	78.66
Genus	69.35	2.09
Species	56.81	1.26
Location	55.91	4.18

Table 3.3: Classification rates for every response variables

4 Naive Bayes' Classifier

This section will revolve around using Naive Bayes (NB) to try to classify the bead data. The NB classifiers were run on a training set first, consisting of a randomly selected 70% of the total data set. Then, the NB model generated by the training data was used to classify the remaining 30% test set.

Response Variable	Training Class. Rate (%)	Test Class. Rate(%)	Gaussian Class. Rate (%)
CLASS	82.44	82.43	86.19
ORDER	81.72	73.22	78.24
GENUS	79.24	66.53	-
SPECIES	-	-	-
LOCATION	53.05	49.79	57.32

Table 4.1: Results of Naive Bayes Classification.

Table 4.1 summarises the results of a NB classifier without kernel density estimation (KDE). It is clear that of the five potential response variables, NB best separates the data according to the Class of the beads, and is worst at classifying beads according to their Location. To an extent, this is to be expected as historically beads were often used as currency, and examining the origins of beads in different locations is a popular way of determining which tribes and countries were trading with each other [1].

The NB classifier in RStudio was unable to classify any information with Species as the response variable - although the NB model would run without issue, all examples of the classification matrix consisted entirely of zero-valued entries[1]. All

[1]We did try to rectify this by testing using KDE estimation and the Poisson distribution, but all outputs remained zero-valued.

research into this issue seems to suggest that this is caused by a strange or difficult distribution in the sample data. Because Species was able to be classified using LDA but gave significantly lower results than all other tests, it is sensible to assume that Species is distributed in such a way that both LDA and NB models struggle to classify.

We were curious about some of the results of table 4.1, particularly for Order and Genus, as the classification rates for training and test data were significantly different for both. This could be a sign that the NB model over-fit the training set in both cases, but just to be sure we also ran a Gaussian NB classifier including KDE for all response variables. Gaussian NB is a particularly helpful model because it works from a core assumption that the data contains no covariance between dimensions, and follows a normal distribution. The third column of table 4.1 shows that this assumption may actually be helpful for classifying Class and Location (as the Gaussian model has a much higher classification rate than the others). The contrast in classification rate for order however, implies that when model order using standard NB, the model did indeed over-fit the training data. Interestingly. the Gaussian NB model for species had the same result as all NB models for Genus - every attempt to run this model resulted in entirely zero-valued confusion matrices. Given that the standard NB classifier does not do this for genus, we believe that this may imply Genus to have strong covariance, but it could also be confirmation that a normality assumption is inappropriate in this instance.

5 Decision Tree and Random Forest

This section looks to analyse the data set using decision tree and random forest methods. The data is split into a 70% training data set and 30% test data set before being analysed to see how well the data classifies within the different models.

When using the decision tree method, the "rpart" function allows us to build the classification tree for each of the training and test data sets. Figure 5.1 implies that the majority of the origins are from the UK, however due to the use of beads as currency many years ago or due to some of the species travelling, it is rather difficult to confidently classify information based on Location. The "unknown" locations does not seem to appear in the classification of either the training set or the test set (see appendix 1), so we can conclude that the beads from unknown locations do not affect the classification of the data significantly. The majority of the classification appears to be dominated by the glutamine/glutamic acid and asparagine/aspartic acid amino acids - many branches of the tree seem to depend on the values of these acids, suggesting that they contribute the most to this classification. From the caret package, we use the "confusionMatrix" function to obtain results from the training

and test data sets in an easy to read format and to calculate the accuracy for each variable.

