Pattern Recognition

Assignment 2

Networks Anomaly Detection using Clustering Techniques

Seif El-Din Ayman Mohamed 6825
Ai Hassan Mekky 6850
Nouine Mohamed Abou El-Waffa 7152

Overview

The exponential growth of network traffic has led to an increase in network anomalies, such as cyberattacks, network failures, and hardware malfunctions. Network anomaly detection is a critical task for maintaining the security and stability of computer networks. The objective of this assignment is to help students understand how K-Means and Normalized Cut algorithms can be used for network anomaly detection.

Importing Data

- This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between bad'' connections, called intrusions or attacks, and good'' normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.
- Data is mounted on google drive to be read.
- Kddcup.data_10percent is imported to be used in training Kmeans model.
- Corrected_gz is imported to be used for testing the Kmeans model.
- Kddcup.data is imported to be used in spectral clustering and agglomerative clustering (additional algorithms) models.
- Data is read using pandas library **pandas.read_csv(path, compression = “gzip”)**

Data Cleaning

- Difference between one hot encoding and label encoding:
  The main difference between label encoding and one hot encoding is that label encoding represents **each category with a unique numerical value**, whereas one hot encoding creates a **separate binary variable for each category**. Label encoding can be problematic for some machine learning algorithms because it implies an order between categories that may not exist. On the other hand, one hot encoding can lead to high-dimensional data and can be computationally expensive for large categorical variables. The choice between label encoding and one hot encoding will depend on the nature of the categorical variable and the machine learning algorithm being used.
- Since clustering algorithms depend on distance measures, then one hot encoding will be used to avoid implying an order to the categorical data. While label encoding will be used with the labels as they are not used in training (only used in testing).
- It was observed that there are categorical values outliers that result in a mismatch between the number of features in training and testing sets, Therefore the rows with these values (red_i, urh_i, icmp) in the service attribute are dropped.

- It was observed that there are label values in the testing set that are not in the training set, therefore, the rows having these values are dropped.
- Training and testing set are encoded using the same label encoder object to avoid mismatch as label encoding depends on the alphabetical order.

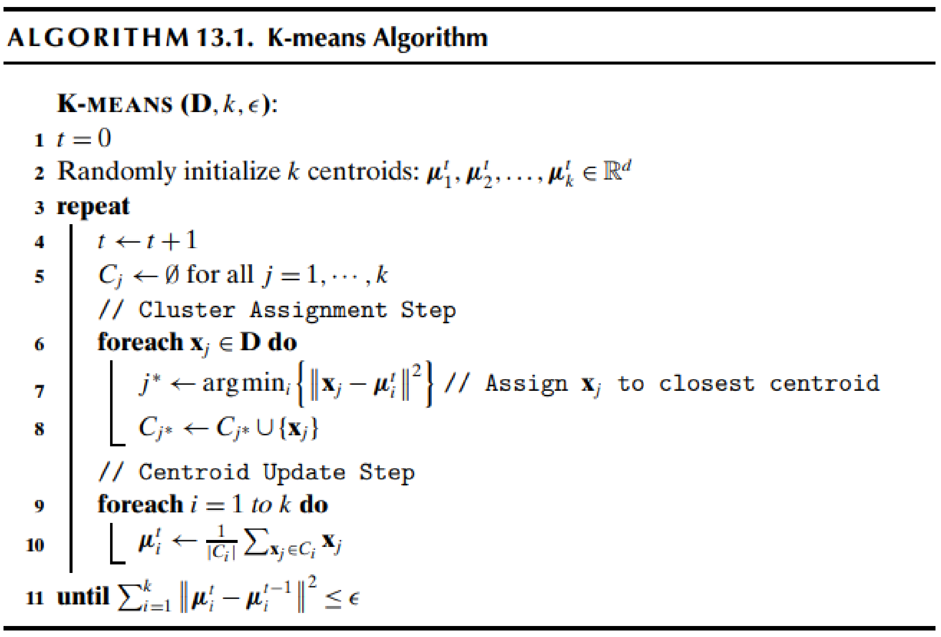
K-means

Description:

K-means is a clustering algorithm used to group a set of data points into k clusters. The algorithm aims to minimize the sum of squared distances between each data point and the centroid of its assigned cluster.

