University of Houston DOWNTOWN

Clustering of Human Urine Proteins

Prepared by:

Shadab Azhar Siddiqui

Table of Contents

Abstract

This report explains the use of the k-means clustering algorithm for the classification of numerous proteins of human urine deposition on catheter surfaces. By the measurements of deposit volume at various time points, the proteins deposit volumes are showing messy and irregular changes and are hard to be classified. To cluster the “behavior” of proteins deposit change, this report used the k-means algorithm as a classifier and used the elbow method, silhouette method, and sum of squares method to get the appropriate value of k-means clusters. The proteins are labeled with seven classes. After that, data mining algorithms such as random forest, decision tree, artificial neural network & support vector machines (SVM) are used to verify the accuracy of classification.

Objective

The objective of this study is to clean the data and analyze the change in the deposited protein volume during 24hour and label it according to the type of change. There is little knowledge or previous study on the trend of deposit value change, the data is messy and unlabeled. Therefore, an unsupervised machine learning algorithm was adopted to do a deep study in finding the appropriate number of clusters that were labeled. The results were further verified using several machine learning algorithms to find the accuracy of classification.

Data Source and Background

Researchers from the Department of Chemistry at the University of Houston did the following experiments in order to study the deposition changes of proteins contained in human urine in the solution: Silicone catheters were incubated in the 500 μL filtered human urine for various time points, including 5 min, 15 min, 30 min, 3 h, 6 h, 18 h, and 24 h. The catheters were rinsed with filtered HPLC grade water to remove any unbound proteins on the catheter surface. The deposited proteins were heat-denatured and tryptic digested. The resulted peptides were analyzed by a Bruker’s timsTOF Pro mass spectrometry system equipped with a NanoElute nanoflow liquid chromatography system. The concentration of the peptides in the extract was determined by Peptide Assay (Thermo Scientific) and converted to the density of the proteins deposited on the catheter.

Data Loading

The original data is in two excel files. The first excel file contains 1500 different proteins with protein name, ID, descriptions and deposited volume of “5min”, “15min”, ”30min”, “3h”, “6h”, “18h”, “24h”, the measurement was reported three times, therefore there are 3 values at each time point. The second data file contains 1210 proteins with ID, name, genes, description, the deposited volume of “30min”, “1h”, “2h”, “3h”, “4h”, reported 3 times along with intensity and amount measurement at each time point. Both excel file was loaded as an original data frame. Figures 1 and 2 show original data frame 1 and data frame 2 respectively.

PG.ProteinDescriptions	PG.UniProtIds	X5min.1	X5min.2	X5min.3	X15min.1	X15min.2	X15min.3	X30min.1	X30min.2
Probable non-functional immunoglobulin kappa variable 3-7	A0A075B6H7	290.457581	1080.80933	175.804047	279.659027	721.84393	625.79285	1302.479614	458.263763
Immunoglobulin lambda variable 8-61	A0A075B6I0	198.499985	293.41687	371.346100	910.212158	857.93866	708.55219	1504.888794	1820.768921
Immunoglobulin lambda variable 2-18	A0A075B6J9	111.354744	86.13438	110.180557	253.748703	163.02072	189.44823	407.094757	498.345581
Immunoglobulin lambda variable 3-10	A0A075B6K4	0.000000	0.00000	0.000000	69.049858	0.00000	0.00000	108.750549	0.000000
Immunoglobulin lambda variable 3-9	A0A075B6K5	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000
Immunoglobulin heavy variable 4-4;Immunoglobulin heavy ...	A0A075B6R2	0.000000	0.00000	12.090642	15.306090	0.00000	0.00000	0.000000	0.000000
Probable non-functional immunoglobulin kappa variable 2D...	A0A075B6R9	0.000000	0.00000	22.600832	35.834705	0.00000	0.00000	0.000000	40.977749
Immunoglobulin kappa variable 2D-29;Immunoglobulin kap...	A0A075B6S2	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000
Immunoglobulin kappa variable 1-27	A0A075B6S5	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000
Probable non-functional immunoglobulinn kappa variable 1...	A0A075B6S9	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000
Immunoglobulin kappa variable 3D-15	A0A087WSY6	61.783802	71.86616	53.166969	113.517219	468.61700	80.20202	144.445236	170.342911
Immunoglobulin kappa variable 2-40;Immunoglobulin kapp...	A0A087WW87	167.952103	126.28728	176.085785	269.841827	139.68526	234.40208	281.096893	325.652924
Immunoglobulin kappa variable 3D-11;Immunoglobulin kap...	A0A0A0MRZ8	1019.478088	702.09210	954.176331	1737.503418	1613.45801	1248.60803	2949.550537	3232.727539
Immunoglobulin heavy variable 3-49	A0A0A0MS15	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	37.678093
Immunoglobulin kappa variable 6D-21;Probable non-functi...	A0A0A0MT36	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000
Immunoglobulin heavy variable 3-15	A0A0B4J1V0	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000
Immunoglobulin heavy variable 3-73	A0A0B4J1V6	0.000000	0.00000	0.000000	0.000000	0.00000	0.00000	0.000000	0.000000

Showing 1 to 17 of 1,500 entries, 31 total columns

Figure 1- the original loaded first data frame with protein name, ID and proteins deposited volume along with time at 5min, 15min, 30min, 3h, 6h, 18h and 24h. Each three repeated measurement values for each case and each time point.

First.Protein.Description	30-1	30-2	30-3	30 min_Intensity	Amount (ng)...10	1-1	1-2	1 h_Intensity	Amount (ng)...14	2-1	2-3
50S ribosomal protein L7/L12	220537.00	389911.00	300813.00	303753.67	11.1766722	228112.00	208700.000	218406.000	11.67358648	317886.00	370477.00
30S ribosomal protein S20	269410.00	185635.00	310807.00	255284.00	9.3932218	461371.00	341575.000	401473.000	21.45833807	687912.00	298993.00
Glyceraldehyde-3-phosphate dehydrogenase A	339187.00	191476.00	231843.00	254168.67	9.3521829	271214.00	316766.000	293990.000	15.71347714	479848.00	314526.00
30S ribosomal protein S16	278215.00	164320.00	238119.00	226884.67	8.3482631	291281.00	248944.000	270112.500	14.43724819	242936.00	157292.00
DNA-binding protein HU-alpha	222425.00	267754.00	168751.00	219643.33	8.0818169	423290.00	178079.000	300684.500	16.07129160	286081.00	315093.00
Glucosamine-6-phosphate deaminase	238091.00	232413.00	117138.00	195880.67	7.2074652	39834.80	72048.000	55941.400	2.99001296	122784.00	102726.00
Protein translocase subunit SecD	284550.00	112253.00	105612.00	167471.67	6.1621508	70250.00	65993.400	68121.700	3.64103805	168783.00	50263.50
Modulator of FtsH protease HflK	286851.00	108127.00	74125.60	156367.87	5.7535845	49067.10	54199.200	51633.150	2.75974122	85645.10	118050.00
30S ribosomal protein S18	158212.00	132008.00	164719.00	151646.33	5.5798548	224441.00	172462.000	198451.500	10.60703802	271032.00	141136.00
3-octaprenyl-4-hydroxybenzoate carboxy-lyase	96483.20	193657.00	NA	145070.10	5.3378811	22742.20	25591.000	24166.600	1.29168107	51830.40	42022.70
Major outer membrane lipoprotein Lpp	175726.00	125987.00	79429.00	127047.33	4.6747301	70295.10	85503.500	77899.300	4.16364118	89756.80	97404.30
30S ribosomal protein S5	134514.00	90640.00	139766.00	121640.00	4.4757662	127423.00	131296.000	129359.500	6.91413839	81496.70	121910.00
Uncharacterized protein YtfJ	66386.00	221590.00	74994.20	120990.07	4.4518518	24552.30	11258.100	17905.200	0.95701538	66718.40	44348.60
50S ribosomal protein L6	109709.00	113566.00	117466.00	113580.33	4.1792093	118385.00	119670.000	119027.500	6.36190313	133507.00	143474.00
50S ribosomal protein L1	118019.00	88608.40	132278.00	112968.47	4.1566956	153447.00	178906.000	166176.500	8.88197093	65379.40	92473.40
50S ribosomal protein L22	124923.00	83915.80	126313.00	111717.27	4.1106574	125403.00	158089.000	141746.000	7.57618467	65076.60	56825.30

Showing 1 to 16 of 1,210 entries, 27 total columns

Figure 2- the original loaded second data frame with protein name, ID and proteins deposited volume along with time at 30min, 1h, 2h, 3h and 4h. Each three repeated measurement values for each case and each time point along with intensity and amount at each time point.

Data Cleaning

Some of original data has a big variant in the three, and many detect result is zero or NA. The NA values was replaced with zero. Here the median value of the three was adopted. Drop the max and min value can better deal with random detect errors, for example, if twice measurements are showing zero, then it will be recognized as zero, and if only one measurement is zero, then the lower value of the other two will be adopted. Figures 3 and 4 show cleaned data frame 1 and data frame 2 respectively.

id	X5min	X15min	X30min	X3h	X6h	X18h	X24h
A0A075B6H7	290.457581	625.79285	458.263763	617.57648	611.960205	267.381653	382.479065
A0A075B6I0	293.416870	857.93866	1504.888794	51.56829	35.289623	0.000000	36.451237
A0A075B6J9	110.180557	189.44823	473.801880	306.96188	1043.298340	313.651489	1564.545654
A0A075B6K4	0.000000	0.00000	100.158310	43.91220	49.874077	35.789711	48.066071
A0A075B6K5	0.000000	0.00000	0.000000	0.00000	41.662849	61.799450	130.281006
A0A075B6R2	0.000000	0.00000	0.000000	0.00000	21.070240	37.714321	49.499237
A0A075B6R9	0.000000	0.00000	0.000000	114.01923	158.603546	194.581451	314.871490
A0A075B6S2	0.000000	0.00000	0.000000	0.00000	23.364923	0.000000	52.503548
A0A075B6S5	0.000000	0.00000	0.000000	132.67885	40.677856	261.490601	30.156401
A0A075B6S9	0.000000	0.00000	0.000000	0.00000	324.642334	471.122681	354.796265
A0A087WSY6	61.783802	113.51722	170.342911	421.41156	297.380371	256.297699	353.195068
A0A087WW87	167.952103	234.40208	325.652924	1933.03003	694.197449	2271.475586	245.599472
A0A0A0MRZ8	954.176331	1613.45801	3232.727539	382.75562	603.753174	421.200500	693.964661

Fig 3- the data frame after 1 with a single median value of each time point. There are 1500 rows and 11 columns.

First.Protein.Description	min_30	X30.min_Intensity	Amount..ng....10	hr_1	X1.h_Intensity	Amount..ng..
50S ribosomal protein L7/L12	300813.00	303753.67	11.1766722	218406.000	218406.000	11.6
30S ribosomal protein S20	269410.00	255284.00	9.3932218	401473.000	401473.000	21.4
Glyceraldehyde-3-phosphate dehydrogenase A	231843.00	254168.67	9.3521829	293990.000	293990.000	15.7
30S ribosomal protein S16	238119.00	226884.67	8.3482631	270112.500	270112.500	14.4
DNA-binding protein HU-alpha	222425.00	219643.33	8.0818169	300684.500	300684.500	16.0
Glucosamine-6-phosphate deaminase	232413.00	195880.67	7.2074652	55941.400	55941.400	2.9
Protein translocase subunit SecD	112253.00	167471.67	6.1621508	68121.700	68121.700	3.6
Modulator of FtsH protease HflK	108127.00	156367.87	5.7535845	51633.150	51633.150	2.7
30S ribosomal protein S18	158212.00	151646.33	5.5798548	198451.500	198451.500	10.6
3-octaprenyl-4-hydroxybenzoate carboxy-lyase	145070.10	145070.10	5.3378811	24166.600	24166.600	1.2
Major outer membrane lipoprotein Lpp	125987.00	127047.33	4.6747301	77899.300	77899.300	4.1
30S ribosomal protein S5	134514.00	121640.00	4.4757662	129359.500	129359.500	6.9
Uncharacterized protein YtfJ	74994.20	120990.07	4.4518518	17905.200	17905.200	0.9

Figure 4- the data frame 2 with a single median value of each time point. There are 1210 rows and 20 columns.

Data Normalization

The original continuous data do not follow the bell curve, we can log transform this data to make it as “normal” as possible so that the statistical analysis results from this data become more valid in other words, the log transformation reduces or removes the skewness of our original data which can be seen in figures 5 and 6 for data frame 1 and figures 7 and 8 for data frame 2.