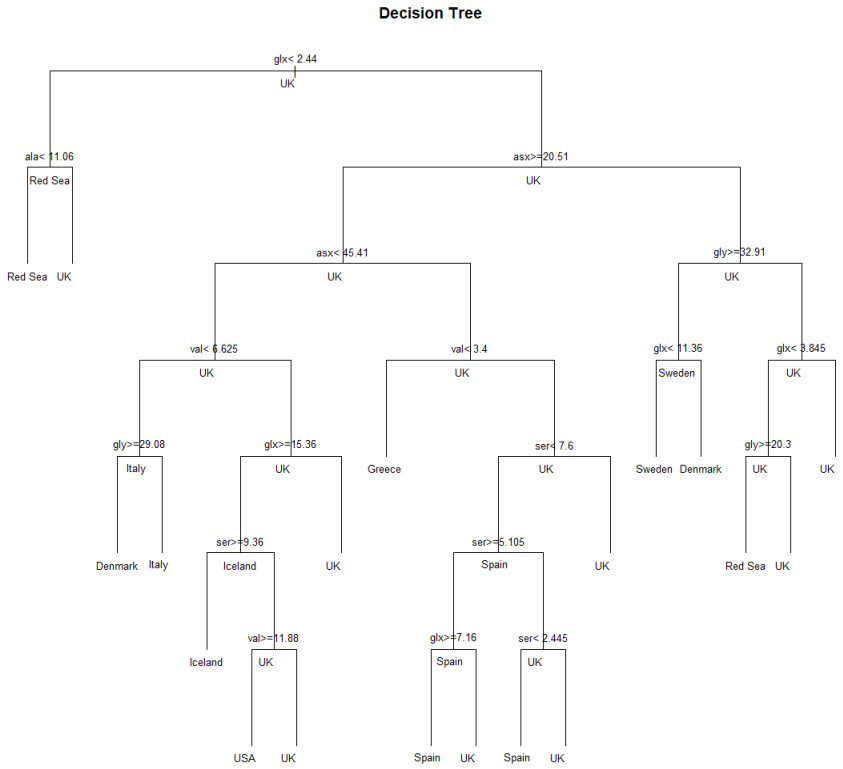

Figure 5.1: Decision tree of Location using a 70% training data set

Table 5.1 (below) shows that the decision trees were best at classifying the data according to Class, and the worst at separating according to Species. Random Forests are a much more reliable method of classifying a data set because they create simulate the classification of variables repeatedly, using different decision trees rather than just one.

Response variable	Training accuracy (%)	Test accuracy (%)
Class	88.38	81.61
Order	82.8	76.99
Genus	71.15	70.71
Species	69.71	66.53
Location	77.42	74.48

Table 5.1: Classification Results from Decision Trees

Table 5.2 (below) gives the classification rate results according to random forest sim-

ulation. If we compare to table 5.1, Class and Order have much more reliable classification rates between the training and test data sets. The Species variable however, has a higher training accuracy but the test accuracy has a significant decrease in terms of accuracy, which tells us that the model may be over fitting the training data, therefore making this particular random forest unreliable. Regardless, the random forests have the best rates of classification across all variables, making them the best option for classifying the beads data.

Response variable	Training accuracy (%)	Test accuracy (%)
Class	95.52	91.21
Order	86.56	86.19
Genus	79.03	75.73
Species	75.09	63.18
Location	78.14	74.48

Table 5.2: The results of classification according to Random Forest

6 Conclusion

Overall, we were quite surprised by the outcomes of our exploratory analysis. While we anticipated the random forest being the most appropriate classifier, we did not expect the principle component analysis (PCA) to be so inconclusive nor did we expect the naive Bayes' (NB) classifier to have difficulty classifying any of the response variables.

We have seen that PCA in particular is not appropriate for this data set, due to the large variance in our data. For every data variable considered, there is never any prominent distinction between the different types of bead, so no conclusions can be made from this information. Especially when analysing Location, the resulting PCA graph is almost indistinguishable.

Although linear discriminant analysis (LDA) is often used in conjunction with PCA, we can argue that LDA may be appropriate for this data, despite PCA not being so. It is clear that LDA cannot be used to classify Genus, Location or Species, but the classification rates for Class and Order are quite similar across training and test sets. It is possible that the LDA model for Order over fits the training data slightly, but the difference between the training and test sets is not enough to warrant concern.

Unsurprisingly, the naive Bayes (NB) classifiers, where they are appropriate, are much more accurate than the LDA models. Even without the Gaussian variant, the NB models for Class and Order consistently classify the data 5-6% more accurately. As with the LDA exploration, the NB classifiers for Genus, Species and Location are not appropriate at all. The standard NB model could be used to classify Genus, however the 66.53% accuracy rate within the test set is not optimal. We do not advise

using the NB classifier to determine Location, as the classification rates are too low. Comparing the results of the Gaussian NB and the LDA analysis, it appears that part of the reason we cannot classify Species and Location is due to the normality assumptions of both models.

The decision tree and random forest classifiers were definitely the most accurate for every potential response variable. As with LDA and NB, Class and Order were modelled most accurately, however unlike the other classifiers the random forest did create classifications for Genus, Species and Location that can be used with confidence. Even the least accurate of the random forests (Species) still has classification rates of 75.09% and 63.18% for the training and test sets respectively, which is significantly better than any of the other results we found, despite the likelihood of this particular forest over fitting the training data.