The algorithm starts by randomly selecting k initial cluster centroids. Each data point is then assigned to the closest centroid based on its Euclidean distance. The centroids are updated by computing the mean of the data points in each cluster. The process of assigning points to clusters and updating centroids is repeated until convergence, which occurs when the assignments of data points to clusters no longer change or a maximum number of iterations is reached.

Algorithm:



ALGORITHM 13.1. K-means Algorithm

```
   K-MEANS ($\mathbf{D}, k, \epsilon$):
1  $t = 0$
2  Randomly initialize $k$ centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$
3  repeat
4      $t \leftarrow t + 1$
5      $C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$
       // Cluster Assignment Step
6      foreach $\mathbf{x}_j \in \mathbf{D}$ do
7          $j^* \leftarrow \arg\min_i \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_i^t\|^2 \right\}$ // Assign $\mathbf{x}_j$ to closest centroid
8          $C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$
       // Centroid Update Step
9      foreach $i = 1$ to $k$ do
10         $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$
11 until $\sum_{i=1}^k \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \leq \epsilon$
```

Implementation:

- Inputs: data frame of the training set and number of clusters k.
- Outputs: cluster vector and final centroids after converging.

- Centroids are chosen randomly chosen from the training set, where the number of centroids is equal to the number of clusters.
- Another approach for centroids initialization: choose random samples from the training set and apply k-means on 10% of the data, and the resulting centroids are used as the centroids for the whole data to avoid converging to a local minimum point.
- For each point in the data set: compute the distance between the point and all the centroids, to assign the point to one of the clusters.
- Assign the point to the cluster with the minimum distance between the point and its centroid.
- Update the centroids using the new clusters after the last epoch.
- Repeat until it converges (no change in the centroids)

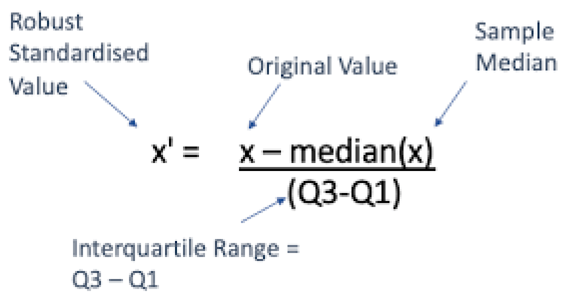
Prediction:

- Given a test set, for each point in the test set compute the distance between the point and the final centroids resulting from the training process, then assign the point to the cluster of the closest centroid.

Data normalization:

- K-means is sensitive to outliers and requires normalization.
- Training and testing set are normalized using robust normalizer and then MinMax normalizer.
- Robust normalizer:
  RobustScaler is a feature scaling technique that scales the data based on the interquartile range (IQR) and median instead of the mean and standard deviation used by StandardScaler. It is a robust method that is less affected by outliers in the data.

  The RobustScaler scales the data by subtracting the median of each feature and then dividing it by the IQR, which is the difference between the 75th percentile and the 25th percentile of the data. This scaling method preserves the shape of the distribution, which is useful when dealing with non-normally distributed data.


Robust Standardised Value
Original Value
Sample Median
x' = x – median(x) / (Q3-Q1)
Interquartile Range = Q3 – Q1

-MinMax normalizer: used to normalize the data in the range [0,1]

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Note: training and testing sets are normalized using the same scalers that are fit using **the training set**.

HyperParameters:

- Hyperparameters of k-means are distance criteria that is chosen to be Euclidean distance and the number of clusters k that will be tuned to choose the appropriate value.
- K values to be implemented: [7, 15, 23, 31, 45]
- Results:
- **At k = 7:**

```
conditional entropy when k is 7 = 0.35894129224731053
precision when k is 7 = [0.9744268077601411, 0.9493218378719378,
0.768194611982308, 1.0, 0.7306805960977109, 0.9490234104446599,
0.6322701688555347]
recall when k is 7 = [0.32608695652173914, 0.3769071820667488,
0.07975204024128071, 0.4380413683705203, 0.09926739162196573,
0.6739130434782609, 0.3918604651162791]
F1 score when k is 7 = 0.4612489954277501
```

