UOW-logo

Informatics Institute of Technology

Department of Computing

Machine Learning and Data Mining Coursework Report

Module : 5DATA001C.2: Machine Learning and Data Mining

Module Leader : Nipuna Senanayake

Date of submission : 04/05/2023

Student ID : 20212196 / w1902278

Student Name : Thejan Vithanage

Table of Content

[1. PARTITIONING CLUSTERING 1](#_Toc134052143)

[1.1. Pre-processing 1](#_Toc134052144)

[Outliers detection/removal 1](#_Toc134052145)

[Scaling 1](#_Toc134052146)

[1.2. Determining The Optimal Number of Clusters 3](#_Toc134052147)

[NbClust() function 3](#_Toc134052148)

[Elbow Method 4](#_Toc134052149)

[Gap statistic method 5](#_Toc134052150)

[Average silhouette method 5](#_Toc134052151)

[Summary of the optimal result 6](#_Toc134052152)

[1.3. K-means analysis 6](#_Toc134052153)

[1.4. Silhouette plot 7](#_Toc134052154)

[5.1. PCA 8](#_Toc134052155)

[Eigenvectors 8](#_Toc134052156)

[Eigenvalues 10](#_Toc134052157)

[Cumulative score per principal components (PC) 10](#_Toc134052158)

[5.2. Determine the number of clusters. 11](#_Toc134052159)

[NBclust 11](#_Toc134052160)

[Elbow 12](#_Toc134052161)

[Gap Statistics 13](#_Toc134052162)

[Average silhouette method 13](#_Toc134052163)

[Summary of the optimal result 14](#_Toc134052164)

[5.3. K-means analysis for this “PCA”-based dataset 14](#_Toc134052165)

[1.8. Silhouette 15](#_Toc134052166)

[1.9. Calinski-Harabasz index 15](#_Toc134052167)

[6. Multi-layer Neural Network 16](#_Toc134052168)

[2.1. Methods used for defining the input vector in electricity load forecasting 16](#_Toc134052169)

[2.2. Data Preprocessing 16](#_Toc134052170)

[Why do we need to normalize data before using them in an Multi-Layer Perceptron (MLP) structure? 16](#_Toc134052171)

[2.3. Implement a number of MLPs for the “AR” approach. 18](#_Toc134052172)

[2.4. comparison matrix for the “AR” case 30](#_Toc134052173)

[2.5. issue of “efficiency” 31](#_Toc134052174)

[2.6. “NARX” approach 32](#_Toc134052175)

[References 33](#_Toc134052176)

[Appendices 34](#_Toc134052177)

[Partitioning\_clustering 34](#_Toc134052178)

[MLP 37](#_Toc134052179)

[MLP\_test 40](#_Toc134052180)

Table of Figures

[Figure 1: Shapiro-Wilk Test used code. 2](#_Toc134052114)

[Figure 3: Normalized and Non-Normalized data Box plot 2](#_Toc134052115)

[Figure 4: NBclust function result bar chart 3](#_Toc134052116)

[Figure 5: NBClust function result graph 4](#_Toc134052117)

[Figure 6: Elbow Method result graph 4](#_Toc134052118)

[Figure 7: Gap statistic method result graph 5](#_Toc134052119)

[Figure 8: Average silhouette method result graph 5](#_Toc134052120)

[Figure 9: kmean clustering plot( x=Cir, y=Rad.Ra) 7](#_Toc134052121)

[Figure 10: Silhouette plot 8](#_Toc134052122)

[Figure 11: PCA Cumulative value graph 11](#_Toc134052123)

[Figure 12: NBClust result graph (PCA Transform data) 12](#_Toc134052124)

[Figure 13: Elbow method result graph (PCA Transform data) 12](#_Toc134052125)

[Figure 14: Gap Statistic result graph (PCA Transform data) 13](#_Toc134052126)

[Figure 15: Silhouette plot (PCA Transform data) 13](#_Toc134052127)

[Figure 16: Silhouette Plot (PCA Transform data) 15](#_Toc134052128)

[Figure 17: Dataset before and after normalization 17](#_Toc134052129)

[Figure 18: Nural network plot (1hidden layer & all input) 18](#_Toc134052130)

[Figure 19: Nural network plot (1hidden layer, 3 neurons & all input) 19](#_Toc134052131)

[Figure 20: Nural network plot (1hidden layer, 2 neurons & all input) 20](#_Toc134052132)

[Figure 21: Nural network plot (1hidden layer, 5 neurons & t1,t7 input) 21](#_Toc134052133)

[Figure 22: Nural network plot (2hidden layer, 2 neurons & t1,t7 input) 22](#_Toc134052134)

[Figure 23: Nural network plot (1hidden layer, 4 neurons & all input) 23](#_Toc134052135)

[Figure 24: Nural network plot (2hidden layer, 6 neurons & t1,t2 input) 24](#_Toc134052136)

[Figure 25: Nural network plot (2hidden layer, 4 neurons & 2 input) 25](#_Toc134052137)

[Figure 26: Nural network plot (2hidden layer, 10 neurons & all input) 26](#_Toc134052138)

[Figure 27: Nural network plot (2hidden layer, 8 neurons & 2 input) 27](#_Toc134052139)

[Figure 28: Nural network plot (2hidden layer,5 neurons & 3 input) 28](#_Toc134052140)

[Figure 29: Nural network plot (3hidden layer, 9 neurons & all input) 29](#_Toc134052141)

[Figure 30: Nural network plot (2hidden layer, 5 neurons & all input) 30](#_Toc134052142)

Table of Table

[Table 1: Shapiro-Wilk Test Result 2](#_Toc134052098)

[Table 2: Accuracy, rmse, mae, smape score (1hidden layer & all input) 18](#_Toc134052099)

[Table 3: Accuracy, rmse, mae, smape score (1hidden layer, 3 neurons & all input) 19](#_Toc134052100)

[Table 4: Accuracy, rmse, mae, smape score (1hidden layer, 2 neurons & all input) 20](#_Toc134052101)

[Table 5: Accuracy, rmse, mae, smape score (1hidden layer, 5 neurons & t1,t7 input) 21](#_Toc134052102)

[Table 6: Accuracy, rmse, mae, smape score (1hidden layer, 2 neurons & t1,t7 input) 22](#_Toc134052103)

[Table 7: Accuracy, rmse, mae, smape score (1hidden layer, 4 neurons & all input) 23](#_Toc134052104)

[Table 8: Accuracy, rmse, mae, smape score (2hidden layer, 6 neurons & t1,t2 input) 24](#_Toc134052105)

[Table 9: Accuracy, rmse, mae, smape score (2hidden layer, 4 neurons & t1,t2 input) 25](#_Toc134052106)

[Table 10: Accuracy, rmse, mae, smape score (2hidden layer, 10 neurons & all input) 26](#_Toc134052107)

[Table 11: Accuracy, rmse, mae, smape score (2hidden layer, 8 neurons & t2,t3 input) 27](#_Toc134052108)

[Table 12: Accuracy, rmse, mae, smape score (2hidden layer, 5 neurons & t1,t4,t7 input) 28](#_Toc134052109)

[Table 13: Accuracy, rmse, mae, smape score(3hidden layer, 9 neurons & all input) 29](#_Toc134052110)

[Table 14: Accuracy, rmse, mae, smape score (2hidden layer, 5 neurons & all input) 30](#_Toc134052111)

[Table 15: RMSE, MAE, MAPE and sMAPE – symmetric MAPE Score 31](#_Toc134052112)

[Table 16: efficiency of models 31](#_Toc134052113)

# PARTITIONING CLUSTERING

## Pre-processing

Data pre-processing is used to prepare raw data for use in machine learning (ML) and data mining (DM). Because in ML and DM performance has an impact on the input data quality

### Outliers detection/removal

In data points there are significant different data points the can be a impact on a performance of the learning model therefore it need to detect and remove.

