

Machine Learning Coursework – John-Alex Yannoulas – w1775197

5DATA002W

Question 1 Discussion, Output, Etc.	Pages 2 – 7
Question 1 Code	Pages 10 - 21
Question 2 Discussion, Output, Etc.	Pages 8 - 9
Question 2 Code	Pages 22 - 25

Question 1

1. Pre-processing tasks (2 marks for scaling and 3 marks for outliers removal)

```

1 ##### W1775197 - JOHN-ALEX YANNOULIAS #####
2 library(factoextra)
3 library(ggplot2)
4 library(cluster)
5 library(NbClust)
6 set.seed(123) # the k-means algorithm uses a random set of initial points to
7 # arrive at the final classification
8
9 library(readxl)
10 # ENTER YOUR PATHNAME BELOW WHEN MARKING
11 W<- read_excel("~/Users/john-alexannoulis/Documents/University/YEAR 2/YEAR
12 W<- as.data.frame(W)
13 names(W)
14 head(W)
15 summary(W)
16 str(W)
17
18 # 1- Pre-processing tasks (2 marks for scaling and 3 marks for outliers rem
19 # .....Checking for missing values
20 sum(is.na(W))
21 # .....Function to find outliers
22 outliers <- function(x) {
23   > summary(W)
24   Q1 <- quantile(x, probs=.10)
25   Q3 <- quantile(x, probs=.90)
26   iqr = Q3 - Q1
27   upper_limit = Q3 + (iqr*1.5)
28   lower_limit = Q1 - (iqr*1.5)
29   x > upper_limit | x < lower_limit
30 }
31 # .....Function to remove outliers
32 remove_outliers <- function(df, cols = names(df)) {
33   for (col in cols) {
34     df <- df[!outliers(df[[col]]),]
35   }
36   df
37 }
38 # .....Function for scale
39 min_max <- function(x) {
40   return ((x - min(x)) / (max(x) - min(x)))
41 }
42 # Removing outliers from all columns except the last (quality)
43 W_clean <- remove_outliers(W[1:12])
44 # .....Splitting the data
45 quality <- W_clean$quality
46 x <- W_clean[, 12]
47 # .....Scaling the data
48 x <- as.data.frame(lapply(x, min_max))
49
50 ##### W1775197 - JOHN-ALEX YANNOULIAS :

```

```

> names(W)
[1] "fixed acidity" "volatile acidity" "citric acid" "residual sugar" "chlorides"
[6] "free sulfur dioxide" "total sulfur dioxide" "density" "pH" "sulphates"
[11] "alcohol" "quality"
> head(W)
  fixed acidity volatile acidity citric acid residual sugar chlorides free sulfur dioxide total sulfur dioxide density
1      8.1      0.27      0.41      1.45      0.033      11      63      0.9908
2      8.6      0.23      0.40      4.20      0.035      17     109      0.9947
3      7.9      0.18      0.37      1.20      0.040      16      75      0.9920
4      8.3      0.42      0.62     19.25      0.040      41     172      1.0002
5      6.5      0.31      0.14      7.50      0.044      34     133      0.9955
6      7.6      0.67      0.14      1.50      0.074      25     168      0.9937
> summary(W)
      pH sulphates alcohol quality
1 2.99      0.56     12.0      5
2 3.14      0.53      9.7      5
3 3.18      0.63     10.8      5
4 2.98      0.67      9.7      5
5 3.22      0.50      9.5      5
6 3.05      0.51      9.3      5
> summary(W)
fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  free sulfur dioxide
Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900   Min.   : 2.00
1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700   1st Qu.:0.03600   1st Qu.: 24.00
Median : 6.800   Median :0.2600   Median :0.3200   Median : 3.300   Median :0.04300   Median : 34.00
Mean   : 6.842   Mean   :0.2744   Mean   :0.3352   Mean   : 6.455   Mean   :0.04561   Mean   : 35.65
3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.:10.000   3rd Qu.:0.05000   3rd Qu.: 46.00
Max.   :14.200   Max.   :0.9650   Max.   :1.6600   Max.   :65.800   Max.   :0.34600   Max.   :131.00
total sulfur dioxide  density
Min.   : 9.0   Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :15.000
1st Qu.:109.0   1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:15.000
Median :134.0   Median :0.9937   Median :3.180   Median :0.4800   Median :10.40   Median :16.000
Mean   :138.7   Mean   :0.9940   Mean   :3.188   Mean   :0.4904   Mean   :10.53   Mean   :15.952
3rd Qu.:167.0   3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:16.000
Max.   :344.0   Max.   :1.0390   Max.   :3.820   Max.   :1.0000   Max.   :14.20   Max.   :18.000
> str(W)
'data.frame': 4710 obs. of 12 variables:
 $ fixed acidity : num 8.1 8.6 7.9 8.3 6.5 7.6 5.8 7.3 6.5 7.3 ...
 $ volatile acidity : num 0.27 0.23 0.18 0.42 0.31 0.67 0.27 0.28 0.39 0.24 ...
 $ citric acid : num 0.41 0.4 0.37 0.62 0.14 0.14 0.2 0.43 0.23 0.39 ...
 $ residual sugar : num 1.45 4.2 1.2 19.25 7.5 ...
 $ chlorides : num 0.033 0.035 0.04 0.04 0.044 0.074 0.044 0.08 0.051 0.057 ...
 $ free sulfur dioxide : num 11 17 16 41 34 25 22 21 25 45 ...
 $ total sulfur dioxide : num 63 109 75 172 133 168 179 123 149 149 ...
 $ density : num 0.991 0.995 0.992 1 0.996 ...
 $ pH : num 2.99 3.14 3.18 2.98 3.22 3.05 3.37 3.19 3.24 3.21 ...
 $ sulphates : num 0.56 0.53 0.63 0.67 0.5 0.51 0.37 0.42 0.35 0.36 ...
 $ alcohol : num 12 9.7 10.8 9.7 9.5 9.3 10.2 12.8 10 8.6 ...
 $ quality : num 5 5 5 5 5 5 5 5 5 ...
> sum(is.na(W))
[1] 0
> # .....Function to find outliers
> outliers <- function(x) {

```

Data

W	4710 obs. of 12 variables
W_clean	4578 obs. of 12 variables
x	4578 obs. of 11 variables

Values

quality	num [1:4578] 5 5 5 5 5 5 5 5 5 5 ...
---------	--------------------------------------