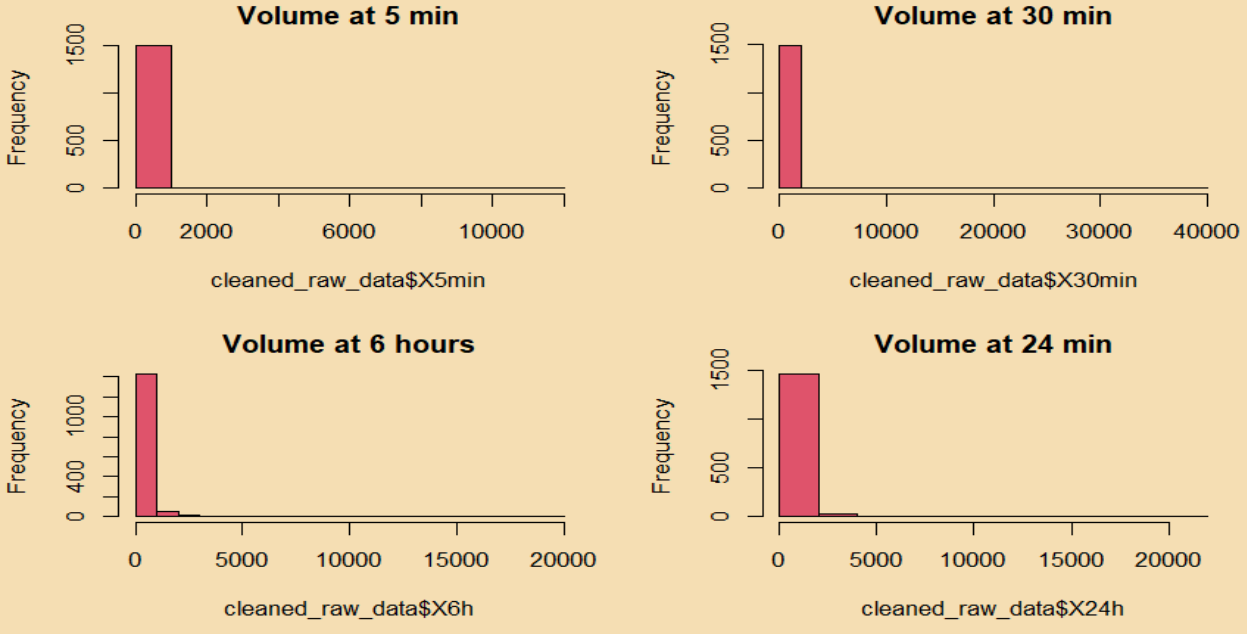

Figure -5 Histogram of the dataframe1 before log transformation is skewed.

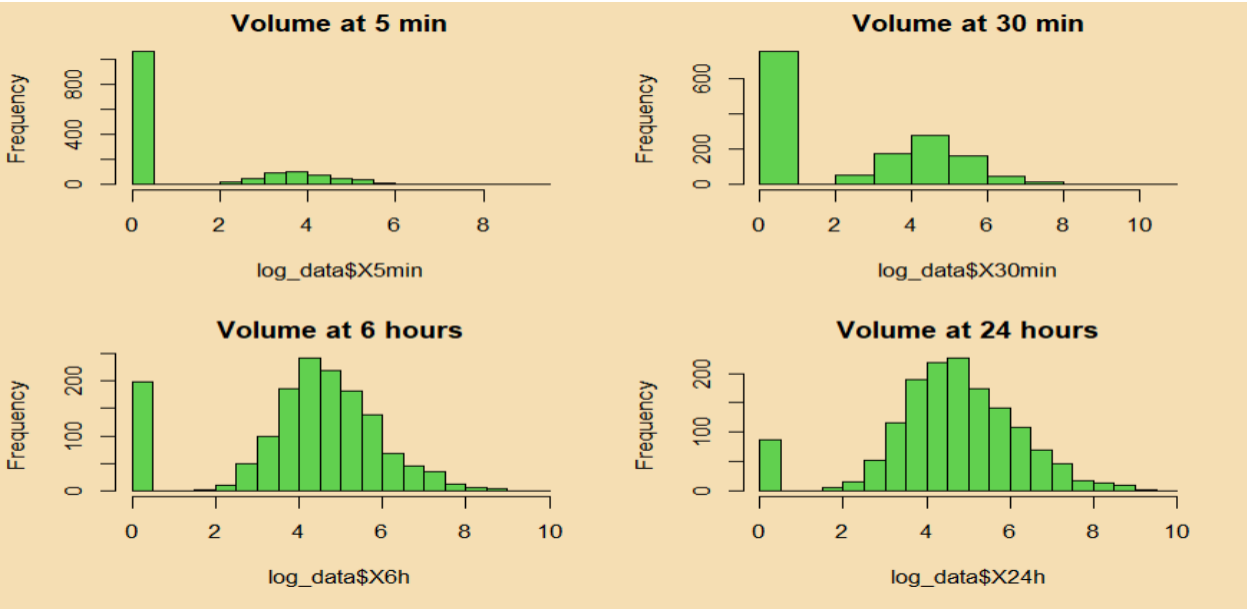

Figure -6 Histogram of the dataframe1 after log transformation shows the normal distribution

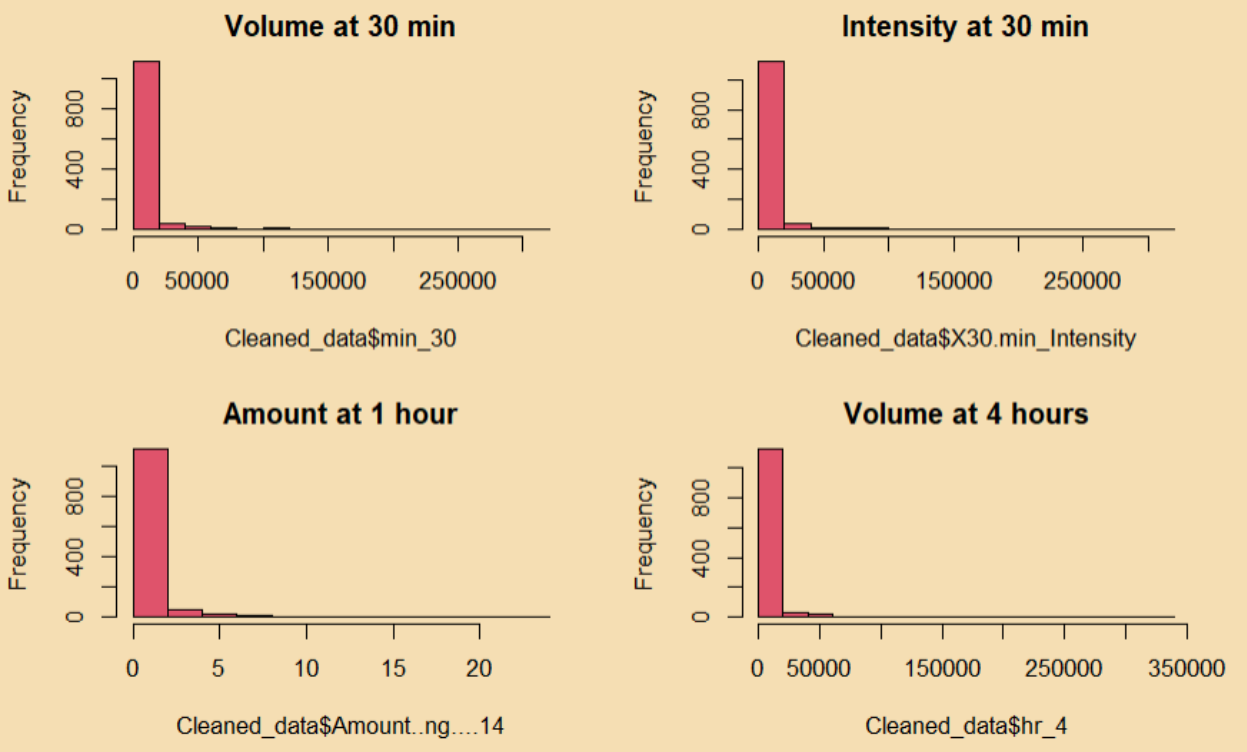

Figure -7 Histogram of the dataframe2 before log transformation is skewed.

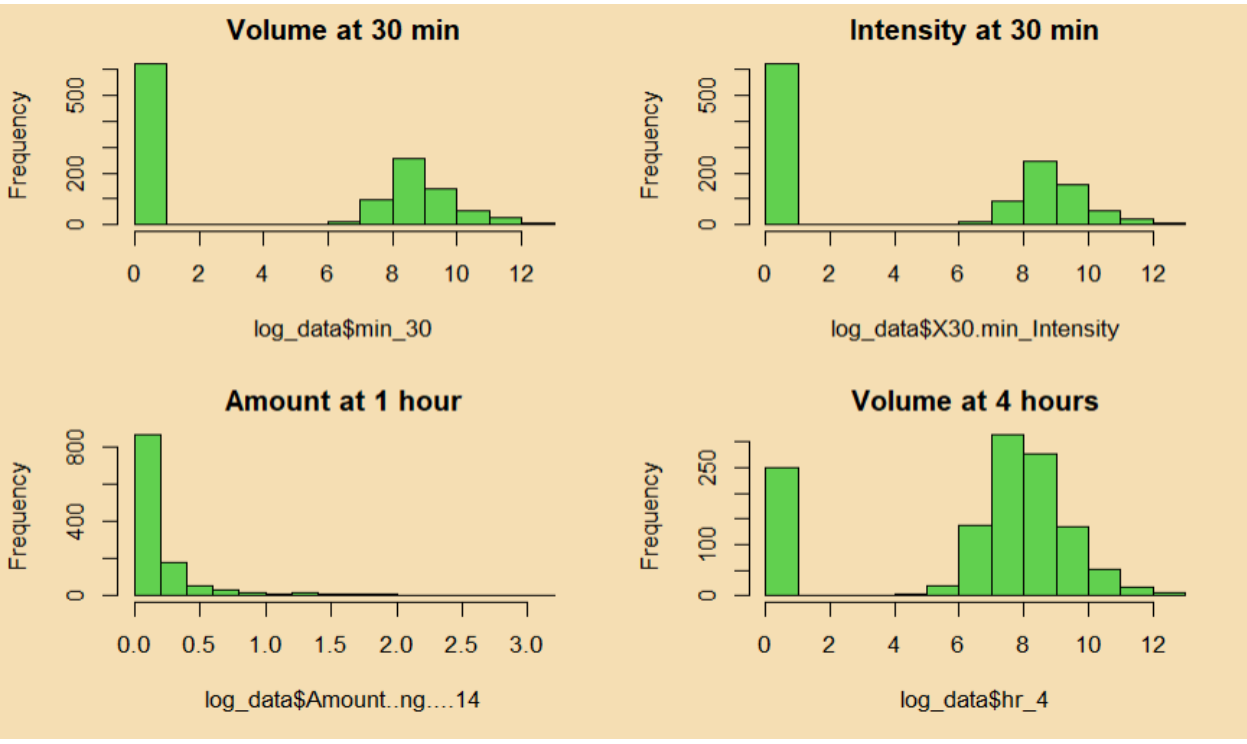

Figure -8 Histogram of the dataframe2 after log transformation shows the normal distribution

After performing the log transformation to eliminate the influence of the data scale, the data should be standardized further. Here, data has been standardized using the min-max standardization technique. The concept to standardize the data is [X-min(x)]/[max(x)-min(x)], then the data range will be converted to [0,1], shown in figure 9 and figure 10.

id	X5min	X15min	X30min	X3h	X6h	X18h	X24h
A0A075B6H7	0.5372848	0.6097814	0.5803375	0.6085321	0.6076686	0.5294754	0.5632634
A0A075B6I0	0.5382412	0.6396125	0.6927683	0.3751229	0.3400372	0.0000000	0.3430203
A0A075B6J9	0.4460404	0.4969982	0.5834877	0.5424998	0.6581128	0.5445343	0.6964467
A0A075B6K4	0.0000000	0.0000000	0.4370963	0.3602203	0.3720213	0.3413330	0.3685954
A0A075B6K5	0.0000000	0.0000000	0.0000000	0.0000000	0.3553557	0.3919597	0.4617743
A0A075B6R2	0.0000000	0.0000000	0.0000000	0.0000000	0.2929539	0.3461608	0.3713212
A0A075B6R9	0.0000000	0.0000000	0.0000000	0.4492541	0.4802698	0.4995163	0.5449007
A0A075B6S2	0.0000000	0.0000000	0.0000000	0.0000000	0.3023189	0.0000000	0.3767926
A0A075B6S5	0.0000000	0.0000000	0.0000000	0.4634880	0.3531442	0.5273740	0.3255978
A0A075B6S9	0.0000000	0.0000000	0.0000000	0.0000000	0.5477850	0.5829519	0.5561695
A0A087WSY6	0.3919361	0.4488400	0.4869895	0.5724182	0.5395073	0.5254822	0.5557425
A0A087WW87	0.4856591	0.5170617	0.5480784	0.7164588	0.6195881	0.7317268	0.5214614
A0A0A0MRZ8	0.6496671	0.6993594	0.7651258	0.5633316	0.6063923	0.5723709	0.6195564
A0A0A0MS15	0.0000000	0.0000000	0.0000000	0.4728556	0.4066862	0.4449204	0.4593302
A0A0A0MT36	0.0000000	0.0000000	0.0000000	0.0000000	0.5180340	0.5173780	0.5537916
A0A0B4J1V0	0.0000000	0.0000000	0.0000000	0.4573495	0.5030956	0.0000000	0.3458934
A0A0B4J1V6	0.0000000	0.0000000	0.0000000	0.0000000	0.2713028	0.3399199	0.3406381

Figure 9 Data frame 1 after min-max normalization scaled between 0 to 1.

min_30	X30.min_Intensity	Amount..ng....10	hr_1	X1.h_Intensity	Amount..ng....14	hr_2	X2.h_Intensity
0.9622451	0.9629871	0.19066954	0.9378246	0.9378246	0.193720680	0.9725188	0.9725188
0.9538346	0.9497262	0.17858883	0.9842640	0.9842640	0.237365039	1.0000000	1.0000000
0.9423790	0.9493922	0.17828703	0.9604949	0.9604949	0.214827668	0.9834453	0.9834453
0.9444165	0.9407299	0.17050570	0.9540333	0.9540333	0.208768396	0.9311524	0.9311524
0.9392156	0.9382556	0.16829990	0.9622125	0.9622125	0.216443542	0.9621877	0.9621877
0.9425663	0.9295213	0.16057782	0.8339251	0.8339251	0.105559163	0.8873917	0.8873917
0.8870514	0.9175686	0.15018558	0.8489518	0.8489518	0.117088758	0.8851734	0.8851734
0.8841947	0.9123354	0.14570498	0.8278119	0.8278119	0.101024605	0.8796308	0.8796308
0.9132298	0.9099966	0.14371701	0.9305160	0.9305160	0.187014818	0.9333948	0.9333948
0.9066147	0.9066147	0.14085884	0.7699006	0.7699006	0.063259898	0.8205210	0.8205210
0.8958561	0.8964954	0.13242800	0.8591828	0.8591828	0.125228389	0.8731732	0.8731732
0.9008517	0.8931776	0.12970542	0.8978712	0.8978712	0.157801648	0.8795227	0.8795227
0.8562836	0.8927689	0.12937154	0.7470261	0.7470261	0.051217570	0.8333669	0.8333669
0.8879384	0.8879481	0.12545803	0.8915214	0.8915214	0.152283963	0.9030741	0.9030741
0.8908723	0.8875360	0.12512571	0.9169764	0.9169764	0.174741006	0.8601820	0.8601820
0.8952091	0.8866864	0.12444162	0.9048465	0.9048465	0.163930047	0.8404674	0.8404674
0.8741548	0.8796445	0.11882961	0.9023152	0.9023152	0.161698222	0.7923422	0.7923422
0.8787354	0.8775949	0.11721628	0.7676014	0.7676014	0.061972474	0.8215636	0.8215636
0.8846379	0.8761901	0.11611585	0.9219855	0.9219855	0.179258007	0.8587634	0.8587634
0.8737552	0.8756186	0.11566943	0.8881722	0.8881722	0.149398350	0.9222186	0.9222186

Figure10 Data frame 2 after min-max normalization scaled between 0 to 1.