For outlier detection and removal, I chose boxplot method it gives an outlier from boxplot function easily. And it can be used in non-normal distribution data sets. After removing outliers and normalizing data then found another outlier.



### Scaling

Scaling is used to transform the value of features to a common scale, in min max normalization its normally 0 to 1, This can improve performance in some algorithms.

For the scaling I used min-max normalization because for z-score normalization dataset need to be normal distribution data set, but this vehicle data set not a normal distributed dataset I check it by using Shapiro-Wilk Test (Gardener, 2012) therefore used min-max normalization.

Text

Description automatically generated

Figure 1: Shapiro-Wilk Test used code.

Table 1: Shapiro-Wilk Test Result



But one of the drawbacks of the min max normalization is its respond to outliers but after removing outliers it is not an issue.



Figure 2: Normalized and Non-Normalized data Box plot

## Determining The Optimal Number of Clusters

### NbClust() function

\*\*\* : The Hubert index is a graphical method of determining the number of clusters.

In the plot of Hubert index, we seek a significant knee that corresponds to a

significant increase of the value of the measure i.e the significant peak in Hubert

index second differences plot.

\*\*\* : The D index is a graphical method of determining the number of clusters.

In the plot of D index, we seek a significant knee (the significant peak in Dindex

second differences plot) that corresponds to a significant increase of the value of

the measure.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Among all indices:

\* 6 proposed 2 as the best number of clusters

\* 13 proposed 3 as the best number of clusters

\* 1 proposed 4 as the best number of clusters

\* 1 proposed 7 as the best number of clusters

\* 3 proposed 10 as the best number of clusters

\*\*\*\*\* Conclusion \*\*\*\*\*

\* According to the majority rule, the best number of clusters is 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Figure 3: NBclust function result bar chart



Figure 4: NBClust function result graph

Using NbClust function it suggests that 3 is the most favored from this function.

### Elbow Method

To identify the optimal number of clusters elbow method plot number of clusters against Total within sum of square. In this plot where the total within sum of square reduction slow down and flattened significantly that is the suggestion k number of elbow method.



Figure 5: Elbow Method result graph

In this chart it reduces slows significantly at k = 3, therefore elbow method suggests that optimal number of clusters 3

### Gap statistic method



Figure 6: Gap statistic method result graph

In gap statics result its suggest k = 3

### Average silhouette method



Figure 7: Average silhouette method result graph

In average silhouette method its suggest k = 2.

### Summary of the optimal result

Using automated optimal clustering methods, NbClust function, Elbow method and Gap statistic suggest 3 as a optimal k and Silhouette suggest 2.

## K-means analysis

between\_cluster\_sums\_of\_squares (BSS) = 378.6251

total\_sum\_of\_Squares (TSS)= 314.8862

between\_SS / total\_SS = 54.6 %

K-means clustering with 3 clusters of sizes 233, 253, 326

Cluster means:

Comp Circ D.Circ Rad.Ra Pr.Axis.Ra Max.L.Ra

1 0.3026250 0.3275008 0.3879351 0.1957323 0.3442356 0.4047210

2 0.6973987 0.7365461 0.8500769 0.6850749 0.5323702 0.6581028

3 0.4363675 0.3292827 0.5213872 0.4503154 0.5603977 0.4754601

Scat.Ra Elong Pr.Axis.Rect Max.L.Rect Sc.Var.Maxis

1 0.2061516 0.6202330 0.1498244 0.3248314 0.2115771

2 0.6661397 0.1550536 0.6316924 0.6577075 0.6098964

3 0.2797342 0.4957932 0.2136085 0.3238826 0.2893531

Sc.Var.maxis Ra.Gyr Skew.Maxis Skew.maxis Kurt.maxis

1 0.1461365 0.3283397 0.6865420 0.3035916 0.2630131

2 0.6056291 0.6343749 0.4583569 0.3598918 0.3864396

3 0.2187884 0.2980090 0.3132121 0.3157895 0.3156363

Kurt.Maxis Holl.Ra

1 0.2306867 0.2130186

2 0.4590627 0.5291173

3 0.6344216 0.6565440

Clustering vector:

[1] 3 3 2 3 2 3 3 3 3 3 3 3 3 3 2 1 3 2 2 1 1 3 3 2 3 1 2 2 1 3

[31] 3 3 2 3 3 1 2 1 2 1 1 3 1 1 1 3 1 3 2 3 2 3 3 1 2 1 2 1 1 1

[61] 3 1 1 2 3 2 2 2 3 1 3 2 3 1 2 1 2 3 1 3 3 1 3 1 2 3 2 3 1 2

[91] 1 1 2 1 3 3 1 2 2 2 1 1 3 3 3 1 1 3 2 2 1 3 1 1 3 3 1 1 3 2

[121] 2 3 1 2 1 3 1 3 3 1 2 1 3 2 3 3 3 3 2 3 3 2 3 2 3 1 3 3 1 2

[151] 3 3 2 2 3 2 1 1 2 2 3 2 3 3 3 3 3 1 2 1 3 1 2 3 3 3 2 3 3 3

[181] 2 3 1 2 1 1 1 3 3 2 2 3 3 3 1 1 2 3 3 3 2 1 3 1 2 1 3 2 1 2

[211] 1 1 3 2 3 2 1 1 1 2 3 1 3 1 2 1 3 3 1 2 1 1 3 3 2 1 1 2 1 3

[241] 3 2 3 3 2 2 1 3 3 3 2 1 1 3 3 1 1 3 3 3 2 3 1 1 2 3 3 1 1 2

[271] 1 3 3 1 2 1 1 3 3 2 3 2 1 3 3 2 3 3 3 1 3 2 2 2 2 2 1 3 2 1

[301] 1 1 3 1 2 2 2 1 2 3 1 2 1 3 3 3 2 2 1 2 2 1 2 3 3 3 1 1 2 2

[331] 2 3 3 3 2 1 3 1 3 3 3 2 3 2 2 2 3 3 1 2 3 1 1 3 3 3 3 3 1 2

[361] 2 1 1 2 1 1 2 3 3 3 3 2 1 3 3 3 3 2 3 3 3 3 2 3 2 3 1 1 3 3

[391] 3 1 1 3 1 2 3 3 1 3 1 2 3 1 3 3 2 3 2 3 2 2 1 1 2 3 1 1 3 2

[421] 2 1 3 2 2 1 2 2 2 3 3 3 3 3 2 1 1 3 2 3 3 2 3 1 2 1 1 2 2 3

[451] 1 2 2 2 1 2 2 3 3 1 2 2 3 3 1 1 2 3 1 2 2 3 1 2 2 3 2 1 2 2

[481] 2 1 1 2 2 3 3 2 1 3 2 1 1 2 1 3 3 1 3 2 3 2 2 3 1 3 2 2 1 1

[511] 3 2 3 2 2 3 3 3 3 1 1 3 3 2 1 1 3 1 2 3 2 1 1 2 2 3 2 3 3 3

[541] 2 3 1 3 2 3 3 1 2 2 2 2 3 1 1 1 2 2 2 3 2 1 3 2 1 1 1 3 1 3

[571] 3 3 3 3 3 3 2 3 3 2 3 3 3 1 2 1 1 3 1 3 3 1 1 2 2 1 3 1 2 3

[601] 3 2 3 1 2 1 2 1 1 3 1 3 2 2 3 2 3 3 1 3 1 2 3 2 1 3 3 3 1 1

[631] 3 2 3 2 1 3 3 3 3 2 3 1 2 3 2 3 3 2 1 2 1 3 3 3 1 2 3 1 3 2

[661] 1 2 3 3 2 1 3 1 3 3 1 3 2 2 3 3 2 2 3 1 3 2 2 2 2 3 2 3 3 2

[691] 2 3 2 3 2 3 1 2 3 1 2 2 2 3 2 1 1 2 2 2 3 2 3 3 2 3 1 3 1 3

[721] 2 3 1 3 3 3 1 2 1 1 1 2 1 2 2 1 3 3 2 3 1 2 2 1 3 3 2 2 2 1

[751] 2 3 2 2 1 1 2 1 2 3 1 3 2 2 3 1 2 3 3 1 3 3 2 1 3 2 1 1 2 1

[781] 3 1 1 3 2 2 3 1 2 3 2 2 1 3 2 1 1 3 3 2 1 1 1 3 3 3 3 3 3 2

[811] 3 1

Within cluster sum of squares by cluster:

[1] 77.9210 108.6242 128.3410

(between\_SS / total\_SS = 54.6 %)

Available components:

[1] "cluster" "centers" "totss" "withinss"

[5] "tot.withinss" "betweenss" "size" "iter"

[9] "ifault"



Figure 8: kmean clustering plot( x=Cir, y=Rad.Ra)

## Silhouette plot

1. cluster size ave.sil.width
2. 1 1 253 0.38
3. 2 2 233 0.31
4. 3 3 326 0.22

In silhouet method its average value range -1 to 1 if its 1 then cluster points vary simlir to own cluster -1 its opposite. In this result value its 0.29. Then it’s a cosiderabaly moderate to good cluster.

Silhouette of 812 units in 3 clusters from silhouette.default

(x = kmeans\_fit$cluster, dist = dist(normalized\_df)) :

Cluster sizes and average silhouette widths:

326 253 233

0.2185616 0.3776430 0.3051266

Individual silhouette widths:

Min. 1st Qu. Median Mean 3rd Qu. Max.

-0.04719 0.18871 0.31144 0.29297 0.41134 0.54793



Figure 9: Silhouette plot

## PCA



### Eigenvectors

> (phi <- vehicle.eigen$vectors)

[,1] [,2] [,3] [,4]

[1,] -0.23624665 -0.07363184 -0.06178410 -0.07375582

[2,] -0.31542815 0.14701255 -0.15377106 0.13105371

[3,] -0.30343195 -0.04445283 0.03935021 -0.09735182

[4,] -0.28321380 -0.19225397 0.08378814 0.15751956

[5,] -0.08980674 -0.20783258 0.04417990 0.40433417

[6,] -0.18493242 -0.08778341 -0.08901054 -0.08283361

[7,] -0.31608426 0.07269880 0.08118065 -0.04326184

[8,] 0.31637030 -0.01502041 -0.08498945 -0.02418334

[9,] -0.33259929 0.09024898 0.07815013 -0.06672657

[10,] -0.26515775 0.11524431 -0.14490752 0.07861075

[11,] -0.27036471 0.06307474 0.09733397 -0.01552691

[12,] -0.31048919 0.07622059 0.07995391 -0.04258020

[13,] -0.24687140 0.19042583 -0.14688582 0.12141565

[14,] 0.04767381 0.50194678 0.06059384 0.01059297

[15,] -0.04280132 -0.02281243 -0.81821562 -0.43341312

[16,] -0.06399448 -0.09591614 0.44473348 -0.74154445

[17,] -0.03608515 -0.49019439 -0.05897039 0.06977418

[18,] -0.09626900 -0.55513911 -0.03631322 -0.01214535

[,5] [,6] [,7] [,8]

[1,] -0.05254921 0.259637485 0.10279334 0.7535298986

[2,] 0.15922746 -0.096664210 -0.42335849 0.0158912733

[3,] 0.10283293 -0.001563963 0.21978435 -0.3052006421

[4,] -0.25747743 -0.192074499 0.17350590 -0.0338941786

[5,] -0.33354761 -0.668620543 0.07702210 0.1544997125

[6,] 0.66770983 -0.298114022 0.41401073 0.1283778683

[7,] -0.08839933 0.119474889 0.10034243 -0.0495882886

[8,] 0.11775123 -0.081292660 -0.13378256 0.1886547694

[9,] -0.07407334 0.142703538 0.09859514 -0.0004919574

[10,] 0.28272256 -0.100885166 -0.35013963 0.2106814391

[11,] -0.15843021 0.143995849 0.08147069 -0.1034792619

[12,] -0.11679801 0.157310396 0.06687641 -0.0054459687

[13,] 0.01785080 -0.094736094 -0.42966332 -0.2377131820

[14,] -0.19011978 -0.120490190 0.01917497 0.3537124618

[15,] -0.30418789 -0.149607205 0.11600908 -0.0624918130

[16,] -0.08423851 -0.359135826 -0.29403135 0.0659982047

[17,] -0.17310278 0.276944275 -0.26194756 0.1191672754

[18,] 0.16133742 0.044338665 -0.17190696 0.0060691552

[,9] [,10] [,11] [,12]

[1,] -0.437683995 -0.196523806 0.009705193 0.024401191

[2,] -0.081213882 0.195505630 0.065119368 -0.048232195

[3,] -0.240328077 -0.184111396 0.744445020 0.259255587

[4,] -0.188178449 0.069223502 -0.132848486 -0.200719863

[5,] 0.003174262 0.041415414 0.059323118 0.119397878

[6,] 0.184654565 -0.202100189 -0.194395173 -0.229643926

[7,] 0.146263284 0.116784591 -0.134867072 0.055655987

[8,] -0.309340881 0.122380102 -0.027654704 0.356466723

[9,] 0.095020283 0.253296857 -0.234387449 0.284280180

[10,] 0.149120445 0.398631123 0.279857255 -0.109780033

[11,] 0.063496698 -0.136385203 0.022367551 -0.161179091

[12,] 0.091589660 0.176688961 -0.187867592 0.182151142

[13,] -0.228480534 -0.605672084 -0.309638115 0.033385013

[14,] 0.517270295 -0.348766345 0.207111233 0.185529475

[15,] 0.065052561 0.051266792 -0.010240594 0.002041453

[16,] -0.040567367 -0.007672065 -0.011846600 -0.095106746

[17,] 0.302650361 -0.164690356 0.208405451 -0.386253691

[18,] 0.310714082 -0.148724957 -0.109793333 0.588741269

[,13] [,14] [,15] [,16]