Functions

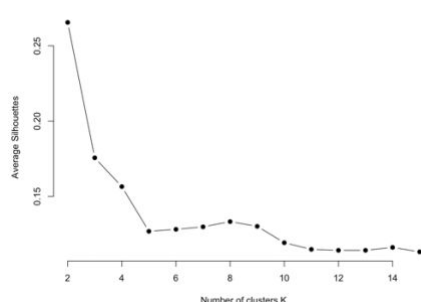
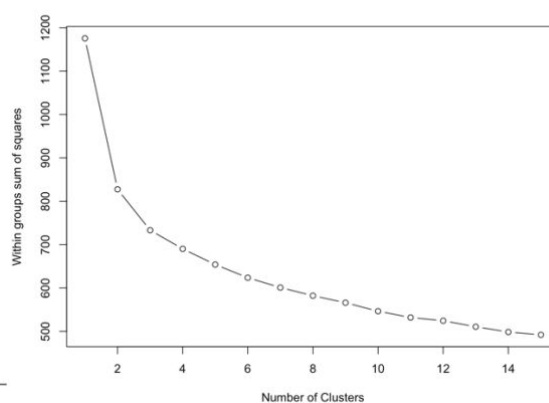
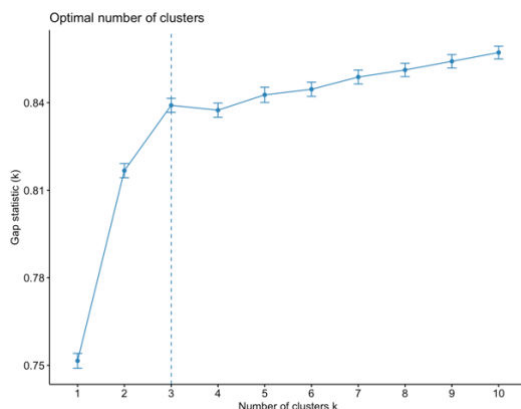
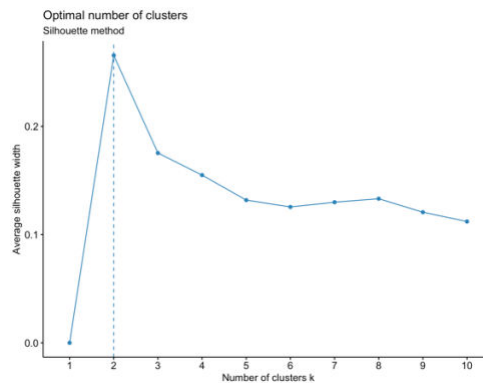
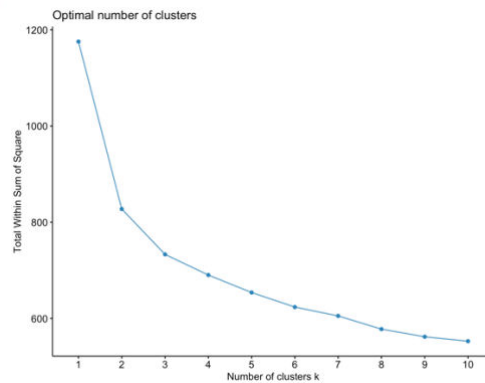
min_max	function (x)
outliers	function (x)
remove_outliers	function (df, cols = names(df))

2. Define the number of cluster centres by showing all necessary steps/methods via manual & automated tools (2 marks for each one of these automated tools)

```

59 # 2- Define the number of cluster centres by showing all necessary
60 # steps/methods via manual & automated tools
61 #.....Elbow Method.....
62 #.....Manual Method.....
63 wss <- 0
64 # For 1 to 15 cluster centers
65 for (i in 1:15) {
66   km.out <- kmeans(x, centers = i)
67   wss[i] <- km.out$tot.withinss
68 }
69 wss
70 plot(1:15, wss, type = "b", xlab = "Number of Clusters",
71      ylab = "Within groups sum of squares")
72 #.....Automated Method
73 # For the automated Elbow method, no needs to program the Elbow Method, has
74 # been implemented already in R.(wss= within-cluster sum of square)
75 fviz_nbclust(x, kmeans, method = "wss")
76
77 #.....Average Silhouette.....
78 #.....Manual Method.....
79 # function to compute average silhouette for k clusters
80 Average_Silhouette <- function(k) {
81   km.res <- kmeans(x, centers = k, nstart = 100)
82   ss <- silhouette(km.res$cluster, dist(x))
83   mean(ss[, 3])
84 }
85 # Compute and plot wss for k = 2 to k = 15
86 k.values <- 2:15
87
88 # extract avg silhouette for 2-15 clusters
89 library(purrr)
90 avg_sil_values <- map_dbl(k.values, Average_Silhouette)
91 # plot
92 plot(k.values, avg_sil_values,
93      type = "b", pch = 19, frame = FALSE,
94      xlab = "Number of clusters K",
95      ylab = "Average Silhouettes")
96 #.....Automated Method
97 fviz_nbclust(x, kmeans, method = "silhouette")+
98   labs(subtitle = "Silhouette method") #Seems to suggest 2
99 # which suggests 2 clusters as the optimal number of clusters.
100
101 #.....Gap-Statistic Method.....
102 #.....Manual Method.....
103 # To compute the gap statistic method we can use the clusGap function
104 # which provides the gap statistic and standard error for an output
105
106 # compute Gap-statistic
107 set.seed(123) # the k-means algorithm uses a random set of initial points to
108 # arrive at the final classification
109 gap_stat <- clusGap(x, FUN = kmeans, nstart = 100,
110                    K.max = 10, B = 50)
111 # Print the result
112 print(gap_stat, method = "firstmax")
113 # visualize the results with fviz_gap_stat
114 fviz_gap_stat(gap_stat)
115 # The Manual Method suggests 3 clusters as the optimal number of clusters.
116 #.....Automated Method
117 fviz_nbclust(x, kmeans, method = "gap_stat")+
118   labs(subtitle = "Gap-Statistics") #Seems to suggest 3
119 # The Automated Method also suggests 3 clusters as the optimal number of clusters.
120
121 # .....Consensus-based algorithm
122 # Because no method is clearly better, a fourth alternative is to run many methods
123 # and take the number of clusters that is the most agreed upon
124 library(parameters)
125 n.clust <- n_clusters(x,
126                      package = c("easystats", "NbClust", "mclust"),
127                      nbclust_method = "kmeans", n_max = 5, standardize = FALSE)
128 n.clust
129 # The optimal number of clusters to retain can also be visualized:
130 plot(n.clust)
131 # The winning Model is the one with k=3, based on the previous analysis
132 #####
133 #####
134 #####
135
136

```



Data	
km.out	List of 9
W	4710 obs. of 12 variables
W_clean	4578 obs. of 12 variables
x	4578 obs. of 11 variables
Values	
i	15L
quality	num [1:4578] 5 5 5 5 5 5 5 5 5 ...
wss	num [1:15] 1176 827 733 690 654 ...
Functions	
min_max	function (x)
outliers	function (x)
remove_outliers	function (df, cols = names(df))

3. K-means analysis for each k attempt (2 marks for each k attempt) (all attributed used)

K-2

```
# 3- K-means analysis for each k attempt (2 marks for each k attempt)
#Notes:
# Because each cluster becomes smaller and tighter, the overall WSS
# decreases as the number of clusters grows. The hope is that once k exceeds
# the ideal number of clusters, the rate at which the WSS drops would slow down.

### .....For K=2
# The "nstart" argument in the kmeans() function allows to run the algorithm
# several times with different initial centers, in order to obtain a
# potentially better partition
kc2 <- kmeans(x,2, nstart = 100)
# displays the class determined by the model for all observations:
kc2$cluster # The output kc2$cluster specifies the group (k=2)
# to which each observation belongs to.

#The cluster for each observation can be stored directly in the dataset
# as a column:
kc2_cluster <- data.frame(x, cluster = as.factor(kc2$cluster))
head(x)
```

