Foundations of Financial Data Science FA-582: Assignment 4

Federica Malamisura

2024-11-27

ASSIGNMENT 4

Problem 1

```
## 'data.frame':    1070 obs. of  18 variables:
##  $ Purchase      : chr  "CH" "CH" "CH" "MM" ...
##  $ WeekofPurchase: int  237 239 245 227 228 230 232 234 235 238 ...
##  $ StoreID       : int  1 1 1 1 7 7 7 7 7 7 ...
##  $ PriceCH       : num  1.75 1.75 1.86 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceMM       : num  1.99 1.99 2.09 1.69 1.69 1.99 1.99 1.99 1.99 1.99 ...
##  $ DiscCH        : num  0 0 0.17 0 0 0 0 0 0 0 ...
##  $ DiscMM        : num  0 0.3 0 0 0 0 0.4 0.4 0.4 0.4 ...
##  $ SpecialCH     : int  0 0 0 0 0 0 1 1 0 0 ...
##  $ SpecialMM     : int  0 1 0 0 0 1 1 0 0 0 ...
##  $ LoyalCH       : num  0.5 0.6 0.68 0.4 0.957 ...
##  $ SalePriceMM   : num  1.99 1.69 2.09 1.69 1.69 1.99 1.59 1.59 1.59 1.59 ...
##  $ SalePriceCH   : num  1.75 1.75 1.69 1.69 1.69 1.69 1.69 1.75 1.75 1.75 ...
##  $ PriceDiff     : num  0.24 -0.06 0.4 0 0 0.3 -0.1 -0.16 -0.16 -0.16 ...
##  $ Store7        : chr  "No" "No" "No" "No" ...
##  $ PctDiscMM     : num  0 0.151 0 0 0 ...
##  $ PctDiscCH     : num  0 0 0.0914 0 0 ...
##  $ ListPriceDiff : num  0.24 0.24 0.23 0 0 0.3 0.3 0.24 0.24 0.24 ...
##  $ STORE         : int  1 1 1 1 0 0 0 0 0 0 ...
```

a. Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

I split the dataset into a training set (800 observations) and a test set (remaining observations) using a random sampling method with a fixed seed for reproducibility. This ensures that the results are consistent across runs.

b. Fit a tree to the training data, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

The classification tree was fitted using the tree package, with Purchase as the response variable and all other variables as predictors. The summary output provides insights into the tree's structure, including: Number of terminal nodes (leaves): This gives an idea of how complex the tree is. Training error rate: This measures how well the tree performs on the training data, providing a baseline for evaluation.

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train_data)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7625 = 603.9 / 792
## Misclassification error rate: 0.165 = 132 / 800
```