[1,] -0.198605465 -0.102739808 -0.007299427 -0.004711970

[2,] -0.126959904 0.221139486 -0.535129713 0.431888371

[3,] 0.113765357 -0.002421975 -0.092062508 -0.079246476

[4,] 0.037038972 0.738466464 0.096922799 -0.248116591

[5,] 0.034996620 -0.400364548 -0.022666802 0.115250681

[6,] 0.207899628 -0.010096263 -0.083360677 0.069451642

[7,] -0.022611845 -0.065976317 -0.073351589 0.030024696

[8,] 0.654908076 0.217206226 0.126717929 0.195297190

[9,] 0.299422480 -0.184659005 -0.150587993 -0.357472590

[10,] -0.009730676 -0.051817028 0.513836145 -0.286614306

[11,] 0.107114578 -0.013397831 0.544425720 0.612249690

[12,] 0.235076405 -0.054132508 -0.054079934 0.206244860

[13,] 0.127738789 -0.125892746 0.106942011 -0.219750558

[14,] 0.068079654 0.295752559 -0.068323287 -0.047129078

[15,] -0.014341735 0.002989417 0.020755977 0.019976486

[16,] -0.001072782 -0.026549633 -0.021960166 0.002538822

[17,] 0.441844232 -0.062920666 -0.209427995 -0.069090916

[18,] -0.288775956 0.188130216 0.141090601 0.073811276

[,17] [,18]

[1,] 0.0028516938 -0.0003818185

[2,] 0.1853895567 -0.0149098297

[3,] -0.0455353177 0.0073277752

[4,] -0.0077394564 0.0263544030

[5,] -0.0026695123 -0.0192824509

[6,] 0.0007464104 0.0122779479

[7,] -0.4049486267 -0.7875783428

[8,] -0.0477185214 -0.2164563356

[9,] 0.5930866962 0.0138457572

[10,] -0.1053167211 0.0239983925

[11,] 0.3404048728 -0.0462582325

[12,] -0.5546056439 0.5721621839

[13,] -0.0901525866 -0.0032686836

[14,] 0.0167022331 0.0073378957

[15,] 0.0052689429 0.0025648010

[16,] 0.0002344129 0.0076745736

[17,] -0.0220477318 -0.0305177668

[18,] 0.0446584089 0.0059490489

### Eigenvalues

[1] 4.538704e-01 1.702475e-01 6.583635e-02 5.466125e-02

[5] 4.143633e-02 2.824432e-02 1.488878e-02 8.852277e-03

[9] 5.088701e-03 3.344046e-03 2.701453e-03 2.081054e-03

[13] 1.330168e-03 1.015179e-03 6.711608e-04 5.514777e-04

[17] 2.939714e-04 1.664103e-05

### Cumulative score per principal components (PC)

[1] 0.5307612 0.7298506 0.8068404 0.8707619 0.9192180

[6] 0.9522472 0.9696583 0.9800103 0.9859610 0.9898716

[11] 0.9930307 0.9954643 0.9970198 0.9982070 0.9989919

[16] 0.9996368 0.9999805 1.0000000



Figure 10: PCA Cumulative value graph

No of PCA attributes used for cumulative score more than 92% is 6. Therefore, need to use PCA1 to PCA 6. Then by using PCA 1 to PCA 6 these new data represent more than 92% of the original data set.

## Determine the number of clusters.

### NBclust

\*\*\* : The Hubert index is a graphical method of determining the number of clusters.

In the plot of Hubert index, we seek a significant knee that corresponds to a

significant increase of the value of the measure i.e the significant peak in Hubert

index second differences plot.

\*\*\* : The D index is a graphical method of determining the number of clusters.

In the plot of D index, we seek a significant knee (the significant peak in Dindex

second differences plot) that corresponds to a significant increase of the value of

the measure.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Among all indices:

\* 7 proposed 2 as the best number of clusters

\* 11 proposed 3 as the best number of clusters

\* 1 proposed 4 as the best number of clusters

\* 2 proposed 7 as the best number of clusters

\* 1 proposed 9 as the best number of clusters

\* 2 proposed 10 as the best number of clusters

\*\*\*\*\* Conclusion \*\*\*\*\*

\* According to the majority rule, the best number of clusters is 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Figure 11: NBClust result graph (PCA Transform data)

### Elbow



Figure 12: Elbow method result graph (PCA Transform data)

In elbow method graph reduction speed reduce in 4 therefore its suggest that optimum number for k 3.

### Gap Statistics



Figure 13: Gap Statistic result graph (PCA Transform data)

### Average silhouette method



Figure 14: Silhouette plot (PCA Transform data)

### Summary of the optimal result

Using automated optimal clustering methods, NbClust function, Elbow method and Gap statistic suggest 3 as a optimal k and Silhouette suggest 2.

## K-means analysis for this “PCA”-based dataset

1. K-means clustering with 3 clusters of sizes 253, 325, 234
2. Cluster means:
3. PC1 PC2 PC3 PC4 PC5
4. 1 0.8760277 0.07524749 -0.0122160208 0.02300791 0.006824829
5. 2 -0.2401719 -0.34194758 0.0008203363 -0.03520882 0.025840480
6. 3 -0.6135861 0.39356987 0.0120685640 0.02402506 -0.043268538
7. PC6
8. 1 -0.02100738
9. 2 -0.00281816
10. 3 0.02662722
11. Clustering vector:
12. [1] 2 2 1 2 1 2 2 2 2 2 2 2 2 2 1 3 2 1 1 3 3 2 2 1 2 3 1 1 3 2
13. [31] 2 2 1 2 2 3 1 3 1 3 3 2 3 3 3 2 3 2 1 2 1 2 2 3 1 3 1 3 3 3
14. [61] 2 3 3 1 2 1 1 1 2 3 2 1 2 3 1 3 1 2 3 2 2 3 2 3 1 2 1 2 3 1
15. [91] 3 3 1 3 2 2 3 1 1 1 3 3 2 2 2 3 3 2 1 1 3 2 3 3 2 2 3 3 2 1
16. [121] 1 2 3 1 3 2 3 2 2 3 1 3 2 1 2 2 2 2 1 2 2 1 2 1 2 3 2 2 3 1
17. [151] 2 2 1 1 2 1 3 3 1 1 2 1 2 2 2 2 2 3 1 3 2 3 1 2 2 2 1 2 2 2
18. [181] 1 2 3 1 3 3 3 2 2 1 1 2 2 2 3 3 1 2 2 2 1 3 2 3 1 3 2 1 3 1
19. [211] 3 3 2 1 2 1 3 3 3 1 2 3 2 3 1 3 2 2 3 1 3 3 2 2 1 3 3 1 3 2
20. [241] 2 1 2 2 1 1 3 2 2 2 1 3 3 2 2 3 3 2 2 2 1 2 3 3 1 2 2 3 3 1
21. [271] 3 2 2 3 1 3 3 2 2 1 2 1 3 2 2 1 2 2 2 3 2 1 1 1 1 1 3 2 1 3
22. [301] 3 3 2 3 1 1 1 3 1 2 3 1 3 2 2 2 1 1 3 1 1 3 1 2 2 2 3 3 1 1
23. [331] 1 2 2 2 1 3 2 3 2 2 2 1 2 1 1 1 2 2 3 1 3 3 3 2 2 2 2 2 3 1
24. [361] 1 3 3 1 3 3 1 2 2 2 2 1 3 2 2 2 2 1 2 2 2 2 1 2 1 2 3 3 2 2
25. [391] 2 3 3 2 3 1 2 2 3 2 3 1 2 3 2 2 1 2 1 2 1 1 3 3 1 2 3 3 2 1
26. [421] 1 3 2 1 1 3 1 1 1 2 2 2 2 2 1 3 3 2 1 2 2 1 2 3 1 3 3 1 1 2
27. [451] 3 1 1 1 3 1 1 2 2 3 1 1 2 2 3 3 1 2 3 1 1 2 3 1 1 2 1 3 1 1
28. [481] 1 3 3 1 1 2 2 1 3 2 1 3 3 1 3 2 2 3 2 1 2 1 1 2 3 2 1 1 3 3
29. [511] 2 1 2 1 1 2 2 2 2 3 3 2 2 1 3 3 2 3 1 2 1 3 3 1 1 2 1 2 2 2
30. [541] 1 2 3 2 1 2 2 3 1 1 1 1 2 3 3 3 1 1 1 2 1 3 2 1 3 3 3 2 3 2
31. [571] 2 2 2 2 2 2 1 2 2 1 2 2 2 3 1 3 3 2 3 2 2 3 3 1 1 3 2 3 1 2
32. [601] 2 1 2 3 1 3 1 3 3 2 3 2 1 1 2 1 2 2 3 2 3 1 2 1 3 2 2 2 3 3
33. [631] 2 1 2 1 3 2 2 2 2 1 2 3 1 2 1 2 2 1 3 1 3 2 2 2 3 1 2 3 2 1
34. [661] 3 1 2 2 1 3 2 3 2 2 3 2 1 1 2 2 1 1 2 3 2 1 1 1 1 2 1 2 2 1
35. [691] 1 2 1 2 1 2 3 1 2 3 1 1 1 2 1 3 3 1 1 1 2 1 2 2 1 2 3 2 3 2
36. [721] 1 2 3 2 2 2 3 1 3 3 3 1 3 1 1 3 2 2 1 2 3 1 1 3 2 2 1 1 1 3
37. [751] 1 2 1 1 3 3 1 3 1 2 3 2 1 1 2 3 1 2 2 3 2 2 1 3 2 1 3 3 1 3
38. [781] 2 3 3 2 1 1 2 3 1 2 1 1 3 2 1 3 3 2 2 1 3 3 3 2 2 2 2 2 2 1
39. [811] 2 3
40. Within cluster sum of squares by cluster:
41. [1] 98.58379 114.05983 69.37615
42. (between\_SS / total\_SS = 57.3 %)
43. Available components:
44. [1] "cluster" "centers" "totss" "withinss"
45. [5] "tot.withinss" "betweenss" "size" "iter"
46. [9] "ifault"

