LIFE748 - Machine learning Assignment

Declaration:

I declare that I have employed a Chat-GPT-3.5, to assist in the creation of this .r script. Specifically, I used it to proactively identify and handle potential exceptions and errors that may arise during execution. I also used it to better understand the machine learning code from the workshops to help me annotate the code further.

Instructions

This assignment aims to assess your understanding and application of machine learning techniques covered in the lectures and workshops, including k-means clustering, hierarchical clustering, logistic regression, Linear Discriminant Analysis (LDA), and Support Vector Machines (SVM).

Submission Requirements:

1. **Quarto Document (.qmd)**: All your answers, code, and explanations must be included within this Quarto document (.qmd)
2. **PDF Output**: Render your Quarto document into a PDF file.
3. **Submission**: Submit both the .qmd file and the generated PDF file.

General Guidelines:

- **Completeness:** Answer all questions thoroughly, providing the code used.
- **Code Comments**: Comment your code effectively to explain each step

- **Programming language:** While R is strongly recommended, you are allowed to use Python for this assignment. Please ensure that all Python code is executed within the Quarto document. No technical support will be provided for Python-related issues.
- **Minimal Text:** Ensure that your explanations are clear and concise. Focus on providing the reasoning behind your results.
- **Reproducibility**: Ensure that your code is reproducible. We must be able to run your .qmd file and obtain the same results as in your PDF.

  Good luck!

Clustering - Unsupervised Learning

Load the unlabelled dataset provided with the assignment.

```
# Loading the unlabelled data
df <- read.csv('Unlabelled_data_Student1.csv', row.names = 1)
# Inspect the data
head(df)
```

```
      gene1     gene2
1 0.6122280 0.7845059
2 0.2951438 1.0236957
3 0.3305224 0.9502435
4 0.6587481 0.6669995
5 0.4107384 0.7300809
6 0.4735187 0.9910019
```

Q1: Apply k-means clustering to the provided dataset. Experiment with at least 3 different values of 'k' (number of clusters).

The number of clusters were experimented with using k=2, 3, and 4.

```
# Load necessary library ggplot2
library(ggplot2)
```

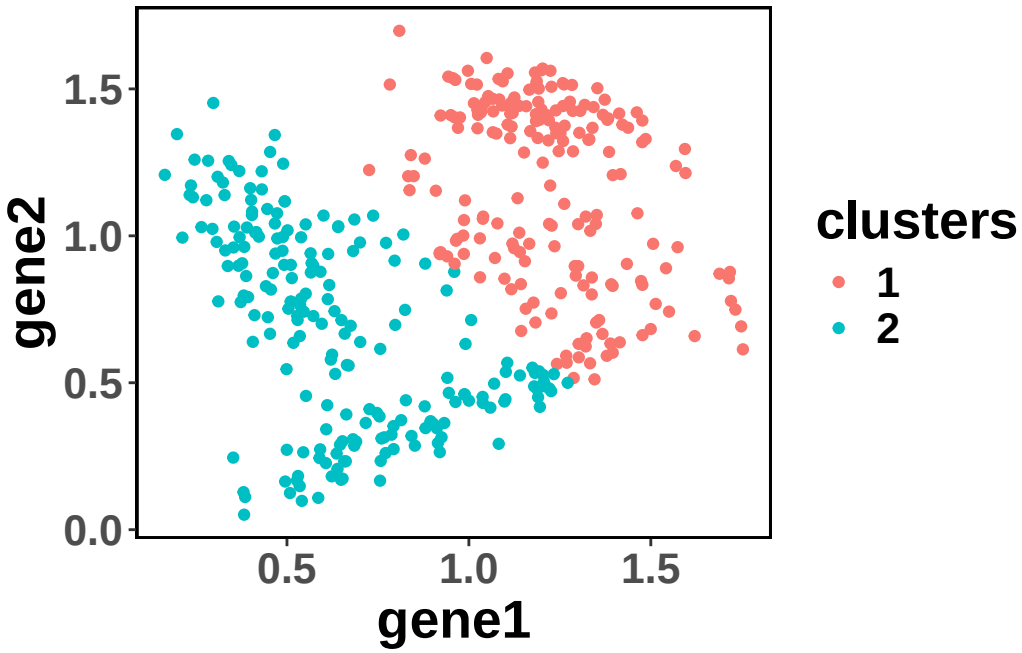
```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```
# Set seed for reproducibility
set.seed(123)
# Apply k means clustering with 2 clusters
out2 <- kmeans(df, centers = 2, iter.max = 15, nstart=5)
out2
```

```
K-means clustering with 2 clusters of sizes 196, 204

Cluster means:
      gene1     gene2
1 1.2205969 1.1587038
2 0.6444082 0.6789026

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
  1   1   1   1   2   1   2   1   1   1   1   1   2   1   1   2   1   1   1   1
221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
  1   1   2   1   1   2   1   1   1   1   2   1   1   1   1   1   2   2   1   1
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
  1   1   1   2   1   1   1   1   1   1   1   1   1   1   2   1   2   1   1   1
261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
  1   1   1   1   1   1   1   1   1   1   2   2   1   2   2   1   1   1   1   1
281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
  1   1   2   1   1   1   1   1   2   1   1   1   2   1   1   1   1   1   2   1
301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
  2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   1   2   2   2   2
321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
  2   2   2   2   2   1   2   1   2   2   2   2   2   2   2   2   2   2   2   2
341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
```

```
  2   2   2   1   2   2   2   2   2   2   2   1   1   2   2   2   2   2   2   2
361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
  2   1   2   1   2   2   2   1   2   1   2   2   2   2   2   2   2   2   2   2
381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
  2   2   2   1   2   1   2   2   2   2   1   2   2   2   1   2   2   2   2   2

Within cluster sum of squares by cluster:
[1] 26.81975 37.28098
 (between_SS / total_SS =  46.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# Setting the clusters as factors for plotting
df$clusters <- as.factor(out2$cluster)
# Plotting the 2 clusters as a scatterplot, coloured by
# cluster
ggplot(df, aes(x=gene1, y=gene2))+geom_point(aes(col=clusters))+
  theme(panel.background = element_rect(fill = "white",colour = "black",
                                        linewidth = 1,linetype = "solid"),
        panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
        text = element_text(size=20,colour = "black",face = "bold"),
        legend.key=element_blank())
```

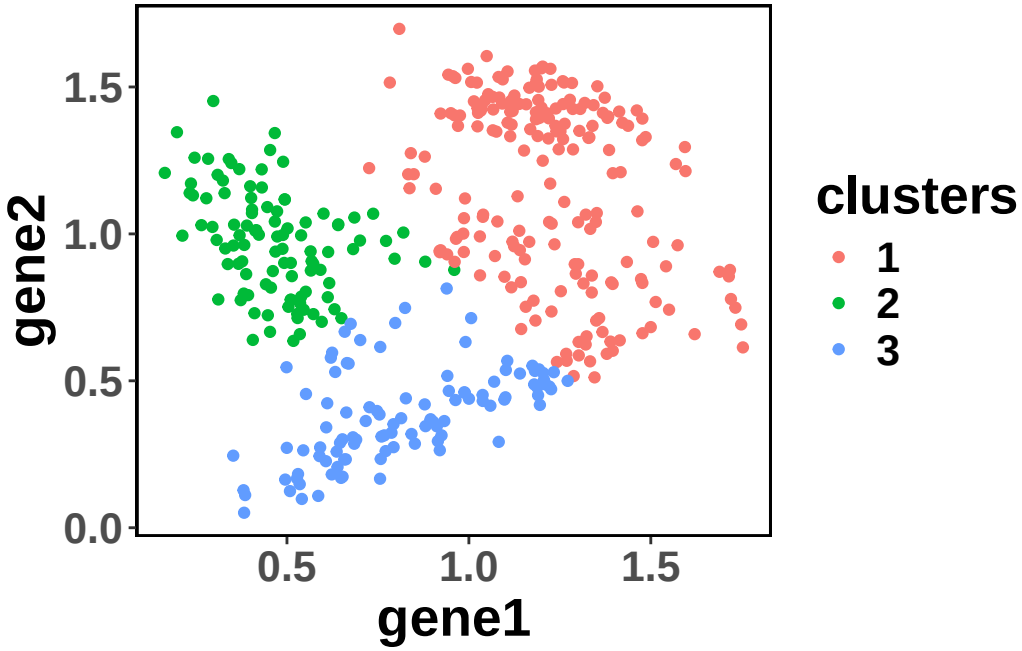



```
# Set seed for reproducibility
set.seed(123)
# Apply k means clustering for 3 clusters
out3 <- kmeans(df, centers = 3, iter.max = 15, nstart=5)
out3
```

```
K-means clustering with 3 clusters of sizes 196, 102, 102

Cluster means:
      gene1     gene2 clusters
1 1.2205969 1.1587038        1
2 0.4693325 0.9699348        2
3 0.8194838 0.3878703        2

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
  2   2   2   3   2   2   2   2   2   2   2   2   2   3   2   2   2   2   2   2
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  2   2   3   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   3   2
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  3   2   2   2   2   2   3   3   2   2   2   2   2   2   2   2   3   2   2   3
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
  2   2   2   2   2   2   2   2   3   3   2   2   2   2   2   2   2   2   2   2
```

```
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
  2   2   3   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
  1   1   1   1   2   1   2   1   1   1   1   1   3   1   1   2   1   1   1   1
221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
  1   1   2   1   1   2   1   1   1   1   3   1   1   1   1   1   2   2   1   1
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
  1   1   1   2   1   1   1   1   1   1   1   1   1   1   2   1   2   1   1   1
261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
  1   1   1   1   1   1   1   1   1   1   2   2   1   2   2   1   1   1   1   1
281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
  1   1   3   1   1   1   1   1   3   1   1   1   3   1   1   1   1   1   3   1
301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
  3   3   3   3   3   1   3   3   3   3   3   3   3   1   3   1   3   3   3   3
321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
  3   3   3   3   3   1   3   1   3   3   3   3   3   3   3   3   3   3   3   3
341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
  3   3   3   1   3   3   3   3   3   3   3   1   1   3   3   3   3   3   3   3
361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
  3   1   3   1   3   3   3   1   3   1   3   3   3   3   3   3   3   3   3   3
381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
  3   3   3   1   3   1   3   3   3   3   1   3   3   3   1   3   3   3   3   3

Within cluster sum of squares by cluster:
[1] 26.819747  5.446974  8.302349
 (between_SS / total_SS =  81.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# Setting the clusters as factors for plotting
df$clusters <- as.factor(out3$cluster)
# Plotting the 3 clusters as a scatterplot,
# coloured by cluster
ggplot(df, aes(x=gene1, y=gene2))+geom_point(aes(col=clusters))+
  theme(panel.background = element_rect(fill = "white",colour = "black",
                                        linewidth = 1,linetype = "solid"),
        panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
        text = element_text(size=20,colour = "black",face = "bold"),
        legend.key=element_blank())
```



