Classfication of Vowel sounds

Final Project_STAT 8090

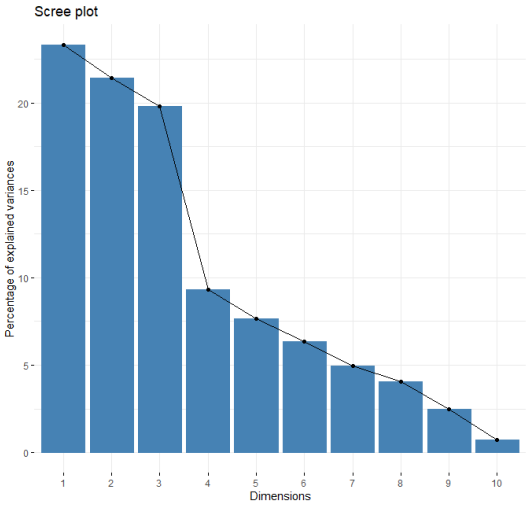
Name: Yukang Xu

Panther ID: 002462280

I. TASK 1

PCA works best for numeric data. Fortunately, all of variable are numeric so there no need to drop any variable. I apply pdcomp() then obtain 10 principal components, which are PC1-10. Each of these explains a percentage of the total variation in the dataset. That is to say: PC1 explains 23% of the total variance, which means that nearly one-fouths of the information in the dataset (10 variables) can be encapsulated by just that one Principal Component. PC2 explains 21% of the variance. So, by knowing the position of a sample in relation to just PC1 and PC2, you can get a rough view on where it stands in relation to other samples, as just PC1 and PC2 can explain 46% of the variance. There will be more than 7 components if we are trying to explain at least 90% of the total sample variance in the training data. To classify our variables more clearly, I also use the eigenvalues to draw a scree plot as below.

```
Importance of components:
                          PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9    PC10
Standard deviation     1.5270 1.4636 1.4075 0.96503 0.87452 0.79565 0.70388 0.63586 0.49604 0.26503
Proportion of Variance 0.2332 0.2142 0.1981 0.09313 0.07648 0.06331 0.04955 0.04043 0.02461 0.00702
Cumulative Proportion  0.2332 0.4474 0.6455 0.73861 0.81509 0.87839 0.92794 0.96837 0.99298 1.00000
```



II. TASK 2

we're trying to describe what qualities in x contributes to whether or not it's in specific classification. After implementing principal component score analysis for x.1-x.9,I decided to drop pca9 and pca10 because they can only explain less than 3% of variation. Now that our data is ready, we can use the lda() function i R to make our analysis which is functionally identical to the lm() and glm() functions. When we train a model based on linear discriminant analysis. We can use this model to get misclassification error rate which is 35% for training dataset. When I apply this model to testing dataset, I get the same rate. It means that this model can represent our data trend.

III. TASK 3

After implements quadratic discriminant analysis and compare misclassification error rate with LDA, QDA model perform a lot more better than LDA because the misclassification error rate in QDA for training data and testing data is 4.9%. LDA methods are closely connected and differ primarily in its fitting procedures. QDA, on the other-hand, provides a non-linear quadratic decision boundary. Thus, when the decision boundary is moderately non-linear, QDA may give better results.

IV. TASK 4

After implementing LDA and QDA in original dataset, we get the similar results that QDA perform better. Misclassification error rate in LDA is 31% while the rate in QDA is 1%. Because 1% is smaller than 4.9% which we get from task2. So we can say, the performance of original dataset perform better in model development in this case. What is important to keep in mind is that no one method will dominate the others in every situation. But in this case, QDA apparently is the better choice.

Misclassification error rate	**Original data**	**Scoring data**
LDA in training data	31%	35%
QDA in training data	1.1%	4.9%
LDA in testing data	31%	35%
QDA in testing data	1.1%	4.9%

V. TASK 5

To find the observations that we need to drop, I create a table to see the classifications we misclassified as below. I picked the most three misclassification types to drop, y=2,6,9. After that , I rerun LDA and QDA again. We can easily find that misclassification error rate going down. Rate of LDA goes down to 20% from 31% while rate of QDA goes down to 0 from 1.1%.

The rate difference was shown in the table below.

