Assignment 4

Lorenzo Ausiello

2023-11-28

PROBLEM 1

The dataset analysed for the problem 1 is OJ dataset, The dataset contains 1070 purchases where the customer either purchased Citrus Hill or Minute Maid Orange Juice. A number of characteristics of the customer and product are recorded.

In this analysis, the dataset is divided into a training set with 800 random observations and a test set containing the remaining observations. A decision tree is then fitted to the training data with "Purchase" (a factor with levels CH and MM indicating whether the customer purchased Citrus Hill or Minute Maid Orange Juice) as the response and other variables as predictors.

The classification tree built on the training subset incorporates two key variables, "LoyalCH" and "PriceDiff," resulting in a tree with 8 terminal nodes. The model's fit is summarized with a residual mean deviance of 0.7625 and a training misclassification error rate of 16.5%. This suggests a reasonably effective predictive performance on the training data. However, 9,24% of true CH are misclassified and 27,29% of true MM are misclassified. Picking one of the terminal nodes as example, the tree in the figure shows that when LoyalCH (loyalty to the brand CH) is greater than 0.5036 but PriceDiff (Sale price of MM less sale price of CH) is lower than 0.39, the Purchase prediction is MM (meaning, in that region the are more MM observations).

In the decision tree, specific rules for predicting purchase outcomes can be derived from the splits. For instance, when customer loyalty ("LoyalCH") is less than 0.5036 (but more than 0.2761) and "PriceDiff" exceeds 0.05, the prediction is CH. This implies that in scenarios where customers exhibit lower loyalty but face a higher price differential in favor of CH stores, the model predicts a CH store preference. These interpretable rules provide valuable insights into the nuanced interplay between loyalty, price differentials, and purchasing decisions, facilitating a targeted understanding of customer behavior within the context of the given predictors.

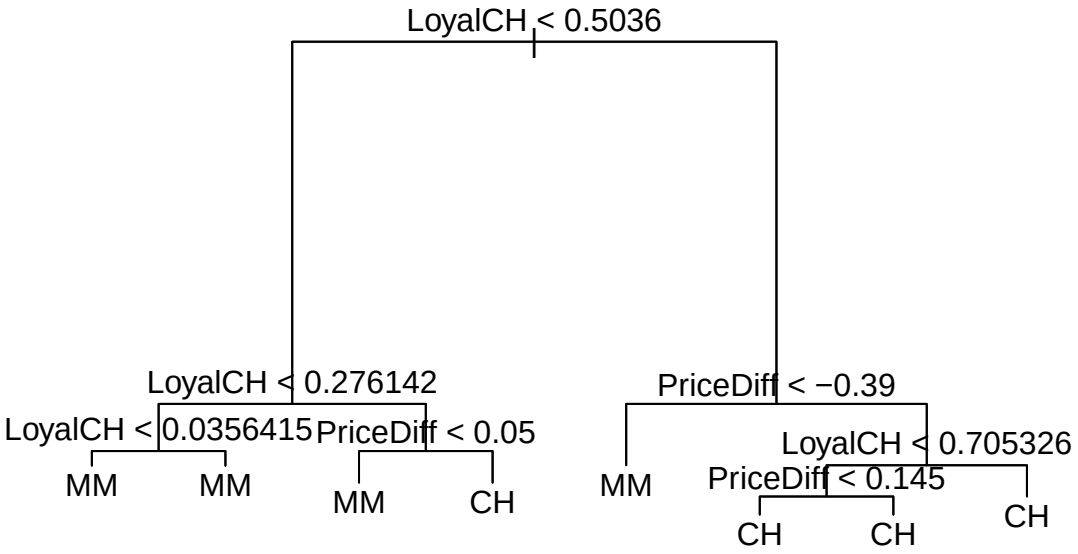
A visual representation of the tree is created using a plot, facilitating a more intuitive understanding of the decision-making process.

The model's performance is assessed on the test set by predicting responses and generating a confusion matrix, allowing computation of the test error rate. The confusion matrix for the test data reveals that out of 270 observations, 150 were correctly classified as CH, 70 as MM, while 34 MM and 16 CH observations were misclassified. The overall error rate, calculated as the sum of misclassifications divided by the total observations, is 18.52%. This indicates that the model accurately predicted the response for approximately 81.48% of the test data. Specifically, the model demonstrated good accuracy in identifying CH observations but had a slightly higher error rate in predicting MM.

Cross-validation is employed using the cv.tree() function on the training set to identify the optimal tree size, and a plot is generated with tree size on the x-axis and cross-validated classification error rate on the y-axis, aiding in the selection of an appropriately sized tree for better generalization performance. Cross-validation suggests that the optimal tree size is 5 nodes, minimizing classification error. The sequence of sizes is 8, 5, 3, 2, and 1. Notably, transitioning from 1 to 2 nodes and further increases in size leads to a reduction in cross-validated error. However, the decline in error is most pronounced between 1 and 2 nodes. This highlights the trade-off between model complexity and accuracy, indicating that a more elaborate tree beyond 2 nodes

may not significantly enhance predictive performance. Hence, selecting a moderately complex tree, such as the one with 8 nodes, strikes a balance between interpretability and accuracy.