We have seen that out of the five potential response variables for this data, there are only two that can be accurately classified: Class and Order. We believe that Genus and Species cannot be accurately predicted by most models because neither can be normally distributed, which is a requirement of LDA and naive Bayes. Overall, we recommend using a random forest to classify this data, as the classification rates are higher for each response variable with less difference between the results for the training sets and test sets.

Appendix

Appendix 1 - Decision Tree for Location, using a 30% test set

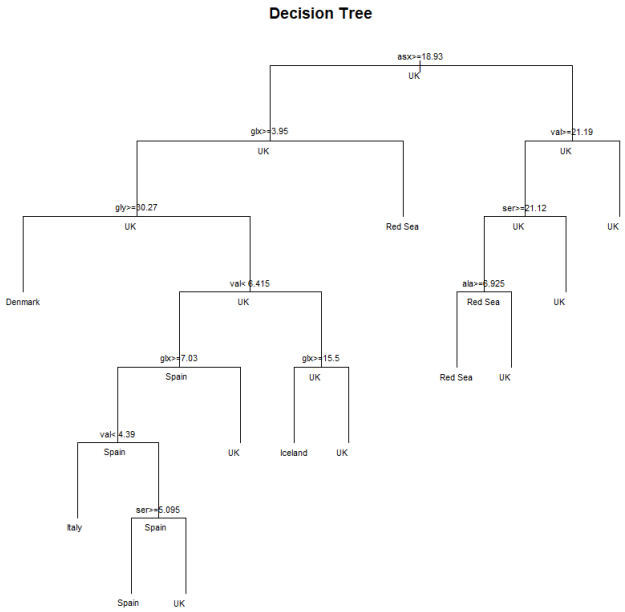

Figure 6.1: Decision tree of Location test data

Appendix 2 - PCA R Code

data = read.csv(file = "bead$_data_1.csv$")

data=data[,-1]

Separating different leveled tests into different datasets:
class=data[,-(2:5)]
order=data[,-c(1,3:5)]
genus=data[,-c(1:2,4:5)]
species=data[,-c(1:3,5)]
location=data[,-(1:4)]

classpca = prcomp(class[,2:7])

plot(classpca$x$[, 1], $classpca$x[,2],type = "p", xlab = "PC1", ylab = "PC2", main = "PCA of Bead Class", cex = 0.6, pch = 16 , col = as.factor(class$class$))

$plot(classpca$x[,2], classpca$x[,3], type = "p", xlab = "PC2", ylab = "PC3", cex = 3, pch = ".", col = as.factor(class$class))
plot(classpca$x[,1], classpca$x[,3],type = "p", xlab = "PC1", ylab = "PC3", cex = 3, pch = "." , col = as.factor(class$class))
$plot(classpca$x[,3], classpca$x[,4], type = "p", xlab = "PC3", ylab = "PC4", cex = 3, pch = ".", col = as.factor(class$class))
plot(classpca$x[,2], classpca$x[,4],type = "p", xlab = "PC2", ylab = "PC4", cex = 3, pch = "." , col = as.factor(class$class))
$plot(classpca$x[,1], classpca$x[,4], type = "p", xlab = "PC1", ylab = "PC4", cex = 3, pch = ".", col = as.factor(class$class))

orderpca = prcomp(order[,2:7])

plot(orderpca$x[,1], $orderpca$x[,2],type = "p", xlab = "PC1", ylab = "PC2", cex = 3, pch = "." , col = as.factor(order$order))
$plot(orderpca$x[,2], orderpca$x[,3], type = "p", xlab = "PC2", ylab = "PC3", cex = 3, pch = ".", col = as.factor(order$order))
plot(orderpca$x[,1], $orderpca$x[,3],type = "p", xlab = "PC1", ylab = "PC3", cex = 3, pch = "." , col = as.factor(order$order))
$plot(orderpca$x[,3], orderpca$x[,4], type = "p", xlab = "PC3", ylab = "PC4", cex = 3, pch = ".", col = as.factor(order$order))
plot(orderpca$x[,2], $orderpca$x[,4],type = "p", xlab = "PC2", ylab = "PC4", cex = 3, pch = "." , col = as.factor(order$order))
$plot(orderpca$x[,1], orderpca$x[,4], type = "p", xlab = "PC1", ylab = "PC4", main = "PCAofBeadOrder", cex = 0.6, pch = 16, col = as.factor(order$order))
genuspca = prcomp(genus[,2:7])