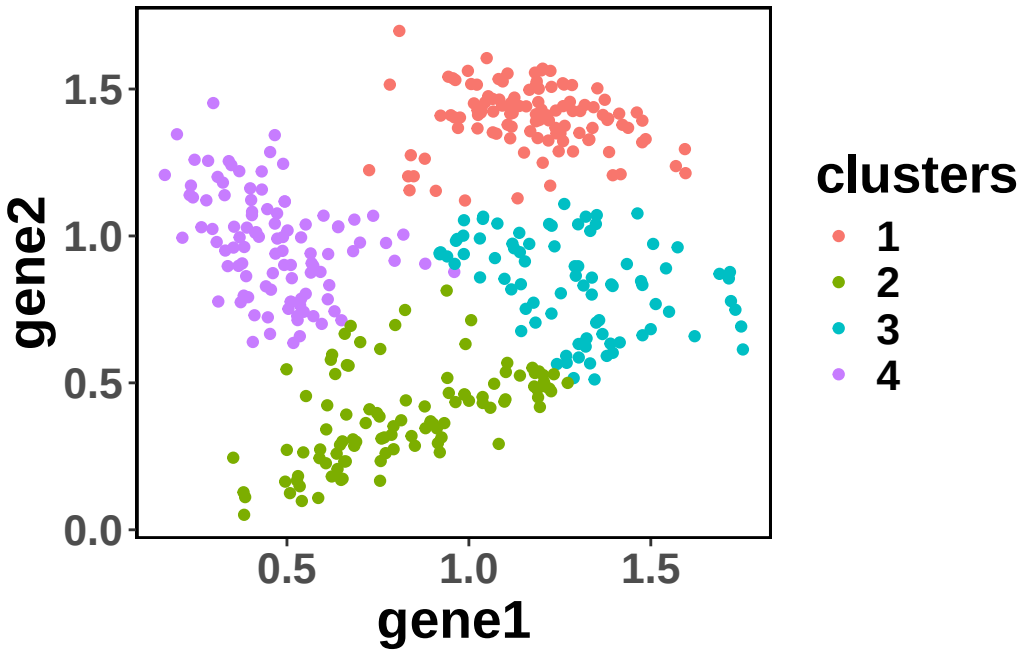

```r
# set seed for reproducibility
set.seed(123)
# Apply k mean clustering with 4 clusters
out4 <- kmeans(df, centers = 4, iter.max = 15, nstart=5)
out4
```

```
K-means clustering with 4 clusters of sizes 112, 102, 84, 102

Cluster means:
      gene1     gene2 clusters
1 1.1690647 1.4001878        1
```

```
2 0.8194838 0.3878703        3
3 1.2893066 0.8367252        1
4 0.4693325 0.9699348        2

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
  4   4   4   2   4   4   4   4   4   4   4   4   4   2   4   4   4   4   4   4
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  4   4   2   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4   2   4
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
  2   4   4   4   4   4   2   2   4   4   4   4   4   4   4   4   2   4   4   2
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
  4   4   4   4   4   4   4   4   2   2   4   4   4   4   4   4   4   4   4   4
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
  4   4   2   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
  1   3   1   1   4   3   4   1   3   3   3   3   2   3   1   4   3   3   3   3
221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
  3   3   4   1   3   4   3   3   3   3   2   3   3   3   3   3   4   4   3   3
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
  3   3   3   4   1   3   3   3   3   3   3   3   3   1   4   3   4   3   1   3
261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
  1   3   3   3   3   3   3   3   3   3   4   4   3   4   4   3   3   3   3   3
281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
  3   3   2   3   3   1   3   1   2   3   3   3   2   3   3   3   3   3   2   3
301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
  2   2   2   2   2   3   2   2   2   2   2   2   2   3   2   3   2   2   2   2
321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
  2   2   2   2   2   3   2   3   2   2   2   2   2   2   2   2   2   2   2   2
341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
  2   2   2   3   2   2   2   2   2   2   2   2   3   3   2   2   2   2   2   2
361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
  2   3   2   3   2   2   2   3   2   3   2   2   2   2   2   2   2   2   2   2
```

```
381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
  2   2   2   3   2   3   2   2   2   2   3   2   2   2   3   2   2   2   2   2

Within cluster sum of squares by cluster:
[1] 4.916547 8.302349 5.969684 5.446974
 (between_SS / total_SS =  93.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

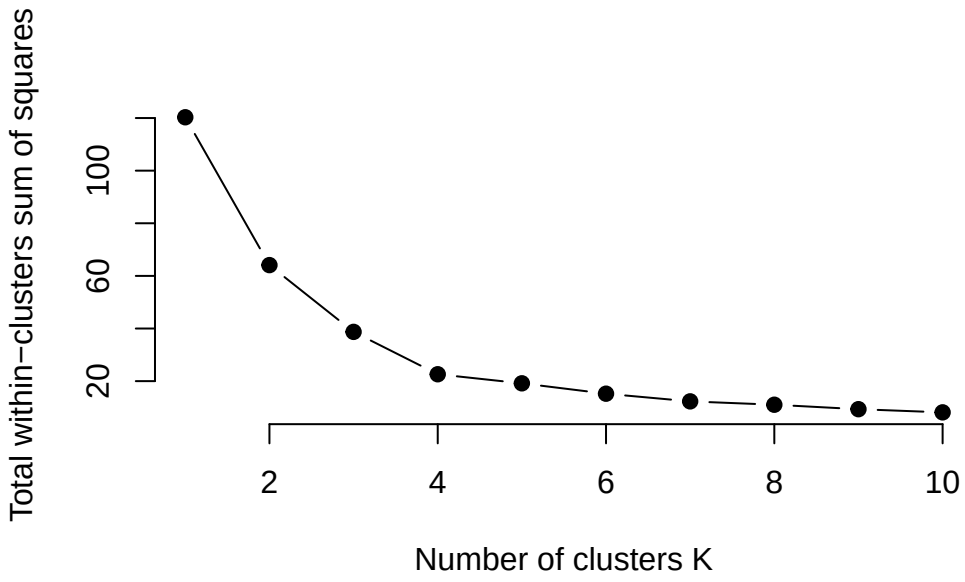
```r
# Setting the clusters as factors for plotting
df$clusters <- as.factor(out4$cluster)
# Plotting the 4 clusters as a scatterplot,
# coloured by cluster
ggplot(df, aes(x=gene1, y=gene2))+geom_point(aes(col=clusters))+
  theme(panel.background = element_rect(fill = "white",colour = "black",
                                        linewidth = 1,linetype = "solid"),
        panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
        text = element_text(size=20,colour = "black",face = "bold"),
        legend.key=element_blank())
```



Q2: Use the elbow method score to determine the optimal number of clusters for this dataset, and then apply k-means with the chosen k.

An elbow graph was used to identify the optimal number of clusters to be between 2 and 4.