K-means for classification

It is hard to find the trend of deposit as it is not a simple linear relationship. To label the behavior of different cases, the k-means algorithm was chosen to perform this unsupervised clustering. To make the clustering result better stable nstart = 25 was added to the k-means function which will generate 25 initial configurations and then average all the centroid results.

The number of clusters (k) needs to be set before we start it can be advantageous to examine several different values of k. The results were plotted using the fviz_cluster() to get the visualization of clusters using several k values. The plots are shown in figure 11 and figure 12 for both data frames.

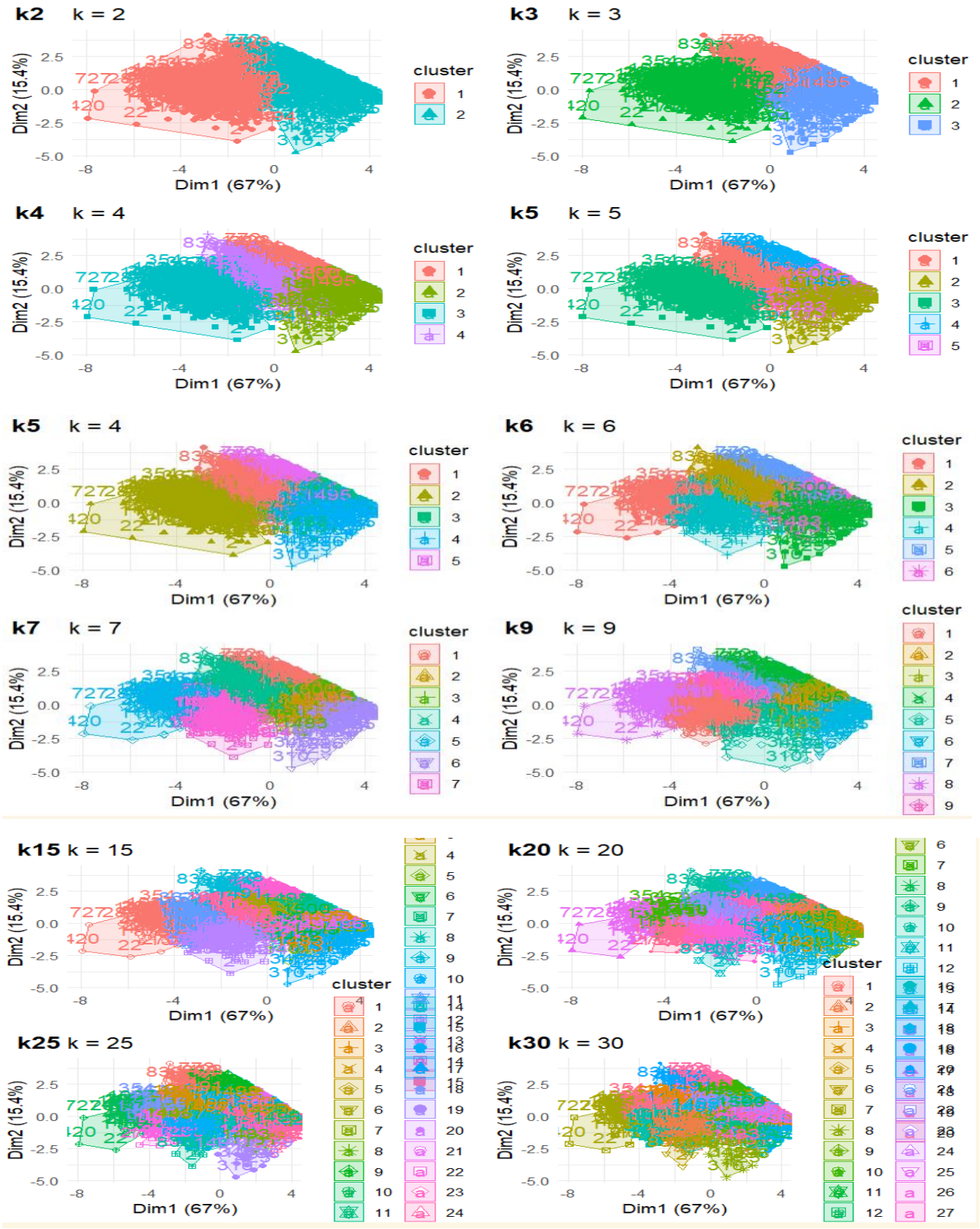

Figure 11 plots of clusters formed by using the k-means algorithm on Data Frame 1. The k values used for the plots is 2,3,4,5,6,7,9,15,20,25,30.

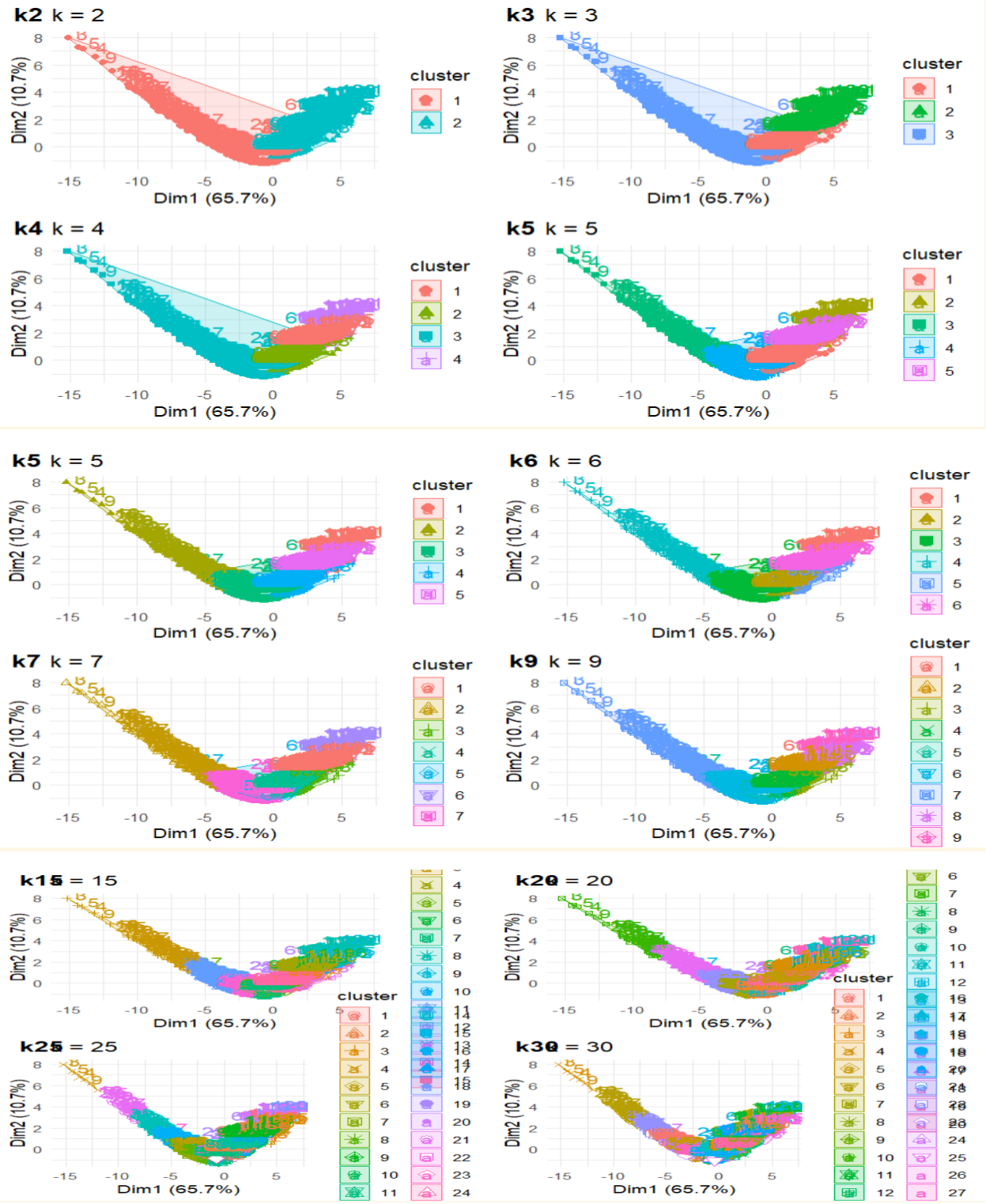

Figure 12 plots of clusters formed by using the k-means algorithm on Data Frame 2. The k values used for the plots is 2,3,4,5,6,7,9,15,20,25,30.

It is observed for data frame 1 that by setting the value k = 2 the K-means function separates cases into two classes. Class 1 has cases of about 534 & class 2 has a maximum of 966 of the total 1500 cases. When the value is set as k = 3, class 1, class 2, and class 3 have 617, 480, and 403 cases respectively. When the value is set as k =7, class 1 has maximum cases of about 377, class 4 & class 7 have 274 & 272 cases and the rest of the classes had cases below 200.

For data frame 2 by setting the value k = 2 the K-means function separates cases into two classes. Class 1 has cases of about 585 & class 2 has a maximum 625 of the total 1210 cases. When the value is set as k = 3, class 1, class 2, and class 3 have 374, 253, and 583 cases respectively. When the value is set as k =7, class 3 has maximum cases of about 464, class 7 has 220 cases and the rest of the classes had cases below 150 cases.

Optimal value of k

Choosing the optimal number of k value while using the k-means algorithm plays a vital role in the overall clustering process. There are several methods to determine the optimal k values.

The Elbow Method

In cluster analysis, the elbow method is one of the well-known methods utilized in determining the optimal number of clusters in a dataset. This method consists of plotting the explained variation as a function of the number of clusters. The number of clusters is often determined by searching for a change of slope from steep to shallow. By this method, from the plot, a point is chosen where the sum of squares value begins to decrease gradually. If the number of clusters is added, it either doesn’t give a better result or may degrade the result. The best value for k determined by this method is 7. Figure 13 and Figure 14 show the plot for the elbow method.