plot(genuspca$x[,1], $genuspca$x[,2],type = "p", xlab = "PC1", ylab = "PC2", cex = 3, pch = "." , col = as.factor(genus$Genus))
$plot(genuspca$x[,2], genuspca$x[,3], type = "p", xlab = "PC2", ylab = "PC3", cex = 3, pch = ".", col = as.factor(genus$Genus))
plot(genuspca$x[,1], $genuspca$x[,3],type = "p", xlab = "PC1", ylab = "PC3", cex = 3, pch = "." , col = as.factor(genus$Genus))
$plot(genuspca$x[,3],genuspca$x[,4], type = "p", xlab = "PC3", ylab = "PC4", cex = 3, pch = ".", col = as.factor(genus$Genus))
plot(genuspca$x[,2], $genuspca$x[,4],type = "p", xlab = "PC2", ylab = "PC4", cex = 3, pch = "." , col = as.factor(genus$Genus))
$plot(genuspca$x[,1], genuspca$x[,4], type = "p", xlab = "PC1", ylab = "PC4", cex = 3, pch = ".", col = as.factor(genus$Genus))

speciespca = prcomp(species[,2:7])

plot(speciespca$x[,1], $speciespca$x[,2],type = "p", xlab = "PC1", ylab = "PC2",

cex = 3, pch = "." , col = as.factor(species$Species$))
$plot(speciespca$x[,2], speciespca$x[,3], type = "p", xlab = "PC2", ylab = "PC3", cex =$
$3, pch = ".", col = as.factor(species$Species))$
plot(speciespca$x[,1], speciespca$x[,3],$type = "p", xlab = "PC1", ylab = "PC3", cex =
3, pch = "." , col = as.factor(species$Species$))
$plot(speciespca$x[,3], speciespca$x[,4], type = "p", xlab = "PC3", ylab = "PC4", cex =$
$3, pch = ".", col = as.factor(species$Species))$
plot(speciespca$x[,2], speciespca$x[,4],$type = "p", xlab = "PC2", ylab = "PC4", cex =
3, pch = "." , col = as.factor(species$Species$))
$plot(speciespca$x[,1], speciespca$x[,4], type = "p", xlab = "PC1", ylab = "PC4", cex =$
$3, pch = ".", col = as.factor(species$Species))$

locationpca = prcomp(location[,2:7])

plot(locationpca$x[,1], locationpca$x[,2],$type = "p", xlab = "PC1", ylab = "PC2",
cex = 3, pch = "." , col = as.factor(location$Location$))
$plot(locationpca$x[,2],$
locationpca$x[,3], type = "p", xlab = "PC2", ylab = "PC3", cex = 0.6, pch = 16, main =$
"PCA of Bead Location", col = as.factor(location$Location))$
plot(locationpca$x[,1],$
locationpca$x[,3],$type = "p", xlab = "PC1", ylab = "PC3", cex = 3, pch = "." , col =
as.factor(location$Location$))
$plot(locationpca$x[,3], locationpca$x[,4], type = "p", xlab = "PC3", ylab = "PC4", cex =$
$3, pch = ".", col = as.factor(location$Location))$
plot(locationpca$x[,2], locationpca$x[,4],$type = "p", xlab = "PC2", ylab = "PC4", cex
= 3, pch = "." , col = as.factor(location$Location$))
$plot(locationpca$x[,1], locationpca$x[,4], type = "p", xlab = "PC1", ylab = "PC4", cex =$
$3, pch = ".", col = as.factor(location$Location))$

Appendix 3 - LDA R Code

n = nrow(Beads)
trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
train = Beads[trainIndex ,]
test = Beads[-trainIndex ,]

ClassLDA = lda(train[,6:11], train[,1])
ClassLDA
pred = predict(ClassLDA, train[,6:11])
table(predicted = pred$class, real = train$[,1])
$sum(diag(prop.table(table(predicted = pred$class, real = train[,1]))))
pred = predict(ClassLDA, test[,6:11])
table(predicted = pred$class, real = test$[,1])

$sum(diag(prop.table(table(predicted = pred$class, real = test[,1]))))$

OrderLDA = lda(train[,6:11], train[,2])
OrderLDA
pred = predict(OrderLDA, train[,6:11])
table(predicted = pred$class, real = train[,2])$
$sum(diag(prop.table(table(predicted = pred$class, real = train[,2]))))$
pred = predict(OrderLDA, test[,6:11])
table(predicted = pred$class, real = test[,2])$
$sum(diag(prop.table(table(predicted = pred$class, real = test[,2]))))$