```
# set seed for reproducibility
set.seed(123)
# Compute and plot within clusters sum of
# squares for k = 1:10
k.max <- 10
wss <- sapply(1:k.max,
              function(k){kmeans(df[,1:2], k, nstart=5,iter.max = 15 )$tot.withinss})
# plots an elbow plot with maximum
# clusters of 10
plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```

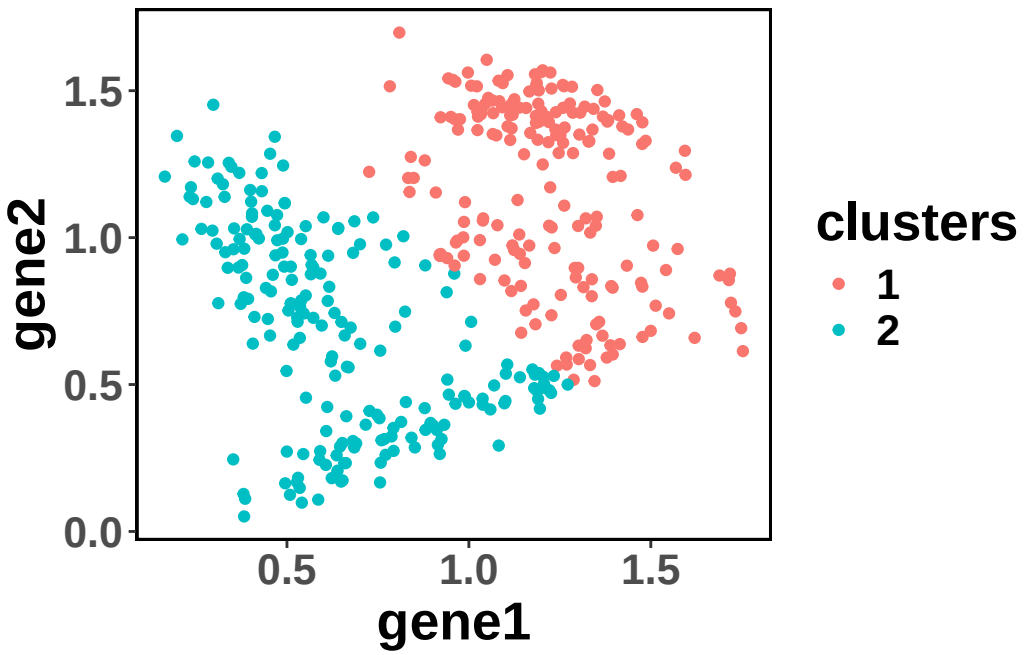


Q3: Visualise the k-means clustering results.

The K-means clustering were visualised by clustering with 2, 3 and 4 clusters as this was the range shown by the elbow graph.

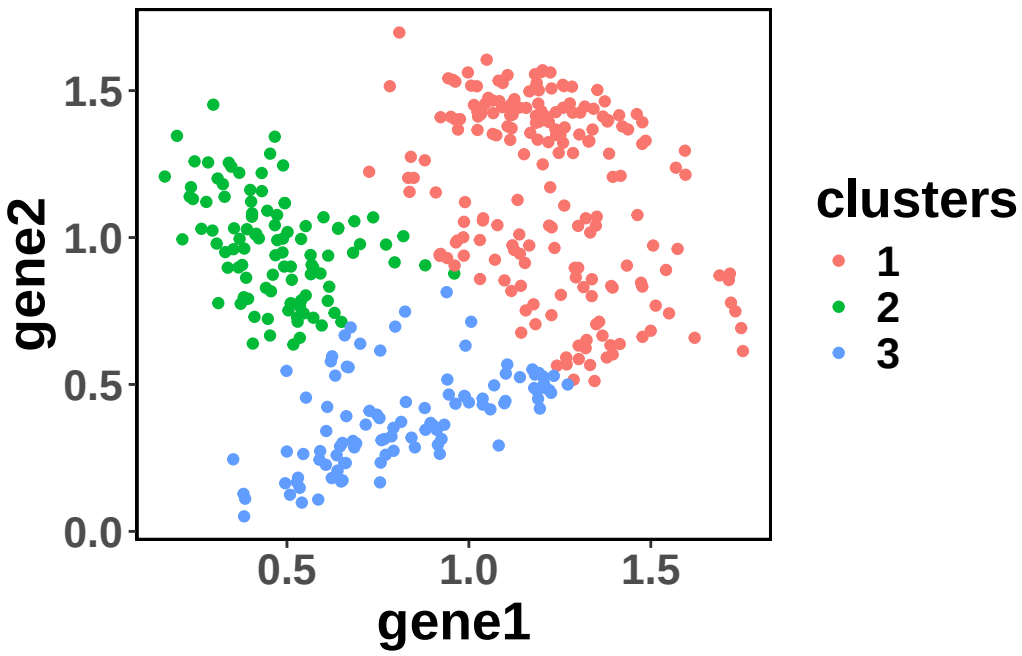
The elbow method was used to find the optimal K-means cluster number (Shahulsyed, 2024).

```
# set seed for reproducibility
set.seed(123)
# Plotting the 2 clusters as a scatterplot,
# coloured by cluster
df$clusters <- as.factor(out2$cluster)
ggplot(df, aes(x=gene1, y=gene2))+geom_point(aes(col=clusters))+
  theme(panel.background = element_rect(fill = "white",colour = "black",
                                        linewidth = 1,linetype = "solid"),
        panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
        text = element_text(size=20,colour = "black",face = "bold"),
        legend.key=element_blank())
```

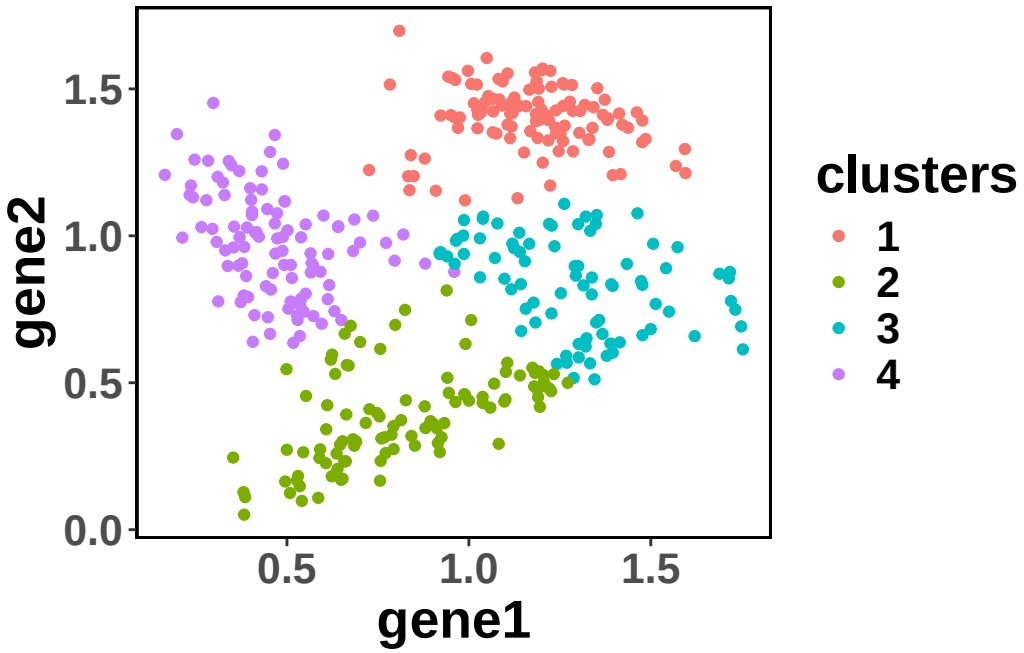



```
# set seed for reproducibility
set.seed(123)
# Plotting the 3 clusters as a scatterplot,
# coloured by cluster
df$clusters <- as.factor(out3$cluster)
ggplot(df, aes(x=gene1, y=gene2))+geom_point(aes(col=clusters))+
  theme(panel.background = element_rect(fill = "white",colour = "black",
                                        linewidth = 1,linetype = "solid"),
        panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
        text = element_text(size=20,colour = "black",face = "bold"),
```

```
        legend.key=element_blank())
```




```
# set seed for reproducibility
set.seed(123)
# Plotting the 2 clusters as a scatterplot,
# coloured by cluster
df$clusters <- as.factor(out4$cluster)
ggplot(df, aes(x=gene1, y=gene2))+geom_point(aes(col=clusters))+
  theme(panel.background = element_rect(fill = "white",colour = "black",
                                        linewidth = 1,linetype = "solid"),
        panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
        text = element_text(size=20,colour = "black",face = "bold"),
        legend.key=element_blank())
```



Q4: Perform hierarchical clustering on the same dataset using at least two different linkage methods (e.g., complete linkage, average linkage).