- **At k = 15:**

```
conditional entropy when k is 15= 0.28530817831O8416
precision when k is 15 = [1.0, 1.0, 0.949210088691796,
0.9995949777237748, 0.9426778687652877, 0.9996862252902416,
0.8085106382978723, 1.0, 0.49001248439450684, 0.6827133479212254,
0.9953488372093023, 0.9231448763250883, 0.9293119698397738,
0.7116883116883117, 0.9963518443453587]
recall when k is 15 = [0.04257894846694914, 0.3936674250172194,
0.6737654928192013, 0.12138500885303954, 0.3700194109912129,
0.06649829892926468, 0.95, 0.15310840055085578, 0.03276909269270105,
1.0, 0.013399845546951639, 0.05139681290576431, 0.02057982509235875,
0.3186046511627907, 0.05130345849596126]
F1 score when k is 15 = 0.3347449968336772
```

- **At k = 23:**

```
conditional entropy when k is 23 = 0.2535268982825815
precision when k is 23 = [0.9829706717123936, 0.4521072796934866,
0.49588607594936707, 0.9505833905284832, 1.0, 1.0, 0.9953488372093023,
1.0, 0.8, 0.8631578947368421, 1.0, 0.9147003745318352,
0.964856455946825, 0.6827133479212254, 0.9824035192961408,
0.9676025917926566, 1.0, 0.9828473413379074, 0.9047619047619048,
0.9993558776167472, 0.9293119698397738, 0.7116883116883117, 1.0]
recall when k is 23 = [0.05110171158764509, 0.6781609195402298,
0.03270647659201436, 0.028907766483688507, 0.043497317943687254,
0.26144309240049257, 0.013399845546951639, 0.12064725555774149,
0.0007513932082402788, 0.09534883720930233, 0.06649829892926468,
0.20389889586942456, 0.6711095809561283, 1.0, 0.10254430089123584,
0.009350671035879025, 0.15305921699783592, 0.035879025693473315, 0.95,
0.1295318402871992, 0.02057982509235875, 0.3186046511627907,
0.04257894846694914]
F1 score when k is 23 = 0.26036820368872193
```

- **At k = 31:**

```
conditional entropy when k is 31= 0.24736919864636803
precision when k is 31 = [0.7063711911357341, 0.9985974754558204, 1.0,
0.9987129987129987, 0.9845916795069337, 1.0, 0.9504132231404959,
0.5778894472361809, 0.999618320610687, 0.8500196309383589,
0.9402618657937807, 0.9588268471517203, 1.0, 0.8954545454545455,
0.9182608695652174, 1.0, 1.0, 0.6428571428571429, 0.9901974513373477,
0.3414154652686763, 1.0, 1.0, 1.0, 1.0, 0.8947368421052632,
0.6815245478036176, 0.9991652754590985, 0.9378125400692396,
0.7415730337078652, 1.0, 0.6827133479212254]
recall when k is 31= [0.29651162790697677, 0.014860887896307738,
0.047167027346055476, 0.016196698044290456, 0.2199980326578792,
0.09285854810151485, 0.03360397403519025, 0.6609195402298851,
0.054663855899480286, 0.10648239228801888, 0.023981966563002232,
0.03548245705579094, 0.01903403501868975, 0.02906747983474326,
0.022040867441714847, 0.06956648786291249, 0.21959466510822148,
0.010356731875719217, 0.34777690340350187, 0.5199600798403193,
0.023397549623259795, 1.0, 0.07525083612040134, 0.11725908455260796,
0.0003485677964364069, 0.022019995408152616, 0.05887271296478458,
0.30531610694829997, 0.0032461144993114303, 0.022270459810899375, 1.0]
F1 score when k is 31 = 0.21402228442257912
```