GenusLDA = lda(train[,6:11], train[,3])
GenusLDA
pred = predict(GenusLDA, train[,6:11])
table(predicted = pred$class, real = train[,3])$
$sum(diag(prop.table(table(predicted = pred$class, real = train[,3]))))$
pred = predict(GenusLDA, test[,6:11])
table(predicted = pred$class, real = test[,3])$
$sum(diag(prop.table(table(predicted = pred$class, real = test[,3]))))$

SpeciesLDA = lda(train[,6:11], train[,4])
SpeciesLDA
pred = predict(SpeciesLDA, train[,6:11])
table(predicted = pred$class, real = train[,4])$
$sum(diag(prop.table(table(predicted = pred$class, real = train[,4]))))$
pred = predict(SpeciesLDA, test[,6:11])
table(predicted = pred$class, real = test[,4])$
$sum(diag(prop.table(table(predicted = pred$class, real = test[,4]))))$

LocationLDA = lda(train[,6:11], train[,5])
LocationLDA
pred = predict(LocationLDA, train[,6:11])
table(predicted = pred$class, real = train[,5])$
$sum(diag(prop.table(table(predicted = pred$class, real = train[,5]))))$
pred = predict(LocationLDA, test[,6:11])
table(predicted = pred$class, real = test[,5])$
$sum(diag(prop.table(table(predicted = pred$class, real = test[,5]))))$

Appendix 4 - Naive Bayes R Code

Read in data
data = read.csv("bead$_data_1.csv", header = T$)

install necessary packages

```
install.packages(c("naivebayes", "caret", "pROC", "ROCR"))
library(naivebayes)
library(pROC)
library(caret)
```

data=data[,-1]

Separating different levelled tests into different datasets:

```
class=data[,-(2:5)]
order=data[,-c(1,3:5)]
genus=data[,-c(1:2,4:5)]
species=data[,-c(1:3,5)]
location=data[,-(1:4)]
```

CLASS n1=nrow(class)

```
trainIndex1=sample(1:n1, size=round(0.7*n1), replace=FALSE)
```

$train_c = class[trainIndex1,]$

$test_c = class[-trainIndex1,]$

$dim(train)$

$dim(test)$

$model_nb_c = naive_bayes(train_c[,2:7], train_c[,1])$

$p_nb_c = predict(model_nb_c, train_c[,2:7])$

$pred_nb_c = predict(model_nb_c, test_c[,2:7])$

$trainclassrate$

$r_c = train_c[,1]$

$table(true = r_c, predicted = p_nb_c)$

$testclassrate$

$real_c = test_c[,1]$

$table(true = real_c, predicted = pred_nb_c)$

ORDER

```
n2=nrow(order)
trainIndex2=sample(1:n2, size=round(0.7*n2), replace=FALSE)
```

$train_o = order[trainIndex2,]$

$test_o = order[-trainIndex2,]$

$dim(train)$

$dim(test)$

$model_nb_o = naive_bayes(train_o[,2:7], train_o[,1])$

$p_nb_0 = predict(model_nb_o, train_o[,2:7])$

$pred_nb_o = predict(model_nb_o, test_o[,2:7])$

$r_o = train_o[,1]$

$table(true = r_o, predicted = p_nb_0)$

$real_o = test_o[,1]$
$table(true = real_o, predicted = pred_nb_o)$

GENUS

n3=nrow(genus)
trainIndex3=sample(1:n3, size=round(0.7*n3), replace=FALSE)
train$_g$ = $genus[trainIndex3,]$
$test_g = genus[-trainIndex3,]$
$dim(train)$
$dim(test)$
$model_nb_g = naive_bayes(train_g[,2:7], train_g[,1])$
$p_nb_g = predict(model_nb_g, train_g[,2:7])$
$pred_nb_g = predict(model_nb_g, test_[,2:7])$
$r_g = train_g[,1]$
$table(true = r_g, predicted = p_nb_g)$
$real_g = test_g[,1]$
$table(true = real_g, predicted = pred_nb_g)$