Misclassification error rate	Original data	Scoring data	Data after dropping observations
LDA in training data	31%	35%	20%
QDA in training data	1.1%	4.9%	0
LDA in testing data	31%	35%	20%
QDA in testing data	1.1%	4.9%	0

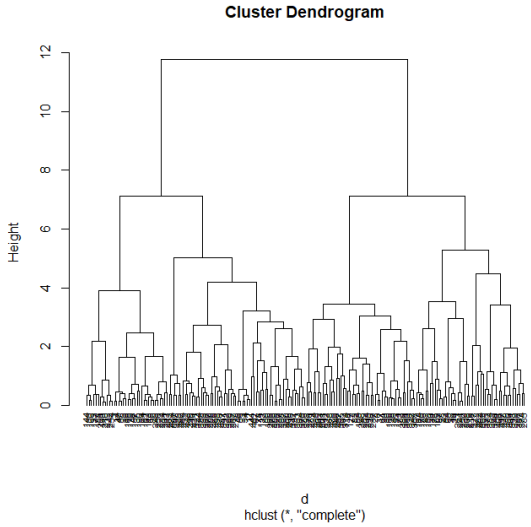
```
> table(a[,1])

 1  2  3  4  5  6  7  8  9 10 11
32 40 14 24 32 56 30 28 38 32 20
```

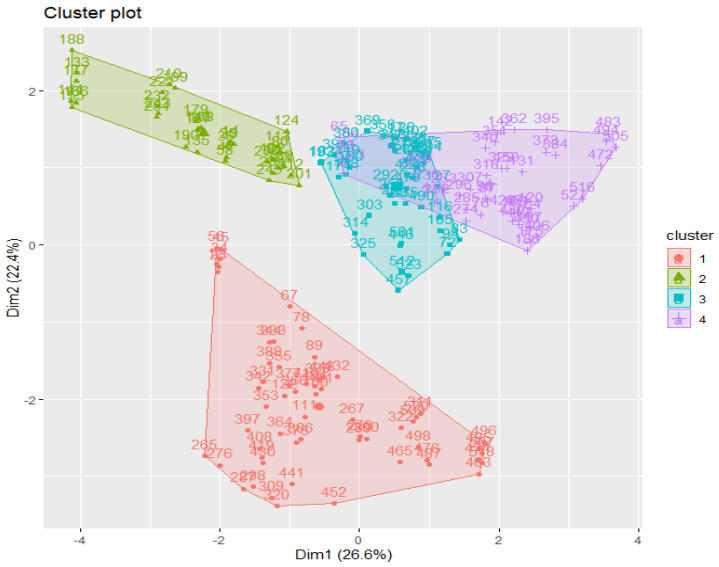
VI. TASK 5

A. hierarchical clustering analysis

Hierarchical clustering is an alternative approach to k-means clustering for identifying groups in the dataset. It does not require us to pre-specify the number of clusters to be generated as is required by the k-means approach. Furthermore, hierarchical clustering has an added advantage over K-means clustering in that it results in an attractive tree-based representation of the observations, called a dendrogram. We can perform agglomerative HC with hclust. First we compute the dissimilarity values with dist and then feed these values into hclust and specify the agglomeration method to be used (i.e. “complete”, “average”, “single”, “ward.D”). We can then plot the dendrogram.



we can also use the fviz_cluster function from the factoextra package to visualize the result in a scatter plot



B. K-mean analysis

K-means clustering is the simplest and the most commonly used clustering method for splitting a dataset into a set of k groups. We can compute k-means in R with the kmeans function. Here will group the data into four clusters (centers = 4). The kmeans function also has an nstart option that attempts multiple initial configurations and reports on the best one. For example, adding nstart = 25 will generate 25 initial configurations. This approach is often recommended. If we print the result we’ll see that our groupings resulted in 4 cluster. We see the cluster centers (means) for the four groups across the 10 variables. We also get the cluster assignment for each observation.