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = dataset, subset = train)
## 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
```




```
## 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 ) *
```

Table 1: Confusion matrix: training data

	CH	MM
CH	442	87
MM	45	226

Table 2: Overall error rate: training data

x
0.165

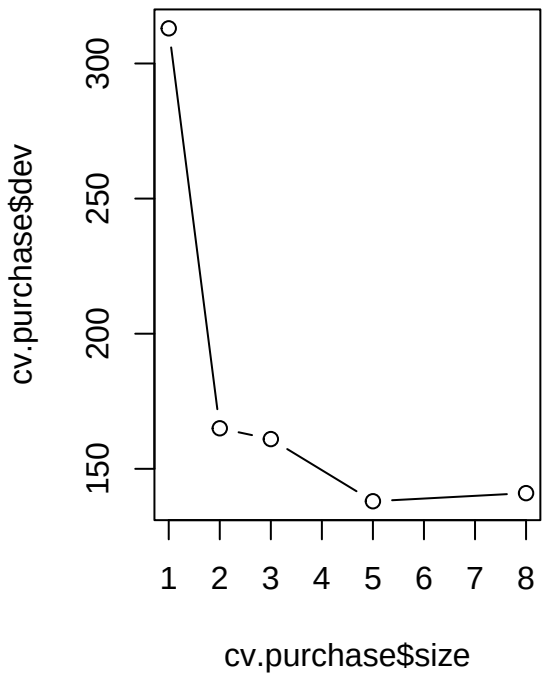
Table 3: Confusion matrix: test data

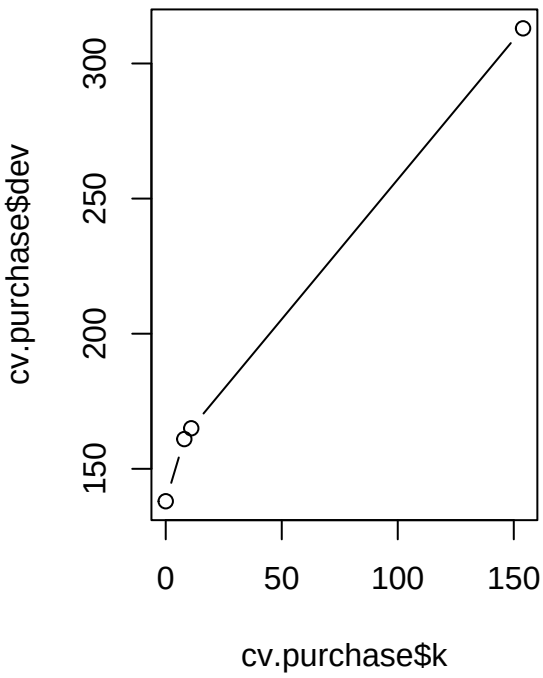
	CH	MM
CH	150	34
MM	16	70

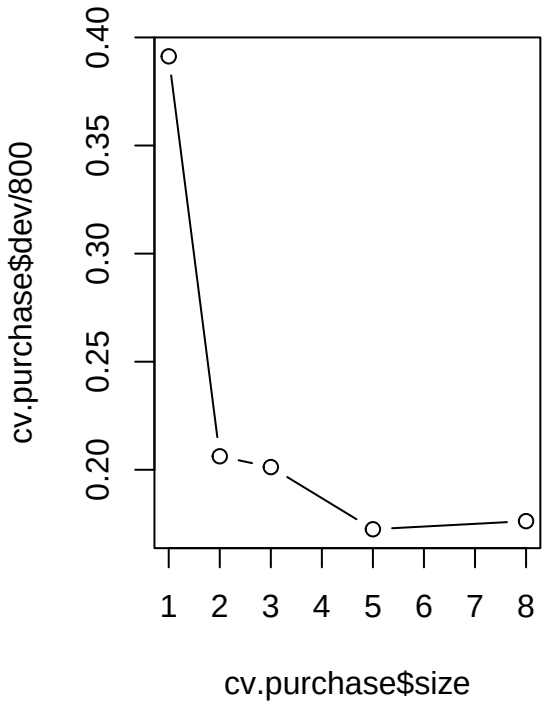
Table 4: Overall error rate: test data

x
0.1851852