SPECIES

n4=nrow(species)
trainIndex4=sample(1:n4, size=round(0.7*n4), replace=FALSE)
train$_s$ = $species[trainIndex4,]$
$test_s = species[-trainIndex4,]$
$dim(train_s)$
$dim(test_s)$
$model_nb_s = naive_bayes(train_s[,2:7], train_s[,1])$
$pred_nb_s = predict(model_nb_s, test_s[,2:7])$
$real_s = test_s[,1]$
$table(true = real_s, predicted = pred_nb_s)$

LOCATION

n5=nrow(location)
trainIndex5=sample(1:n5, size=round(0.7*n5), replace=FALSE)
train$_l$ = $location[trainIndex5,]$
$test_l = location[-trainIndex5,]$
$dim(train)$
$dim(test)$
$model_nb_l = naive_bayes(train_l[,2:7], train_l[,1])$
$p_nb_l = predict(model_nb_l, train_l[,2:7])$
$pred_nb_l = predict(model_nb_l, test_l[,2:7])$
$r_l = train_l[,1]$
$table(true = r_l, predicted = p_nb_l)$

$real_l = test_l[,1]$
$table(true = real_l, predicted = pred_n b_l)$

Appendix 5 - Gaussian Naive Bayes R Code

data = read.csv("bead$_d ata_1.csv", header = T)$
$installallnecessarypackages$
$install.packages(c("naivebayes","caret","pROC","ROCR"))$
$library(naivebayes)$
$library(pROC)$

data=data[,-1]

separate dataset for relevant tests
class=data[,-(2:5)]
order=data[,-c(1,3:5)]
genus=data[,-c(1:2,4:5)]
species=data[,-c(1:3,5)]
location=data[,-(1:4)]

CLASS
n1=nrow(class)
trainIndex1=sample(1:n1, size=round(0.7*n1), replace=FALSE)
train$_c = class[trainIndex1,]$
$test_c = class[-trainIndex1,]$

$dim(train)$

$dim(test)$

$model_{gc} = naive_b ayes(train_c[,2:7], train_c[,1], usekernel = TRUE)$

$pred_{gc} = predict(model_{gc}, test_c[,2:7])$

$real_c = test_c[,1]$

$table(true = real_c, predicted = pred_{gc})$

ORDER
n2=nrow(order)
trainIndex2=sample(1:n2, size=round(0.7*n2), replace=FALSE)

$train_o = order[trainIndex2,]$
$test_o = order[-trainIndex2,]$
$dim(train)$
$dim(test)$
$model_{go} = naive_bayes(train_o[,2:7], train_o[,1], usekernel = TRUE)$
$pred_{go} = predict(model_{go}, test_o[,2:7])$
$real_o = test_o[,1]$
$table(true = real_o, predicted = pred_{go})$

GENUS

n3=nrow(genus)
trainIndex3=sample(1:n3, size=round(0.7*n3), replace=FALSE)
$train_g = genus[trainIndex3,]$
$test_g = genus[-trainIndex3,]$
$dim(train)$
$dim(test)$
$model_{gg} = naive_bayes(train_g[,2:7], train_g[,1], usekernel = TRUE)$
$pred_{gg} = predict(model_{gg}, test_g[,2:7])$
$real_g = test_g[,1]$
$table(true = real_g, predicted = pred_{gg})$

SPECIES

n4=nrow(species)
trainIndex4=sample(1:n4, size=round(0.7*n4), replace=FALSE, usekernel=TRUE)
$train_s = species[trainIndex4,]$
$test_s = species[-trainIndex4,]$
$dim(train_s)$
$dim(test_s)$
$model_{gs} = naive_bayes(train_s[,2:7], train_s[,1])$
$pred_{gs} = predict(model_{gs}, test_s[,2:7])$
$real_s = test_s[,1]$
$table(true = real_s, predicted = pred_nb_s)$

LOCATION

n5=nrow(location)
trainIndex5=sample(1:n5, size=round(0.7*n5), replace=FALSE)
$train_l = location[trainIndex5,]$
$test_l = location[-trainIndex5,]$
$dim(train)$
$dim(test)$
$model_{gl} = naive_bayes(train_l[,2:7], train_l[,1], usekernel = TRUE)$
$pred_nb_l = predict(model_nb_l, test_l[,2:7])$

$real_l = test_l[,1]$
$table(true = real_l, predicted = pred_n b_l)$

Appendix 6 - Decision Trees and Random Forest R Code

```
require(randomForest)
install.packages("randomForest")
install.packages("tree")
install.packages("caTools")
install.packages("rpart")
library(randomForest)
library(tree)
library(caTools)
library(rpart)
library(caret)
```