```
## Training Error Rate: 0.165
```

```
## Number of Terminal Nodes: 8
```

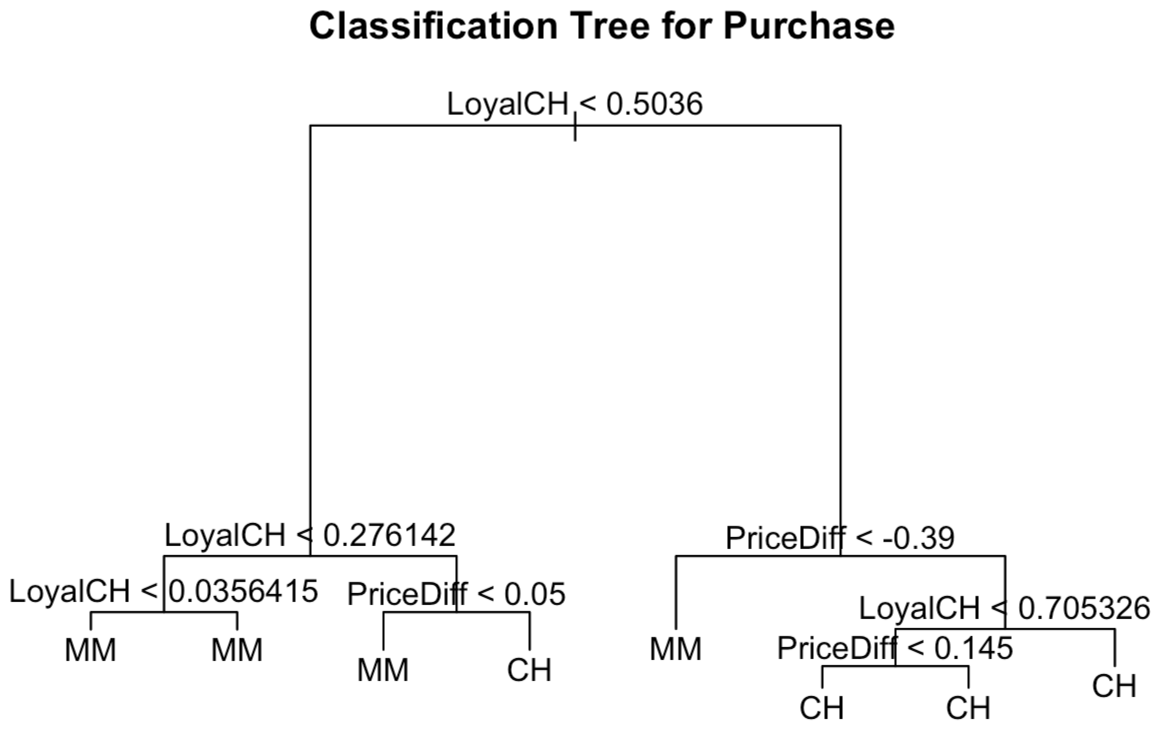
c. Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

Interpreting a terminal node involves understanding the conditions leading to that node and the majority class (or response) it predicts. This output is crucial for understanding the decision-making process within the tree.

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1071.00 CH ( 0.60875 0.39125 )
##    2) LoyalCH < 0.5036 350  415.10 MM ( 0.28000 0.72000 )
##      4) LoyalCH < 0.276142 170  131.00 MM ( 0.12941 0.87059 )
##        8) LoyalCH < 0.0356415 56   10.03 MM ( 0.01786 0.98214 ) *
##        9) LoyalCH > 0.0356415 114  108.90 MM ( 0.18421 0.81579 ) *
##      5) LoyalCH > 0.276142 180  245.20 MM ( 0.42222 0.57778 )
##       10) PriceDiff < 0.05 74   74.61 MM ( 0.20270 0.79730 ) *
##       11) PriceDiff > 0.05 106  144.50 CH ( 0.57547 0.42453 ) *
##    3) LoyalCH > 0.5036 450  357.10 CH ( 0.86444 0.13556 )
##      6) PriceDiff < -0.39 27   32.82 MM ( 0.29630 0.70370 ) *
##      7) PriceDiff > -0.39 423  273.70 CH ( 0.90071 0.09929 )
##       14) LoyalCH < 0.705326 130  135.50 CH ( 0.78462 0.21538 )
##         28) PriceDiff < 0.145 43   58.47 CH ( 0.58140 0.41860 ) *
##         29) PriceDiff > 0.145 87   62.07 CH ( 0.88506 0.11494 ) *
##       15) LoyalCH > 0.705326 293  112.50 CH ( 0.95222 0.04778 ) *
```

d. Create a plot of the tree, and interpret the results.

The plot of the tree helps visualize the splits, showcasing how variables partition the dataset. It provides an intuitive way to understand the tree’s logic.



e. Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

I predicted outcomes on the test set and compared them to actual outcomes using a confusion matrix. This step calculates the test error rate, indicating how well the model generalizes to unseen data.

```
## [1] "Confusion Matrix:"
```

```
##       Predicted
## Actual  CH  MM
##     CH 150  16
##     MM  34  70
```