Data	
gap_stat	List of 6
kc2_cluster	4578 obs. of 12 variables
kc2	List of 9
kn_out	List of 9
n_cluster	29 obs. of 4 variables
W	4738 obs. of 12 variables
W_clean	4578 obs. of 12 variables
x	4578 obs. of 11 variables

```
Values
avg_sil_values num [1:14] 0.266 0.176 0.156 0.127 0.128 ...
bss_k2 348.11556680182
cm_k2 "table" int [1:4, 1:2] 791 826 155 26 682 1310 721 147
i 15L
k_values int [1:14] 2 3 4 5 6 7 8 9 10 11 ...
quality num [1:4578] 5 5 5 5 5 5 5 5 5 ...
score_k2 0.296115462308078
tss_k2 1175.68060283239
wss num [1:15] 1176 827 733 690 654 ...
wss_k2 827.487125952289

Functions
Average.Silhouette function(x)
min_max function(x)
outliers function(x)
remove_outliers function(df, cols = names(df))
```

K-3

```
### .....For K3
kc3 <- kmeans(quality,3, nstart = 100)
kc3
kc3$centers
# how many Observations are in each Cluster
kc3$size # Generally, a good clustering will be well-balanced
# (N table)
cm_k3 = table(quality, kc3$cluster)
cm_k3
wss_k3 = kc3$tot.withinss
wss_k3
bss_k3 = kc3$betweenss
bss_k3
tss_k3 = wss_k3 + bss_k3
score_k3 = bss_k3 / tss_k3
# The winning Model is the one with k=3, based on the previous indices
195- #####
```

Data	
gap_stat	List of 6
kc3_cluster	4578 obs. of 12 variables
kc3	List of 9
kn_out	List of 9
n_cluster	29 obs. of 4 variables
W	4738 obs. of 12 variables
W_clean	4578 obs. of 12 variables
x	4578 obs. of 11 variables

```
Values
avg_sil_values num [1:14] 0.266 0.176 0.156 0.127 0.128 ...
bss_k2 348.11556680182
bss_k3 2818.1430486272
cm_k2 "table" int [1:4, 1:2] 791 826 155 26 682 1310 721 147
cm_k3 "table" int [1:4, 1:3] 0 876 173 1393 0 0 0 2136 ...
i 15L
k_values int [1:14] 2 3 4 5 6 7 8 9 10 11 ...
quality num [1:4578] 5 5 5 5 5 5 5 5 5 ...
score_k2 0.296115462308078
score_k3 0.296115462308078
tss_k2 1175.68060283239
tss_k3 2954.61271297521
wss num [1:15] 1176 827 733 690 654 ...
wss_k2 827.487125952289
wss_k3 144.40818112488

Functions
Average.Silhouette function(x)
min_max function(x)
outliers function(x)
remove_outliers function(df, cols = names(df))
```

4. Evaluation of the produced outputs against 12th column (1 mark for the “CM” table, 6 marks for calculation of these requested indices)

(READ COMMENTS BELOW OR IN ATTACHED CODE FOR DISCUSSION)

1 Mark for CM Matrix Table

```
198 # 4-Evaluation of the produced outputs against 12th column
199 # a- (1 mark for the "CM" table)
200 # Evaluation of the produced outputs against 12th column (quality variable)
201 # Confusion Matrix "CM" table
202 cm_k2 = table(quality, kc2$cluster)
203 cm_k2
204 cm_k3 = table(quality, kc3$cluster)
205 cm_k3
206
207 # The quality of a k-means partition is found by calculating the percentage
208 # of the TSS "explained" by the partition using the following formula:
209 # (BSS/TSS)*100
210
211 # The higher the percentage, the better the score (and thus the quality)
212 # because it means that BSS is large and/or WSS is small.
213 # BSS and TSS are extracted from the model and stored
214
215 bss_k2 = kc2$betweenss # the between sum of squares BSS(k)
216 bss_k2
217 TSS_k2 <- kc2$totss
218 TSS_k2
219 # calculate the quality of the partition
220 bss_k2 / TSS_k2 * 100
221 # The quality of the partition is 29.61%. This value has no real
222 # interpretation in absolute terms except that a higher quality means a higher
223 # explained percentage.
224
225
226 bss_k3 = kc3$betweenss # the between sum of squares BSS(k)
227 bss_k3
228 TSS_k3 <- kc3$totss
229 TSS_k3
230 # calculate the quality of the partition
231 bss_k3 / TSS_k3 * 100
232 # The quality of the partition is 95.11%. This value has no real
233 # interpretation in absolute terms except that a higher quality means a higher
234 # explained percentage.
235
236 #Conclusion: As can be seen, classifying the data into three sections provides for a larger explained
237 # percentage and improved quality.
238 # We conclude that the K=3 is the optimal clustering strategy
239
240
```

```
quality      1      2
5      791  602
6      826 1310
7      155   721
8       26   147
> cm_k3 = table(quality, kc3$cluster)
> cm_k3

quality      1      2      3
5       0 1393   0
6       0   0 2136
7      876   0   0
8      173   0   0

> # The quality of a k-means partition is found by calculating the percentage
> # of the TSS "explained" by the partition using the following formula:
> # (BSS/TSS)*100
>
> # The higher the percentage, the better the score (and thus the quality)
> # because it means that BSS is large and/or WSS is small.
> # BSS and TSS are extracted from the model and stored
>
> bss_k2 = kc2$betweenss # the between sum of squares BSS(k)
> bss_k2
[1] 348.1136
> TSS_k2 <- kc2$totss
> TSS_k2
[1] 1175.601
> # calculate the quality of the partition
> bss_k2 / TSS_k2 * 100
[1] 29.61155
> # The quality of the partition is 29.61%. This value has no real
> # interpretation in absolute terms except that a higher quality means a higher
> # explained percentage.
>
> bss_k3 = kc3$betweenss # the between sum of squares BSS(k)
> bss_k3
[1] 2810.144
> TSS_k3 <- kc3$totss
> TSS_k3
[1] 2954.613
> # calculate the quality of the partition
> bss_k3 / TSS_k3 * 100
[1] 95.11039
> # The quality of the partition is 95.11%. This value has no real
> # interpretation in absolute terms except that a higher quality means a higher
```