Loading the file into rstudio

beads1 = read.csv("bead$_data_1.csv$", $header = TRUE$)

Removes the numbering column from the data set

```
beads = beads1[,-1]
```

divides the dataset into seperate tables

```
class = beads[,-(2:5)]
order = beads[,-c(1,3:5)]
genus = beads[,-c(1:2,4:5)]
species = beads[,-c(1:3,5)]
location = beads[,-c(1:4)]
```

CLASS

```
dimbeads=dim(beads)
obtains number of rows in class table
n = nrow(class)
```

sets a seed so results can be replicated

```
set.seed(101)
```

Sample allows for a 70sample = sample(1:n, size = round(0.7*n))

seperating the class dataset into train and test data

```
trainclass = class[sample,]
```

testclass = class[-sample,]

uses the rpart function to create a tree
fitclasstrain <- rpart(class asx+glx+ser+gly+ala+val, method="class", data=trainclass,
control =rpart.control(minsplit=10, minbucket=5, cp=0.0001))
fitclasstest <- rpart(class asx+glx+ser+gly+ala+val, method="class", data=testclass,
control =rpart.control(minsplit=10, minbucket=5, cp=0.0001))
Grow decision tree
plot(fitclasstrain, uniform=TRUE, main="Decision Tree")
text(fitclasstrain, use.n=TRUE, all=TRUE, cex=0.5)

plot(fitclasstest, uniform=TRUE, main="Decision Tree")
text(fitclasstest, use.n=TRUE, all=TRUE, cex=.5)

create training and test confusion matrices tree.pred=predict(fitclasstrain,trainclass[,-
1],type="class") confusionMatrix(tree.pred,as.factor(trainclass$class$))

tree.pred=predict(fitclasstest,testclass[,-1],type="class")
confusionMatrix(tree.pred,as.factor(testclass$class$))

printcp(fitclasstrain) display the results
summary(fitclasstrain) detailed summary of splits

RANDOMFOREST
trainclass$class$ = $factor$($trainclass$class)
rf.beadtrainclass = randomForest(class asx+glx+ser+gly+ala+val, data = trainclass)
rf.beadtrainclass

testclass$class$ = $factor$($testclass$class)
rf.beadtestclass = randomForest(class asx+glx+ser+gly+ala+val, data = testclass)
rf.beadtestclass

ORDER
obtains number of rows in the order table
n1 = nrow(order)

70sample1 = sample(1:n1, size = round(0.7*n1))
seperating the class dataset into train and test data
trainorder = order[sample1,]
testorder = order[-sample1,]
uses the rpart function to create a tree
fitordertrain <- rpart(order asx+glx+ser+gly+ala+val, method="class", data=trainorder
)

```
fitordertest <- rpart(order  asx+glx+ser+gly+ala+val, method="class", data=testorder)
Grow decision tree
plot(fitordertrain, uniform=TRUE, main="Decision Tree")
text(fitordertrain, use.n=TRUE, all=TRUE, cex=.8)

    plot(fitordertest, uniform=TRUE, main="Decision Tree")
text(fitordertest, use.n=TRUE, all=TRUE, cex=.8)

    create training and test confusion matrices
tree.pred=predict(fitordertrain,trainorder[,-1],type="class")
confusionMatrix(tree.pred,as.factor(trainorder$order$))

    tree.pred=predict(fitordertest,testorder[,-1],type="class")
confusionMatrix(tree.pred,as.factor(testorder$order$))

    RANDOMFOREST
trainorder$order$ = $factor$($trainorder$order)
rf.beadtrainorder = randomForest(order  asx+glx+ser+gly+ala+val, data = trainorder)
rf.beadtrainorder

    testorder$order$ = $factor$($testorder$order)
rf.beadtestorder = randomForest(order  asx+glx+ser+gly+ala+val, data = testorder)
rf.beadtestorder

    GENUS
obtains number of rows in the genus table
n2 = nrow(genus)

    70sample2 = sample(1:n2, size = round(0.7*n2))
seperates into train and test set
traingenus = genus[sample2,]
testgenus = genus[-sample2,]
generates decision tree
fitgenustrain <- rpart(Genus  asx+glx+ser+gly+ala+val, method="class", data=traingenus
)
fitgenustest <- rpart(Genus  asx+glx+ser+gly+ala+val, method="class", data=testgenus)
Grow decision tree
plot(fitgenustrain, uniform=TRUE, main="Decision Tree")
text(fitgenustrain, use.n=TRUE, all=TRUE, cex=.8)

    plot(fitgenustest, uniform=TRUE, main="Decision Tree")
text(fitgenustest, use.n=TRUE, all=TRUE, cex=.8)
```

generate confusion matrix
tree.pred=predict(fitgenustrain,traingenus[,-1],type="class")
confusionMatrix(tree.pred,as.factor(traingenus$Genus$))

tree.pred=predict(fitgenustest,testgenus[,-1],type="class")
confusionMatrix(tree.pred,as.factor(testgenus$Genus$))