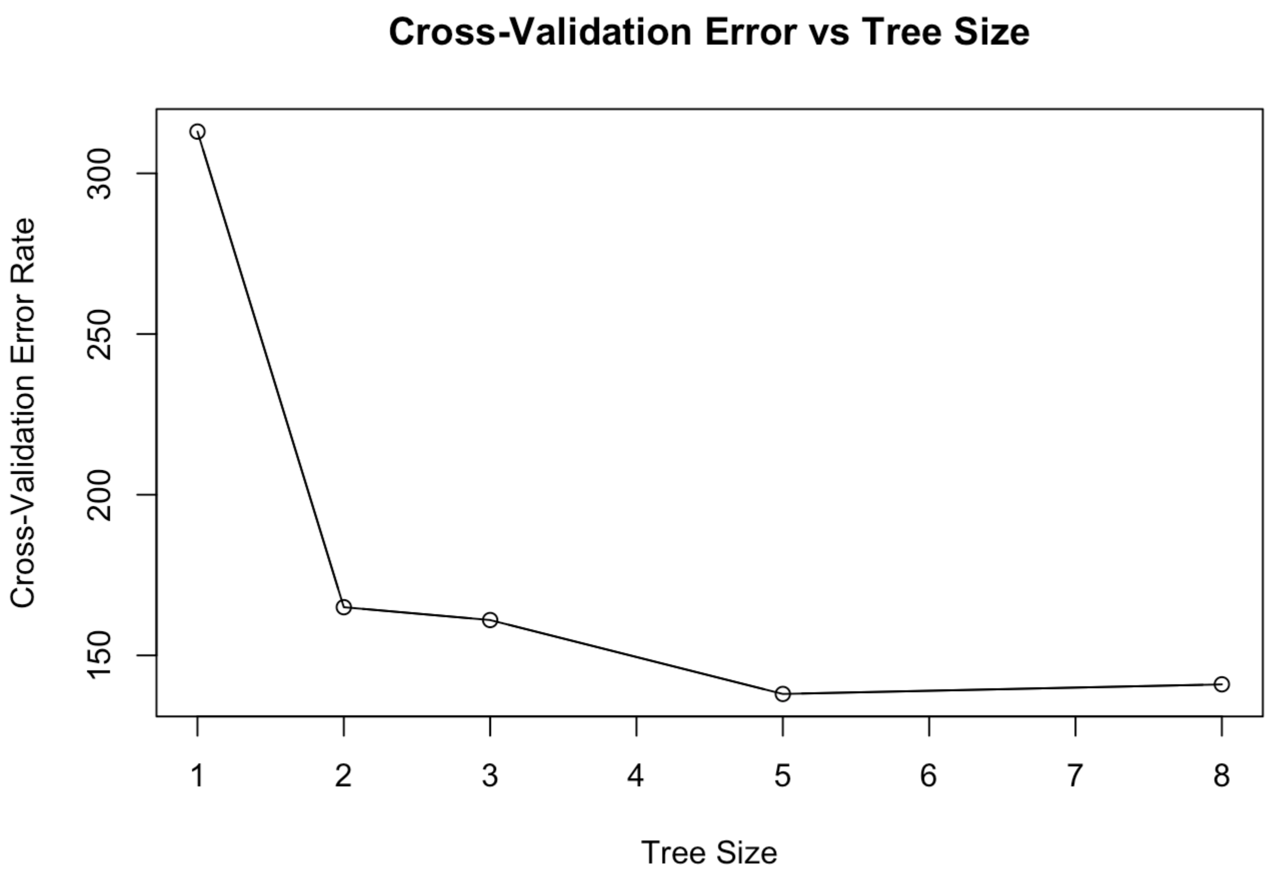
```
##
## Test Error Rate: 0.1851852
```