```
## $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"
```


cv.purchase$dev
150
200
250
300
1
2
3
4
5
6
7
8
cv.purchase$size


cv.purchase$dev
150
200
250
300
0
50
100
150
cv.purchase$k



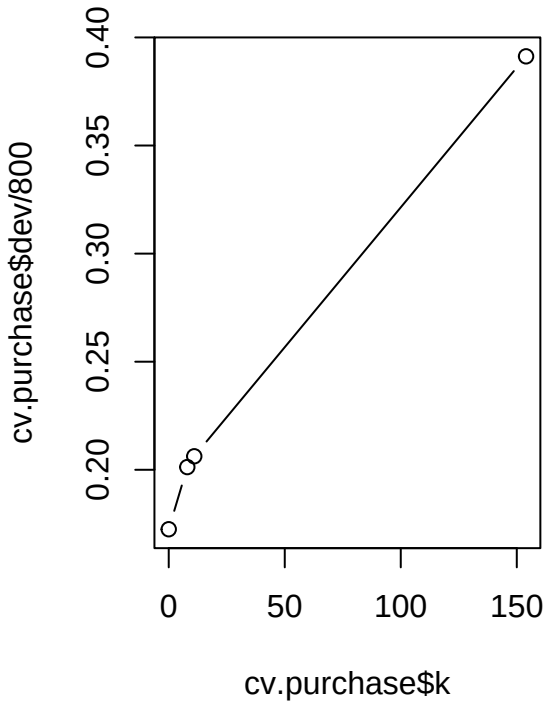
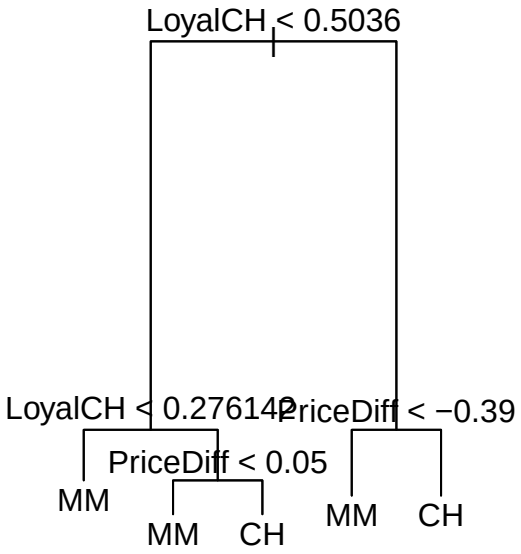


Table 5: Confusion matrix: test data with best=5

	CH	MM
CH	150	34
MM	16	70

Table 6: Overall error rate: test data with best=5

x
0.1851852

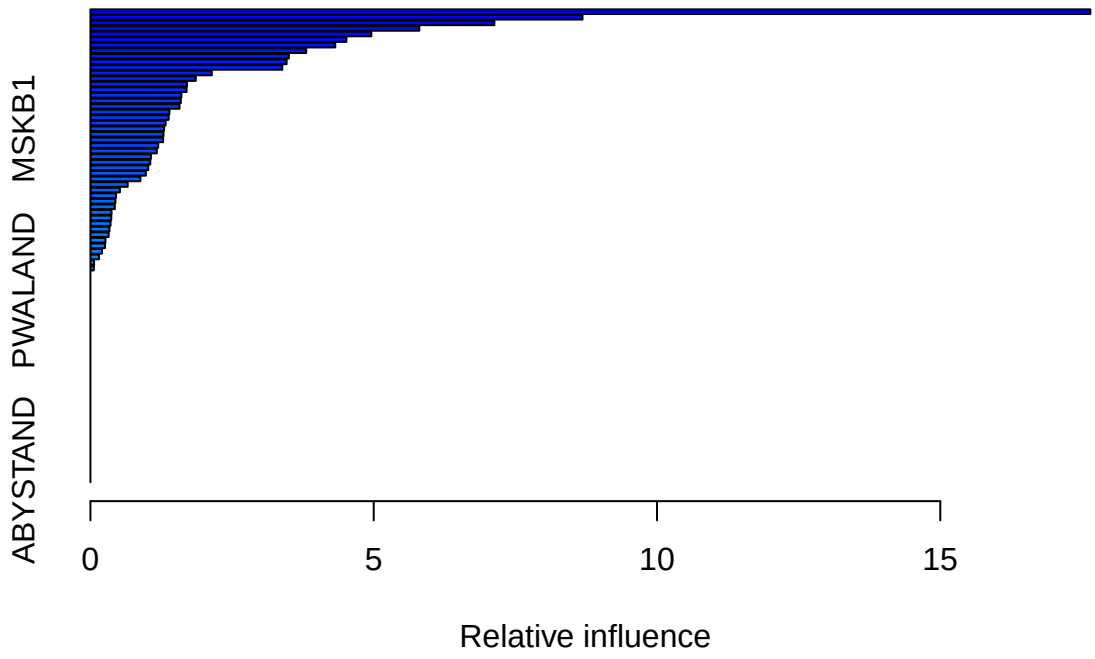


PROBLEM 2

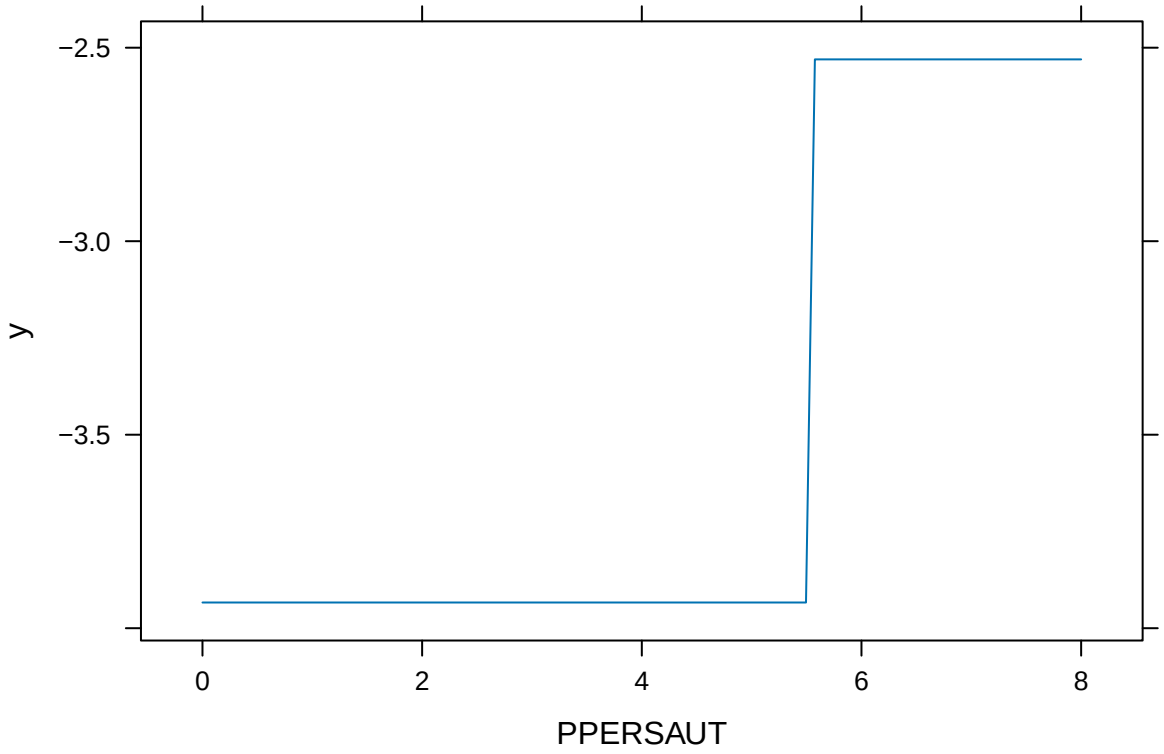
A boosting model was fitted to the training set with the response variable "Purchase" and 85 predictors. The model used 1,000 trees and a shrinkage value of 0.01. The boosting model identified 47 predictors with non-zero influence. The top predictors, ranked by relative influence, include PPERSAUT (17.65%), MAUT2 (8.68%), ALEVEN (7.13%), MBERMIDD (5.81%), and MINKGEM (4.96%). The error rate for the test data at the 20% threshold is approximately 8.38%. 40 people out of 188 predicted make one purchase. However, only 40 out of 296 that make at least one purchase are correctly classified. Logistic regression was applied to the training set, and the resulting model coefficients are presented. The logistic model identified predictors with associated coefficients, along with their standard errors and p-values.The error rate for the logistic model on the test data at the 20% threshold is approximately 11.68%. K-nearest neighbors (KNN) algorithm was applied to the data with a 20% threshold for predicted probabilities. The error rate for the KNN model on the test data at the 20% threshold is approximately 7.49%. The boosting model identified important predictors for predicting purchases, with PPERSAUT, MAUT2, and ALEVEN being the most influential. The boosting model's error rate is comparable to KNN and lower than logistic regression at the 20% threshold.