RANDOMFOREST
traingenus$Genus = factor(traingenus$Genus)
rf.beadtraingenus = randomForest(Genus asx+glx+ser+gly+ala+val, data = traingenus)
rf.beadtraingenus

testgenus$Genus = factor(testgenus$Genus)
rf.beadtestgenus= randomForest(Genus asx+glx+ser+gly+ala+val, data = testgenus)
rf.beadtestgenus

SPECIES
obtains number of rows in species
n3 = nrow(species)
samples data into 70sample3 = sample(1:n3, size = round(0.7*n3))
generates training and test set
trainspecies = species[sample3,]
testspecies = species[-sample3,]
generates decision tree
fitspeciestrain <- rpart(Species asx+glx+ser+gly+ala+val, method="class", data=trainspecies)
fitspeciestest <- rpart(Species asx+glx+ser+gly+ala+val, method="class", data=testspecies)
Grow decision tree
plot(fitspeciestrain, uniform=TRUE, main="Decision Tree")
text(fitspeciestrain, use.n=FALSE, all=TRUE, cex=.8)

plot(fitspeciestest, uniform=TRUE, main="Decision Tree")
text(fitspeciestest, use.n=TRUE, all=TRUE, cex=.8)

generates confusion matrices
tree.pred=predict(fitspeciestrain,trainspecies[,-1],type="class")
confusionMatrix(tree.pred,as.factor(trainspecies$Species$))

tree.pred=predict(fitspeciestest,testspecies[,-1],type="class")
confusionMatrix(tree.pred,as.factor(testspecies$Species$))

RANDOMFOREST

trainspecies$Species = factor(trainspecies$Species)
rf.beadtrainspecies = randomForest(Species asx+glx+ser+gly+ala+val, data = train-
species)
rf.beadtrainspecies

testspecies$Species = factor(testspecies$Species)
rf.beadtestspecies= randomForest(Species asx+glx+ser+gly+ala+val, data = test-
species)
rf.beadtestspecies

LOCATION
obtains number of rows in location table
n4 = nrow(location)

gets sample for 70sample4 = sample(1:n4, size = round(0.7*n4))
seperates into 70trainlocation = location[sample4,]
testlocation = location[-sample4,]
generates decision tree
fitlocationtrain <- rpart(Location asx+glx+ser+gly+ala+val, method="class", data=trainlocation
)
fitlocationtest <- rpart(Location asx+glx+ser+gly+ala+val, method="class", data=testlocation)
Grow decision tree
plot(fitlocationtrain, uniform=TRUE, main="Decision Tree")
text(fitlocationtrain, use.n=FALSE, all=TRUE, cex=0.8)
plot(fitlocationtest, uniform=TRUE, main="Decision Tree")
text(fitlocationtest, use.n=FALSE, all=TRUE, cex=.66)
generates confusion matrix
tree.pred=predict(fitlocationtrain,trainlocation[,-1],type="class")
confusionMatrix(tree.pred,as.factor(trainlocation$Location$))

tree.pred=predict(fitlocationtest,testlocation[,-1],type="class")
confusionMatrix(tree.pred,as.factor(testlocation$Location$))

printcp(fitlocationtrain) display the results
summary(fitlocationtrain) detailed summary of splits

RANDOMFOREST
trainlocation$Location = factor(trainlocation$Location)
rf.beadtrainlocation = randomForest(Location asx+glx+ser+gly+ala+val, data = train-
location)
rf.beadtrainlocation

testlocation$Location = factor(testlocation$Location)

rf.beadtestlocation = randomForest(Location asx+glx+ser+gly+ala+val, data = test-location)
rf.beadtestlocation
plot(rf.beadtestlocation)

References

[1] Ramli, Z. et al. 2009. *Beads trade in Peninsula Malaysia: based on archaeological evidences*. European Journal of Social Sciences, 10(4), pp.585-593.