The different linkage methods used were average linkage and ward D linkage (Nielsen, 2016).

```
# set seed for reproducibility
set.seed(123)
# Computation of the pairwise euclidean
# distance matrix for the dataset
dist_mat <- dist(df, method = 'euclidean')
# Performing hierarchical clustering using
# average linkage
hclust_average<- hclust(dist_mat, method = 'average')
# Performing hierarchical clustering using
# ward D linkage
hclust_wardD<- hclust(dist_mat, method='ward.D')
```

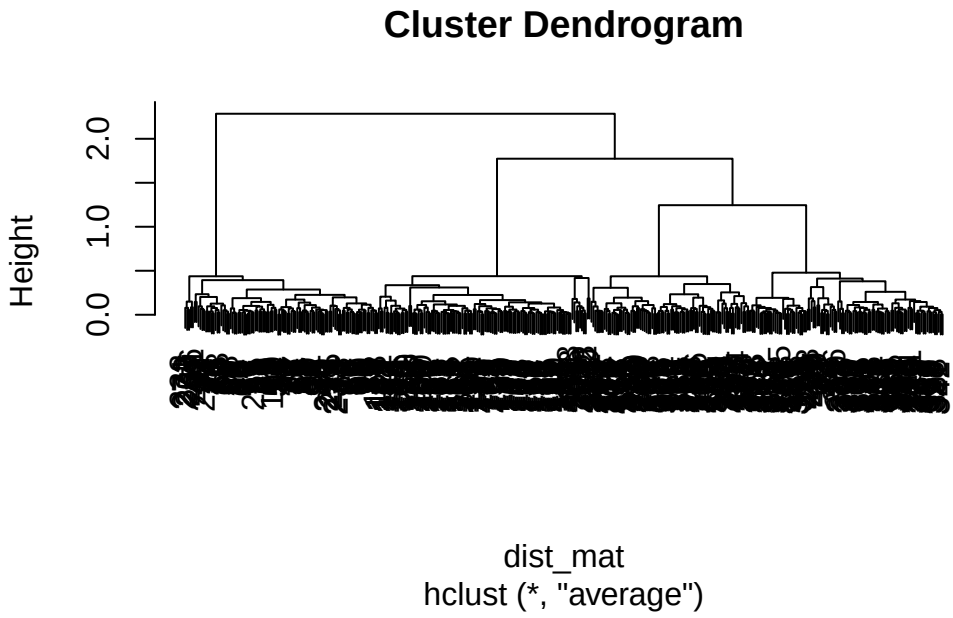
Q5: Generate dendrograms for each linkage method. Compare the resulting dendrograms. Which linkage method seems to produce more distinct and interpretable clusters?

Both linkage methods provide distinct clustering, however, the ward D linkage method appears more refined to 3 overall clusters, whereas the average linkage method seems more ambiguous.

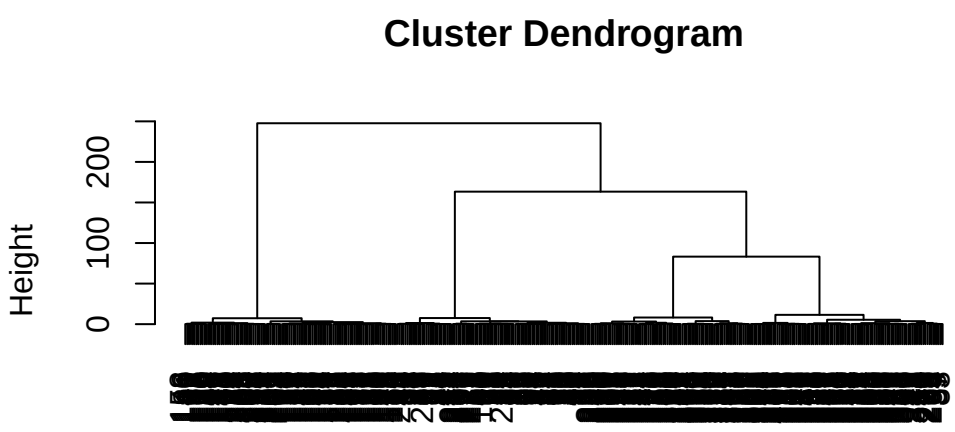
Average linkage calculates the distance between two clusters as the as the average of all the pairwise distances between elements in the first cluster when compared to the second cluster (Xu et al., 2021).

Ward's linkage method minimises the total variance within clusters to produce clusters of equal size (Murtagh and Legendre, 2011).

```
# set seed for reproducibility
set.seed(123)
# Plot the dendrogram for average linkage
plot(hclust_average)
```



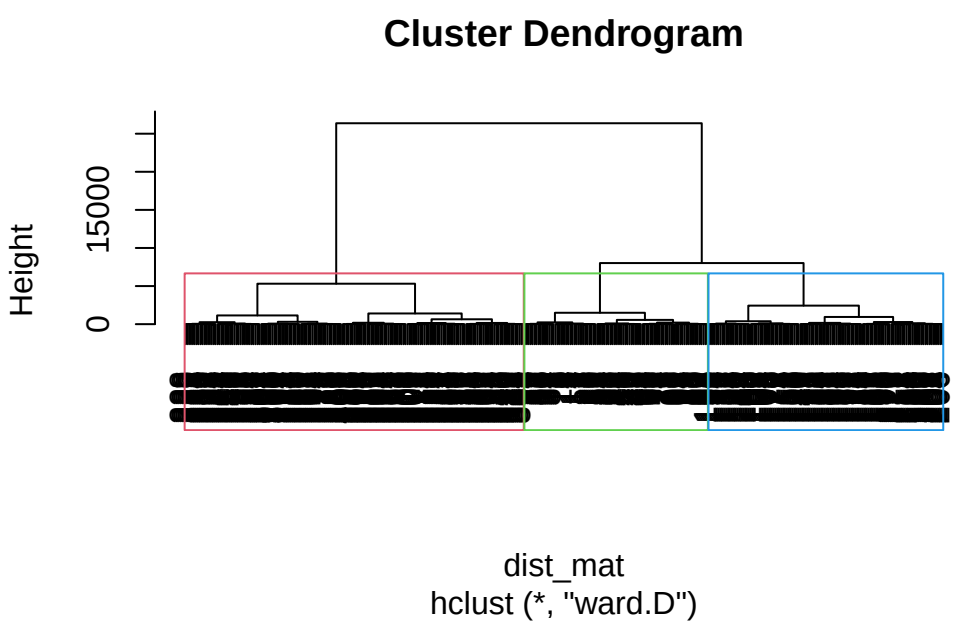

```
# set seed for reproducibility
set.seed(123)
# Plot the dendrogram for ward D linkage
plot(hclust_wardD)
```



Q6: Cut the dendrograms to produce clusters, and compare the resulting clusters to the k-means results. Provide the rationale you used to decide how to cut the tree.

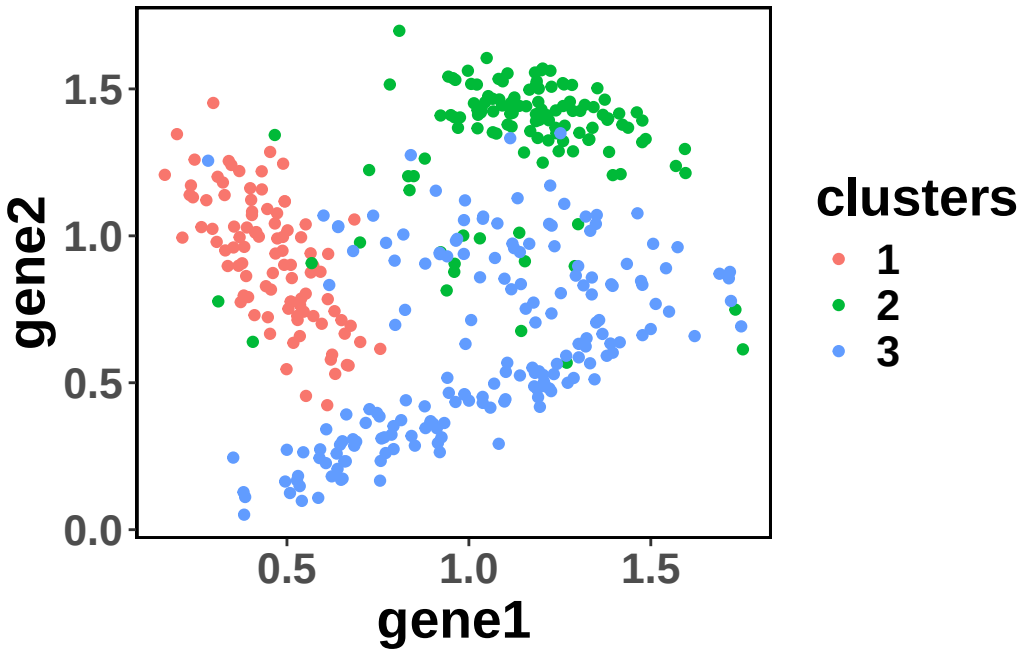
The k-means results align reasonably well with the clustered dendrograms (CDs),. However, whilst cluster 3 (blue) appears to overlap with clusters 1 and 2 in the k-means clustering, cluster 3 is the furthest cluster in the CDs. This suggests some variability between the clustering methods. Moreover, 3 clusters were chosen to cut the trees, as this was the crux in the elbow graph, suggesting optimal clustering. Additionally, the ward D linkage method was used to cut the trees due to more distinct clustering of branches.

```
set.seed(123)
# ward D linkage
# reloading dataset and recalculating
# clusters for reuse, just incase there
# was modification
df <- read.csv('Unlabelled_data_Student1.csv')
dist_mat <- dist(df, method = 'euclidean')
hclust<- hclust(dist_mat, method = 'ward.D')
plot(hclust)
rect.hclust(hclust , k = 3, border = 2:6)
```




```
clusters <- cutree(tree = hclust, k=3)
```

```
# Plot the dendrogram with the clustered
# labels for 3 clusters using ward D clustering
df$clusters <- as.factor(clusters)
ggplot(df, aes(x=gene1, y=gene2))+geom_point(aes(col=clusters))+
  theme(panel.background = element_rect(fill = "white",colour = "black",
                                        linewidth = 1,linetype = "solid"),
        panel.grid.major = element_blank(),panel.grid.minor = element_blank(),
        text = element_text(size=20,colour = "black",face = "bold"),
        legend.key=element_blank())
```



missing the rationale used to decide how to cut the trees

Supervised Learning/Classifiers

Load the labelled dataset provided with the assignment.

```
set.seed(123)
df_all <- read.csv('Labelled_data_Student24.csv', row.names = 1)
#head(df_all)
#colnames(df_all)
#tail(df_all)
dim(df_all)
```

```
[1]  100 2371
```

Logistic Regression:

Q7: Consider only class c1 and class c2. Split the dataset into training and testing sets using a 70/30 split (70% for training).