Function used to plot the Elbow method:

```
fviz_nbclust(norm_data[,5:11], kmeans, method = "wss", k.max = 24) + theme_minimal() + ggtitle("The Elbow Method")
```

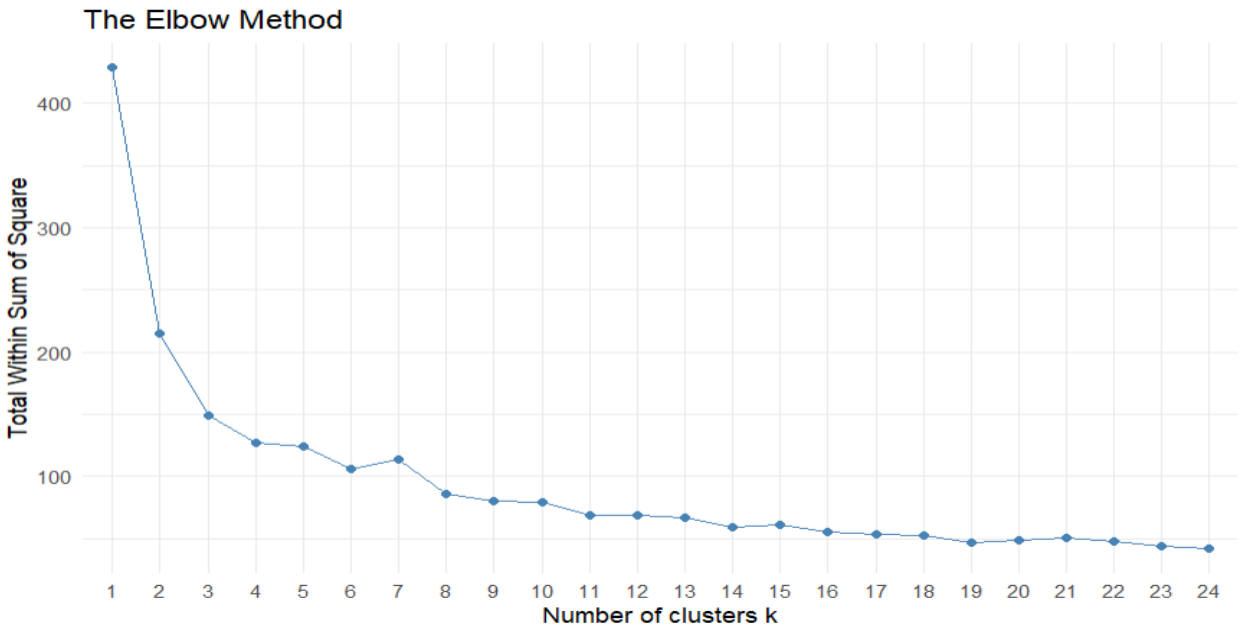

Figure 13 shows the elbow method plot for data frame 1 with number of clusters on X axis and total within sum of square on the Y axis

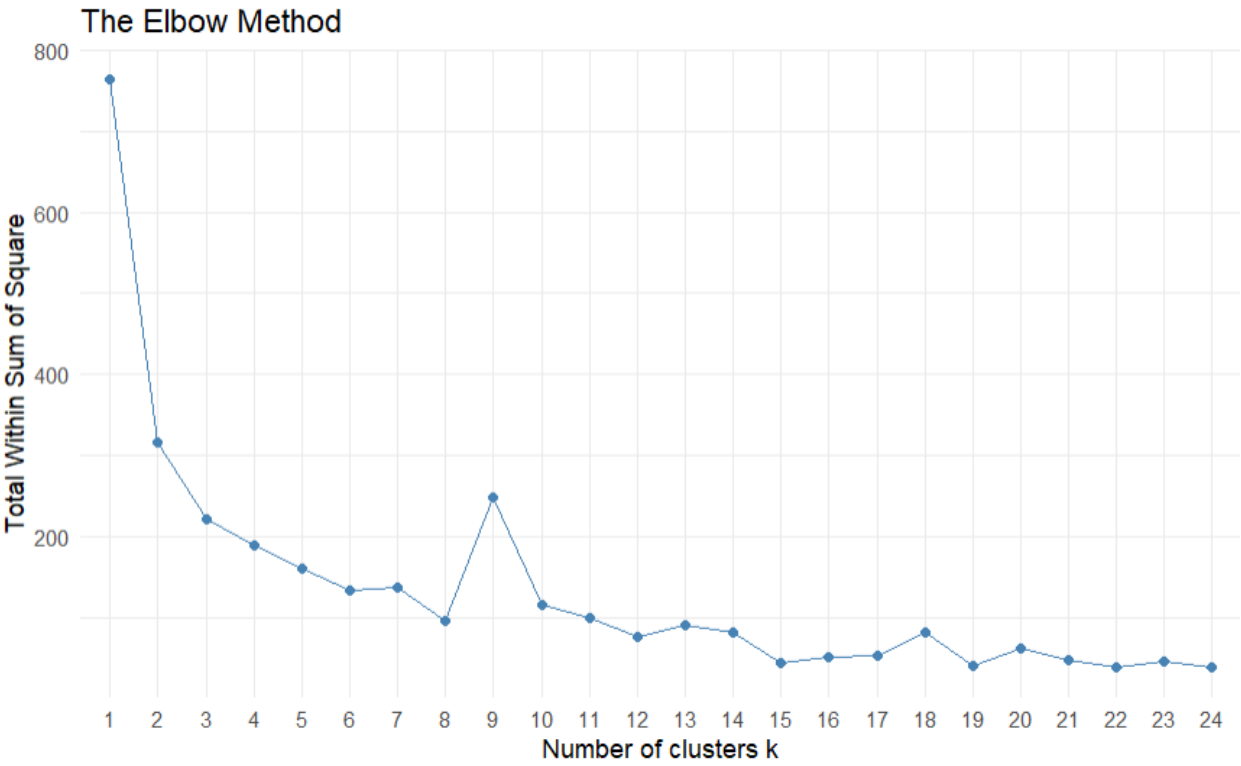

Figure 14 shows the elbow method plot for data frame 2 with number of clusters on X axis and total within sum of square on the Y axis

The Silhouette Plot

The silhouette method is also used to find the optimal number of clusters. It computes silhouette coefficients of each and every point that measure the similarity of a point to its own cluster. After computing the silhouette coefficients for each point its average is calculated to get the silhouette score. The value of the silhouette score lies between [1, -1]. A value close to 1 indicates the object is much similar to its own cluster and not similar to other clusters. The highest silhouette score value can be seen at k = 2 but a considerable silhouette score is at k = 8 & k = 9 which are very close to k =7 derived from the previous elbow method. The plots for the Silhouette score are shown in figure 15 and figure 16.

Function used to plot the Silhouette method:

fviz_nbclust(norm_data[,5:11], kmeans, method = "silhouette", k.max = 24) + theme_minimal() + ggtitle("The Silhouette Plot")

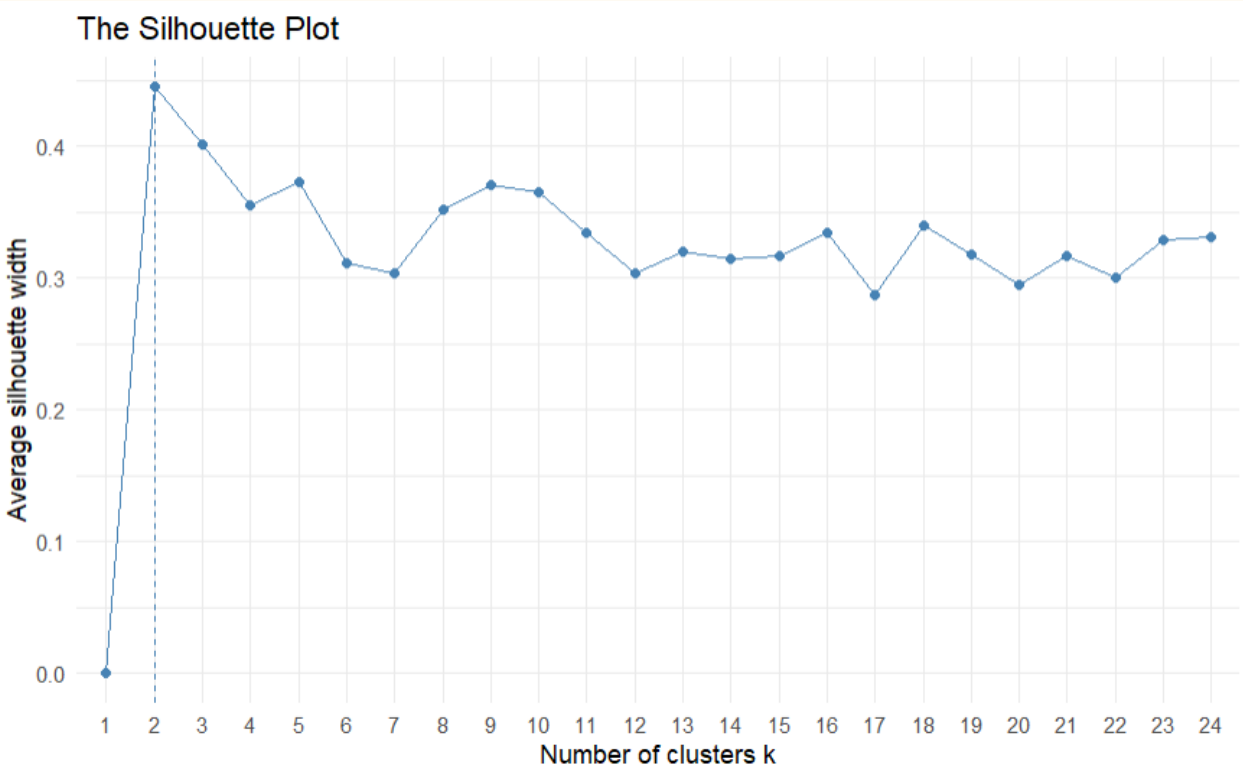

Figure 15 shows the silhouette plot for dataframe1 with number of clusters on X axis and average silhouette width on the Y axis

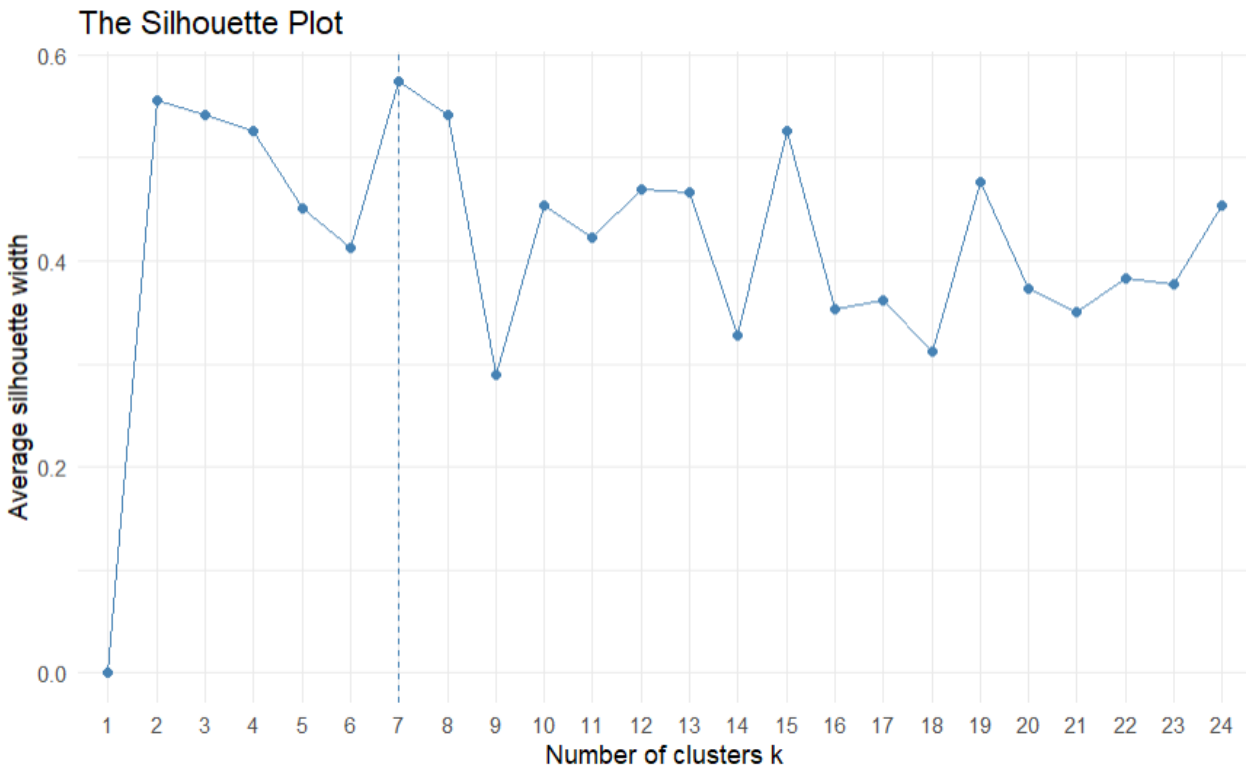

Figure 16 shows the silhouette plot for dataframe2 with number of clusters on X axis and average silhouette width on the Y axis

The Sum of Squares Method

The sum of squares method is used to find the optimized number of clusters by finding out the minimum sum of square distance within the cluster which represents the tightness of clusters and finding the maximum sum of squares distances between the clusters which shows how far apart the clusters are with each other. By using this method, the optimal value of k lies around 7 or 9. The plots for sum of sum of squares distance are shown in figure 17 and figure 18.

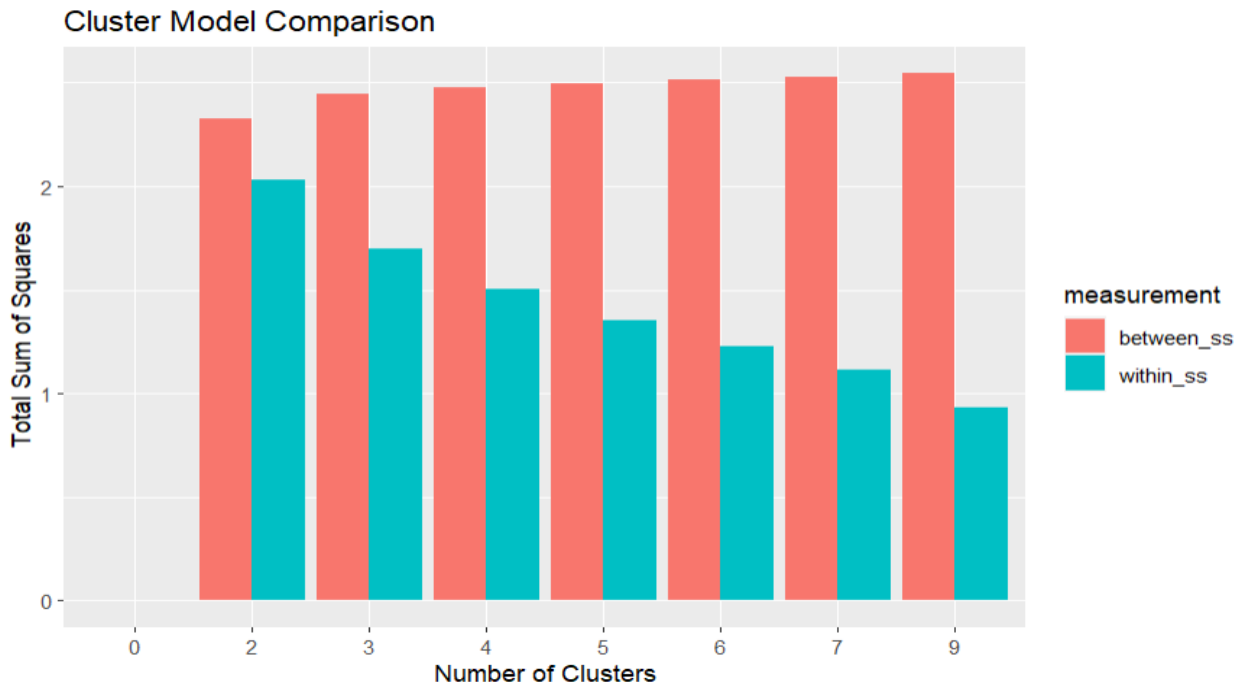

Figure 17 shows the bar plot of cluster model comparison using data frame 1. The X axis represents the total sum of squares and Y axis is for the number of clusters. The blue bars show the total sum of squares within the clusters while the pink bars show the total sum of squares between the clusters.