```
## Loaded gbm 2.1.8.1
```

```
## gbm(formula = Purchase ~ ., distribution = "bernoulli", data = training,
##     n.trees = 1000, shrinkage = 0.01, verbose = F)
## A gradient boosted model with bernoulli loss function.
## 1000 iterations were performed.
## There were 85 predictors of which 47 had non-zero influence.
```




```
##                var   rel.inf
## PPERSAUT PPERSAUT 17.648214
## MAUT2       MAUT2  8.684638
## ALEVEN     ALEVEN  7.129549
## MBERMIDD MBERMIDD  5.805132
## MINKGEM   MINKGEM  4.957738
## MGODGE     MGODGE  4.518531
## MBERHOOG MBERHOOG  4.321225
## MHKOOP     MHKOOP  3.807980
## MHHUUR     MHHUUR  3.502367
## PBRAND     PBRAND  3.462513
```


−2.5
−3.0
−3.5
y
0
2
4
6
8
PPERSAUT

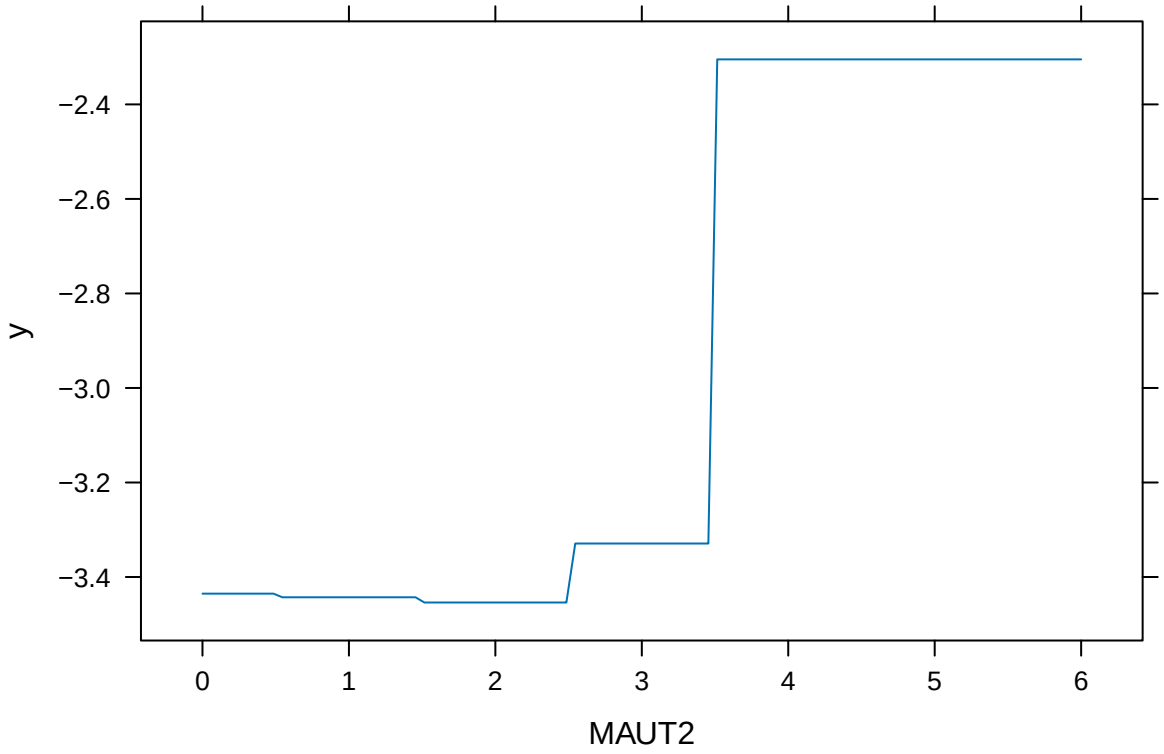


Table 7: Confusion matrix: test data, treshold=0.2

	0	1
0	4378	256
1	148	40

Table 8: Error rate: test data, treshold=0.2

x
0.0837827