- **At k = 45:**

```
conditional entropy when k is 45= 0.20687376379107345
precision when k is 45 = [1.0, 0.9796167957602935, 1.0,
0.6798444588464031, 1.0, 1.0, 1.0, 1.0, 0.9215909090909091, 1.0,
0.6153846153846154, 0.9370932754880694, 0.9388609715242882, 1.0,
0.9926470588235294, 0.9836556169429097, 0.9715846994535519, 1.0, 0.86,
1.0, 0.6827133479212254, 1.0, 1.0, 1.0, 0.9992896718283847,
0.34571997345719974, 0.6370510396975425, 1.0, 0.9171240985900334,
0.995835646862854, 0.9382716049382716, 1.0, 0.8709677419354839,
0.9831908831908832, 0.9984496124031008, 0.9708454810495627,
0.9713541666666666, 0.7209302325581395, 0.9769784172661871, 1.0, 1.0,
0.9976958525345622, 0.9, 1.0]
recall when k is 45 = [0.09236671257131615, 0.11818807790674798,
0.043664294212185095, 0.021894763206779237, 0.011115482982490656,
0.0905428815929536, 0.01335810147982718, 0.017699484460771013,
0.016927219218968503, 0.023272317421886415, 0.14507772020725387,
0.018033436997766692, 0.023397549623259795, 0.01849262173613575,
0.47093023255813954, 0.21016132205390517, 0.01855523783682244,
0.07589022230965965, 0.00634467833956325, 0.03359236671257131, 1.0,
0.05689716349063889, 0.05593704994677631, 0.03531548078729311,
0.34595711194176665, 0.5199600798403193, 0.3918604651162791,
0.017052451420341883, 0.17785059798376157, 0.0748679843877189, 0.95,
0.0474621286641747, 0.0941860465116279, 0.07202938782325562,
0.0134415896140761, 0.034751935881112896, 0.007785268518711778,
0.7126436781609196, 0.04251633236626245, 0.05813495966948652,
0.008703637995449896, 0.009037590532445575, 0.00036559358180156395,
0.06597649809020893]
F1 score when k is 45 = 0.18001787411451572
```

Clustering evaluation(External measures):

External evaluation measures attempt to capture how many points from the same partition appear in the same cluster and how many points from different partitions appear in different clusters. Typically, there is a trade-off between these two goals, which is either explicitly captured by a measure or implicitly captured in its computation. The contingency matrix is used in all of the external measures.

Contingency matrix:

If there are m classes, Cm ∈ ℜm × m and each element Cm(i, j) represents the number of samples with true label= j that have been assigned to the cluster i. Hence, a perfect contingency matrix is diagonal, while the presence of elements in all the other cells indicates a clustering error.
In our project the contingency matrix ∈ ℜm × n where m is the number of clusters computed and n is the number of true clusters

Conditional Entropy:

The conditional entropy of T given clustering C is then defined as the weighted sum:

$$H(\mathcal{T}|\mathcal{C}) = \sum_{i=1}^{r} \frac{n_i}{n} H(\mathcal{T}|C_i) = -\sum_{i=1}^{r}\sum_{j=1}^{k} \frac{n_{ij}}{n} \log\left(\frac{n_{ij}}{n_i}\right)$$
$$= -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log\left(\frac{p_{ij}}{p_{C_i}}\right)$$

Implementation:

-input: contingency matrix
-output: conditional entropy of clustering
-from contingency matrix (cm) each row cm[i] represents a cluster so

$$H(T|Ci) = -\sum_{i=1}^{k} \frac{cm[i,:]}{sum(cm[i])} * log(\frac{cm[i,:]}{sum(cm[i])})$$

$$H(T|C) = \sum_{i=1}^{k} \frac{sum(cm[i])}{sum(cm)} * H(T|Ci)$$

Where Ci is cluster computed and k number of clusters
I-n particular, H(T|C) = 0 if and only if T is completely determined by C, corresponding to the ideal clustering.
-Best entropy calculated with k-means was when k =15

Confusion matrix:

TP: True Positive (within cluster): Measured within cluster, pairs that must be in the same cluster, and are in the same cluster.We use combinatorics
From contingency matrix (cm):

$$\sum_{i=1}^{k}\sum_{j=1}^{T} = \binom{cm[i][j]}{2}$$

Where k is number of clusters and T number of true clusters
FP: False Positive (within cluster): Measured within cluster, pairs that must not be in the same cluster, but are.
From contingency matrix (cm):

$$FP = (\sum_{i=1}^{k}\sum_{j=1}^{T} = (sum(cm[i]) - cm[i][j]) * cm[i][j])$$

$$FP = \frac{FP}{2}$$

Where k is number of clusters and T number of true clusters

FN: False Negative (between cluster) :Measured between clusters, pairs that should be in the same cluster, but are not.