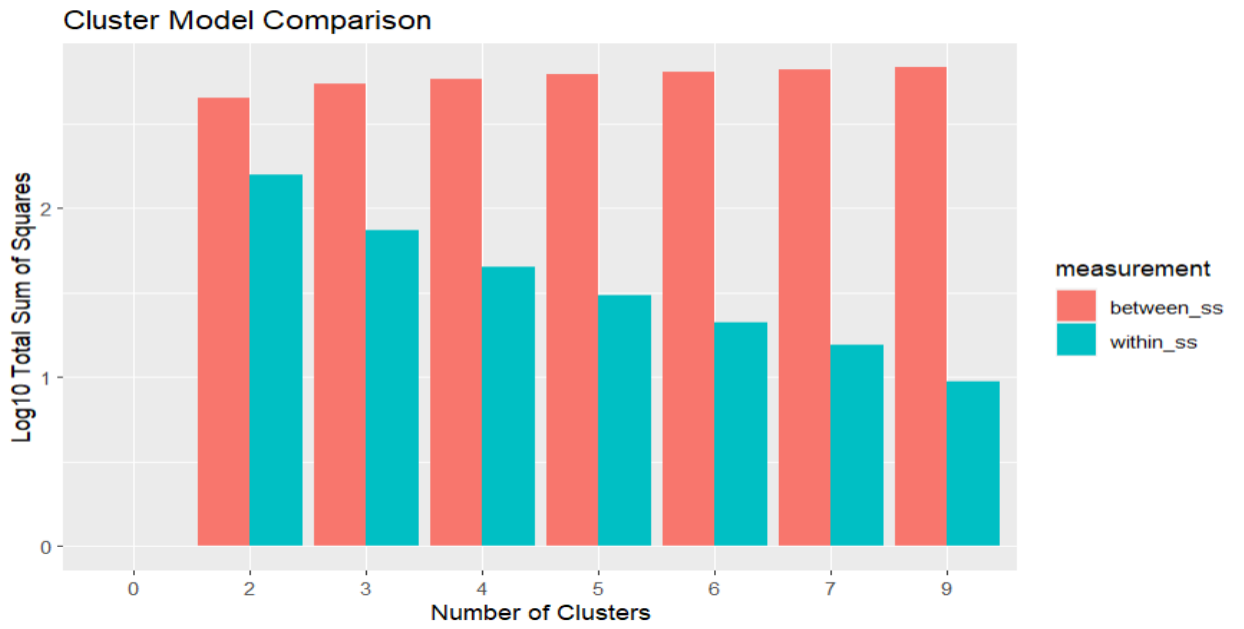

Figure 18 shows the bar plot of cluster model comparison using data frame 2. The X axis represents the total sum of squares and Y axis is for the number of clusters. The blue bars show the total sum of squares within the clusters while the pink bars show the total sum of squares between the clusters.

Based on the above three methods to find the optimal number of clusters the finalized value for the cluster was chosen as k = 7 for both the data frames.

Data Labeling

Data labeling is the process of identifying the patterns of the raw data and adding more informative labels to provide context so that supervised machine learning algorithms can be deployed to make a model from it. A labeled set of data is necessary for supervised learning algorithms to create a model to make correct decisions. Here, all the proteins of both data frames are labeled with the 7 classes determined by the k-means clustering method. Figure 19 and Figure 20 shows the labeled data for both the data frames.

name	id	X5min	X15min	X30min	X3h	X6h	X18h	X24h	clusters
IGKV3-7	A0A075B6H7	291.457581	626.79285	459.263763	618.57648	612.960205	268.381653	383.479065	7
IGLV8-61	A0A075B6I0	294.416870	858.93866	1505.888794	52.56829	36.289623	1.000000	37.451237	5
IGLV2-18	A0A075B6J9	111.180557	190.44823	474.801880	307.96188	1044.298340	314.651489	1565.545654	7
IGLV3-10	A0A075B6K4	1.000000	1.00000	101.158310	44.91220	50.874077	36.789711	49.066071	2
IGLV3-9	A0A075B6K5	1.000000	1.00000	1.000000	1.00000	42.662849	62.799450	131.281006	3
IGHV4-4	A0A075B6R2	1.000000	1.00000	1.000000	1.00000	22.070240	38.714321	50.499237	3
IGKV2D-24	A0A075B6R9	1.000000	1.00000	1.000000	115.01923	159.603546	195.581451	315.871490	6
IGKV2D-29	A0A075B6S2	1.000000	1.00000	1.000000	1.00000	24.364923	1.000000	53.503548	3
IGKV1-27	A0A075B6S5	1.000000	1.00000	1.000000	133.67885	41.677856	262.490601	31.156401	6
IGKV1-37	A0A075B6S9	1.000000	1.00000	1.000000	1.00000	325.642334	472.122681	355.796265	3
IGKV3D-15	A0A087WSY6	62.783802	114.51722	171.342911	422.41156	298.380371	257.297699	354.195068	7
IGKV2-40	A0A087WW87	168.952103	235.40208	326.652924	1934.03003	695.197449	2272.475586	246.599472	7
IGKV3D-11	A0A0A0MRZ8	955.176331	1614.45801	3233.727539	383.75562	604.753174	422.200500	694.964661	7
IGHV3-49	A0A0A0MS15	1.000000	1.00000	1.000000	147.58173	73.368103	109.873131	127.935349	6
IGKV6D-21	A0A0A0MT36	1.000000	1.00000	1.000000	1.00000	237.832031	236.189850	346.971558	3
IGHV3-15	A0A0B4J1V0	1.000000	1.00000	1.000000	125.28658	203.117050	1.000000	38.605164	6
IGHV3-73	A0A0B4J1V6	1.000000	1.00000	1.000000	1.00000	17.558647	36.244675	36.520657	3
IGHV3-43	A0A0B4J1X8	1.000000	1.00000	75.065056	303.37695	337.526550	1.000000	284.331635	2
IGHV3-72	A0A0B4J1Y9	1.000000	556.30774	307.369202	652.62537	633.015625	266.358368	834.634582	5
IGKV1D-13	A0A0B4J2D9	1.000000	1.00000	1.000000	1.00000	1690.563232	665.573059	1.000000	3
IGKV6-21	A0A0C4DH24	1.000000	1.00000	1.000000	116.76285	90.192993	131.159698	1.000000	6

Figure 19 shows the column "clusters" of data frame 1 that contains the class label of all the 1500 proteins.

Both the data frames are divided into training data set and test data set. The proportion of data frame 1 class labels for each dataset is given below.

Training Data Labels

Classes	1	2	3	4	5	6	7
Proportion	0.11	0.15	0.05	0.17	0.25	0.15	0.12

Test Data Labels

Classes	1	2	3	4	5	6	7
Proportion	0.10	0.19	0.04	0.12	0.29	0.17	0.09

Amount..ng....18	hr_3	X3.h_Intensity	Amount..ng....23	hr_4	X4.h_Intensity	Amount..ng....27	clusters
0.00000000	951.7790	1190.0797	0.10751241	0.0000	0.0000	0.000000000	5
0.15453349	1279.2100	1182.9723	0.10687033	0.0000	0.0000	0.000000000	1
0.11098047	1048.5000	1175.4523	0.10619097	0.0000	0.0000	0.000000000	1
0.19052032	1170.7800	1170.7800	0.10576887	0.0000	0.0000	0.000000000	2
0.11503894	1164.1500	1164.6900	0.10521870	0.0000	0.0000	0.000000000	2
0.08574202	1159.4020	1159.4020	0.10474098	0.0000	0.0000	0.000000000	2
0.11847631	1147.9200	1147.9200	0.10370369	0.0000	0.0000	0.000000000	2
0.08395384	1221.5800	1145.0027	0.10344013	0.0000	0.0000	0.000000000	2
0.06756216	1143.0320	1143.0320	0.10326210	0.0000	0.0000	0.000000000	1
0.00000000	1138.9825	1138.9825	0.10289627	0.0000	0.0000	0.000000000	5
0.11672629	1132.1250	1132.1250	0.10227676	0.0000	0.0000	0.000000000	2
0.12555094	1129.9050	1129.9050	0.10207620	0.0000	0.0000	0.000000000	2
0.05998168	1055.0800	1105.8333	0.09990156	0.0000	0.0000	0.000000000	1
0.15520353	1104.5000	1104.5000	0.09978110	0.0000	0.0000	0.000000000	2
0.07373822	913.2740	1069.9287	0.09665791	0.0000	0.0000	0.000000000	1
0.08907261	1053.7300	1053.7300	0.09519451	0.0000	0.0000	0.000000000	2
0.05378398	1042.8000	1042.8000	0.09420709	0.0000	0.0000	0.000000000	1
0.12321105	984.8590	1036.7500	0.09366053	0.0000	0.0000	0.000000000	1
0.10432495	1001.8000	1029.4570	0.09300168	0.0000	0.0000	0.000000000	1
0.00000000	1018.2900	1018.2900	0.09199285	0.0000	0.0000	0.000000000	5

Figure 20 shows the column "clusters" of data frame 2 that contains the class label of all the 1210 proteins.

The proportion of data frame 2 class labels for each dataset is given below.

Training Data Labels

Classes	1	2	3	4	5	6	7
Proportion	0.08	0.18	0.09	0.09	0.38	0.10	0.06

Test Data Labels

Classes	1	2	3	4	5	6	7
Proportion	0.08	0.17	0.14	0.10	0.38	0.08	0.05

Random Forest

The random forest is an algorithm based on the ensemble method which is built on the idea of bagging. It is used for classification and regression by constructing multiple decision trees at the time of training. It generates a number of training datasets from the original training data. At every internal node of decision trees, it randomly does the sample of all the attributes and chooses the attributes that give the highest info gain. These datasets are then used to generate a set of decision trees. The model's predictions are combined using voting (for classification) or averaging (for numerical prediction).

The random forest algorithm is used to create the model for data frame 1. For creating the model, the algorithm used 500 different trees and the number of variables tried at each split was 2. The mean squared residuals are 0.227 and the overall accuracy of the model is 86.67%. The model has been created 3 times to get the average accuracy, and the average accuracy calculated is 86%. The various other details of the model are shown in figure 21. The plot of the predicted labels and ground truth is shown in figure 22.

```
Call:
 randomForest(formula = clusters ~ ., data = train_data[, 5:12])
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 0.2276266
                    % Var explained: 94.35
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3  4  5  6  7
         1 13  0  0  0  0  0  0
         2  1 13  1  0  0  0  0
         3  0  2 26  1  0  0  0
         4  0  0  0  4  0  0  0
         5  0  0  1  1 15  5  0
         6  0  0  0  0  3 39  5
         7  0  0  0  0  0  0 20

Overall Statistics

               Accuracy : 0.8667
                 95% CI : (0.8016, 0.9166)
    No Information Rate : 0.2933
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.836
```

Figure 21 The summary of the output of random forest model based on data frame 1.