6 marks for calculation of these requested indices

```
242 # b- (6 marks for calculation of these requested indices)
243 # k2 <- kmeans(x,2, nstart = 50)
244 library("cluster")
245 #Structure of each K
246 str(kc2)
247
248 # b- (6 marks for calculation of these requested indices)
249 # k2
250 k2 <- kmeans(x,2, nstart = 50)
251 # The output of kmeans is a list with several bits of information
252 kc2$cluster # indicating the cluster to which each point is allocated.
253 kc2$centers # A matrix of cluster centers.
254 kc2$totss # The total sum of squares.
255 kc2$withinss # Vector of within-cluster sum of squares, one component per cluster.
256 kc2$tot.withinss
257 kc2$betweenss # The between-cluster sum of squares, i.e. $totss-tot.withinss$.
258 kc2$size # The number of points in each cluster.
259 table(kc2$cluster) #Give a count of data points in each cluster
260 tss_k2 = wss_k2 + bss_k2
261 score_k2 = bss_k2 / tss_k2
262 score_k2
263
264 # k=3
265 k3 <- kmeans(x,3, nstart = 50)
266 kc3$cluster
267 kc3$centers
268 kc3$totss
269 kc3$withinss
270 kc3$betweenss
271 kc3$size
272 table(kc3$cluster)
273 tss_k3 = wss_k3 + bss_k3
274 score_k3 = bss_k3 / tss_k3
275 score_k3
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

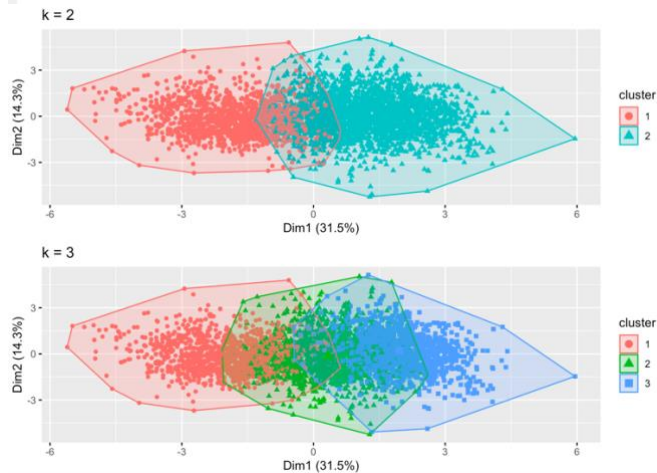
5. Define the final “winner” cluster case and provide brief explanation of evaluation indices

(READ COMMENTS BELOW OR IN ATTACHED CODE FOR DISCUSSION)

```
# 4- Define the final "winner" cluster case and provide brief
# explanation of evaluation indices
# plots to compare
p2 <- fviz_cluster(kc2, geom = "point", data = x) + ggtitle("k = 2")
p3 <- fviz_cluster(kc3, geom = "point", data = x) + ggtitle("k = 3")
library(gridExtra)
grid.arrange(p2, p3, nrow = 2)

# Although this visual assessment tells us where true delineations occur
# (or do not occur) between clusters.
# However, it does not tell us what the optimal number of clusters is.
```

Spellcheck



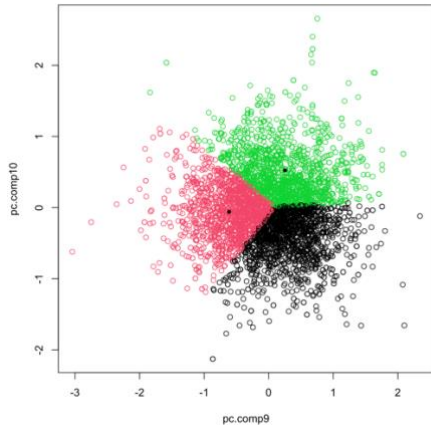
6. Apply a PCA for this white wine dataset (2 marks). Create a new dataset with those PCs with a cumulative score > 96%, as attributes (2 marks).

```
21 # 6- Apply a PCA for this white wine dataset (2 marks). Create a new
22 # dataset with those PCs with a cumulative score > 96%, as attributes (2 marks).
23
24 PC<-princomp(x, cor=T)
25
26 pc.comp <- PC$scores
27 head(PC$scores)
28 # To compute the proportion of variance explained (PVE) by each principal component,
29 # The variance explained by each principal component is obtained by squaring values:
30 # simply divide the variance explained by each principal component by the total
31 # variance explained by all four principal components:
32
33
34 PVE<-cumsum(PC$dev^2 / sum(PC$dev^2))
35 # Observe the cumulative scores, the cumulative scores above 0.96 (96%) will be accepted
36 round(PVE, 2)
37 # Comp9, Comp10, Comp11 have the the cumulative scores above 0.96 (96%)
38 # will be set as attributes
39
40
41 # Create a new dataset with
42 # those PCs with a cumulative score > 96%, as attributes
43 pc.comp9 <- -1*pc.comp[,9] # principal component 9 has a score>96%
44 pc.comp10 <- -1*pc.comp[,10] # principal component 10 has a score>96%
45 pc.comp11 <- -1*pc.comp[,11] # principal component 11 has a score>96%
46 PC_DataFrame <- cbind(pc.comp9, pc.comp10, pc.comp11)
47
48 > # 6- Apply a PCA for this white wine dataset (2 marks). Create a new
49 > # dataset with those PCs with a cumulative score > 96%, as attributes (2 marks).
50
51 > PC<-princomp(x, cor=T)
52
53 > pc.comp <- PC$scores
54 head(PC$scores)
55
56 [1,] -2.413434 2.7045799 0.5493323 -0.2420975 1.0500507 -0.3732729 0.4331744 -0.12729467 0.89704293 0.2516618
57 [2,] -0.1470334 2.1318533 0.7021910 -0.8441022 0.6067261 -0.4650888 0.1888694 1.2383035 -0.65472016 0.2987566
58 [3,] -1.5948264 1.402884 1.4730492 -1.3020585 0.8022722 -0.230277 0.6057766 0.39497375 -0.82963306 0.4262875
59 [4,] 3.5684978 2.1566413 0.4253989 1.4662884 2.0820458 -2.0877433 -0.7722490 -0.37203788 0.24122872 0.2349325
60 [5,] 0.3537640 -1.0395570 -1.212925 -0.7067451 0.1295306 -0.2097067 0.9304714 0.32979184 -0.37945821 0.2894737
61 [6,] 0.9103847 -0.2151164 -0.3216366 -0.2505572 0.7838295 2.7109786 0.1075647 -0.07595554 -0.36589576 0.5563338
62
63 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
64 [1,] -0.02425662 [2,] 0.16483866 [3,] 0.15960922 [4,] 0.05251293 [5,] -0.03967170 [6,] 0.13855377
65
66 > # To compute the proportion of variance explained (PVE) by each principal component,
67 > # The variance explained by each principal component is obtained by squaring values:
68 > # simply divide the variance explained by each principal component by the total
69 > # variance explained by all four principal components:
70
71
72 > PVE<-cumsum(PC$dev^2 / sum(PC$dev^2))
73 # Observe the cumulative scores, the cumulative scores above 0.96 (96%) will be accepted
74 round(PVE, 2)
75
76 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10 Comp.11
77 0.31 0.46 0.57 0.66 0.75 0.82 0.89 0.94 0.97 1.00 1.00
78
79 > # Comp9, Comp10, Comp11 have the the cumulative scores above 0.96 (96%)
80 > # will be set as attributes
81
82
83 > # Create a new dataset with
84 > # those PCs with a cumulative score > 96%, as attributes
85 pc.comp9 <- -1*pc.comp[,9] # principal component 9 has a score>96%
86 pc.comp10 <- -1*pc.comp[,10] # principal component 10 has a score>96%
87 pc.comp11 <- -1*pc.comp[,11] # principal component 11 has a score>96%
88 PC_DataFrame <- cbind(pc.comp9, pc.comp10, pc.comp11)
89
90 >
```

Data			
gap.stat	List of 6		
kc_cluster	4578 obs. of 12 variables		
kc2	List of 9		
kc3	List of 9		
kn.out	List of 9		
n.clust	29 obs. of 4 variables		
p2	List of 9		
p3	List of 9		
PC	List of 7		
PC_DataFrame	num [1:4578, 1:3] -0.097 0.6547 0.0298 -0.2412 0.3795 ...		
pc.comp	num [1:4578, 1:11] -2.413 -0.147 -1.595 3.568 0.354 ...		
score	4578 obs. of 12 variables		
u_clean	4578 obs. of 12 variables		
x	4578 obs. of 11 variables		
Values			
avg.all.values	num [1:14] 0.266 0.176 0.156 0.127 0.128 ...		
bss.a2	348.115566808182		
bss.a3	2810.14369486272		
cm.k2	'table' int [1:4, 1:2] 791 826 155 26 682 1318 721 147		
cm.k3	'table' int [1:4, 1:3] 0 0 876 173 1393 0 0 0 2336 ...		
t	tsr		
k.values	int [1:14] 2 3 4 5 6 7 8 9 10 11 ...		
pc.comp10	num [1:4578] -0.252 -0.299 -0.426 -0.235 -0.289 ...		
pc.comp11	num [1:4578] -0.252 -0.299 -0.426 -0.235 -0.289 ...		
pc.comp9	num [1:4578] -0.097 0.6547 0.0298 -0.2412 0.3795 ...		
PVE	Named num [1:11] 0.315 0.458 0.569 0.664 0.752 ...		
quality	num [1:4578] 5 5 5 5 5 5 5 5 ...		
score.k2	0.296115482510876		
score.k3	0.9510390095943		
tsr.a2	1175.60009283239		
tsr.a2	1175.60009283239		
tsr.a3	2954.612721297521		
tsr.a3	2954.612721297521		
wss	num [1:15] 1176 827 241 690 654		
wss.a2	827 487125952109		
wss.a3	344.460018112488		
Functions			
Average_Silhouette	Function (x)		
min_max	Function (x)		
outliers	Function (x)		
remove_outliers	Function (df, cols = names(df))		

7. Apply kmeans on this new “PCA-based” dataset