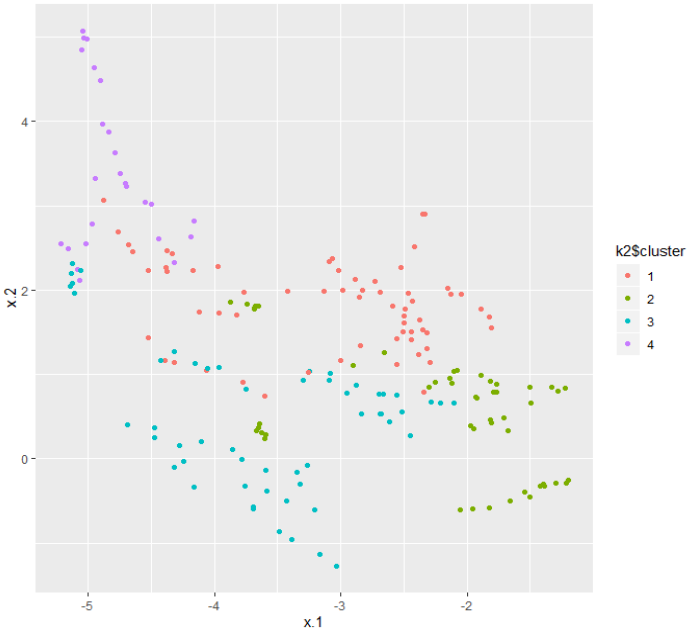
```
K-means clustering with 4 clusters of sizes 63, 50, 94, 25

Cluster means:
          y       x.1       x.2       x.3        x.4        x.5        x.6        x.7        x.8        x.9       x.10
1 0.6115724  0.1101235 0.4104204 0.1056907 -0.34897179 -0.9224185 -0.2217652 0.3541672 0.03090136 0.6210564 0.3334720
2 -0.5176424 0.8880068 -0.6090804 -1.0944489 0.03273906 -0.1399416 1.0262648 -0.1051058 0.96723834 -0.2507775 -0.7388488
3 -0.9150244 -0.3142349 -0.6872033 0.6994156 0.88259999 0.9103444 -0.3602780 -0.5163629 -1.04086833 -0.2637681 0.6923429
4 1.4705749 -1.3747775 1.6682605 0.4118197 -1.09248520 0.6380340 -0.7154808 0.4330541 0.23592747 -0.4937682 -0.8581124

Clustering vector:
  1   3   6  10  12  14  17  21  23  25  28  32  34  36  39  43  45  47  50  54  56  58  61  65  67  69  72  76  78  80  83  87  89  91
 94  98
  2   2   1   2   2   2   1   2   2   2   1   2   2   2   1   2   2   2   1   2   2   2   1   2   3   2   1   1   3   2   1   1   3   2
  1   1
100 102 105 109 111 113 116 120 122 124 127 131 133 135 138 142 144 146 149 153 155 157 160 164 166 168 171 175 177 179 182 186 188 190
193 197
  3   2   1   1   3   2   1   1   3   2   1   4   2   2   1   1   2   2   1   1   2   2   1   1   2   2   1   1   2   2   1   1   2   2
  1   1
199 201 204 208 210 212 215 219 221 223 226 230 232 234 237 241 243 245 248 252 254 256 259 263 265 267 270 274 276 278 281 285 287 289
```

We can also view our results by using visulization. This provides a nice illustration of the clusters. If there are more than

two dimensions (variables) will perform principal component analysis (PCA) and plot the data points according to the first two principal components that explain the majority of the variance.



C. Model based analysis

Model based analysis consider the data as coming from a distribution that is mixture of two or more clusters, Unlike k-means, the model-based clustering uses a soft assignment, where each data point has a probability of belonging to each cluster. But after I implement this method, the recommendation classification is 9 which make analysis more complicated. So I will not choose this method.

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VEV (ellipsoidal, equal shape) model with 9 components:

 log-likelihood   n  df       BIC       ICL
       266.865 192 621 -2731.175 -2731.175

Clustering table:
 1  2  3  4  5  6  7  8  9
 6  6 78 24 30  6 12  6 24
```

D. Assessment

Because model-based analysis is not that suitable for this case, I will make a choice between K-mean and hierarchical. I am trying to develop confusion matrix for these two methods and compare misclassification rate. From the table below, hierarchical clustering classify more observations to suitable group, So we choose hierarchical as our model

```
> table(data5[,1],k2$cluster)

    1  2  3  4
1   0 18 30  0
2   0 24 24  0
3 46  2  0  0
4 17  6  0 25
> table(data5[,1],sub_grp)
   sub_grp
    1  2  3  4
1  36 12  0  0
2  24 24  0  0
3   0  0 48  0
4   0  0  0 48
```

VII. TASK 7

Form the analysis above, I will draw a conclusion that hierarchical analysis perform best because it can put our observations to the suitable group.

R Code:

```
# 1
data <- read.table("C:/Users/xuyuk/OneDrive - Georgia State University/Data import/vowel-train.txt",
                   header = TRUE, sep = ",", na.strings = 'null')
pca <- prcomp(data[,c(3:12)], center = TRUE,scale. = TRUE)
summary(pca)
library(factoextra)
fviz_eig(pca)