f. Apply the cv.tree() function to the training set in order to determine the optimal tree size. Perform cross-validation

```
## $size
## [1] 8 5 3 2 1
##
## $dev
## [1] 141 138 161 165 313
##
## $k
## [1] -Inf    0    8   11  154
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

g. Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis. Create plot

Using cv.tree, I determined the optimal tree size by examining cross-validation error rates. The plot of tree size vs. error rate helps identify whether pruning can improve the model's performance.



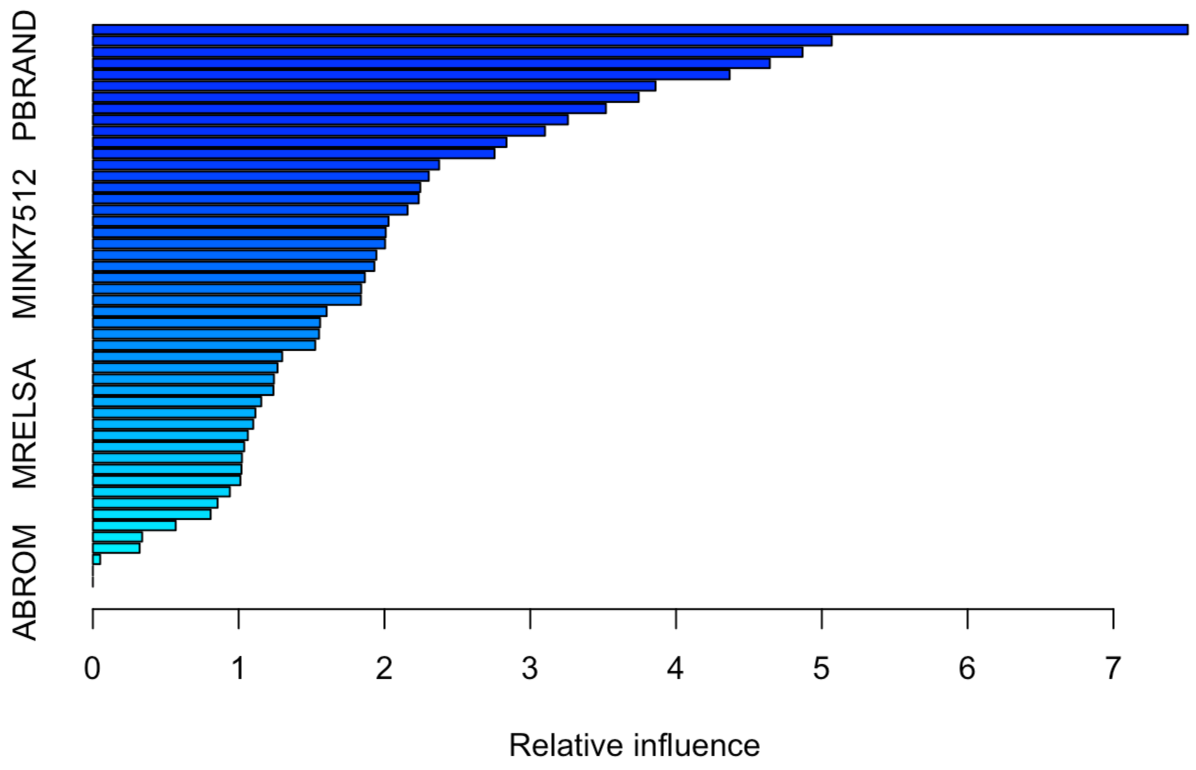
Problem 2

a. Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

I separated the dataset into training (first 1,000 observations) and test sets, addressing: Zero-variance predictors: Removing them ensures that irrelevant features do not affect the model. Scaling: Standardizing predictors facilitates comparison across models like boosting, KNN, and logistic regression.

b. Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

I fitted a boosting model using 1,000 trees and a shrinkage value of 0.01. Boosting leverages weak learners iteratively to improve performance. The summary function provided variable importance measures, highlighting which predictors are most influential in determining purchases.



```
##               var    rel.inf
## PPERSAUT PPERSAUT 7.51042075
## MGODGE     MGODGE 5.06804320
## MKOOPKLA MKOOPKLA 4.86775231
## MOPLHOOG MOPLHOOG 4.64352366
## PBRAND     PBRAND 4.36767371
## MOSTYPE   MOSTYPE 3.85945155
## MBERMIDD MBERMIDD 3.74397480
## MGODPR     MGODPR 3.51912267
## MBERARBG MBERARBG 3.25858707
## MINK3045 MINK3045 3.10123152
## MAUT2       MAUT2 2.83703658
## MSKC         MSKC 2.75511835
## MSKA         MSKA 2.37500422
## MRELGE     MRELGE 2.30394686
## MFGEKIND MFGEKIND 2.24580240
## MFWEKIND MFWEKIND 2.23498965
## MSKB1       MSKB1 2.15872914
## MBERARBO MBERARBO 2.02808658
## PWAPART   PWAPART 2.00896017
## MINK7512 MINK7512 2.00441921
## MBERHOOG MBERHOOG 1.94440336
## MINKM30   MINKM30 1.93064021
## MRELOV     MRELOV 1.86565454
## MAUT1       MAUT1 1.84031060
## MOPLMIDD MOPLMIDD 1.83856434
## MSKB2       MSKB2 1.60340619
## MINKGEM   MINKGEM 1.55866524
## MGODOV     MGODOV 1.55136541
## MHKOOP     MHKOOP 1.52521547
## MGODRK     MGODRK 1.29852335
## ABRAND     ABRAND 1.26674035
## MINK4575 MINK4575 1.24206294
## MFALLEEN MFALLEEN 1.23835434
## MAUT0       MAUT0 1.15494321
## MSKD         MSKD 1.11464417
## MRELSA     MRELSA 1.09990518
## MGEMLEEF MGEMLEEF 1.06307894
## MZFONDS   MZFONDS 1.03809346
## MZPART     MZPART 1.02253142
## MGEMOMV   MGEMOMV 1.01912402
## MHHUUR     MHHUUR 1.01165817
## MOPLLAAG MOPLLAAG 0.93980719
## MBERZELF MBERZELF 0.85562897
## APERSAUT APERSAUT 0.80771825
## MOSHOOFD MOSHOOFD 0.56827492
## MBERBOER MBERBOER 0.33807239
## MINK123M MINK123M 0.32075012
## MAANTHUI MAANTHUI 0.04998887
## AWAPART   AWAPART 0.00000000
## ABROM       ABROM 0.00000000
```

c. Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 %. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying random forest model, KNN or logistic regression to this data set?

Using boosting, random forests, KNN, and logistic regression, I compared predictions on the test set. Precision (fraction of correctly predicted positive purchases) was a key metric. This analysis highlights the strengths and weaknesses of different methods:

```
Boosting: Often excels when capturing non-linear relationships.
Random Forests: Robust to overfitting and interpretable via variable importance.
KNN: Effective in capturing local patterns but sensitive to scaling and distance metr
ics.
Logistic Regression: A simple baseline model suitable for linearly separable data.
```

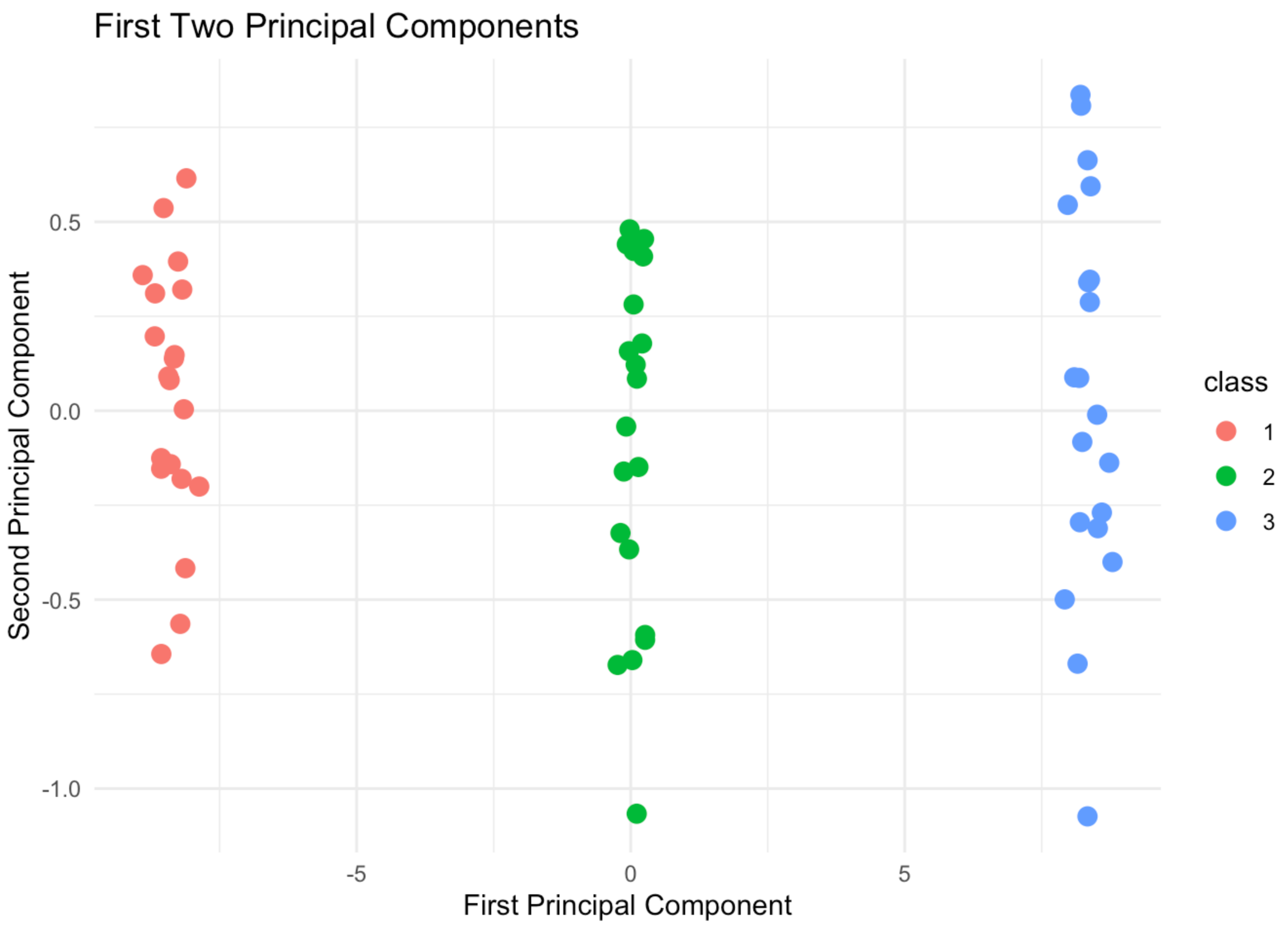
Problem 3

a. Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables. Hint: There are a number of functions in R that you can use to generate data. One example is the rnorm() function; runif() is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

I generated three classes of data with distinct means, ensuring sufficient separation for classification and clustering tasks.

b. Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.

PCA was performed to reduce dimensionality, plotting the first two principal components to visually assess class separability. This step is critical for understanding whether the simulated data reflects distinct clusters.



c. Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels? Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.