```
349 # 7- Apply kmeans on this new "PCA-based" dataset
350 # set the k=3 as the previous Winner model
351 PCs <- kmeans(PC_DataFrame,3, nstart = 100)
352 PCs$cluster
353 plot(PC_DataFrame, col=PCs$cluster)
354 points(PCs$centers, pch=20)
355
356 # According to the plot the PCs clustering is similar to the one
357 # with k=3, meaning that the pca succeeded to capture the right clustered data.
358
359
```



8. Discuss the performance for this “PCA-based” dataset though the calculation of WSS, BSS, BSS/TSS indices (3 marks) and compare them against the ones produced from the previous “winner” model utilising all attributes (3 marks).

(READ COMMENTS BELOW OR IN ATTACHED CODE FOR DISCUSSION)

```

361 # R - Discuss the performance for this "PCA-based" dataset though the
362 # calculation of WSS, BSS, BSS/TSS indices (3 marks)
363 # and compare them against the ones produced from the previous "winner" model utilising all
364 # attributes (3 marks).
365 # CxCluster
366 PCs$centers
367 PCs$totals
368 PCs$withinss
369 PCs$betweenss
370 # Clusters size (not a
371 PCs$size
372 # According to the clustering size the results between the k=3, and PCs
373 # are closely similar
374
375 # Cluster
376 table(PCs$cluster)
377 # Also the Cluster table are similar to the one with k=3
378
379 wss_PCs = PCs$total.withinss
380 bss_PCs = PCs$betweenss
381
382 on_PCs = table(quality, PCs$cluster)
383 on_PCs
384
385 wss_PCs = PCs$total.withinss
386 wss_PCs
387 bss_PCs = PCs$betweenss
388 bss_PCs
389 tss_PCs = wss_PCs + bss_PCs
390 # The quality of the partition defined by the following indice
391 score_PCs = bss_PCs / tss_PCs
392 score_PCs
393
394 # According to the quality of the partition there is a drop in the
395 # proportion from 95.11 to 53.59, however, both models are significantly similar
396
397
398 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
[96] 2 3 1 3 3 2 1 2 1 1 3 3 2 3 3 3 3 3 3 2 1 3 1 1 2 1 1 3 2 3 3 1 1 2 1 1 3 2 1 1 1 1
[ reached getOption("max.print") -- omitted 3578 entries ]
> PCs$centers
  pc.comp1 pc.comp10 pc.comp11
1 0.3785777 -0.46931226 -0.46931226
2 -0.6146123 -0.06018884 -0.06018884
3 0.2528624 0.52285960 0.52285960
> PCs$totals
[1] 4431.828
> PCs$withinss
[1] 660.1820 573.1604 823.1237
> PCs$betweenss
[1] 2374.642
> # Clusters size indice
> PCs$size
[1] 1540 1485 1553
> # According to the clustering size the results between the k=3, and PCs
> # are closely similar
>
> # Cluster
> table(PCs$cluster)
  1 2 3
1540 1485 1553
> # Also the Cluster table are similar to the one with k=3
>
> wss_PCs = PCs$total.withinss
> bss_PCs = PCs$betweenss
>
> on_PCs = table(quality, PCs$cluster)
> on_PCs
  quality 1 2 3
1 5 573 122 498
2 6 693 708 735
3 7 235 367 274
8 8 39 88 46
>
> wss_PCs = PCs$total.withinss
> wss_PCs
[1] 2056.386
> bss_PCs = PCs$betweenss
> bss_PCs
[1] 2374.642
> tss_PCs = wss_PCs + bss_PCs
> score_PCs = bss_PCs / tss_PCs
> score_PCs
[1] 0.5359122
>
> # According to the the quality of the partition there is a drop in the
> # proportion from 95.11 to 53.59, however, both models are significantly similar

```