# 2
data1=data[,3:10]
R<-cor(data1)
e<-eigen(R)
zdat<-scale(data1)
pca.scores<- zdat %*% e$vectors
colnames(pca.scores)<-c('pca1','pca2','pca3','pca4','pca5','pca6','pca7','pca8')
head(pca.scores)
data2=cbind.data.frame(data[,2],pca.scores[,1:8])
colnames(data2)<-c('y','pca1','pca2','pca3','pca4','pca5','pca6','pca7','pca8')
library(MASS)
lda <- lda(y ~ pca1+pca2+pca3+pca4+pca5+pca6, data = data2)
trainpred=predict(lda,data2)
mean(trainpred$class !=data2$y)
test <- read.table("C:/Users/xuyuk/OneDrive - Georgia State University/Data import/vowel-train.txt",
                   header = TRUE, sep = ",", na.strings = 'null')
test1=test[,3:10]
R1<-cor(test1)
e1<-eigen(R1)
zdat<-scale(test1)
pca.scores<- zdat %*% e1$vectors
colnames(pca.scores)<-c('pca1','pca2','pca3','pca4','pca5','pca6','pca7','pca8')
head(pca.scores)
test2=cbind.data.frame(test[,2],pca.scores[,1:8])
colnames(test2)<-c('y','pca1','pca2','pca3','pca4','pca5','pca6','pca7','pca8')
library(MASS)
trainpred1=predict(lda,test2)
mean(trainpred1$class !=test2$y)


# 3
qda.model = qda (y ~ pca1+pca2+pca3+pca4+pca5+pca6+pca7+pca8, data = data2)
trainpred=predict(qda.model,data2)
mean(trainpred$class !=data2$y)
trainpred1=predict(qda.model,test2)
mean(trainpred1$class !=test2$y)


# 4
data=data[,2:12]
ldaori <- lda(y~., data = data)
qdaori= qda (y~., data = data)
test3=test[,2:12]
trainpred=predict(ldaori,data)
mean(trainpred$class !=data$y)
trainpred=predict(ldaori,test3)
mean(trainpred$class !=test3$y)
trainpred=predict(qdaori,data)
mean(trainpred$class !=data$y)
trainpred=predict(qdaori,test3)
mean(trainpred$class !=test3$y)


# 5
trainpred2=predict(ldaori,data)
trainpred3=predict(ldaori,test3)
trainpred4=predict(qdaori,data)
trainpred5=predict(qdaori,test3)
errorline1=data[which(trainpred2$class !=data$y),]
errorline2=test3[which(trainpred3$class !=test3$y),]
errorline3=data[which(trainpred4$class !=data$y),]
errorline4=test3[which(trainpred5$class !=test3$y),]
a=rbind(errorline1,errorline2,errorline3,errorline4)
table(a[,1])
data3=subset(data, y!=2 & y!=6 & y!=9)
data4=subset(test3, y!=2 & y!=6 & y!=9)
ldaori <- lda(y~., data = data3)
qdaori= qda (y~., data = data3)
trainpred=predict(ldaori,data3)
mean(trainpred$class !=data3$y)
trainpred=predict(ldaori,data4)
```

```
mean(trainpred$class !=data4$y)
trainpred=predict(qdaori,data3)
mean(trainpred$class !=data3$y)
trainpred=predict(qdaori,data4)
mean(trainpred$class !=data4$y)


# 6
library(dendextend)
library(cluster)
library(factoextra)
data5=subset(data, y!=2 & y!=4 & y!=5 & y!=7 & y!=8 & y!=9 & y!=11)
data6=subset(test3, y!=2 & y!=4 & y!=5 & y!=7 & y!=8 & y!=9 & y!=11)
# hierarchical clustering analysis
d <- dist(data5, method = "euclidean")
hc1 <- hclust(d, method = "complete" )
plot(hc1, cex = 0.6, hang = -1)
hc5 <- hclust(d, method = "ward.D2" )
sub_grp <- cutree(hc5, k = 4)
fviz_cluster(list(data = data5, cluster = sub_grp))
# K-mean method
k2 <- kmeans(scale(data5), centers = 4, nstart = 25)
k2$cluster <- as.factor(k2$cluster)
ggplot(data5, aes(x.1, x.2, color = k2$cluster)) + geom_point()
# Model-based clustering
fit <- Mclust(data5)
summary(fit)
# assessment of clustering
data5$y[data5$y == 3] <- 2
data5$y[data5$y == 6] <- 3
data5$y[data5$y == 10] <- 4
table(data5[,1],k2$cluster)
table(data5[,1],sub_grp)
```