BSS

WSS = 282.0198

## 1.8. Silhouette

cluster size ave.sil.width

1 1 253 0.40

2 2 325 0.24

3 3 234 0.33

In this new transform data set average silhouette score more than zero and every cluster silhouette value also more than 0 therefore its can consider has a average to good cluster.

Silhouette of 812 units in 3 clusters from silhouette.default(x = kmeans\_fit$cluster, dist = dist(pca\_result$x[, 1:6])) :

Cluster sizes and average silhouette widths:

234 325 253

0.3268898 0.2385897 0.3992239

Individual silhouette widths:

Min. 1st Qu. Median Mean 3rd Qu. Max.

-0.03488 0.20606 0.33778 0.31409 0.43826 0.57491



Figure 15: Silhouette Plot (PCA Transform data)

## Calinski-Harabasz index

1. # Calinski-Harabasz index
2. library(fpc)
3. Calinski <- calinhara(pca\_result$x, kmeans\_fit$cluster)
4. Calinski

Calinski-Harabasz index value: 489.3797

Calinski-Harabasz index is also like other clustering quality measure, when a index value is higher separation between clusters better and small variation between clusters.

# Multi-layer Neural Network

## Methods used for defining the input vector in electricity load forecasting

In load furcating its divide in to three main categories,

1. Short-term load forecasting
2. Medium-term load forecasting
3. Long-term load forecasting

Because of category model train differently by using input data. In Long term load forecasting use micro, macro economy of the county and a household as well as GDP, population Electricity. (Tartibu & Kabengele, 2018). Load forecasting uses various input vectors for trained neural network models. Some of these input vectors are, temperature, seasonal input such as air conditioning and heaters, and historical load data, weather data, (Mystakidis, et al., 2022). In regression analysis used past observations of input variables like electricity load as well as humidity and time. Cooperation Long term forecasting considers using 25 to 35 years of data for training the model. (Singla, et al., 2019). Artificial Network uses lagged variables like lagged load for regression. And temperature and calendar variable like seasonal data. (Groß, et al., 2021). Input data detemine by the category o the forcast as well as available data. (Hammad, et al., 2020)

## Data Preprocessing

### Why do we need to normalize data before using them in an Multi-Layer Perceptron (MLP) structure?

Data normalization is a one of method use in data pre prosessing to scale all features on a similar scale. Nomalization, avoid bias, improve generalization and improve convgence is the reason to use normalization in MLP by result of this its return greate proformance and also grate accuracy result from the model.

* Avoiding bias: Normalization scale input features to same range therefore its give similar ground for all data. If isnt high value range feauters dominate against other input features.
* Better generalization: By ovoiding overfitting its give better accurate prediction.
* Faster convergence: Normalization speed up training process of MLP.

Thereare varius normalization methods and min-max normalization most popular two. When using normalization its need to cosider about data set and choose correct method.



Figure 16: Dataset before and after normalization

Funtion use in code for normalization and de-normalization

min\_max\_norm <- function(x) {

  (x - min(x)) / (max(x) - min(x))

}

 #function for de-normalized data

  deNorm<-function(y,min,max){

    return((max-min)\*y+min)

  }

## Implement a number of MLPs for the “AR” approach.



Figure 17: Nural network plot (1hidden layer & all input)

Table 2: Accuracy, rmse, mae, smape score (1hidden layer & all input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9965172 | 9.086114e+17 | 2.48461 | 8.515509 |



Figure 18: Nural network plot (1hidden layer, 3 neurons & all input)

Table 3: Accuracy, rmse, mae, smape score (1hidden layer, 3 neurons & all input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9972133 | 1.516948e+18 | 2.485899 | 8.518373 |



Figure 19: Nural network plot (1hidden layer, 2 neurons & all input)

Table 4: Accuracy, rmse, mae, smape score (1hidden layer, 2 neurons & all input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9980563 | 1.221641e+18 | 2.450448 | 8.53423 |



Figure 20: Nural network plot (1hidden layer, 5 neurons & t1,t7 input)

Table 5: Accuracy, rmse, mae, smape score (1hidden layer, 5 neurons & t1,t7 input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9921136 | 9.328024e+17 | 2.506818 | 8.505532 |



Figure 21: Nural network plot (2hidden layer, 2 neurons & t1,t7 input)

Table 6: Accuracy, rmse, mae, smape score (1hidden layer, 2 neurons & t1,t7 input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9932443 | 9.781572e+17 | 2.496189 | 8.508399 |



Figure 22: Nural network plot (1hidden layer, 4 neurons & all input)

Table 7: Accuracy, rmse, mae, smape score (1hidden layer, 4 neurons & all input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9952463 | 1.58693e+18 | 2.422108 | 8.51397 |



Figure 23: Nural network plot (2hidden layer, 6 neurons & t1,t2 input)

Table 8: Accuracy, rmse, mae, smape score (2hidden layer, 6 neurons & t1,t2 input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9918452 | 6.217438e+17 | 2.931957 | 8.49697 |



Figure 24: Nural network plot (2hidden layer, 4 neurons & 2 input)

Table 9: Accuracy, rmse, mae, smape score (2hidden layer, 4 neurons & t1,t2 input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9882761 | 6.599503e+17 | 2.864306 | 8.493118 |



Figure 25: Nural network plot (2hidden layer, 10 neurons & all input)

Table 10: Accuracy, rmse, mae, smape score (2hidden layer, 10 neurons & all input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9952289 | 1.689011e+18 | 2.450552 | 8.514344 |



Figure 26: Nural network plot (2hidden layer, 8 neurons & 2 input)