```
##
## true_labels  1  2  3
##           1  0  0 20
##           2  0 20  0
##           3 20  0  0
```

d) Perform K-means clustering with K = 2. Describe your results.

```
##
## true_labels  1  2
##           1  0 20
##           2 20  0
##           3 20  0
```

e) Now perform K-means clustering with K = 4, and describe your results.ù

```
##
## true_labels  1  2  3  4
##           1  0  0 20  0
##           2  8 12  0  0
##           3  0  0  0 20
```

Using K-means with K=3,2,4, I evaluated how well the clusters align with the true labels. The comparisons highlighted the importance of choosing an appropriate KK based on domain knowledge and visualizations.

f. Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 × 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

Clustering on PCA-reduced data emphasizes how dimensionality reduction affects cluster quality. Often, using fewer dimensions can improve clustering efficiency without significant information loss.

```
##
## true_labels  1  2  3
##           1  0  0 20
##           2 20  0  0
##           3  0 20  0
```

g. Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain.

Standardizing variables before clustering ensures that all features contribute equally. The results were compared to previous clustering attempts, showcasing the impact of scaling.

```
##
## true_labels  1  2  3
##           1 20  0  0
##           2  0  0 20
##           3  0 20  0
## R version 4.4.1 (2024-06-14)
## Platform: x86_64-apple-darwin20
## Running under: macOS 15.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRbla
s.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRlapa
ck.dylib;  LAPACK version 3.12.0
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] caret_6.0-94         lattice_0.22-6       class_7.3-22
##  [4] randomForest_4.7-1.2 gbm_2.2.2            MASS_7.3-61
##  [7] tree_1.0-43          lubridate_1.9.3      forcats_1.0.0
## [10] stringr_1.5.1        dplyr_1.1.4          purrr_1.0.2
## [13] readr_2.1.5          tidyr_1.3.1          tibble_3.2.1
## [16] ggplot2_3.5.1        tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1     timeDate_4041.110    farver_2.1.2
##  [4] fastmap_1.2.0        pROC_1.18.5          digest_0.6.37
##  [7] rpart_4.1.23         timechange_0.3.0     lifecycle_1.0.4
## [10] survival_3.7-0       magrittr_2.0.3       compiler_4.4.1
## [13] rlang_1.1.4          sass_0.4.9           tools_4.4.1
## [16] utf8_1.2.4           yaml_2.3.10          data.table_1.16.2
## [19] knitr_1.48           labeling_0.4.3       plyr_1.8.9
## [22] withr_3.0.1          nnet_7.3-19          grid_4.4.1
## [25] stats4_4.4.1         fansi_1.0.6          colorspace_2.1-1
## [28] future_1.34.0        globals_0.16.3       scales_1.3.0
## [31] iterators_1.0.14     cli_3.6.3            rmarkdown_2.28
## [34] generics_0.1.3       rstudioapi_0.17.1    future.apply_1.11.3
## [37] reshape2_1.4.4       tzdb_0.4.0           cachem_1.1.0
## [40] splines_4.4.1        parallel_4.4.1       vctrs_0.6.5
## [43] hardhat_1.4.0        Matrix_1.7-0         jsonlite_1.8.8
## [46] hms_1.1.3            listenv_0.9.1        foreach_1.5.2
## [49] gower_1.0.1          jquerylib_0.1.4      recipes_1.1.0
## [52] glue_1.8.0           parallelly_1.39.0    codetools_0.2-20
## [55] stringi_1.8.4        gtable_0.3.5         munsell_0.5.1
## [58] pillar_1.9.0         htmltools_0.5.8.1    ipred_0.9-15
## [61] lava_1.8.0           R6_2.5.1             evaluate_0.24.0
## [64] highr_0.11           bslib_0.8.0          Rcpp_1.0.13
## [67] nlme_3.1-166         prodlim_2024.06.25   xfun_0.47
## [70] pkgconfig_2.0.3      ModelMetrics_1.2.2.2
```