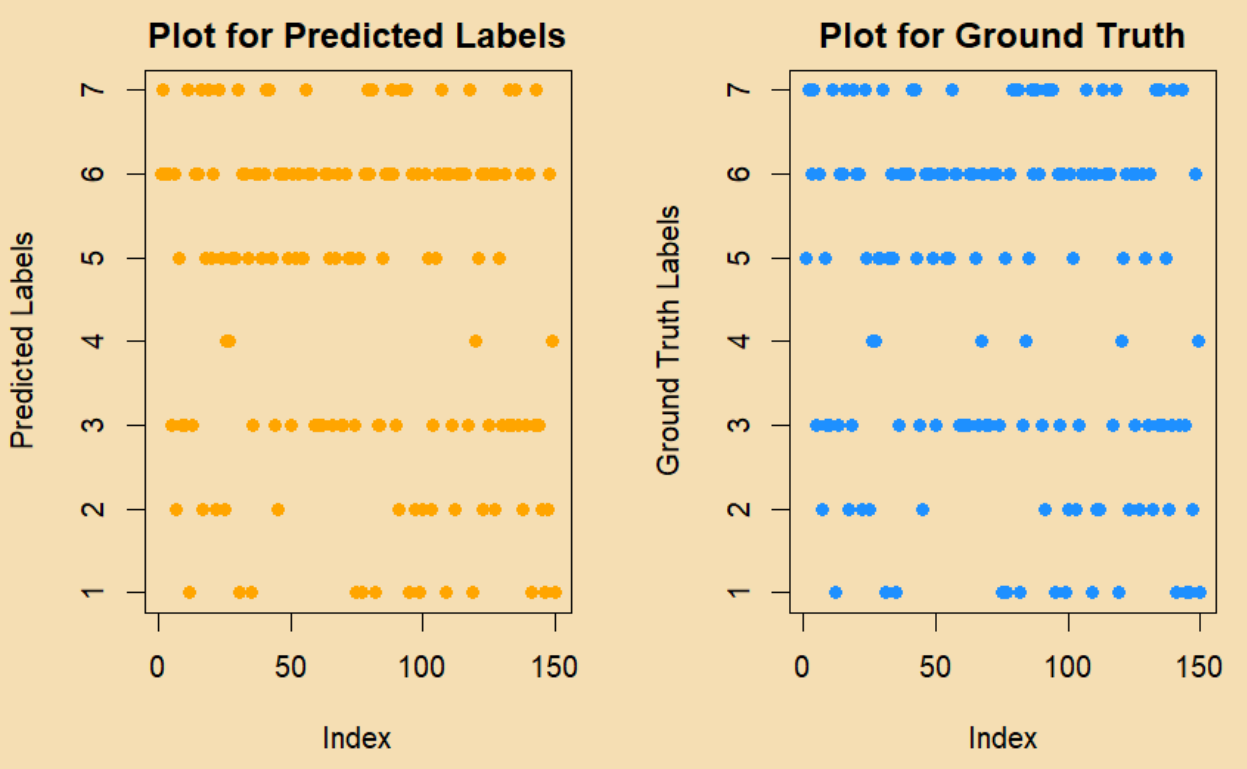

Figure 22 Side by side plots of the predicted labels using random forest algorithm using data frame 1 with the ground truth.

The random forest algorithm is used to create the model for data frame 2. For creating the model, the algorithm used 500 different trees, and the number of variables tried at each split was 5. The mean squared residuals are 0.036 and the overall accuracy of the model is 94.21%. The model has been created 3 times to get the average accuracy, the average accuracy calculated is 95.59%. The various other details of the model are shown in figure 23. The plot of the predicted labels and ground truth is shown in figure 24.

```
Call:
 randomForest(formula = clusters ~ ., data = train_data[, 6:21])
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 5

          Mean of squared residuals: 0.03609935
                    % Var explained: 98.88
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3  4  5  6  7
         1  6  0  0  0  0  0  0
         2  0 10  0  0  0  0  0
         3  0  0 20  0  0  0  0
         4  0  0  0 14  0  0  0
         5  0  0  0  3 12  3  0
         6  0  0  0  0  0 43  1
         7  0  0  0  0  0  0  9

Overall Statistics

               Accuracy : 0.9421
                 95% CI : (0.8844, 0.9764)
    No Information Rate : 0.3802
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9266
```

Figure 23 The summary of the output of random forest model based on data frame 2.

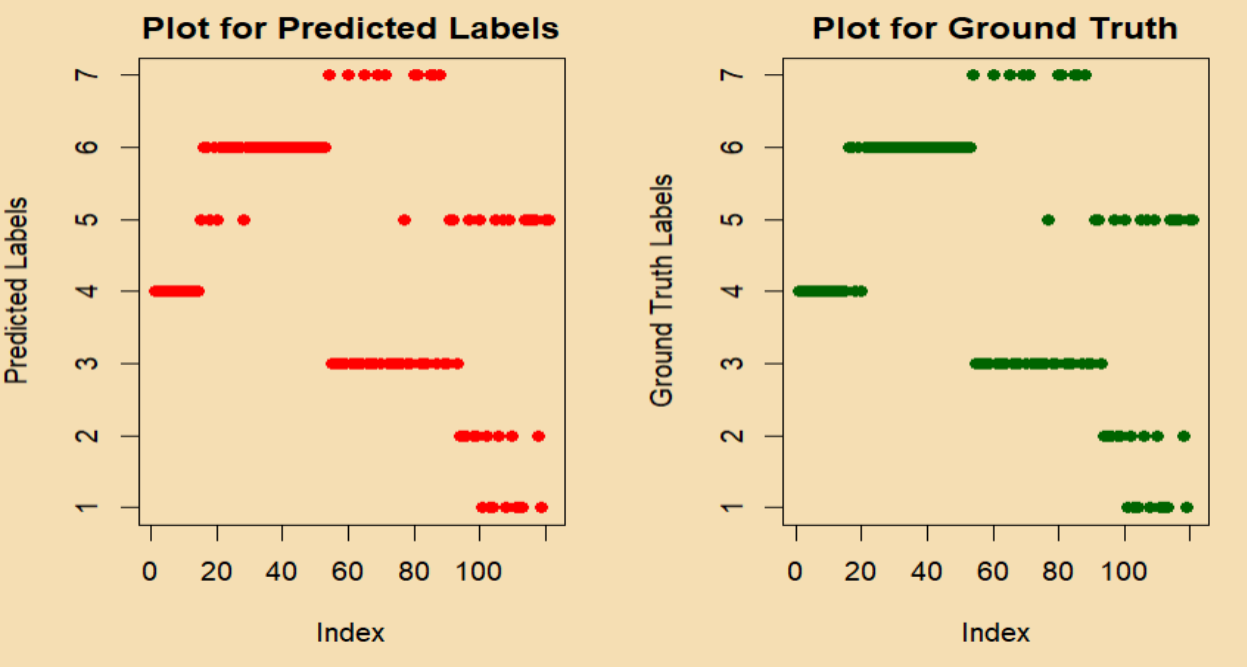

Figure 24 Side by side plots of the predicted labels using random forest algorithm using data frame 2 with the ground truth.

Artificial Neural Network (ANN)

The artificial neural network algorithm is inspired by attempts to emulate biological neural systems. The ANN model involves computations that simulates human brain processes. The ANN models consist of nodes in a complex and nonlinear form and can have multiple hidden layers. The nodes are connected to each other by weighted links. ANN model is trained by feeding a large amount of data, along with the input, output is also provided to the model so that it can learn the underlying pattern. The model adjusts its internal weights to learn the pattern present in the training data. Once the model is created, it can be deployed for the test data set to get the predicted values. It can be used for both classification and regression problems.

Here, the artificial neural network algorithm is used for the classification of the labels of dataframe1. The hidden layer for the ANN model can be adjusted through the code. Several numbers of hidden layers were adjusted to get the most accurate model. The accuracy of the model with one hidden layer was 60%. The accuracy of the model with 2 hidden layers has reached 73.5%. The layer has been increased up to 6 but the accuracy of the model hasn't been improved much. The plot for the neural network with 2 hidden layers is shown below in figure 25. The plot of the predicted labels and ground truth is shown in figure 26.

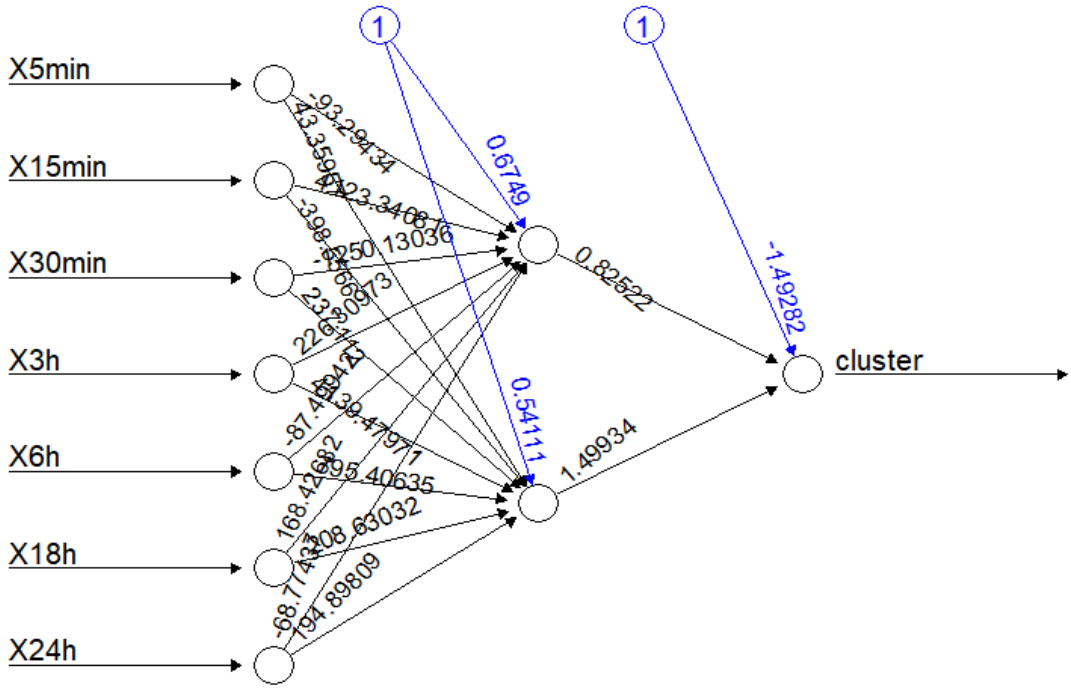

Figure 25: The neural network plot with 2 hidden layers for the data frame 1.

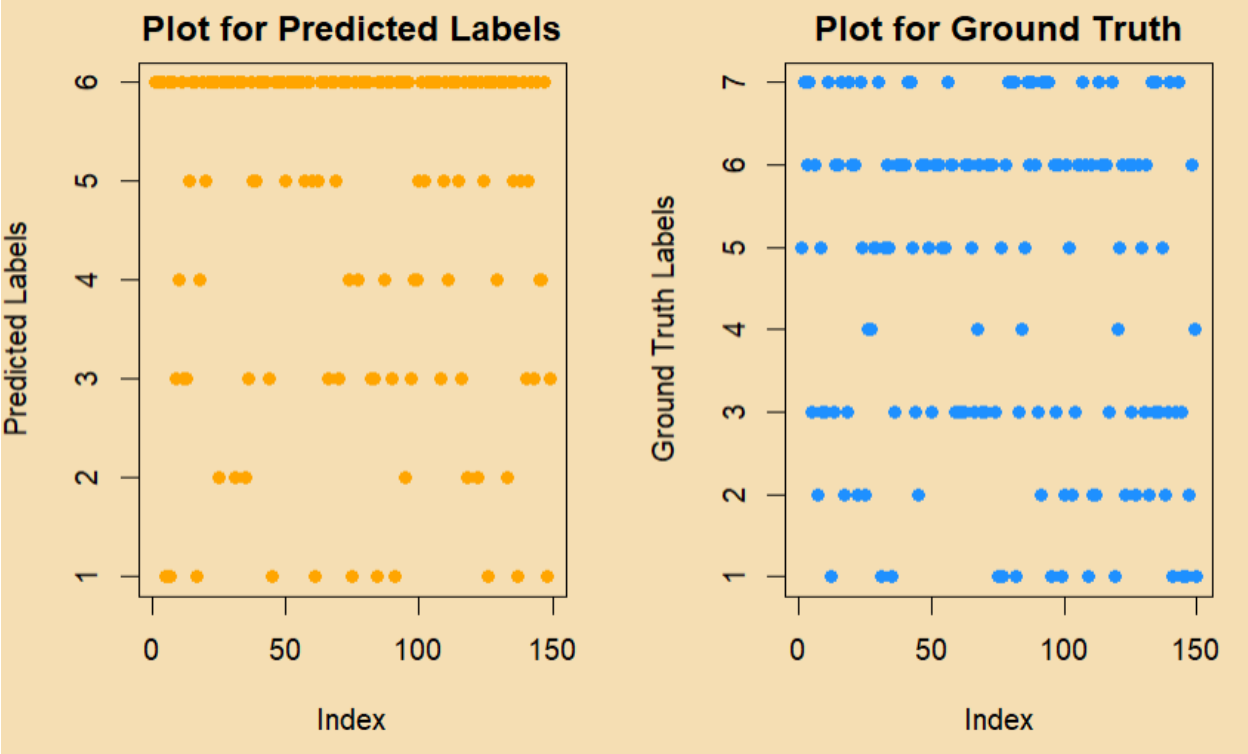

Figure 26 Side by side plots of the predicted labels using artificial neural network using data frame 1 with the ground truth.

Here, the artificial neural network algorithm is used the classification of the labels of dataframe2. The hidden layer for the ANN model can be adjusted through the code. Several numbers of hidden layers were adjusted to get the most accurate model. The accuracy of the model with one hidden layer was 41.4%. The accuracy of the model with 2 hidden layers has reached to 77.5%. The maximum accuracy of the model is 88.7% which is observed with 5 hidden layers. The plot for the neural network with 5 hidden layers is shown below in figure 27. The plot of the predicted labels and ground truth is shown in figure 28.