Table 11: Accuracy, rmse, mae, smape score (2hidden layer, 8 neurons & t2,t3 input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9813733 | 2.811663e+17 | 2.977977 | 8.489833 |



Figure 27: Nural network plot (2hidden layer,5 neurons & 3 input)

Table 12: Accuracy, rmse, mae, smape score (2hidden layer, 5 neurons & t1,t4,t7 input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9929431 | 5.394419e+17 | 2.596023 | 8.503299 |



Figure 28: Nural network plot (3hidden layer, 9 neurons & all input)

Table 13: Accuracy, rmse, mae, smape score(3hidden layer, 9 neurons & all input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9915509 | 2.15414e+18 | 2.435437 | 8.506993 |



Figure 29: Nural network plot (2hidden layer, 5 neurons & all input)

Table 14: Accuracy, rmse, mae, smape score (2hidden layer, 5 neurons & all input)

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9992423 | 1.30815e+18 | 2.430985 | 8.530734 |

## comparison matrix for the “AR” case

Briefly explain the meaning of these four stat

* RMSE (Root Mean Squared Error)

RMSE is a measure of the average squared error between the predicted and actual values. This is good evaluation method for accuracy in regression models because it penalize large error more than small error

* MAE (Mean Absolute Error)

MAE is a measure of the average absolute error between the predicted and actual values. This is good evaluation method for accuracy in regression models because it easy to understand and interpret.

* MAPE (Mean Absolute Percentage Error)

MAPE is a measure of the average absolute percentage error between the predicted and actual values. This matrix is good for forecasting models’ evaluation.

* sMAPE (Symmetric Mean Absolute Percentage Error)

MAPE variation is sMAPE. It’s robust for outliers.

Other than accuracy all other mention evaluation matrix nose of the time best its score low)

In summary all evaluation matrix good at evaluating data but need to choose which one choose.

Table 15: RMSE, MAE, MAPE and sMAPE – symmetric MAPE Score

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | No Inputs | No. Hidden layers | No neurones | Accuracy | RMSE | MAE | sMAPE |
| 1 | 2 | 1 | 5 | 0.9921136 | 9.328024 | 2.506818 | 8.505532 |
| 2 | 2 | 2 | 4 | 0.9886238 | 7.99E+00 | 2.873717 | 8.493922 |
| 3 | 2 | 2 | 6 | 0.9918452 | 6.22E+00 | 2.931957 | 8.49697 |
| 4 | 2 | 2 | 8 | 0.9813733 | 2.81E+00 | 2.977977 | 8.489833 |
| 5 | 3 | 2 | 5 | 0.9929431 | 5.394419 | 2.596023 | 8.503299 |
| 6 | 5 | 1 | 1 | 0.9965172 | 9.086114 | 2.48461 | 8.515509 |
| 7 | 5 | 1 | 2 | 0.9980563 | 1.221641 | 2.450448 | 8.53423 |
| 8 | 5 | 1 | 3 | 0.9972133 | 1.516948 | 2.485899 | 8.518373 |
| 9 | 5 | 1 | 4 | 0.9952463 | 1.58693 | 2.422108 | 8.51397 |
| 10 | 5 | 2 | 10 | 0.9952289 | 1.689011 | 2.450552 | 8.514344 |
| 11 | 5 | 2 | 5 | 0.9992423 | 1.30815 | 2.430985 | 8.530734 |
| 12 | 5 | 3 | 9 | 0.9965152 | 2.508707 | 2.450498 | 8.518585 |

## issue of “efficiency”

Table 16: efficiency of models

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | No Inputs | No. Hidden layers | No neurones | H1 | H2 | total number of weight parameters |
| 1 | 2 | 1 | 5 | 5 |  | 21 |
| 2 | 2 | 2 | 4 | 2 | 2 | 15 |
| 3 | 2 | 2 | 6 | 3 | 3 | 25 |
| 4 | 2 | 2 | 8 | 4 | 4 | 37 |
| 5 | 3 | 2 | 5 | 3 | 2 | 23 |
| 6 | 5 | 1 | 1 | 1 |  | 8 |
| 7 | 5 | 1 | 2 | 2 |  | 15 |
| 8 | 5 | 1 | 3 | 3 |  | 22 |
| 9 | 5 | 1 | 4 | 4 |  | 29 |
| 10 | 5 | 2 | 10 | 5 | 5 | 66 |
| 11 | 5 | 2 | 5 | 2 | 3 | 25 |

Using this table for one hidden layer model most efficient one index six. And for two hidden layer its index 2. But when considering accuracy and weight parameters index 11 and considering 2 hidden layer index 9.

## “NARX” approach



|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9941274 | 3.102795e+19 | 2.565544 | 8.316933 |

Diagram

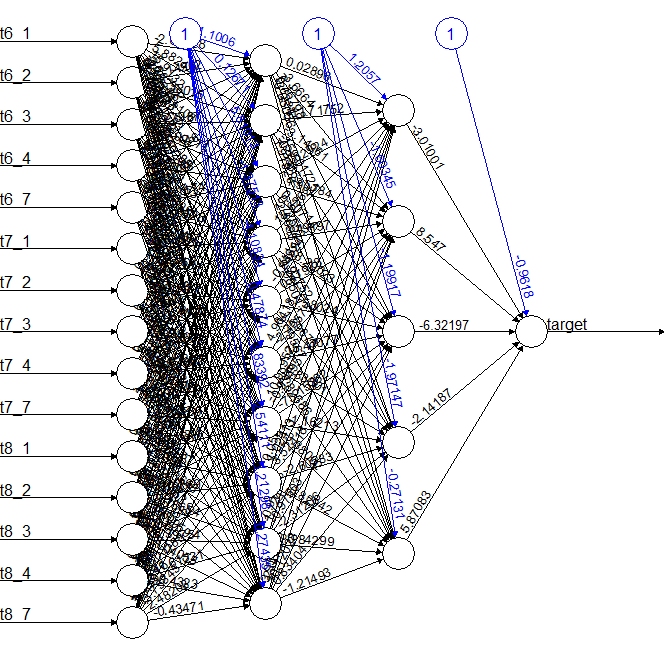
Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.988028 | 5.799374e+20 | 2.644774 | 8.257579 |

Chart, diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9985096 | 1.468904e+19 | 2.747622 | 8.348707 |



|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9930611 | 7.022138e+20 | 2.901472 | 8.482227 |

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9861303 | 2.644392e+19 | 2.453417 | 8.245021 |

Chart, radar chart

Description automatically generated

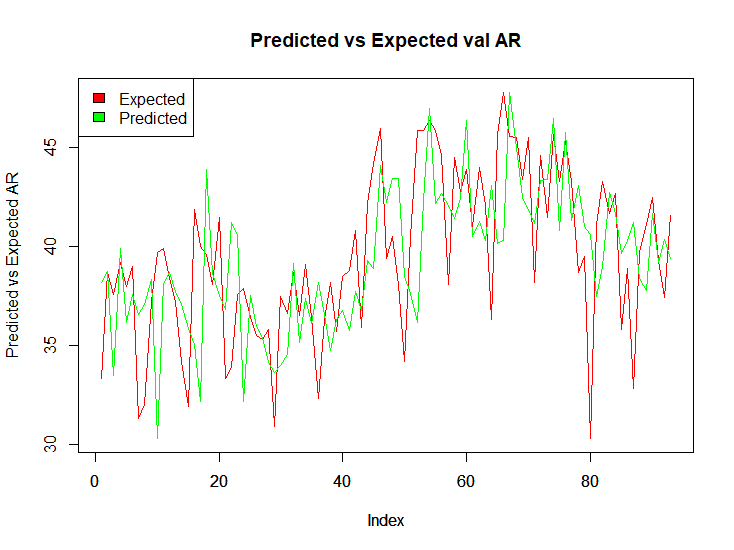
|  |  |  |  |
| --- | --- | --- | --- |
| accuracy | rmse | mae | smape |
| 0.9936031 | 5.600032e+19 | 2.587178 | 8.319871 |