```
##
## Call:
## glm(formula = Purchase ~ ., family = binomial, data = training)
##
## Coefficients: (2 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.773e+02  1.162e+05   0.002  0.99809
## MOSTYPE      4.088e-02  1.350e-01   0.303  0.76202
## MAANTHUI    -1.174e-01  5.688e-01  -0.206  0.83651
## MGEMOMV      2.753e-01  4.295e-01   0.641  0.52151
## MGEMLEEF     5.079e-01  2.905e-01   1.748  0.08039 .
## MOSHOOFD    -1.015e-01  6.112e-01  -0.166  0.86811
```

```
## MGODRK      -1.839e-01  3.243e-01  -0.567  0.57068
## MGODPR      -3.478e-01  3.747e-01  -0.928  0.35319
## MGODOV      -1.911e-01  3.056e-01  -0.625  0.53173
## MGODGE      -4.328e-01  3.445e-01  -1.256  0.20899
## MRELGE       5.669e-01  4.554e-01   1.245  0.21323
## MRELSA       4.724e-01  4.388e-01   1.077  0.28162
## MRELOV       5.041e-01  4.584e-01   1.100  0.27142
## MFALLEEN    -1.970e-01  4.002e-01  -0.492  0.62263
## MFGEKIND    -6.129e-01  3.814e-01  -1.607  0.10806
## MFWEKIND    -6.754e-01  4.246e-01  -1.591  0.11171
## MOPLHOOG    -3.486e-01  4.057e-01  -0.859  0.39013
## MOPLMIDD    -6.638e-01  4.229e-01  -1.570  0.11648
## MOPLLAAG    -9.499e-01  4.254e-01  -2.233  0.02555 *
## MBERHOOG    -6.184e-02  2.919e-01  -0.212  0.83224
## MBERZELF    -3.451e-01  3.254e-01  -1.060  0.28893
## MBERBOER    -4.321e-01  3.440e-01  -1.256  0.20915
## MBERMIDD    -6.226e-02  2.780e-01  -0.224  0.82280
## MBERARBG    -3.130e-01  2.781e-01  -1.126  0.26033
## MBERARBO    -8.800e-02  2.821e-01  -0.312  0.75510
## MSKA        -2.225e-02  3.194e-01  -0.070  0.94447
## MSKB1        1.406e-02  3.023e-01   0.047  0.96291
## MSKB2        6.868e-02  2.645e-01   0.260  0.79517
## MSKC         6.041e-01  3.093e-01   1.953  0.05080 .
## MSKD        -3.819e-02  2.981e-01  -0.128  0.89807
## MHHUUR      -1.456e+01  5.937e+03  -0.002  0.99804
## MHKOOP      -1.439e+01  5.937e+03  -0.002  0.99807
## MAUT1        8.327e-02  3.967e-01   0.210  0.83374
## MAUT2        3.762e-01  3.554e-01   1.059  0.28980
## MAUT0        7.795e-02  3.751e-01   0.208  0.83538
## MZFONDS     -1.674e+01  1.146e+04  -0.001  0.99883
## MZPART      -1.683e+01  1.146e+04  -0.001  0.99883
## MINKM30      2.983e-01  3.076e-01   0.970  0.33223
## MINK3045     3.568e-01  3.045e-01   1.172  0.24125
## MINK4575     2.945e-01  3.133e-01   0.940  0.34721
## MINK7512     1.406e-02  3.400e-01   0.041  0.96702
## MINK123M     1.391e-01  4.401e-01   0.316  0.75202
## MINKGEM      4.170e-01  3.429e-01   1.216  0.22393
## MKOOPKLA     9.477e-02  1.334e-01   0.711  0.47738
## PWAPART     -1.133e-01  1.264e+00  -0.090  0.92859
## PWABEDR      2.000e-01  1.259e+00   0.159  0.87376
## PWALAND     -1.882e-01  5.760e+03   0.000  0.99997
## PPERSAUT     5.141e-01  1.673e-01   3.072  0.00213 **
## PBESAUT     -3.334e+00  3.489e+03  -0.001  0.99924
## PMOTSCO      2.390e-01  3.760e+03   0.000  0.99995
## PVRAAUT      2.367e+00  5.263e+03   0.000  0.99964
## PAANHANG    -1.133e+00  9.501e+03   0.000  0.99990
## PTRACTOR    -5.887e+00  6.712e+03  -0.001  0.99930
## PWERKT       1.403e+01  2.229e+04   0.001  0.99950
## PBROM        1.528e+01  1.001e+03   0.015  0.98782
## PLEVEN      -7.152e-01  5.408e-01  -1.323  0.18598
## PPERSONG     1.087e+00  1.627e+04   0.000  0.99995
## PGEZONG      2.922e+00  1.887e+04   0.000  0.99988
## PWAOREG      1.508e+00  9.888e-01   1.525  0.12716
## PBRAND       5.098e-01  2.457e-01   2.075  0.03802 *
```