Logistic regression is a statistical method for data analysis where one or more independent variables determine a dichotomous outcome (only two possible outcomes) (Shevade and Keerthi, 2003).

```
# Import caret for createDataPartition
library(caret)
```

```
Warning: package 'caret' was built under R version 4.4.2
```

```
Loading required package: lattice
```

```
# Setting the seed ensures reproducibility
# for random operators
set.seed(123)

# Filtering to c1 and c2
# New generalised linear model (GLM)
# dataframe (df) is created which contains
# a subset of the df that contains class c1
# or c2
df_glm <- subset(df_all, class %in% c("c1", "c2"))
# Class is converted to a factor for usage
# in the model
df_glm$class <- factor(df_glm$class)

# 70/30 split
# The create Data Partition function splits
# the dataset into training and test data,
# whilst preserving the proportions of each class
# The target variable (class) is chosen and the
# percentage of the data used for training is set
# at 70% (p=0.7), and returns a simple vector used
# for indexing rows
train_idx <- createDataPartition(df_glm$class, p = 0.7, list = FALSE)
# The generated indices are used to split the datasets
# into 70 % training and 30% test data
train_df <- df_glm[train_idx, ]
test_df <- df_glm[-train_idx, ]

#check the split datasets
table(train_df$class)
```

```
c1 c2
31 40
```

```
table(test_df$class)
```

```
c1 c2
13 16
```

Q8: Train a logistic regression model on the training set, using only one variable as predictor. Justify your choice.

To balance the risk of over fitting with the need for informative predictors, one variable was chosen out of the top 20 associated with class separation, based on the lowest p-values. The p-values were derived from a two-sample t-test comparing each metabolite's intensity between c1 and c2. This identified variables with more information about which class the sample is. The top 20 was used to reduce the chance of over fitting with the most significant samples, and random selection was made reproducible by setting the seed as 123.

```
# Setting the seed ensures reproducibility
# for random operators (e.g. random number
# generation)
set.seed(123)

# Removing the class column to retain numeric
# variable only for use in t-tests
intensity_glm <- train_df[, colnames(train_df) != "class"]

# Empty numeric vector is made to store each
# metabolites p-value with the metabolite colnames
# as the vectors name
ttest_pvals <- numeric(length = ncol(intensity_glm))
names(ttest_pvals) <- colnames(intensity_glm)

# For loop over each metabolite using the seq_along
# function to loop over indices of a vector
for (i in seq_along(intensity_glm)) {
  # Variable name of the metabolite is stored for i
  var_name <- colnames(intensity_glm)[i]
  # Two-sample t-test used to ask:
  # Does the mean value of the metabolite vary
```

```
  # between classes?
  test_result <- t.test(intensity_glm[[i]] ~ train_df$class)
  # P-values are stored in the pval vector
  ttest_pvals[var_name] <- test_result$p.value
}

# False Discovery Rate (FDR) correction applied
# using Benjamini-Hochberg method to account for
# multiple hypothesis testing, helping to reduce
# false positives
ttest_fdr <- p.adjust(ttest_pvals, method = "BH")

# Filtering variables to include FDR-adjusted
# p-values < 0.05 as this is a standard threshold
# for significance to ensures only statistically
# meaningful features are retained
sig_vars <- names(ttest_fdr[ttest_fdr < 0.05])

# One variable is randomly selected significant variables
selected_variable <- sample(sig_vars, 1)

# Selected variable is printed to ensure everything
# has worked
print(selected_variable)
```

```
[1] "Metabolite1678"
```

```
# The selected variable is made into a new column
# for use in the models
train_df$selected_var <- train_df[[selected_variable]]

# GLM is made using the selected variable using
# biomial family for binary logistic regression
# (e.g. 0/1,true/false, c1/c2)
glm_model <- glm(class ~ selected_var, data = train_df, family = "binomial")

# The model is output as a summary
summary(glm_model)
```

```
Call:
```

```
glm(formula = class ~ selected_var, family = "binomial", data = train_df)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   3.261e-01  2.642e-01   1.235  0.21698
selected_var -1.300e-06  4.152e-07  -3.132  0.00174 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 97.283  on 70  degrees of freedom
Residual deviance: 84.360  on 69  degrees of freedom
AIC: 88.36

Number of Fisher Scoring iterations: 4
```

Q9: Predict the classes for the testing dataset. Estimate the performance of the model, reporting the following: Confusion matrix, Accuracy, Sensitivity and Specificity.

```r
# The variable is added to the test data as a new column
test_df$selected_var <- test_df[[selected_variable]]

# Predicting the probabilities from the model with
# c2 as the response
pred_probs_glm <- predict(glm_model, newdata = test_df, type = "response")
# Converting probabolities into class labels
# if p>0.5 then the class is c2, otherwise its c1
glm_pred <- ifelse(pred_probs_glm >= 0.5, "c2", "c1")
# Converting to factor and creating levels for classes
glm_pred <- factor(glm_pred, levels = levels(test_df$class))

# Generating a confusion matrix that compares
# predicted vs actual classes provides accuracy,
# sensitivity, and specificity
glm_cm <- confusionMatrix(glm_pred, test_df$class)
glm_cm
```

```
Confusion Matrix and Statistics

          Reference
Prediction c1 c2
```

```
        c1 10  7
        c2  3  9

               Accuracy : 0.6552
                 95% CI : (0.4567, 0.8206)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : 0.1756

                  Kappa : 0.3224

 Mcnemar's Test P-Value : 0.3428

            Sensitivity : 0.7692
            Specificity : 0.5625
         Pos Pred Value : 0.5882
         Neg Pred Value : 0.7500
             Prevalence : 0.4483
         Detection Rate : 0.3448
   Detection Prevalence : 0.5862
      Balanced Accuracy : 0.6659

       'Positive' Class : c1
```

Q10: Perform a 7-fold cross-validation storing the 7 different values of Accuracy, Sensitivity and Specificity calculated for each fold. Plot the results as boxplots.

```
# Setting seed for reproducibility of folds
set.seed(123)
# Load library for k fold splitting
library(groupdata2)
```

```
Warning: package 'groupdata2' was built under R version 4.4.3
```