### Comparison Matrix for NARX

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | No Inputs | No. Hidden layers | No neurones | Accuracy | RMSE | MAE | sMAPE |
| 1 | 15 | 2 | 8 | 0.994127 | 3.102795e | 2.56554 | 8.31693 |
| 2 | 6 | 2 | 8 | 0.988028 | 5.799374 | 2.64477 | 8.25757 |
| 3 | 6 | 2 | 14 | 0.998506 | 1.468904e | 2.74762 | 8.34870 |
| 4 | 15 | 2 | 15 | 0.993061 | 7.022138e | 2.90147 | 8.48222 |
| 5 | 6 | 2 | 10 | 0.986130 | 2.644392e | 2.45341 | 8.24502 |
| 6 | 15 | 2 | 18 | 0.993603 | 5.600032e | 2.58717 | 8.31987 |

## Best MLP network

Considering AR model best model its index 9 and NARX model best one is 4 its in a second place considering to accuracy.



# References

Gardener, M., 2012. *Chapter 5: Data: Distribution | Beginning R: The Statistical Programming Language.* [Online]   
Available at: https://learning.oreilly.com/library/view/beginning-r-the/9781118239377/164303c05.html#x164303c05-para-0107  
[Accessed 30 04 2023].

Groß, A. et al., 2021. Comparison of short-term electrical load forecasting methods for different building types. *Energy Informatics,* 4(3), p. 13.

Hammad, M. A., Jereb, B., Rosi, B. & Dragan, D., 2020. Methods and Models for Electric Load Forecasting: A Comprehensive Review. *Logistics & Sustainable Transport,* 11(1), pp. 51-76.

Mystakidis, A. et al., 2022. One Step Ahead Energy Load Forecasting: A Multi-model approach utilizing Machine and Deep Learning. *2022 57th International Universities Power Engineering Conference: Big Data and Smart Grids, UPEC 2022 - Proceedings.*

Singla, M. K., Gupta, J., Nijhawan, P. & Oberoi, A. S., 2019. Electrical load forecasting using machine learning. *International Journal of Advanced Trends in Computer Science and Engineering,* 8(3), pp. 615-619.

Tartibu, L. & Kabengele, K., 2018. *Forecasting Net Energy Consumption of South Africa using Artificial Neural Network.* Cape Town, s.n.

# Appendices

## Partitioning\_clustering

#Removed all the existing objects

rm(list = ls())

# load package

library(readxl)

library(ggplot2)

library(factoextra)

library(NbClust)

library(cluster)

#Load the dataset

df <- read\_excel("Data/vehicles.xlsx")

head(df)

df.lable <- df$Class

df <- df[,2:19]

plot.new()

par(mfrow = c(1, 2))

boxplot(df)

title("Box Plot before outlier remove")

func\_outliers <- function(x){

  outlier\_values <- boxplot(x, plot=FALSE)$out

  data\_clean <- replace(x, x %in% outlier\_values, NA)

  data\_clean

}

# remove outliers

clean\_df <- apply(df, 2, func\_outliers)

clean\_df <- as.data.frame(na.omit(clean\_df))

boxplot(clean\_df)

title("Box Plot after outlier remove")

# Min max normalization

normalize <- function(x) {

  (x - min(x)) / (max(x) - min(x))

}

# normalize data

normalized\_df <- as.data.frame(apply(clean\_df,2, normalize))

boxplot(normalized\_df)

clean\_df <- apply(normalized\_df, 2, func\_outliers)

clean\_df <- as.data.frame(na.omit(clean\_df))

normalized\_df <- as.data.frame(apply(clean\_df,2, normalize))

boxplot(normalized\_df)

title("Box Plot after Normalization")

library(NbClust)

# using NbClust determine the optimal number of clusters

set.seed(123)

clusterNo = NbClust(normalized\_df, distance="euclidean", min.nc=2, max.nc = 10, method="kmeans")

table(clusterNo$Best.n[1,])

barplot(table(clusterNo$Best.n[1,]), xlab="Numer of Clusters",ylab="Number of Criteria",

        main="Number of Clusters

Chosen by 30 Criteria") # NBClust suggest 3

library(factoextra)

library(ggplot2)

library(factoextra)

# silhouette

fviz\_nbclust(normalized\_df, kmeans, method = 'silhouette')+

  labs(subtitle = "Silhouette method") # silhouette suggest 2

# Elbow Method

fviz\_nbclust(normalized\_df, kmeans, method = 'wss') # Elbow suggest 3

# gap

fviz\_nbclust(normalized\_df, kmeans, method = 'gap\_stat') # gap Suggest 3

set.seed(1234)

# use k as a 3

k <- 3

kmeans\_fit  = kmeans(normalized\_df, centers = k, nstart = 2, iter.max = 17)

kmeans\_fit

#The between-cluster sum of squares

bss <- kmeans\_fit $betweenss

bss

# The total sum of squares.

tss <- kmeans\_fit $totss

tss

# Total within-cluster sum of squares

wss <- kmeans\_fit $tot.withinss

wss

plot(normalized\_df[c("Comp", "Rad.Ra")], main="K-means Clustering Plot", col=kmeans\_fit $cluster)

points(kmeans\_fit $centers[,c("Comp", "Rad.Ra")], col=1:3, pch=23, cex=3)

sil <- silhouette(kmeans\_fit$cluster, dist(normalized\_df))

fviz\_silhouette(sil)

summary(sil)

library(tidyverse)

library(gridExtra)

library(ggplot2)

library(rlang)

pca\_result <- prcomp(normalized\_df, scale = FALSE)

cumulative\_variance <- cumsum(pca\_result$sdev^2 / sum(pca\_result$sdev^2))

cumulative\_variance

plot(cumulative\_variance, xlab = "Number of PCs", ylab = "Cumulative Variance Explained", type = "b")

abline(h=0.92, col="red")

title("Cumalative value > 92%")

pca\_result

eigenvector <- pca\_result$rotation

eigenvector

pca\_result$rotation <- -pca\_result$rotation

pca\_result$x <- - pca\_result$x

eigenvalues <- pca\_result$sdev^2

eigenvalues

set.seed(1233)

nb <- NbClust(pca\_result$x[,1:6], method="kmeans", distance = "euclidean",

              min.nc = 2, max.nc = 10, index = "all") # 3

# Elbow Method

fviz\_nbclust(pca\_result$x[,1:6], kmeans, method = 'wss') # Elbow suggest 3

# gap

fviz\_nbclust(pca\_result$x[,1:6], kmeans, method = 'gap\_stat') # gap Suggest 3

fviz\_nbclust(pca\_result$x, kmeans, method = 'silhouette')+

  labs(subtitle = "Silhouette method") # silhouette suggest 2

k <- 3

kmeans\_fit  = kmeans(pca\_result$x[,1:6], centers = k, nstart = 2, iter.max = 17)

kmeans\_fit

kmeans\_fit$betweenss

wss <- kmeans\_fit $tot.withinss

wss

library(cluster)

sil <- silhouette(kmeans\_fit$cluster, dist(pca\_result$x[,1:6]))

fviz\_silhouette(sil)

summary(sil)

kmeans\_fit <- function(data, k) {

  kmeans(data, centers = k, nstart = 25)