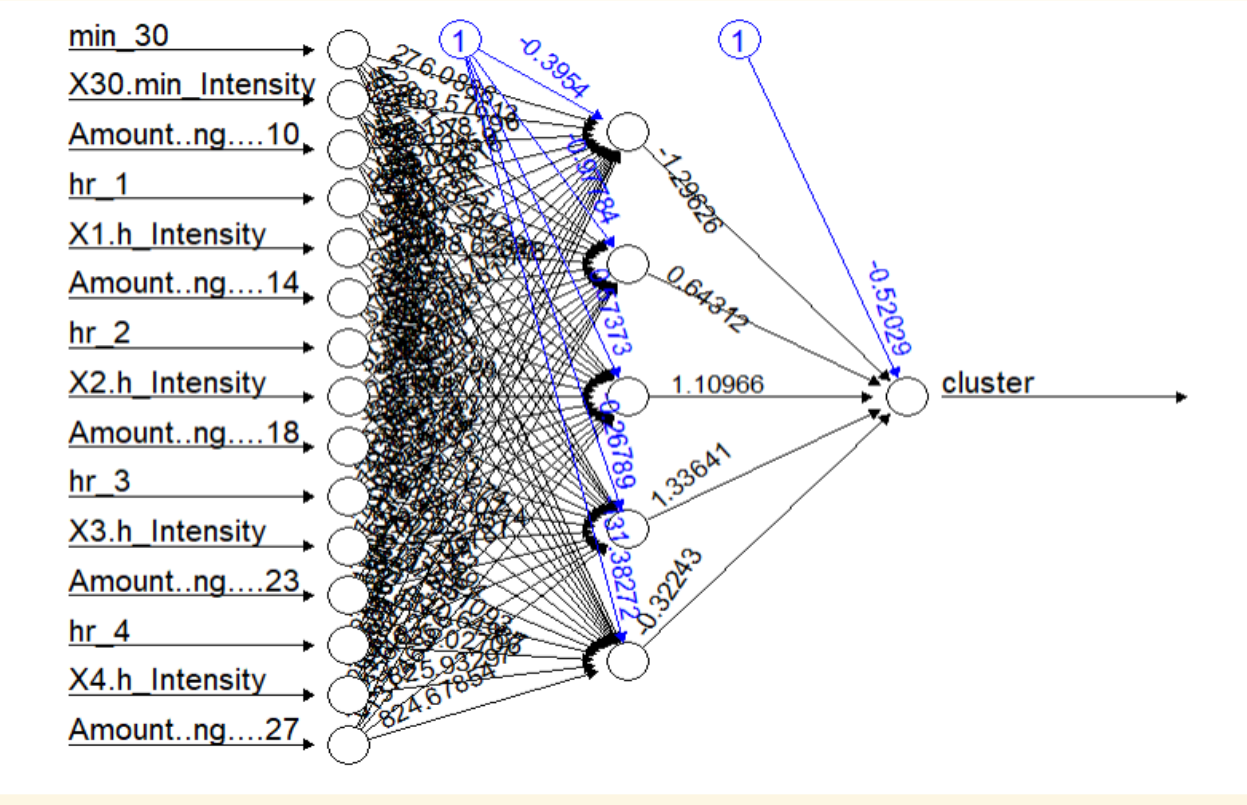

Figure 27 The neural network plot with 5 hidden layers for the data frame 2

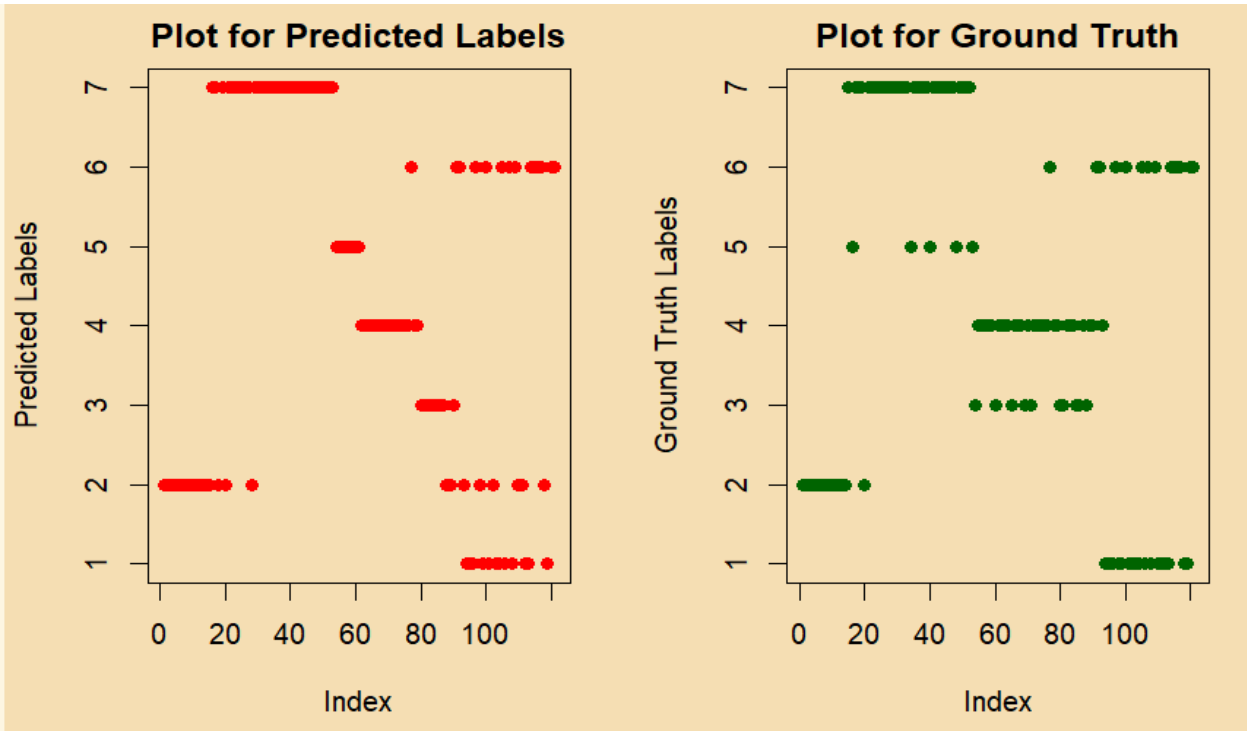

Figure 28 Side by side plots of the predicted labels an using artificial neural network using data frame 2 with the ground truth.

Support Vector Machine (SVM)

The support vector machine algorithm is a supervised learning algorithm used for classification, regression, and outlier detection. The objective of the support vector machine algorithm is to find a hyperplane (decision boundary) in a multi-dimensional space that distinctly classifies the data points. The mathematical definition of a hyperplane is quite simple. In two dimensions, a hyperplane is defined by the equation for the parameters β0, β1, and β2.

$$\beta 0 + \beta 1X1 + \beta 2X2 = 0$$

There are various SVM kernels available for the classification of data points. Here for both the data frames linear kernel "Vanilla" and non-linear kernel "Gaussian radial basis" were used to classify the labels.

1. Linear Kernel

The linear kernel is used when the data is linearly separable so that it can be separated using a single line. It is one of the most common kernels to be used. Here, for the dataframe1 linear kernel "vanilla" is used, creating 1155 support vectors to do the classification. The linear kernel has wrongly classified 38.67% (error rate) of the labels and the accuracy is 61.33 %. It wasn't able to classify any labels of class 2 and class 3. The cross table of the output is shown in figure 29. The plot demonstrating the predicted labels and the ground truth is shown in figure 30.

	predicted cluster					
actual cluster	1	4	5	6	7	Row Total
1	12	0	0	2	0	14
	0.080	0.000	0.000	0.013	0.000	
2	0	0	0	15	0	15
	0.000	0.000	0.000	0.100	0.000	
3	3	0	0	25	0	28
	0.020	0.000	0.000	0.167	0.000	
4	0	2	3	1	0	6
	0.000	0.013	0.020	0.007	0.000	
5	0	0	15	2	1	18
	0.000	0.000	0.100	0.013	0.007	
6	0	0	1	43	0	44
	0.000	0.000	0.007	0.287	0.000	
7	0	1	4	0	20	25
	0.000	0.007	0.027	0.000	0.133	
Column Total	15	3	23	88	21	150

Figure 29 Cross table of the labels predicted by SVM using linear kernel for the data frame 1

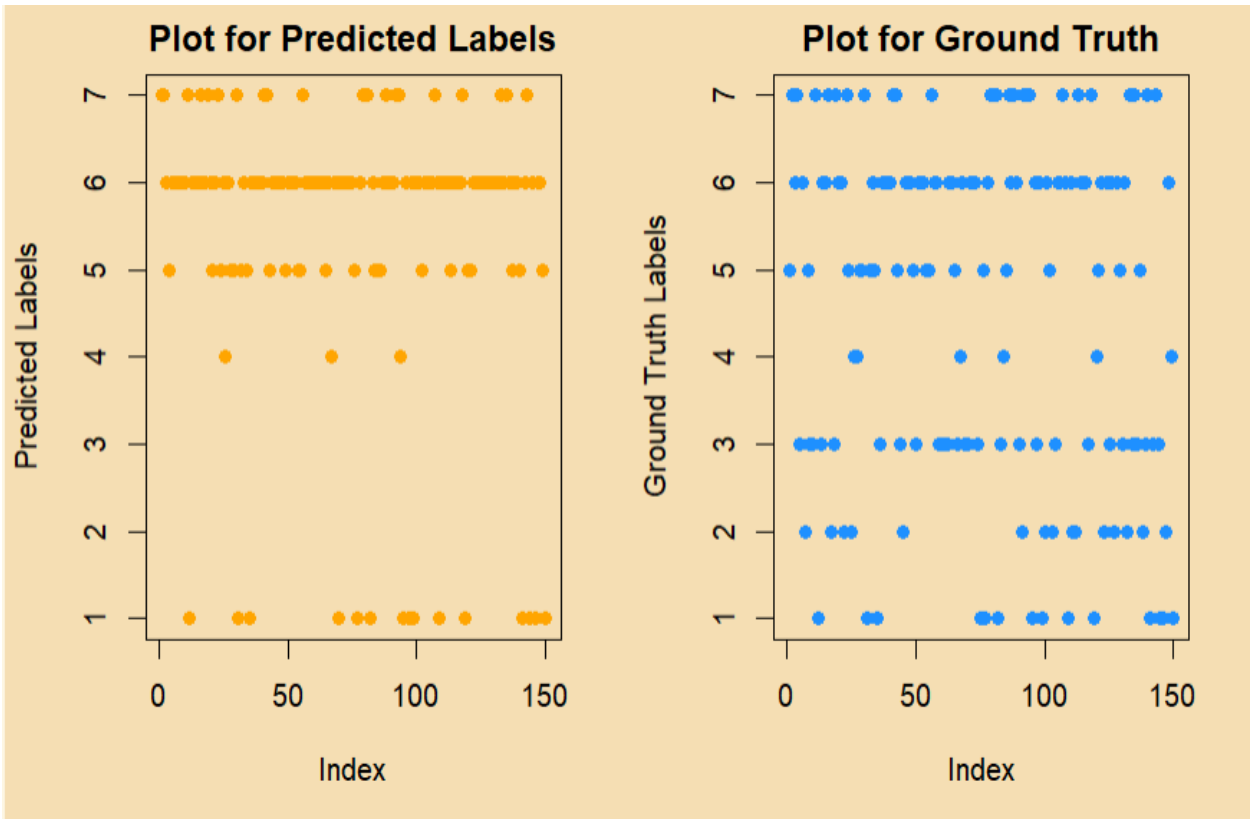

Figure 30 Side by side plots of the predicted labels using SVM algorithm using data frame 1 with the ground truth.

Here, for the dataframe2 linear kernel “vanilla” is used creating 652 support vectors to do the classification. The linear kernel has wrongly classified 27.27% (error rate) of the labels and the accuracy is 72.73 %. It wasn’t able to classify any labels of class 1 and class 2. The cross table of the output is shown in figure 31. The plot demonstrating the predicted labels and the ground truth is shown in figure 32.

	predicted cluster				
actual cluster	3	4	5	6	Row Total
1	0 0.000	0 0.000	8 0.066	0 0.000	8
2	6 0.050	0 0.000	3 0.025	0 0.000	9
3	23 0.190	0 0.000	4 0.033	0 0.000	27
4	0 0.000	16 0.132	0 0.000	1 0.008	17
5	0 0.000	0 0.000	14 0.116	0 0.000	14
6	0 0.000	1 0.008	0 0.000	35 0.289	36
7	7 0.058	0 0.000	3 0.025	0 0.000	10
Column Total	36	17	32	36	121

Figure 31 Cross table of the labels predicted by SVM using linear kernel for the data frame 2.

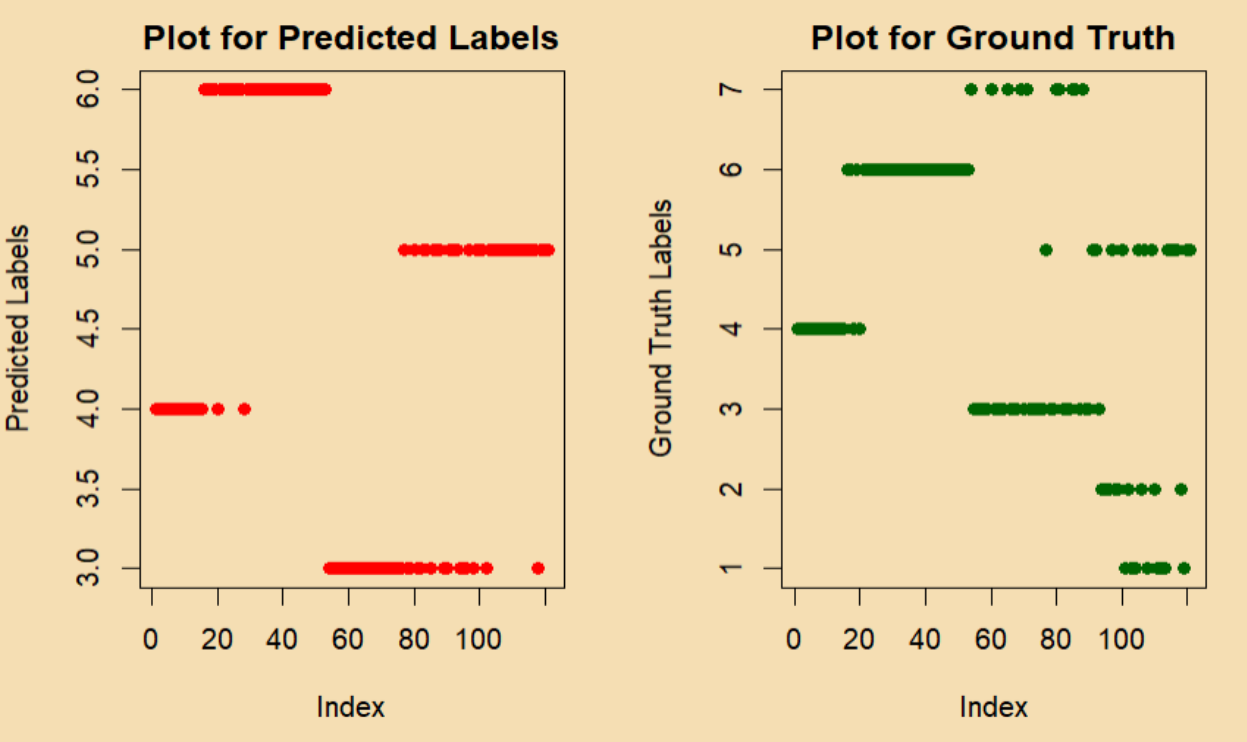

Figure 32 Side by side plots of the predicted labels using SVM algorithm using data frame 2 with the ground truth.