```

> Global Environment:
> kcc_cluster
4578 obs. of 12 variables
> kcc2
List of 9
> kcc3
List of 9
> kn_out
List of 9
> kn_cust
29 obs. of 4 variables
> pc2
List of 9
> pc3
List of 9
> PC
List of 7
> PC_dataframe
num [1:4578, 1:3] -0.097 0.6547 0.8298 -0.2412 0.3795 ...
> comp
num [1:4578, 1:11] -2.413 -0.147 -1.595 3.568 0.394 ...
> PCs
num [1:4578, 1:11] -2.413 -0.147 -1.595 3.568 0.394 ...
> R
4718 obs. of 12 variables
> R_cust
4578 obs. of 12 variables
> R_kn
4578 obs. of 13 variables
Values
avg_sil_values
num [1:34] 0.266 0.176 0.156 0.127 0.128 ...
tsr.tss2
tbl[,1]5564080272
tsr.tss3
tbl[,1]281434068272
tsr.tsc1
2374.64260314066
cm.tss2
"table" int [1,4, 1:2] 791 826 155 26 682 1310 723 147
cm.tss3
"table" int [1,4, 1:3] 876 173 1200 0 0 0 2356 ...
cm.tsc1
"table" int [1,4, 1:3] 753 635 239 332 708 367 88 698 735 ...
i
15L
k.values
int [1:34] 2 3 4 5 6 7 8 9 10 11 ...
pc.compl1
tbl[,1]45780.252 -0.299 -0.426 -0.235 -0.289 ...
pc.compl11
tbl[,1]45780.252 -0.299 -0.426 -0.235 -0.289 ...
pc.compl111
tbl[,1]45780.252 -0.299 -0.426 -0.235 -0.289 ...
PVE
tbl[,1]45780.252 -0.299 -0.426 -0.235 -0.289 ...
quality
num [1:4578] 5 5 5 5 5 5 5 5 5 5 ...
score_x2
0.29615482508078
score_x3
0.0151900059343
score_PCs
0.53591218605532
tsr.tss2
1175.6800208329
tsr.tss2.1
1175.6800208329
tsr.tss3
2994.6171297501
tsr.tss3.1
2994.6171297501
tsr.tsc1
6411.8202038267
ws.tss2
num [1:15] 3176 827 733 609 654 ...
ws.tss2.1
817.481259526209
ws.tss3
144.40098112448
ws.tss3.1
2861.3861385461
Functions
Average_Silhouette
function() x
min_max
function() x
outliers
function() x
remove_outliers
function() x, cols = names(df)

```

Question 2

1. Brief discussion of the various methods used for defining the input vector in electricity load forecasting problems

The process of defining the input vector is used to identify the most useful inputs from a larger set of input variables/data.

There are various methods to do this:

- The Autoregressive (AR) method defines the forecasted electric load, in terms of the previous loads and a random noise signal.
- The Moving-Average (MA) method defines the forecasted electric load in terms of the current and previous random noise signals. The noise series is constructed from the previous forecast errors
- The Autoregressive Moving-Average (ARMA) method defines the forecasted load combining the AR and MA methods.

Reference

Taylor, Eric Lynn, "Short-term Electrical Load Forecasting for an Institutional/Industrial Power System Using an Artificial Neural Network. " Master's Thesis, University of Tennessee, 2013.

https://trace.tennessee.edu/utk_gradthes/2468

2. Evidence of various adopted input vectors and the related input/output matrices for both AR (5 marks) and NARX (4 marks) based approaches

3. Evidence of correct normalisation (3 marks) and brief discussion of its necessity (3 marks)

Data is normalised so that the model to be more specific. It is only required when features have different ranges. For example, consider a data set with 2 features – age (0-100) and Savings (0-10000000). The Savings range is far greater than the age range thus we need to normalize and transform the data to the range of [0, 1]. In this case Normalisation is a necessity as the data follows a non-normal distribution, so then normalization to a 0-1 range is appropriate.

```
12 #.....Function Declaration for use
13 #..norm func
14 ▾ normalize<- function(x) {
15     return((x - min(x)) / (max(x) - min(x)))
16 ▴ }
17 #..UNnorm FUNC
18 ▾ unnormalize <- function(x, min, max) {
19     return( (max - min)*x + min )
20 ▴ }

67
68 #.....Normalizing
69 DS_norm<- as.data.frame(lapply(DS_clean, normalize))
70 summary(DS_norm)
71 str(DS_norm)
72 boxplot(DS_norm)
--
```

4. Discussion of the meaning of these stat. indices

- errors measures indices such as above mentioned are used for evincing the degree of spreading of data.
 - RMSE = Root Mean Squared Error. This gives more weight to the highest outliers, which means it is more sensitive to outliers.
 - MAE = refers to Mean Absolute Error. This gives less weight to outliers, which means it is more robust to outliers.
 - MAPE = refers to Mean Absolute Percentage Error. Similar to MAE, but normalized by true observation.
- towardsdatascience. 2019. *Forecast KPI: RMSE, MAE, MAPE & Bias*. [online] Available at: <<https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d>> [Accessed 28 April 2022].

CODE –

Q1 code – (Ends Page 21)

```
##### W1775197 - JOHN-ALEX YANNOULIAS #####
```

```
library(factoextra)
```

```
library(ggplot2)
```

```
library(cluster)
```

```
library(NbClust)
```

```
set.seed(123)# the k-means algorithm uses a random set of initial points to
```

```
# arrive at the final classification
```

```
library(readxl)
```

```
#ENTER YOUR PATHNAME BELOW WHEN MARKING
```

```
W<- read_excel("/Users/john-alexynnoulias/Documents/University/YEAR 2/YEAR 2 SEM 2/Machine  
Learning/COURSEWORK/Whitewine_v2.xlsx")
```

```
W<-as.data.frame(W)
```

```
names(W)
```

```
head(W)
```

```
summary(W)
```

```
str(W)
```

```
# 1- Pre-processing tasks (2 marks for scaling and 3 marks for outliers removal)
```

```
#.....Checking for missing values
```

```
sum(is.na(W))
```

```
#.....Function to find outliers
```

```
outliers <- function(x) {
```

```
  Q1 <- quantile(x, probs=.10)
```

```
  Q3 <- quantile(x, probs=.90)
```

```
  iqr = Q3-Q1
```

```
  upper_limit = Q3 + (iqr*1.5)
```

```
  lower_limit = Q1 - (iqr*1.5)
```

```
  x > upper_limit | x < lower_limit
```

```
}
```

```
#.....Function to remove outliers
```

```
remove_outliers <- function(df, cols = names(df)) {  
  for (col in cols) {  
    df <- df[!outliers(df[[col]]),]  
  }  
  df  
}
```

```
#.....Function for scale
```

```
min_max <- function(x) {  
  return ((x - min(x)) / (max(x) - min(x)))  
}
```

```
# Removing outliers from all columns except the last (quality)
```

```
W_clean <- remove_outliers(W[1:12])
```

```
#.....Splitting the data
```

```
quality <- W_clean$quality
```

```
x <- W_clean[,-12]
```

```
#.....Scaling the data
```

```
x <- as.data.frame(lapply(x, min_max))
```

```
View(x)
```

```
# 2- Define the number of cluster centres by showing all necessary
```

```
# steps/methods via manual & automated tools
```

```
#.....Elbow Method.....
```

```
#.....Manual Method.....
```

```
wss <- 0
```

```
# For 1 to 15 cluster centers
```

```
for (i in 1:15) {
```

```

km.out <- kmeans(x, centers = i)
wss[i] <- km.out$tot.withinss
}

wss

plot(1:15, wss, type = "b", xlab = "Number of Clusters",
     ylab = "Within groups sum of squares")

#.....Automated Method

# For the automated Elbow method, no needs to program the Elbow Method, has
# been implemented already in R.(wss= within-cluster sum of square)
fviz_nbclust(x, kmeans, method = "wss")

#.....Average Silhouette.....
#.....Manual Method.....

# function to compute average silhouette for k clusters
Average_Silhouette <- function(k) {
  km.res <- kmeans(x, centers = k, nstart = 100)
  ss <- silhouette(km.res$cluster, dist(x))
  mean(ss[, 3])
}

# Compute and plot wss for k = 2 to k = 15
k.values <- 2:15

# extract avg silhouette for 2-15 clusters
library(purrr)
avg_sil_values <- map_dbl(k.values, Average_Silhouette)
# plot
plot(k.values, avg_sil_values,
     type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Average Silhouettes")

#.....Automated Method

fviz_nbclust(x, kmeans, method = 'silhouette')+
  labs(subtitle = "Silhouette method") #Seems to suggest 2
# which suggests 2 clusters as the optimal number of clusters.

```

```

#.....Gap-Statistic Method.....
#.....Manual Method.....
# To compute the gap statistic method we can use the clusGap function
# which provides the gap statistic and standard error for an output

# compute Gap-statistic
set.seed(123)# the k-means algorithm uses a random set of initial points to
# arrive at the final classification
gap_stat <- clusGap(x, FUN = kmeans, nstart = 100,
                    K.max = 10, B = 50)
# Print the result
print(gap_stat, method = "firstmax")
# visualize the results with fviz_gap_stat
fviz_gap_stat(gap_stat)
# The Manual Method suggests 3 clusters as the optimal number of clusters.
#.....Automated Method
fviz_nbclust(x, kmeans, method = 'gap_stat')+
  labs(subtitle = "Gap-Statistics") #Seems to suggest 3
# The Automated Method also suggests 3 clusters as the optimal number of clusters.