}

# Calinski-Harabasz index

library(fpc)

Calinski <- calinhara(pca\_result$x, kmeans\_fit$cluster)

Calinski

## MLP

#Removed all the existing objects

rm(list = ls())

library(readxl)

#Load the dataset

data <- read\_excel("Data/uow\_consumption.xlsx")

date <- data$date

data <- data[,4]

names(data)[1] <- "twenty"

acf(data)

pacf(data)

boxplot(data$twenty)

#Data pre-processing

min\_max\_norm <- function(x) {

  (x - min(x)) / (max(x) - min(x))

}

data\_nor <- as.data.frame(lapply(data, min\_max\_norm))

boxplot(data\_nor)

date\_t <- factor(date)

date\_t <- as.Date(date\_t)

data\_nor$date <- date\_t

plot(data\_nor$twenty~data\_nor$date,type="l",col="blue")

library(xts)

library(zoo)

library(quantmod)

library(neuralnet)

library(forecast)

library(NeuralNetTools)

timeser <- xts(data\_nor$twenty,order.by = as.Date(data\_nor$date))

# rm(normalized\_df,date,lable)

rtimeserall <- cbind(timeser,lag(timeser, 1),lag(timeser, 2),lag(timeser, 3),

                     lag(timeser, 4),lag(timeser, 7)

)

colnames(rtimeserall) <- c('t', 't1', 't2', 't3', 't4','t7')

rtimeserall <- rtimeserall[complete.cases(rtimeserall),]

data\_train <- rtimeserall[1:370]

data\_test <- rtimeserall[371:463]

library(Metrics)

# Remove missing values from the training data

#creating various kinds of neural networks and plotting them with the results matrix

set.seed(235)

consumsion\_nnar\_h1\_all<- neuralnet(t ~ ., hidden = 1, data = data\_train,

                            linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h1\_all$result.matrix

plot(consumsion\_nnar\_h1\_all, dimension = 8)

set.seed(236)

consumsion\_nnar\_h2\_all<- neuralnet(t ~ ., hidden = 2, data = data\_train,

                               linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h2\_all$result.matrix

plot(consumsion\_nnar\_h2\_all)

set.seed(237)

consumsion\_nnar\_h3\_all<- neuralnet(t ~ ., hidden = 3, data = data\_train,

                            linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h3\_all$result.matrix

plot(consumsion\_nnar\_h3\_all)

set.seed(238)

consumsion\_nnar\_h4\_all<- neuralnet(t ~ ., hidden = 4, data = data\_train,

                            linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h4\_all$result.matrix

plot(consumsion\_nnar\_h4\_all)

set.seed(239)

consumsion\_nnar\_h5\_t1.t7<- neuralnet(t ~ t1+t7, hidden = 5, data = data\_train,

                            linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h5\_t1.t7$result.matrix

plot(consumsion\_nnar\_h5\_t1.t7)

set.seed(240)

consumsion\_nnar\_h1.1\_t1.t7<- neuralnet(t ~ t1+t7, hidden = c(1,1), data = data\_train,

                            linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h1.1\_t1.t7$result.matrix

plot(consumsion\_nnar\_h1.1\_t1.t7)

set.seed(241)

consumsion\_nnar\_h2.2\_t1.t2<- neuralnet(t ~ t1+t2, hidden = c(2,2), data = data\_train,

                               linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h2.2\_t1.t2$result.matrix

plot(consumsion\_nnar\_h2.2\_t1.t2, dimension = 8)

set.seed(242)

consumsion\_nnar\_h3.3\_t1.t2<- neuralnet(t ~ t1+t2, hidden = c(3,3), data = data\_train,

                                 linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h3.3\_t1.t2$result.matrix

plot(consumsion\_nnar\_h3.3\_t1.t2)

set.seed(243)

consumsion\_nnar\_h4.4\_t2.t3<- neuralnet(t ~ t2+t3, hidden = c(4,4), data = data\_train,

                                  linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h4.4\_t2.t3$result.matrix

plot(consumsion\_nnar\_h4.4\_t2.t3)

set.seed(244)

consumsion\_nnar\_h5.5\_all<- neuralnet(t ~ ., hidden = c(5,5), data = data\_train,

                                  linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h5.5\_all$result.matrix

plot(consumsion\_nnar\_h5.5\_all)

set.seed(245)

consumsion\_nnar\_h2.3\_all<- neuralnet(t ~ ., hidden = c(2,3), data = data\_train,

                                        linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h2.3\_all$result.matrix

plot(consumsion\_nnar\_h2.3\_all)

test(consumsion\_nnar\_h2.3\_all)

set.seed(246)

consumsion\_nnar\_h5.3\_t3.t4<- neuralnet(t ~ t3+t4+t7, hidden = c(3,2), data = data\_train,

                                      linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h5.3\_t3.t4$result.matrix

plot(consumsion\_nnar\_h5.3\_t3.t4)

test(consumsion\_nnar\_h5.3\_t3.t4)

set.seed(246)

consumsion\_nnar\_h5.3\_t3.t4<- neuralnet(t ~ ., hidden = c(4,3,2), data = data\_train,

                                       linear.output=TRUE,threshold= 0.01)

consumsion\_nnar\_h5.3\_t3.t4$result.matrix

plot(consumsion\_nnar\_h5.3\_t3.t4)

test(consumsion\_nnar\_h5.3\_t3.t4)

## MLP\_test

(Function use for get score and accuracy of the model)

test <- function(variables) {

  table\_result <- c()

  consumsion\_nnar <- variables

  #Model performance evluation

  load\_model<-predict(consumsion\_nnar,data\_test)

  # get test and train data set without normalization

  uow\_loadtrain <- data[7:377,1]

  uow\_loadtest <- data[378:470,1]

  #finding min and max vaues of trained dataset

  train\_min<-min(uow\_loadtest)

  train\_max<-max(uow\_loadtest)

  #function for unormalized data

  unNorm<-function(y,min,max){

    return((max-min)\*y+min)

  }

  #calculate accuracy

  predict=load\_model\*abs(diff(range(data\_test$t)))+min(data\_test$t)

  actual=data\_test$t\*abs(diff(range(data\_test$t)))+min(data\_test$t)

  compare=data.frame(predict,actual)

  deviation=((actual-predict)/actual)

  is.na(deviation)<-sapply(deviation,is.infinite)

  dev\_NAomit<-na.omit(deviation)

  compare= data.frame(predict,actual,deviation)

  accuracy=1-abs(mean(dev\_NAomit))

  table\_result$accuracy <- accuracy

  uow\_load\_pred\_unnorm<- unNorm(load\_model,train\_min,train\_max)

  #testing pperformance of RMSE

  #RMSE

  table\_result$rmse <- rmse(exp(uow\_load\_pred\_unnorm),uow\_loadtest$twenty)

  #MAE

  table\_result$mae <- mean(abs(uow\_load\_pred\_unnorm - uow\_loadtest$twenty))

  # sMAPE

  table\_result$smape <- (1/nrow(uow\_loadtest)) \* sum(200 \*

                                                       abs(mean(uow\_load\_pred\_unnorm) - uow\_loadtest) /

                                                       (abs(mean(uow\_load\_pred\_unnorm)) +

                                                          abs(uow\_loadtest)))

  print(table\_result)

}