```
## PZEILPL     -1.641e+01  1.773e+04  -0.001  0.99926
## PPLEZIER    -4.691e+00  2.487e+03  -0.002  0.99850
## PFIETS       1.803e+01  1.252e+04   0.001  0.99885
## PINBOED     -9.919e-01  6.568e+03   0.000  0.99988
## PBYSTAND     2.023e+01  7.687e+03   0.003  0.99790
## AWAPART     -1.436e-02  2.536e+00  -0.006  0.99548
## AWABEDR     -5.686e-02  3.980e+00  -0.014  0.98860
## AWALAND     -1.561e+01  2.033e+04  -0.001  0.99939
## APERSAUT    -9.787e-01  8.215e-01  -1.191  0.23354
## ABESAUT      1.994e+00  1.855e+04   0.000  0.99991
## AMOTSCO     -1.873e+01  1.644e+04  -0.001  0.99909
## AVRAAUT     -1.251e+01  1.813e+04  -0.001  0.99945
## AAANHANG    -1.549e+01  1.701e+04  -0.001  0.99927
## ATRACTOR     2.198e+00  1.976e+04   0.000  0.99991
## AWERKT      -3.364e+01  4.589e+04  -0.001  0.99942
## ABROM       -7.385e+01  5.005e+03  -0.015  0.98823
## ALEVEN       1.346e+00  8.151e-01   1.651  0.09867 .
## APERSONG    -2.044e+01  3.411e+04  -0.001  0.99952
## AGEZONG     -2.485e+01  4.040e+04  -0.001  0.99951
## AWAOREG     -4.121e+00  5.478e+00  -0.752  0.45189
## ABRAND      -1.230e+00  9.839e-01  -1.250  0.21140
## AZEILPL             NA         NA      NA       NA
## APLEZIER            NA         NA      NA       NA
## AFIETS      -1.838e+01  1.252e+04  -0.001  0.99883
## AINBOED     -1.534e+01  1.469e+04  -0.001  0.99917
## ABYSTAND    -7.744e+01  3.075e+04  -0.003  0.99799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 408.73  on 999  degrees of freedom
## Residual deviance: 281.34  on 916  degrees of freedom
## AIC: 449.34
##
## Number of Fisher Scoring iterations: 19
```

Table 9: Logistic Confusion matrix: test data, treshold=0.2

	0	1
0	4207	244
1	319	52

Table 10: Logistic Error rate: test data, treshold=0.2

x
0.1167565

Table 11: Knn Confusion matrix: test data, treshold=0.2

	0	1
0	4441	276
1	85	20

Table 12: Knn Error rate: test data, treshold=0.2