# .....Consensus-based algorithm
# Because no method is clearly better, a fourth alternative is to run many methods
# and take the number of clusters that is the most agreed upon
library(parameters)
n_clust <- n_clusters(x,
                     package = c("easystats", "NbClust", "mclust"),
                     nbclust_method = "kmeans",n_max=5,standardize = FALSE)
n_clust
# The optimal number of clusters to retain can also be visualized:
plot(n_clust)
# The winning Model is the one with k=3, based on the previous analysis
#####

```

#####

3- K-means analysis for each k attempt (2 marks for each k attempt)

#Notes:

Because each cluster becomes smaller and tighter, the overall WSS

decreases as the number of clusters grows. The hope is that once k exceeds

the ideal number of clusters, the rate at which the WSS drops would slow down.

.....For K=2

The "nstart" argument in the kmeans() function allows to run the algorithm

several times with different initial centers, in order to obtain a

potentially better partition

kc2 <- kmeans(x,2, nstart = 100)

displays the class determined by the model for all observations:

kc2\$cluster # The output kc2\$cluster specifies the group (k=2)

to which each observation belongs to.

#The cluster for each observation can be stored directly in the dataset

as a column:

kc_cluster <- data.frame(x,cluster = as.factor(kc2\$cluster))

head(x)

kc2\$centers

how many Observations are in each Cluster

kc2\$size

wss_k2 = kc2\$tot.withinss

wss_k2

bss_k2 = kc2\$betweenss # the between sum of squares BSS(k)

bss_k2

Where the $BSS(k) = TSS - WSS(k)$

```

tss_k2 = wss_k2 + bss_k2 # the total sum of squares
score_k2 = bss_k2 / tss_k2
score_k2
# CM table
cm_k2 = table(quality, kc2$cluster)
cm_k2

###.....For K3
kc3 <- kmeans(quality,3, nstart = 100)
kc3
kc3$centers
# how many Observations are in each Cluster
kc3$size # Generally, a good clustering will be well-balanced
# CM table
cm_k3 = table(quality, kc3$cluster)
cm_k3
wss_k3 = kc3$tot.withinss
wss_k3
bss_k3 = kc3$betweenss
bss_k3
tss_k3 = wss_k3 + bss_k3
score_k3 = bss_k3 / tss_k3
score_k3
# The winning Model is the one with k=3, based on the previous indices
#####

# 4-Evaluation of the produced outputs against 12th column
# a- (1 mark for the "CM" table)
# Evaluation of the produced outputs against 12th column (quality variable)

```

```
# Confusion Matrix "CM" table
```

```
cm_k2 = table(quality, kc2$cluster)
```

```
cm_k2
```

```
cm_k3 = table(quality, kc3$cluster)
```

```
cm_k3
```

```
# The quality of a k-means partition is found by calculating the percentage
```

```
# of the TSS "explained" by the partition using the following formula:
```

```
#      (BSS/TSS)*100
```

```
# The higher the percentage, the better the score (and thus the quality)
```

```
# because it means that BSS is large and/or WSS is small.
```

```
# BSS and TSS are extracted from the model and stored
```

```
bss_k2 = kc2$betweenss # the between sum of squares BSS(k)
```

```
bss_k2
```

```
TSS_k2 <- kc2$totss
```

```
TSS_k2
```

```
# calculate the quality of the partition
```

```
bss_k2 / TSS_k2 * 100
```

```
# The quality of the partition is 29.61%. This value has no real
```

```
# interpretation in absolute terms except that a higher quality means a higher
```

```
# explained percentage.
```

```
bss_k3 = kc3$betweenss # the between sum of squares BSS(k)
```

```
bss_k3
```

```
TSS_k3 <- kc3$totss
```

```
TSS_k3
```

```
# calculate the quality of the partition
```

```
bss_k3 / TSS_k3 * 100
```

```
# The quality of the partition is 95.11%. This value has no real
```

```
# interpretation in absolute terms except that a higher quality means a higher
```

```
# explained percentage.
```

#Conclusion: As can be seen, classifying the data into three sections provides for a larger explained
percentage and improved quality.
We conclude that the K=3 is the optimal clustering strategy

b- (6 marks for calculation of these requested indices)

```
library("cluster")
```

#Structure of each K

```
str(kc2)
```

b- (6 marks for calculation of these requested indices)

k=2

```
kc2 <- kmeans(x,2, nstart = 50)
```

The output of kmeans is a list with several bits of information

kc2\$cluster # indicating the cluster to which each point is allocated.

kc2\$centers # A matrix of cluster centers.

kc2\$totss # The total sum of squares.

kc2\$withinss # Vector of within-cluster sum of squares, one component per cluster.

kc2\$tot.withinss

kc2\$betweenss # The between-cluster sum of squares, i.e. \$totss-tot.withinss\$.

kc2\$size # The number of points in each cluster.

```
table(kc2$cluster) #Give a count of data points in each cluster
```

```
tss_k2 = wss_k2 + bss_k2
```

```
score_k2 = bss_k2 / tss_k2
```

```
score_k2
```

k=3

```
kc3 <- kmeans(x,3, nstart = 50)
```

```
kc3$cluster
```

```
kc3$centers
```

```
kc3$totss
```

```
kc3$withinss
kc3$betweenss
kc3$size
table(kc3$cluster)
tss_k3 = wss_k3 + bss_k3
score_k3 = bss_k3 / tss_k3
score_k3
```

```
# Model Evaluation using the visualization method
# How the clustering comparing to the real data observations
# K=2: Plot 1
```

```
fviz_cluster(list(data = x, cluster = kc2$cluster),
              ellipse.type = "norm", geom = "point", stand = FALSE,
              palette = "jco", ggtheme = theme_classic())
```

```
# k=3: Plot 2.
```

```
fviz_cluster(list(data = x, cluster = kc3$cluster),
              ellipse.type = "norm", geom = "point", stand = FALSE,
              palette = "jco", ggtheme = theme_classic())
```

```
# Plot 3: comparing to the real data observations (quality variable)
```

```
fviz_cluster(list(data = x, cluster = quality),
              ellipse.type = "norm", geom = "point", stand = FALSE,
              palette = "jco", ggtheme = theme_classic())
```

```
# By comparing plots 1 and 2 with the plot 3, we see clearly that
# the k=3 is the optimal for clustering strategy.
```

```
# 4- Define the final "winner" cluster case and provide brief
# explanation of evaluation indices
# plots to compare
p2 <- fviz_cluster(kc2, geom = "point", data = x) + ggtitle("k = 2")
p3 <- fviz_cluster(kc3, geom = "point", data = x) + ggtitle("k = 3")
library(gridExtra)
grid.arrange(p2, p3, nrow = 2)