2. Non-linear kernel

The radial basis function kernel also called the RBF kernel, or Gaussian kernel is a kernel in the form of a radial basis function (more specifically, a Gaussian function). The radial basis kernel uses an exponential equation which works well to classify the non-linear data points. Here, for the dataframe1 “Gaussian radial basis kernel” is used creating 937 support vectors to do the classification. The accuracy of the output has reached 85.33% using the non-linear kernel. The cross table of the output is shown in figure 33. The plot demonstrating the predicted labels and the ground truth is shown in figure 34.

	predicted cluster							
actual cluster	1	2	3	4	5	6	7	Row Total
1	13	0	1	0	0	0	0	14
	0.087	0.000	0.007	0.000	0.000	0.000	0.000	
2	0	10	0	0	0	5	0	15
	0.000	0.067	0.000	0.000	0.000	0.033	0.000	
3	1	0	26	0	0	1	0	28
	0.007	0.000	0.173	0.000	0.000	0.007	0.000	
4	0	0	0	1	1	0	4	6
	0.000	0.000	0.000	0.007	0.007	0.000	0.027	
5	0	0	0	0	15	0	3	18
	0.000	0.000	0.000	0.000	0.100	0.000	0.020	
6	0	1	0	1	0	40	2	44
	0.000	0.007	0.000	0.007	0.000	0.267	0.013	
7	0	0	0	0	2	0	23	25
	0.000	0.000	0.000	0.000	0.013	0.000	0.153	
Column Total	14	11	27	2	18	46	32	150

Figure 33 Cross table of the labels predicted by SVM using radial basis function kernel for the data frame 1.

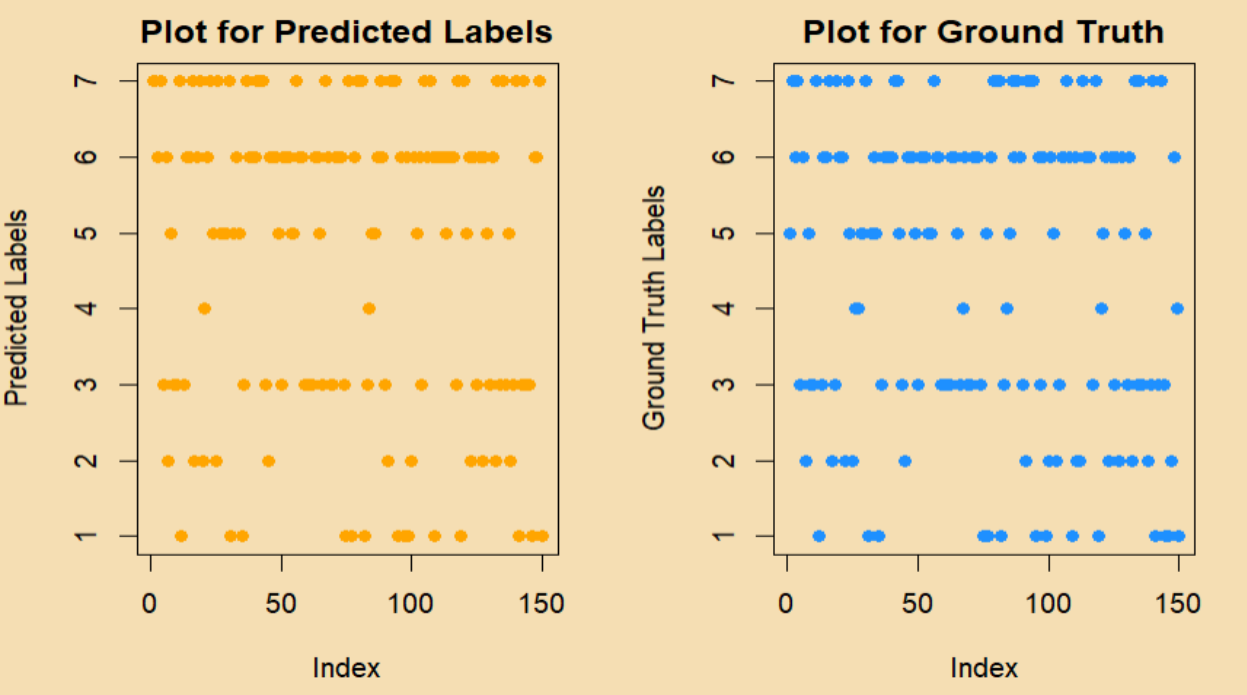

Figure 34 Side by side plots of the predicted labels using SVM algorithm using data frame 1 with the ground truth.

Here, for the dataframe1 “Gaussian radial basis kernel” is used creating 669 support vectors to do the classification. The accuracy of the output has reached 91.73% using the non-linear kernel. The cross table of the output is shown in figure 35. The plot demonstrating the predicted labels and the ground truth is shown in figure 36.

	predicted cluster							
actual cluster	1	2	3	4	5	6	7	Row Total
1	5 0.041	0 0.000	0 0.000	0 0.000	3 0.025	0 0.000	0 0.000	8
2	1 0.008	8 0.066	0 0.000	0 0.000	0 0.000	0 0.000	0 0.000	9
3	0 0.000	1 0.008	25 0.207	0 0.000	0 0.000	0 0.000	1 0.008	27
4	0 0.000	0 0.000	0 0.000	17 0.140	0 0.000	0 0.000	0 0.000	17
5	0 0.000	0 0.000	0 0.000	1 0.008	13 0.107	0 0.000	0 0.000	14
6	0 0.000	0 0.000	0 0.000	2 0.017	0 0.000	34 0.281	0 0.000	36
7	0 0.000	0 0.000	0 0.000	0 0.000	1 0.008	0 0.000	9 0.074	10
Column Total	6	9	25	20	17	34	10	121

Figure 35 Cross table of the labels predicted by SVM using radial basis function kernel for the data frame 2.

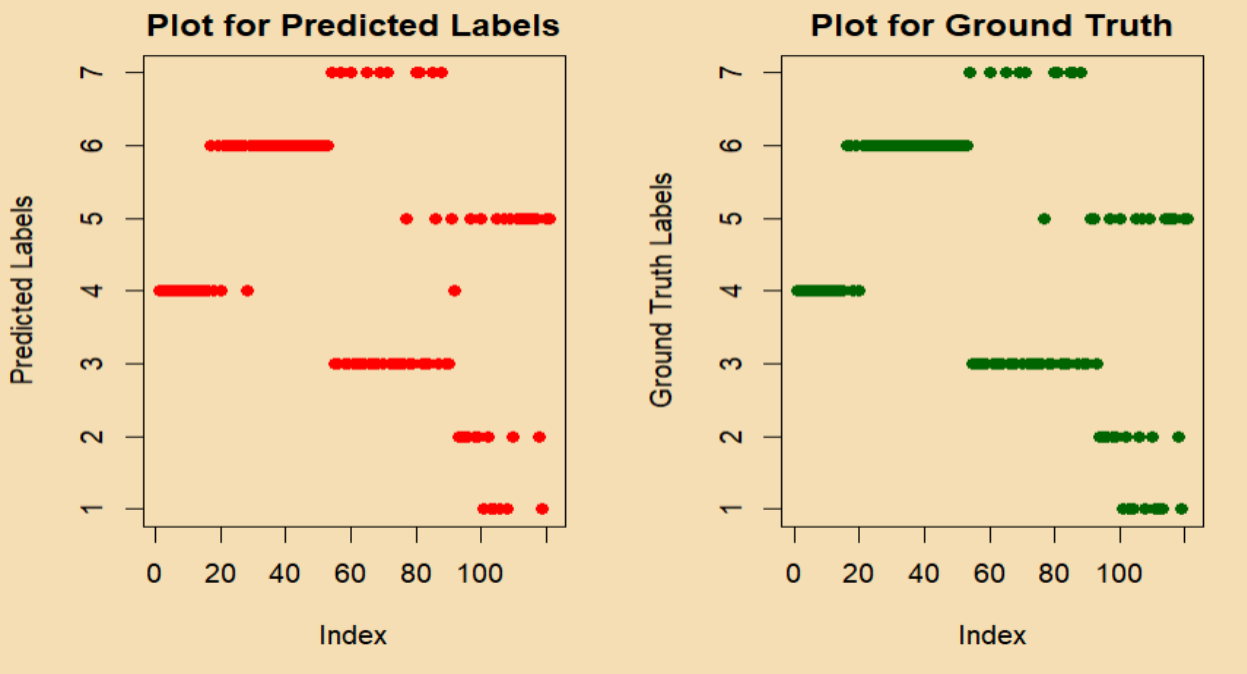

Figure 36 Side by side plots of the predicted labels using SVM algorithm using data frame 2 with the ground truth.

Summary

1. Standardization of data using the log transformation method was important as the data were highly skewed.
2. Normalization of data using the min-max normalization method was also important as there was a huge difference in the range of the columns.
3. Elbow method, silhouette method, and sum of squares method is used in this study to determine the optimal number of clusters of the 1500 proteins for data frame 1 and 1210 proteins for data frame 2.
4. After doing a deep study to determine the number of clusters, data is classified into 7 classes and labeled.
5. Several supervised machine learning algorithms like the random forest, artificial neural network, and support vector machine are used to train the model and find the accuracy of the trained model on test data to verify the results.

The accuracy (test data) of the supervised machine learning algorithms on data frame 1 is below:

Function	Random Forest	ANN	Linear SVM	Radial SVM
Accuracy	86%	73.5%	61.33%	85.33%

The accuracy (test data) of the supervised machine learning algorithms on data frame 2 is below:

Function	Random Forest	ANN	Linear SVM	Radial SVM
Accuracy	95.59%	88.7%	72.73%	91.73%

Future Work

1. Different values of k for the k-means clustering preferably 6,8,9 can be used to label the data and results can be verified using a number of the supervised learning algorithms.
2. Different clustering algorithms like DBSCAN clustering, Gaussian mixture model, and BIRCH clustering can be used in place of k-means and a comparison can be drawn and best-suited algorithm can be deployed to get the results.
3. XINA: a workflow for the integration of multiplexed proteomics kinetics data with network analysis is a package designed especially for protein analysis.
4. Both the k-mean labels and supervised machine learning algorithms can be further fine-tuned to get better results.
5. There are several other methods for data standardization and normalization like z-score, scaling to a range, and clipping normalization which can be used in place of min-max normalization.

References

Lantz, Brett. “Black Box Methods - Neural Networks and Support Vector Machines.” *Machine Learning with r: Expert Techniques for Predictive Modeling*, 3rd ed., Packt Publishing, Birmingham, 2019.

Lantz, Brett. “Improving Model Performance – Understanding ensembles – Random Forests.” *Machine Learning with r: Expert Techniques for Predictive Modeling*, 3rd ed., Packt Publishing, Birmingham, 2019.

Matt.0. “10 Tips for Choosing the Optimal Number of Clusters.” *Medium*, Towards Data Science, 28 Jan. 2019, https://towardsdatascience.com/10-tips-for-choosing-the-optimal-number-of-clusters-277e93d72d92.

Yiu, Tony. “Understanding Random Forest.” *Medium*, Towards Data Science, 29 Sept. 2021, https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Oleszak, Michal; “SVM Kernels: What Do They Actually Do?” *Medium*, Towards Data Science, 21 Sept. 2021, https://towardsdatascience.com/svm-kernels-what-do-they-actually-do-56ce36f4f7b8.

Bajaj, Prateek. “Creating Linear Kernel SVM in Python.” *GeeksforGeeks*, 20 June 2018, https://www.geeksforgeeks.org/creating-linear-kernel-svm-in-python/.

Ankit. “Radial Basis Function Kernel - Machine Learning.” *GeeksforGeeks*, 22 July 2021, https://www.geeksforgeeks.org/radial-basis-function-kernel-machine-learning/.

Technical report of Human Urine Proteins Clustering by Machine Learning Algorithm by Jinding Huang.

Appendix 1: Code of this Study

This code used in the document is written in R studio in the R programming language. The code files are submitted separately.

Appendix 2: Data Source

The data source excel files are submitted separately.