x
0.0748652

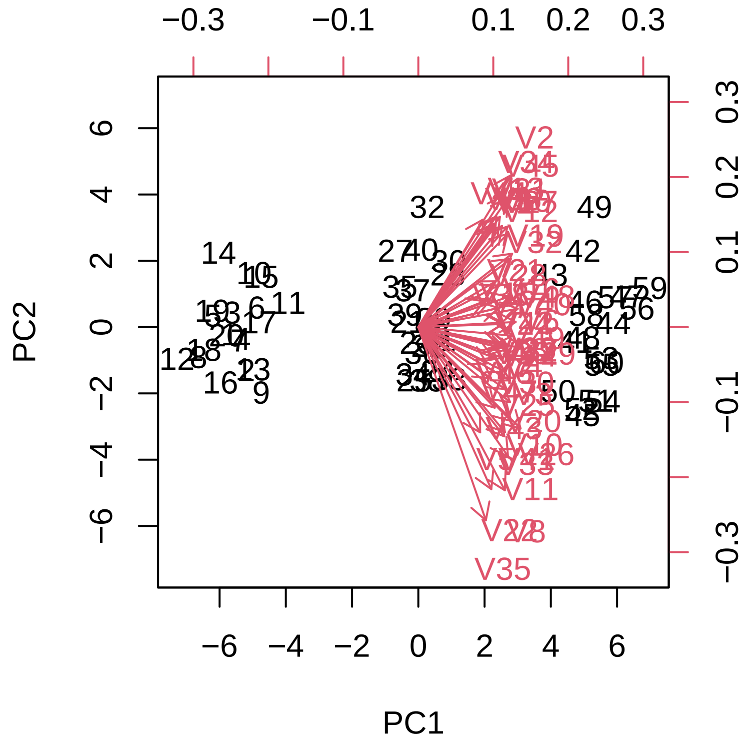
PROBLEM 3

A simulated dataset with 60 observations (20 in each of three classes) and 50 variables was generated. A mean shift was added to create three distinct classes. PCA was performed on the simulated data. The first two principal component score vectors were plotted, with different colors indicating observations in each of the three classes. K-means clustering was applied to the original data with K = 3. The clusters obtained were compared with the true class labels using a confusion matrix. Results indicate perfect clustering, with each class correctly identified by K-means. Then, K-means clustering was performed with K = 2 and K = 4. K-means clustering was applied to the first two principal component score vectors. The clustering results were compared with true class labels. Results indicate perfect clustering. K-means clustering was performed on the data after scaling each variable to have standard deviation one. The clustering results were compared with true class labels using a confusion matrix. Results indicate perfect clustering, aligning with the true class labels.

The first two principal component score vectors were crucial for clear class separation in the PCA analysis. K-means clustering on the original data, on the first two PC and on scaled variables resulted in perfect clustering when K = 3, suggesting strong separation between the classes. The analysis demonstrates the effectiveness of PCA in capturing the most relevant information for clustering and the impact of scaling on K-means clustering results. The ideal choice of K may vary based on the dataset and the nature of the classes.

Table 13: PC: 6 out of 86

	PC1	PC2	PC3	PC4	PC5	PC6
V1	0.1274534	0.1667633	-0.0588011	0.1317275	-0.0679477	0.2006661
V2	0.1551264	0.2534071	0.0507287	0.0928177	0.1506128	0.0489057