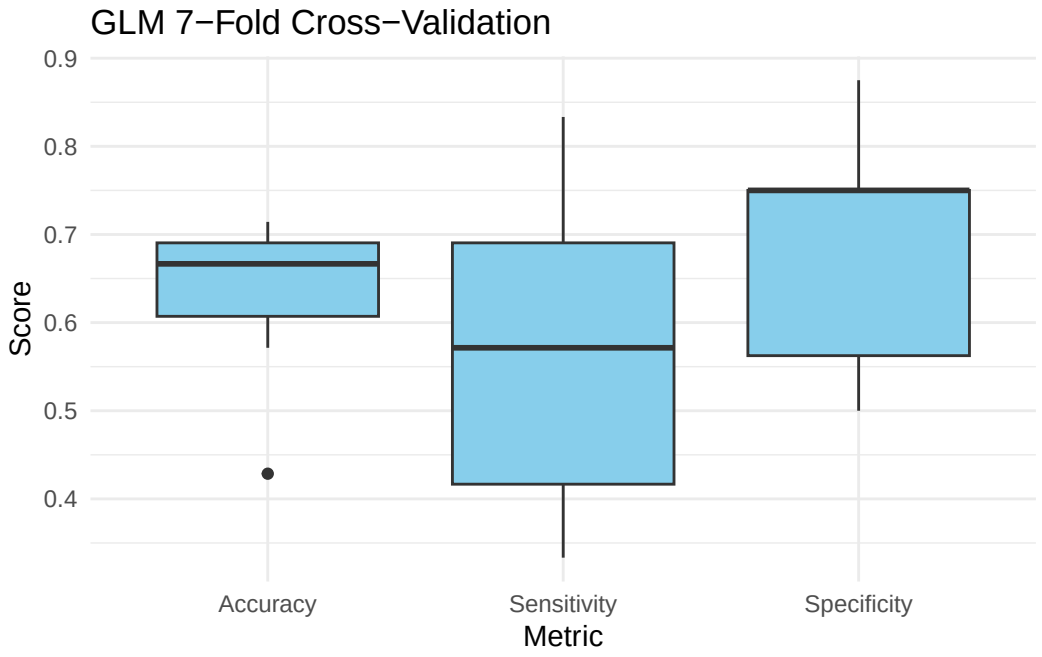
```
# Copy the data for cross validation (cv)
df_cv_glm <- df_glm

# 7 folds are applied to cv data
# handle_existing_fold_cols used for data
# tidying purposes, preventing reuse of folds
df_cv_glm <- fold(df_cv_glm, k = 7, cat_col = "class", handle_existing_fold_cols = "remove")
```

```r
# Folds are saved
folds_glm <- df_cv_glm$.folds
# Empty dataframe created for fold performance metrics
results_glm <- data.frame()

# For k in 7 folds
for (k in 1:7) {
  # All used for training but the k'th fold
  train_fold <- df_cv_glm[folds_glm != k, ]
  # k'th fold used for test set
  test_fold <- df_cv_glm[folds_glm == k, ]

  # The selected predictor column is added to
  # training and test sets
  train_fold$selected_var <- train_fold[[selected_variable]]
  test_fold$selected_var <- test_fold[[selected_variable]]
  # Model is fitted to training fold
  model <- glm(class ~ selected_var, data = train_fold, family = "binomial")
  # Probabilities predicted using test fold data
  probs <- predict(model, newdata = test_fold, type = "response")
  # Probabilities converted to binary classes and
  # set as factors
  preds <- ifelse(probs >= 0.5, "c2", "c1")
  preds <- factor(preds, levels = levels(df_cv_glm$class))
  # Confusion matrix created and results are stored
  cm <- confusionMatrix(preds, test_fold$class)
  results_glm <- rbind(results_glm, data.frame(
    Fold = k,
    Accuracy = cm$overall["Accuracy"],
    Sensitivity = cm$byClass["Sensitivity"],
    Specificity = cm$byClass["Specificity"]
  ))
}
print(results_glm)
```

```
          Fold  Accuracy Sensitivity Specificity
Accuracy     1 0.5714286   0.3333333       0.750
Accuracy1    2 0.6428571   0.8333333       0.500
Accuracy2    3 0.7142857   0.5000000       0.875
Accuracy3    4 0.6666667   0.5714286       0.750
Accuracy4    5 0.7142857   0.6666667       0.750
```

```
Accuracy5    6 0.4285714   0.3333333       0.500
Accuracy6    7 0.6666667   0.7142857       0.625
```

```r
set.seed(123)
# Importing necessary libraries for plotting
# and reformatting
library(ggplot2) # ggplot
library(reshape2) # melt
# Results are converted to long format for plotting
results_long_glm <- melt(results_glm, id.vars = "Fold", variable.name = "Metric", value.name
# Boxplot showing the distribution of each metric
# across all the folds
ggplot(results_long_glm, aes(x = Metric, y = Value)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "GLM 7-Fold Cross-Validation", x = "Metric", y = "Score") +
  theme_minimal()
```



Linear Discriminant Analysis (LDA):

Important note: LDA can be easily applied to multi-class problems. Feel free to try it!

Q11: Using the same train and validation sets (considering only c1 and c2) obtained in **Q7,** train an LDA model with the training data.

Linear discriminant analysis (LDA) is a statistical method utilising pattern recognition and machine learning to find linear combinations of features that seperates two or more classes/groups (Xanthopoulos, Pardalos and Trafalis, 2012).

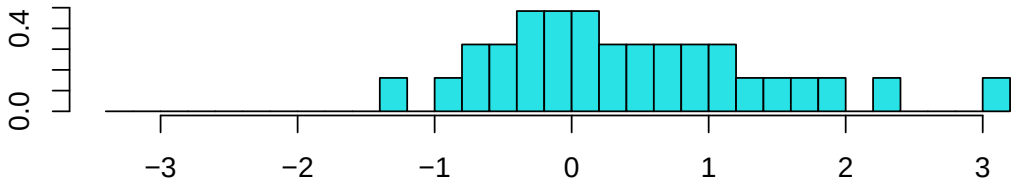
```
# Import caret for createDataPartition and MASS for LDA
library(caret)
library(MASS)
set.seed(123) # Setting the seed ensures reproducibility
# for random operators

# The selected variable is made into a new column
# for use in the models
train_df$selected_var <- train_df[,"Volume.cm.3."]

# LDA model is trained using the selected predictor variable
# Class is the response variable
# LDA finds the linear combination (LD1) of the
# best predictor
lda_model <- lda(class ~ selected_var, data = train_df)

# Model output displays group probabilities and means,
# in addition to LD1 coefficients
lda_model
```

you needed to use the whole dataset, not just the selected var

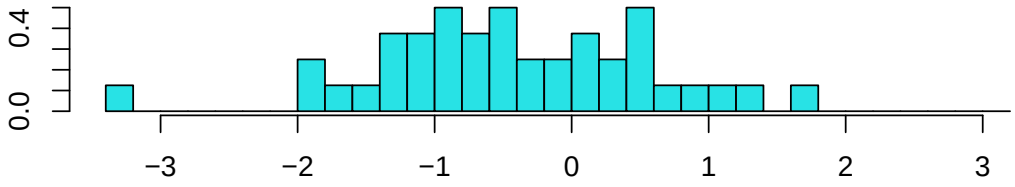
```
Call:
lda(class ~ selected_var, data = train_df)

Prior probabilities of groups:
       c1        c2
0.4366197 0.5633803

Group means:
   selected_var
c1     379442.4
c2    -253206.2

Coefficients of linear discriminants:
                     LD1
selected_var 1.39502e-06
```

Q12: Visualize the data in the reduced dimensional space produced by LDA.

```r
# Plotting the base lda structure- usually
# isn't as informative for single variable models
try(plot(lda_model))
```



group c1

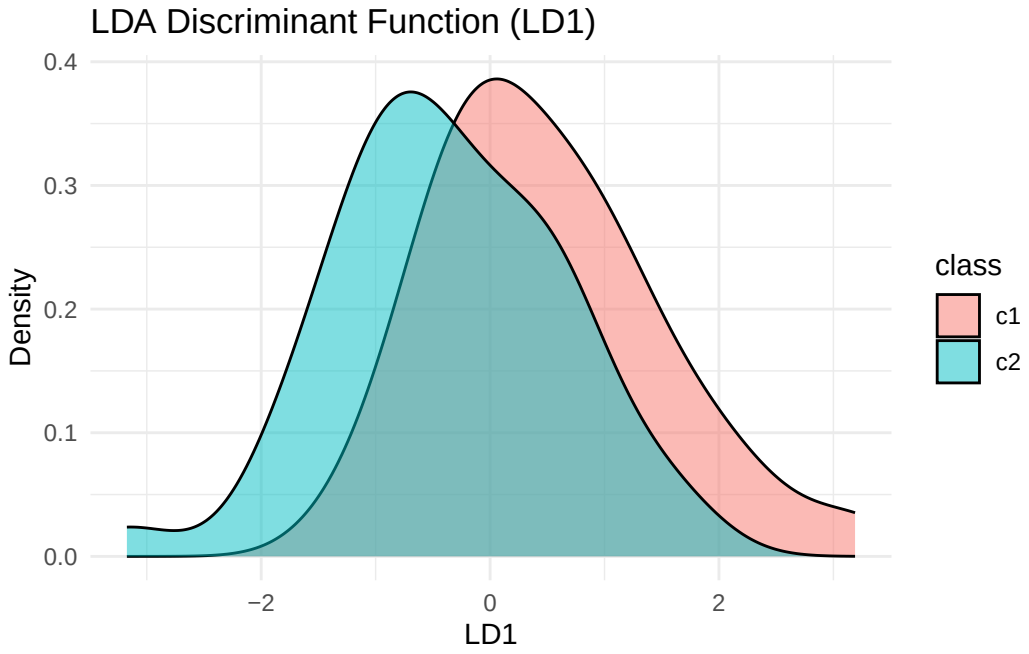


group c2

everything after this was not needed

```r
# Using the trained LDA model to predict values on
# training data
# returns predicted class, posterior probabilities
# for each class, and the LD1 score (x)
lda_pred <- predict(lda_model)
# Extracts LD1 values from the prediction to store
# them in the training data
# The LD1 scores represent the position on the
# linear discriminant axis (LD1), with a direction
# that best seperated the classes
train_df$LD1 <- lda_pred$x[, 1]

# LD1 values are visualised between the classes
```

```r
# This helps visualise how well the LD1 splits the classes
ggplot(train_df, aes(x = LD1, fill = class)) +
  geom_density(alpha = 0.5) +
  labs(title = "LDA Discriminant Function (LD1)", x = "LD1", y = "Density") +
  theme_minimal()
```



Q13: Predict the classes for the testing dataset. Estimate the performance of the model, reporting the following: Confusion matrix, Accuracy, Sensitivity and Specificity.

```r
set.seed(123)
# The variable is added to the test data as a new column
test_df$selected_var <- test_df[[selected_variable]]

# Predicting the probabilities from the model
lda_pred_test <- predict(lda_model, newdata = test_df, type="class")

# Generating a confusion matrix that compares
# predicted vs actual classes provides accuracy,
# sensitivity, and specificity
lda_cm <- confusionMatrix(lda_pred_test$class, test_df$class)
lda_cm
```

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction c1 c2
        c1 10  7
        c2  3  9

               Accuracy : 0.6552
                 95% CI : (0.4567, 0.8206)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : 0.1756