# Although this visual assessment tells us where true dilineations occur
# (or do not occur) between clusters.
# However, it does not tell us what the optimal number of clusters is.
```

```
# 6- Apply a PCA for this white wine dataset (2 marks). Create a new
# dataset with those PCs with a cumulative score > 96%, as attributes (2 marks).
```

```
PC<-princomp(x, cor=T)
```

```
pc.comp <- PC$scores
```

```
head(PC$scores)
```

```
# To compute the proportion of variance explained (PVE) by each principal component,
# The variance explained by each principal component is obtained by squaring values:
# simply divide the variance explained by each principal component by the total
# variance explained by all four principal components:
```

```
PVE<-cumsum(PC$sdev^2 / sum(PC$sdev^2))
```

```
# Observe the cumulative scores, the cumulative scores above 0.96 (96%) will be accepted
```

```
round(PVE, 2)
```

```
# Comp9, Comp10, Comp11 have the the cumulative scores above 0.96 (96%)
```

will be set as attributes

Create a new dataset with

those PCs with a cumulative score > 96%, as attributes

```
pc.comp9 <- -1*pc.comp[,9] # principal component 9 has a score>96%
```

```
pc.comp10 <- -1*pc.comp[,10] # principal component 10 has a score>96%
```

```
pc.comp11 <- -1*pc.comp[,11] # principal component 11 has a score>96%
```

```
PC_DataFrame <- cbind(pc.comp9, pc.comp10, pc.comp11)
```

7- Apply kmeans on this new "PCA-based" dataset

set the k=3 as the previous Winner model

```
PCs <- kmeans(PC_DataFrame, 3, nstart = 100)
```

```
PCs$cluster
```

```
plot(PC_DataFrame, col=PCs$cluster)
```

```
points(PCs$centers, pch=20)
```

According to the plot the PCs clustering is similar to the one

with k=3, meaning that the pca succeeded to capture the right clustered data.

8- Discuss the performance for this "PCA-based" dataset though the

calculation of WSS, BSS, BSS/TSS indices (3 marks)

and compare them against the ones produced from the previous "winner" model utilising all

attributes (3 marks).

```
PCs$cluster
```

```
PCs$centers
```

```
PCs$totss
```

```
PCs$withinss
```

```
PCs$betweenss
```

Clusters size indice

```
PCs$size
```

```
# According to the clustering size the results between the k=3, and PCs
```

```
# are closely similar
```

```
# Cluster
```

```
table(PCs$cluster)
```

```
# Also the Cluster table are similar to the one with k=3
```

```
wss_PCs = PCs$tot.withinss
```

```
bss_PCs = PCs$betweenss
```

```
cm_PCs = table(quality, PCs$cluster)
```

```
cm_PCs
```

```
wss_PCs = PCs$tot.withinss
```

```
wss_PCs
```

```
bss_PCs = PCs$betweenss
```

```
bss_PCs
```

```
tss_PCs = wss_PCs + bss_PCs
```

```
# The quality of the partition deifined by the following indice
```

```
score_PCs = bss_PCs / tss_PCs
```

```
score_PCs
```

```
# According to the the quality of the partition there is a drop in the
```

```
# proportion from 95.11 to 53.59, however, both models are significantly similar
```

Q2 code – (Ends Page 25)

W1775197 - JOHN-ALEX YANNOULIAS

```
library("ggplot2")
```

```
library("gridExtra")
```

```
library("grid")
```

```
library("MASS")
```

```
library("neuralnet")
```

```
library("MLmetrics")
```

```
library("neuralnet")
```

```
library("readxl")
```

```
library("tidyr")
```

```
#.....Function Declaration for use
```

```
#..norm func
```

```
normalize<- function(x) {  
  return((x - min(x)) / (max(x) - min(x)))  
}
```

```
#..UNnorm FUNC
```

```
unnormalize <- function(x, min, max) {  
  return( (max - min)*x + min )  
}
```

```
#..RMSE func
```

```
RMSEtest<- function(error)  
{  
  dataSquared<- error^2  
  dataMean = mean(dataSquared)  
  dataSqrt = sqrt(dataMean)  
  return(dataSqrt)  
}
```

```
#..MAPE func
```

```
MAPEtest<- function(error, load_test_original_tomorrow)  
{  
  newData<- error/ load_test_original_tomorrow  
  dataAbs<- abs(newData)
```

```

dataMean<- sum(dataAbs) / length(dataAbs)
return(dataMean)
}
#..MAE func
MAEtest<- function(error)
{
  dataAbs<- abs(error)
  dataMean<- mean(dataAbs)
  return(dataMean)
}
##### END OF FUNC #####

#.....Load this data (ENTER YOUR PATHNAME WHEN MARKING)
DS<- read_excel('/Users/john-alexynnoulias/Documents/University/YEAR 2/YEAR 2 SEM 2/Machine
Learning/COURSEWORK/UoW_load.xlsx')

names(DS)
str(DS)
summary(DS)

#.....Checking for missing values
sum(is.na(DS))

#.....Deleting rows with missing values (none in this case)
DS_NMV<- DS %>% drop_na()
head(DS_NMV)

#.....Splitting the 9th and 10th features
DS_clean<- DS_NMV[,c(-1,-2,-3)]
colnames(DS_clean)[1] <- "T"

str(DS_clean)
summary(DS_clean)
boxplot(DS_clean)

```

```

#.....Normalizing
DS_norm<- as.data.frame(lapply(DS_clean, normalize))
summary(DS_norm)
str(DS_norm)
boxplot(DS_norm)

#.....Train And Test ratio split as stated in brief
train<- DS_norm[1:430,]
test<-DS_norm[431:500,]

predict<- test

#.....Neural Network attempts

set.seed(12345)

#.....ATTEMPT 1 - 1 layer , 2 nodes
#...This part is broken I couldn't get the input vector file in the correct time series vector format using the
lag function.
nnplot1 <- neuralnet(T.1.1 ~ T + T.1 + T.2 , data = train, hidden = 2)
plot(nnplot1)
nnplot1_results <- compute(nnplot1, test[1:3])
nnplot1_predicted_output <- nnplot1_results$net.result
nnplot1_cor <- cor(nnplot1_predicted_output, predict)
error1 <- (predict - nnplot1_predicted_output)

#..STATISTICAL INDICES MAE RMSE
MAE1 <- MAEtest(error1)
RMSE1 <- RMSEtest(error1)

set.seed(12345)
nnplot2 <- neuralnet(T.1.1 ~ T + T.1 + T.2 , data = train, hidden = 2, linear.output=FALSE)
plot(nnplot2)
nnplot2_results <- compute(nnplot2, test[1:3])
nnplot2_predicted_output <- nnplot2_results$net.result

```

```
nnplot2_cor <- cor(nnplot2_predicted_output, predict)
error2 <- (predict - nnplot2_predicted_output)

#..STATISTICAL INDICES MAE RMSE
MAE2 <- MAEtest(error2)
RMSE2 <- RMSEtest(error2)

comparison1 <- cbind(nnplot1_cor, nnplot2_cor, MAE1, MAE2, RMSE1, RMSE2)
colnames(comparison1) <- c("M2","M3","MAE1","MAE2","RMSE1", "RMSE2")
```