V3	0.1527464	-0.0889333	0.1007438	-0.0096783	0.0667688	0.1449309
V4	0.1399330	-0.0511287	-0.0391817	-0.1157673	0.2191128	-0.2868141
V5	0.1031090	-0.1754824	-0.2950898	-0.0055306	0.1336590	0.0515192
V6	0.1624167	0.0085599	0.0125950	-0.1258323	0.0196811	0.0781587


PC1
PC2

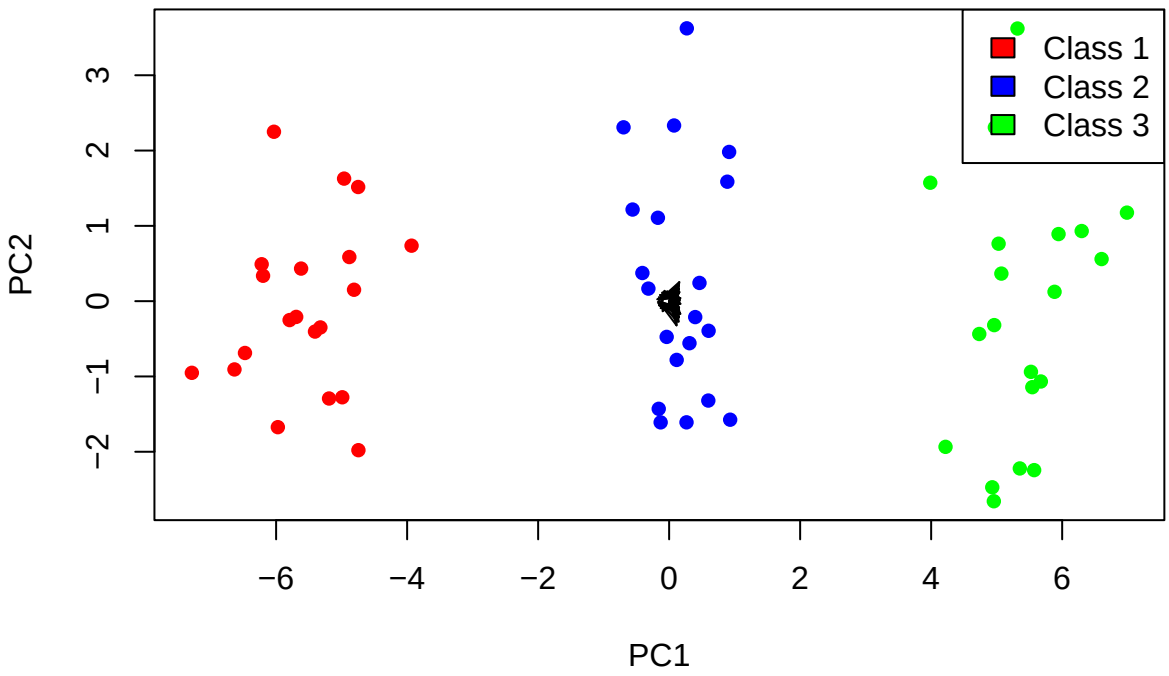
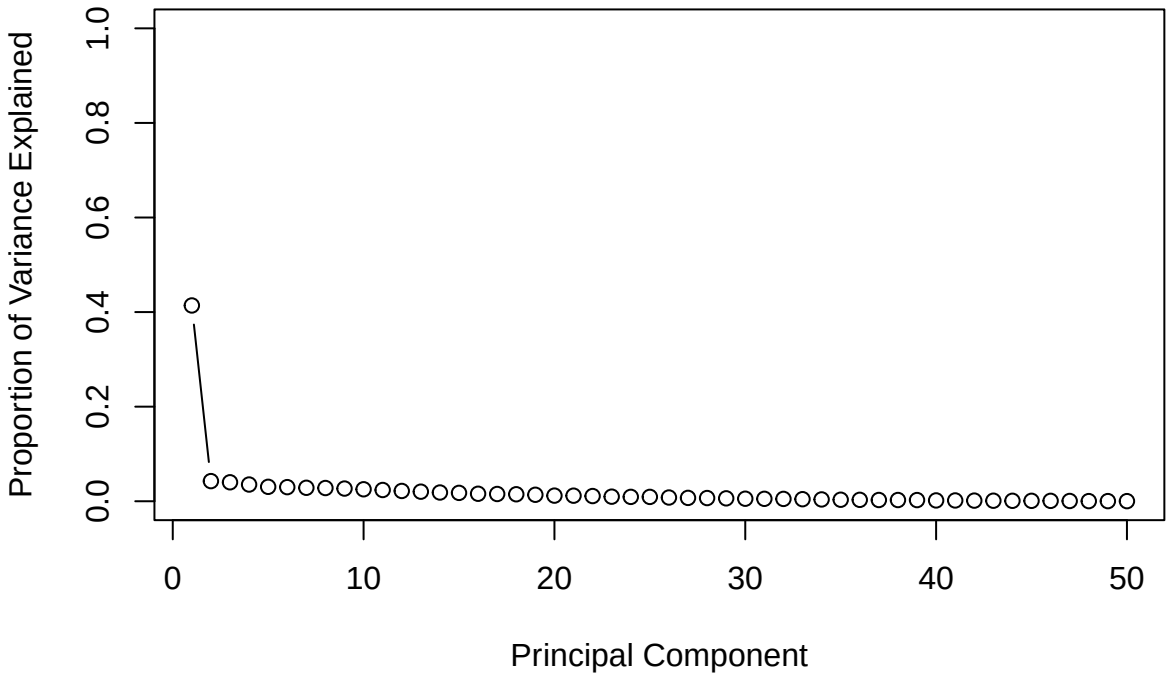


Table 14: Standard deviation of the first 6 PC

x
4.549604
1.459136
1.416016
1.329359
1.234823
1.221348

Table 15: Variance of the first 6 PC

x
20.698897
2.129079
2.005101
1.767196
1.524789
1.491691


Proportion of Variance Explained
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
Principal Component

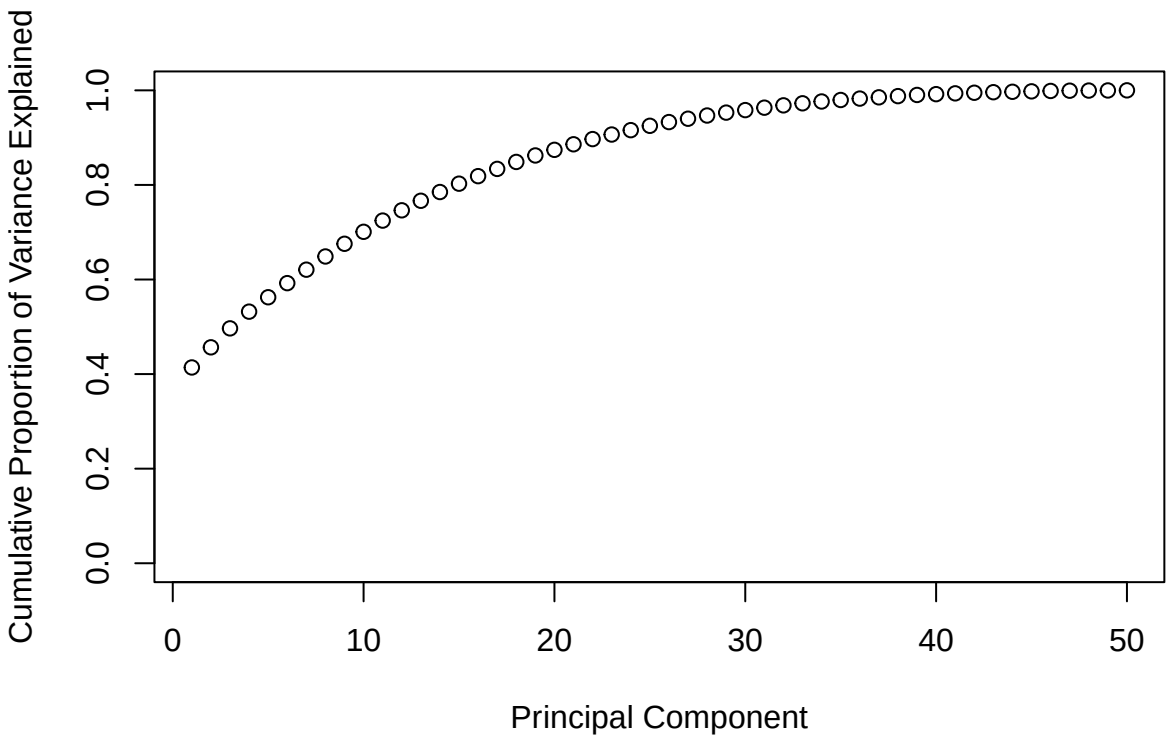


Table 16: K-Means Clustering, k=3

3	3	3	3	3	3	3	2	2	2	2	2	2	2	1	1	1	1	1	1
3	3	3	3	3	3	3	2	2	2	2	2	2	1	1	1	1	1	1	1
3	3	3	3	3	3	2	2	2	2	2	2	2	1	1	1	1	1	1	1

Table 17: Class predicted vs True classes

1	2	3
20	0	0
0	20	0
0	0	20

Table 18: TOT Withinss

x
2822.535

Table 19: K-Means Clustering, k=2

2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1

2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 20: TOT Withinss

x
3306.083

Table 21: K-Means Clustering, k=4

3	3	3	3	3	3	3	1	2	2	1	1	1	1	4	4	4	4	4	4
3	3	3	3	3	3	3	2	2	2	2	2	2	4	4	4	4	4	4	4
3	3	3	3	3	3	1	2	2	1	2	1	2	4	4	4	4	4	4	4

Table 22: TOT Withinss

x
2711.278

Table 23: K-Means Clustering on the first two PC, k=3

3	3	3	3	3	3	3	1	1	1	1	1	1	1	2	2	2	2	2	2
3	3	3	3	3	3	3	1	1	1	1	1	1	2	2	2	2	2	2	2
3	3	3	3	3	3	1	1	1	1	1	1	1	2	2	2	2	2	2	2

Table 24: Class predicted vs True classes

1	2	3
20	0	0
0	20	0
0	0	20

Table 25: TOT Withinss

x
243.2963

Table 26: K-Means Clustering on scaled variable, k=3

1	1	1	1	1	1	1	3	3	3	3	3	3	3	2	2	2	2	2	2
1	1	1	1	1	1	1	3	3	3	3	3	3	2	2	2	2	2	2	2
1	1	1	1	1	1	3	3	3	3	3	3	3	2	2	2	2	2	2	2

Table 27: Class predicted vs True classes

1	2	3
20	0	0
0	20	0
0	0	20

Table 28: TOT Withinss

x
1730.152