                  Kappa : 0.3224

 Mcnemar's Test P-Value : 0.3428

            Sensitivity : 0.7692
            Specificity : 0.5625
         Pos Pred Value : 0.5882
         Neg Pred Value : 0.7500
             Prevalence : 0.4483
         Detection Rate : 0.3448
   Detection Prevalence : 0.5862
      Balanced Accuracy : 0.6659

       'Positive' Class : c1
```

Q14: Perform a 7-fold cross-validation storing the 7 different values of Accuracy, Sensitivity and Specificity calculated for each fold. Plot the results as boxplots.

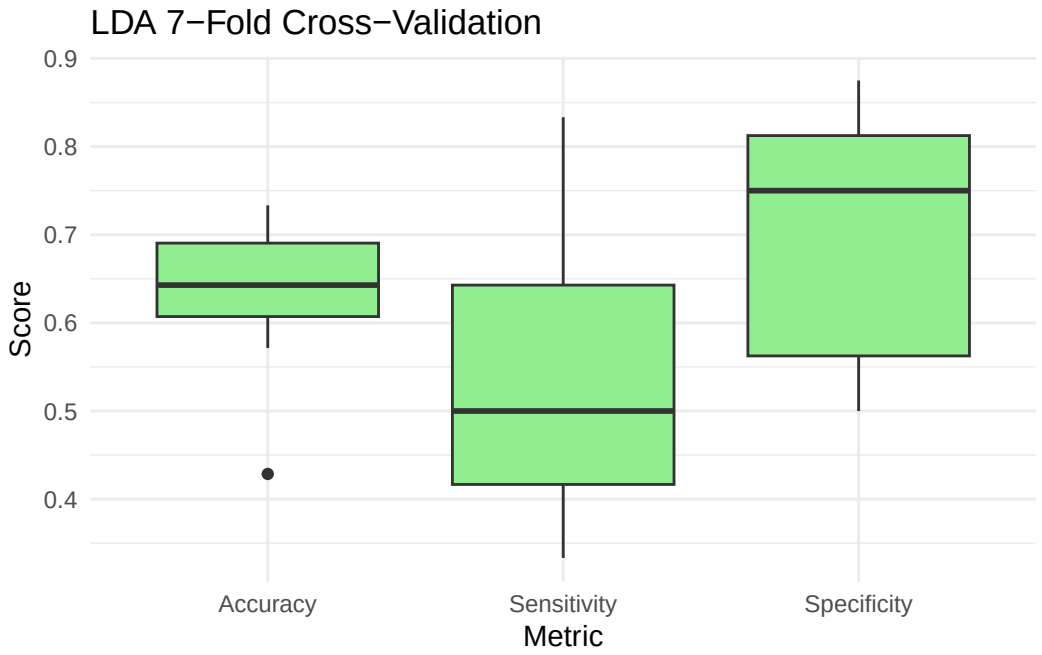
```
# Setting seed for reproducibility of folds
set.seed(123)
# Load library for k fold splitting
library(groupdata2)

# Copy the data for cross validation (cv)
df_cv_lda <- df_glm
# 7 folds are applied to cv data
# handle_existing_fold_cols used for data
# tidying purposes, preventing reuse of folds
df_cv_lda <- fold(data = df_cv_lda, k = 7, cat_col = "class", handle_existing_fold_cols = "re
# Folds are saved
```

```r
folds_lda <- df_cv_lda$.folds
# Empty dataframe created for fold performance metrics
results_lda <- data.frame()


# For k in 7 folds
for (k in 1:7) {
  # All used for training but the k'th fold
  lda_train_fold <- df_cv_lda[folds_lda != k, ]
  # k'th fold used for test set
  lda_test_fold <- df_cv_lda[folds_lda == k, ]

  # The selected predictor column is added to training
  # and test sets
  lda_train_fold$selected_var <- lda_train_fold[[selected_variable]]
  lda_test_fold$selected_var <- lda_test_fold[[selected_variable]]

  # Model is fitted to training fold
  model <- lda(class ~ selected_var, data = lda_train_fold)
                  again, using only one variable is wrong
  # Probabilities predicted using test fold data
  preds <- predict(model, newdata = lda_test_fold)$class
  # Probabilities converted to binary classes and
  # set as factors
  preds <- factor(preds, levels = levels(df_cv_lda$class))

  # Confusion matrix (cm) is created and results are stored
  cm <- confusionMatrix(preds, lda_test_fold$class)
  results_lda <- rbind(results_lda, data.frame(
    Fold = k,
    Accuracy = cm$overall["Accuracy"],
    Sensitivity = cm$byClass["Sensitivity"],
    Specificity = cm$byClass["Specificity"]
  ))
}
print(results_lda)
```

```
          Fold  Accuracy Sensitivity Specificity
Accuracy     1 0.5714286   0.3333333       0.750
Accuracy1    2 0.6428571   0.8333333       0.500
Accuracy2    3 0.7142857   0.5000000       0.875
Accuracy3    4 0.7333333   0.5714286       0.875
```

```
Accuracy4    5 0.6428571   0.5000000       0.750
Accuracy5    6 0.4285714   0.3333333       0.500
Accuracy6    7 0.6666667   0.7142857       0.625
```

```r
set.seed(123)
# Importing necessary libraries for plotting
# and reformatting
library(ggplot2) # ggplot
library(reshape2) # melt
# Results are converted to long format for plotting
results_long_lda <- melt(results_lda, id.vars = "Fold", variable.name = "Metric", value.name

# Boxplot showing the distribution of each metric across
# all the folds
ggplot(results_long_lda, aes(x = Metric, y = Value)) +
  geom_boxplot(fill = "lightgreen") +
  labs(title = "LDA 7-Fold Cross-Validation", x = "Metric", y = "Score") +
  theme_minimal()
```



Support Vector Machines (SVM):

Q15: Using the same train and validation sets (considering only c1 and c2) obtained in **Q7,** train a SVM model with the training data.

Support vector machines (SVMs) are supervised learning models used for classification and regression analysis. They are highly effective in high-dimensional spaces, with more dimensions than variables (Huang et al., 2018).

```
#install.packages('e1071')

# Import necessary libraries
library(e1071)
```

```
Warning: package 'e1071' was built under R version 4.4.3
```

```
library(caret)
set.seed(123)

# The variable is added to the training data as a new column
train_df$selected_var <- train_df[[selected_variable]]

# SVM model is made using the selected variable
svm_model <- svm(class~selected_var, data = train_df)
summary(svm_model)
```

also for SVM, using only one variable does

```
Call:
svm(formula = class ~ selected_var, data = train_df)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  59

 ( 29 30 )


Number of Classes:  2

Levels:
 c1 c2
```

Q16: Predict the classes for the testing dataset. Estimate the performance of the model, reporting the following: Confusion matrix, Accuracy, Sensitivity and Specificity.

```
set.seed(123)
# The variable is added to the test data as a new column
test_df$selected_var <- test_df[[selected_variable]]
# Predicting the probabilities from the model as type class
svm_pred <- predict(svm_model, newdata = test_df, type="class")
# Converting to factor and creating levels for classes
svm_cm <- confusionMatrix(as.factor(svm_pred), as.factor(test_df$class))
svm_cm
```

```
Confusion Matrix and Statistics

          Reference
Prediction c1 c2
        c1 11  7
        c2  2  9

               Accuracy : 0.6897
                 95% CI : (0.4917, 0.8472)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : 0.09437

                  Kappa : 0.3944

 Mcnemar's Test P-Value : 0.18242

            Sensitivity : 0.8462
            Specificity : 0.5625
         Pos Pred Value : 0.6111
         Neg Pred Value : 0.8182
             Prevalence : 0.4483
         Detection Rate : 0.3793
   Detection Prevalence : 0.6207
      Balanced Accuracy : 0.7043

       'Positive' Class : c1
```

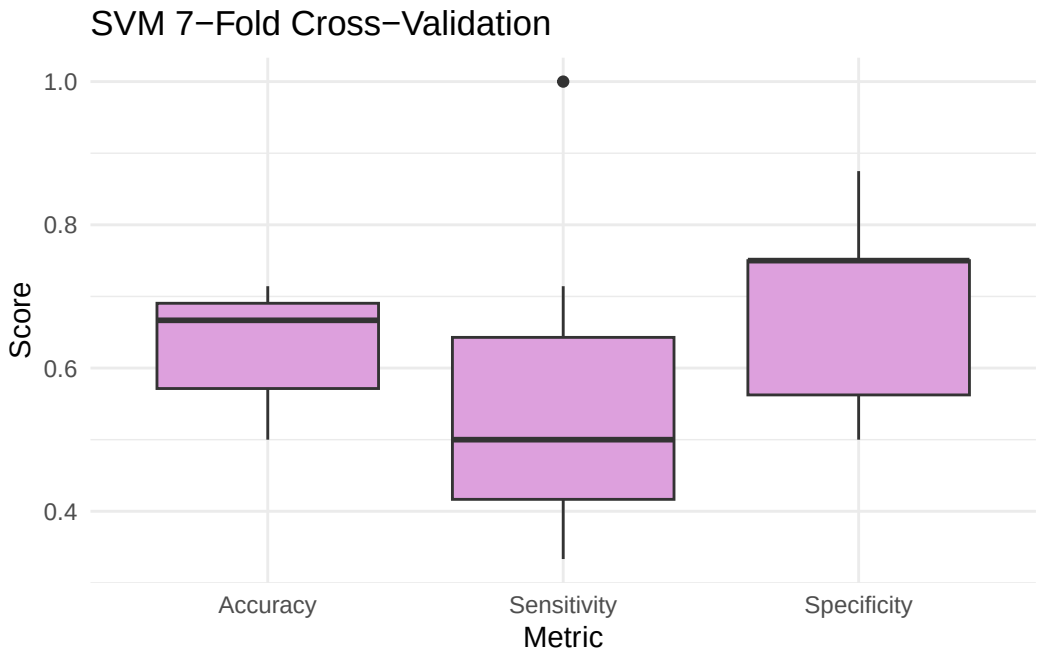
Q17: Perform a 7-fold cross validation storing the 7 different values of Accuracy, Sensitivity and Specificity calculated for each fold. Plot the results as boxplots.

```r
# Load library
library(groupdata2)
library(caret)
library(e1071)
library(ggplot2)
library(reshape2)
set.seed(123) # Setting seed for reproducibility of folds

# Copy the data for cross validation (cv)
df_cv_svm <- df_glm
# 7 folds are applied to cv data
# handle_existing_fold_cols used for data tidying
# purposes, preventing reuse of folds
df_cv_svm <- fold(data = df_cv_svm, k = 7, cat_col = "class", handle_existing_fold_cols = "re

# Folds are saved
folds_svm <- df_cv_svm$.folds
# Empty dataframe created for fold performance metrics
results_svm <- data.frame()

# For k in 7 folds
for (k in 1:7) {
  # All used for training but the k'th fold
  svm_train_fold <- df_cv_svm[folds_svm != k, ]
  # k'th fold used for test set
  svm_test_fold <- df_cv_svm[folds_svm == k, ]

  # The selected predictor column is added to training
  # and test sets                                          same here
  svm_train_fold$selected_var<-svm_train_fold[[selected_variable]]
  svm_test_fold$selected_var<-svm_test_fold[[selected_variable]]
  # Model is fitted to training fold
  model <-svm(class ~ selected_var, data = svm_train_fold)
  # Probabilities predicted using test fold data
  # and converted into a factor with class levels
  preds <-predict(model, newdata = svm_test_fold)
  preds <-factor(preds, levels = levels(df_cv_svm$class))

  # Confusion matrix created and results are stored
  # for plotting
  cm <- confusionMatrix(preds, svm_test_fold$class)
  results_svm <- rbind(results_svm, data.frame(
```

```
    Fold = k,
    Accuracy = cm$overall["Accuracy"],
    Sensitivity = cm$byClass["Sensitivity"],
    Specificity = cm$byClass["Specificity"]
  ))
}
print(results_svm)
```

```
          Fold  Accuracy Sensitivity Specificity
Accuracy     1 0.5714286   0.3333333       0.750
Accuracy1    2 0.7142857   1.0000000       0.500
Accuracy2    3 0.7142857   0.5000000       0.875
Accuracy3    4 0.6666667   0.5714286       0.750
Accuracy4    5 0.5714286   0.3333333       0.750
Accuracy5    6 0.5000000   0.5000000       0.500
Accuracy6    7 0.6666667   0.7142857       0.625
```

```
set.seed(123) # Consistant seed
# Importing necessary libraries for plotting
# and reformatting
library(ggplot2) # ggplot
library(reshape2) # melt
# Results are converted to long format for plotting
results_long_svm <- melt(results_svm, id.vars = "Fold", variable.name = "Metric", value.name

# Boxplot showing the distribution of each metric across
# all the folds
ggplot(results_long_svm, aes(x = Metric, y = Value)) +
  geom_boxplot(fill = "plum") +
  labs(title = "SVM 7-Fold Cross-Validation", x = "Metric", y = "Score") +
  theme_minimal()
```



Q18: Using the values and boxplots generated in **Logistic Regression - Q10, Q14** and **Q17**, compare and discuss the performances of the models.

Each model was trained on the same training data, filtered for classes c1 and c2, and assessed with 7-fold cross-validation.

Both the generalised linear model (GLM) and the linear discriminant analysis (LNA) model performed remarkably similarly across the 7-fold cross-validation. This is likely due to both models being linear models, the dataset being small (training set) and simple (one variable predictor).

The GLM had moderate accuracy, with an interquartile range (IQR) between around 0.61 and 0.69 across folds, averaging at 0.67. The model's specificity was high, with an average of around 0.75 with an IQR between 0.56 and 0.75. The average sensitivity was 0.57, with an IQR between 0.42 and 0.69.

The LDA model had moderate accuracy, with an interquartile range (IQR) between around 0.61 and 0.69 across folds, averaging at 0.64. The model's specificity was high, with an average of around 0.75 with an IQR between 0.56 and 0.81. The average sensitivity was 0.5, with an IQR between 0.41 and 0.64.

Overall, both of these models were reasonably accurate (0.66), with a bias toward predicting c1, with 10/13 correct c1 predictions and 9/16 correct c2 predictions. However, the main distinction between the models was observed in folds 4, where the GLM had an accuracy of 0.67, sensitivity of 0.57, and specificity 0.75, compared to LDA model's accuracy of 0.73, sensitivity of 0.57, and specificity of 0.88. Additionally, there was differences in folds 5, where

the GLM had an accuracy of 0.71, sensitivity of 0.67, and specificity 0.75, compared to LDA model's accuracy of 0.64, sensitivity of 0.50, and specificity of 0.75.

The support vector machine (SVM) model had an average accuracy of 0.67, and an IQR of around 0.58-0.69. This models specificity averaged at 0.74, with an IQR of 0.57-0.74. The sensitivity averaged at 0.5, with an IQR of 0.41-0.64. Overall, this model had good accuracy (0.69) but continued to be biased towards c1 classification, with 11/13 correct c1 predictions and 9/16 correct c2 predictions.

In conclusion, these models demonstrated similar performances across the 7-fold validation, with accruacies averaging around 0.64 to 0.67. This similarity likely reflects the simplicity of the single predictor variable and a relatively small dataset. Additionally, Pohar et al. (2004) found logistic regression to yield similar results to LDA when assumptions were not severely violated, suggesting this is a trend across models. Across all models, specificity was consistently higher than sensitivity, indicating there is a systematic bias towards predicting observations as c1.

Although none of the models clearly outperformed the others, the GLM offered the best trade-off between accuracy, sensitivity, and specificity, with a balanced and consistent performance. GLMs are typically well-suited to small datasets with few predictors, making them appropriate for using a single metabolite as a predictor. In contrast, LDA and SVM models typically perform better when leveraging the full dataset or complete training data. Therefore, given the simplicity of this task, the GLM was potentially better suited to this scenario. Moreover, logistic regression is assumed to be more flexible and robust method in cases of assumption violation (Pohar, Blas and Turk, 2004).

decent description.

Reference list

Huang, S., Cai, N., Pacheco, P.P., Narrandes, S., Wang, Y. and Xu, W. (2018). Applications of Support Vector Machine (SVM) Learning in Cancer Genomics. *Cancer Genomics & Proteomics*, [online] 15(1), pp.41–51. Available at: https://cgp.iiarjournals.org/content/15/1/41.short.

Murtagh, F. and Legendre, P. (2011). *Ward's Hierarchical Clustering Method: Clustering Criterion and Agglomerative Algorithm*. [online] Research Gate. Available at: https://www.researchgate.net/publication/51962445_Ward%27s_Hierarchical_Clustering_Method_Clusterin

Nielsen, F. (2016). Hierarchical Clustering. *Introduction to HPC with MPI for Data Science*, pp.195–211. doi:https://doi.org/10.1007/978-3-319-21903-5_8.

Pohar, M., Blas, M. and Turk, S. (2004). Comparison of logistic regression and linear discriminant analysis. *Advances in Methodology and Statistics*, 1(1), pp.143–161. doi:https://doi.org/10.51936/ayrt6204.

Shahulsyed (2024). *Understanding the Clustering Algorithms Explained: K-Means, K-Means++, Hierarchical Clustering, and the Elbow Method in Unsupervised Learning.*

[online] Medium. Available at: https://medium.com/%40shahulsyed777/understanding-the-clustering-algorithms-explained-k-means-k-means-hierarchical-clustering-and-3dfd17e89949 [Accessed 1 Apr. 2025].

Shevade, S.K. and Keerthi, S.S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17), pp.2246–2253. doi:https://doi.org/10.1093/bioinformatics/

Xanthopoulos, P., Pardalos, P.M. and Trafalis, T.B. (2012). Linear Discriminant Analysis. *SpringerBriefs in Optimization*, pp.27–33. doi:https://doi.org/10.1007/978-1-4419-9878-1_4.

Xu, N., Finkelman, R.B., Dai, S., Xu, C. and Peng, M. (2021). Average Linkage Hierarchical Clustering Algorithm for Determining the Relationships between Elements in Coal. *ACS Omega*, 6(9), pp.6206–6217. doi:https://doi.org/10.1021/acsomega.0c05758.