$$FN = \sum_{i=1}^{k} \sum_{j=1}^{T} = cm[i][j] * cm[i+1:, j]$$

Where k is number of clusters and T number of true clusters

TN: True Negative (between clusters):Measured between clusters, pairs that must not be in the same cluster, and are not in the same cluster.

$$TN = \binom{sum(cm)}{2} - (TP + FP + FN)$$

Precision:

For each of the cluster, we obtain the gold standard class with the maximum number of objects assigned. Then, we sum the maximum number of objects for each cluster and divide it by the total number of clustered objects. The resulting value is the precision and using the m x n contingency matrix(cm),it is calculated as below:

$$P(Ci) = \frac{max(cm[i])}{sum(cm[i])}$$

Where k is number of clusters

Recall:

For each gold standard class, we obtain the cluster with the maximum number of objects assigned. Then, we sum the maximum number of objects for each gold standard class and divide it by the total number of clustered objects and unclustered objects. The resulting value is the recall (also known as sensitivity) and using the m x n contingency matrix (cm),it is calculated as below:

$$R(Ci) = \frac{max(cm[i])}{sum(cm[:, index(max(cm[i])))}$$

F1-score:

The F1-score is the harmonic mean of precision and recall and is calculated as shown below:

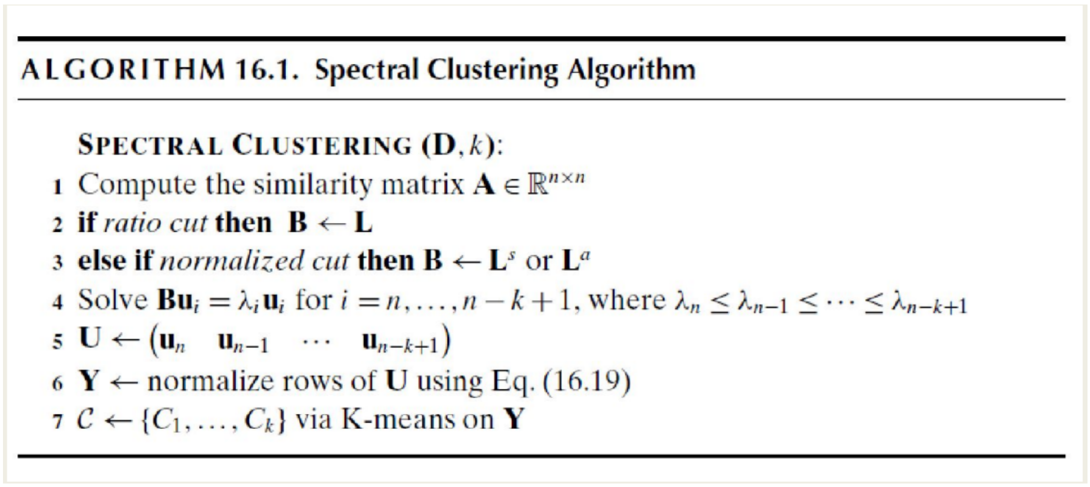
$$\frac{1}{k}\sum_{i=1}^{k} \frac{2*P(Ci)*R(Ci)}{P(Ci)+R(Ci)}$$

Spectral Clustering

Description:

Spectral clustering is a technique with roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them. Spectral clustering uses information from the eigenvalues (spectrum) of special matrices built from the graph or the data set.

Algorithm:



ALGORITHM 16.1. Spectral Clustering Algorithm

```
SPECTRAL CLUSTERING ($\mathbf{D}, k$):
1 Compute the similarity matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$
2 if ratio cut then $\mathbf{B} \leftarrow \mathbf{L}$
3 else if normalized cut then $\mathbf{B} \leftarrow \mathbf{L}^s$ or $\mathbf{L}^a$
4 Solve $\mathbf{B}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ for $i = n, \ldots, n-k+1$, where $\lambda_n \le \lambda_{n-1} \le \cdots \le \lambda_{n-k+1}$
5 $\mathbf{U} \leftarrow (\mathbf{u}_n \quad \mathbf{u}_{n-1} \quad \cdots \quad \mathbf{u}_{n-k+1})$
6 $\mathbf{Y} \leftarrow$ normalize rows of $\mathbf{U}$ using Eq. (16.19)
7 $\mathcal{C} \leftarrow \{C_1, \ldots, C_k\}$ via K-means on $\mathbf{Y}$
```

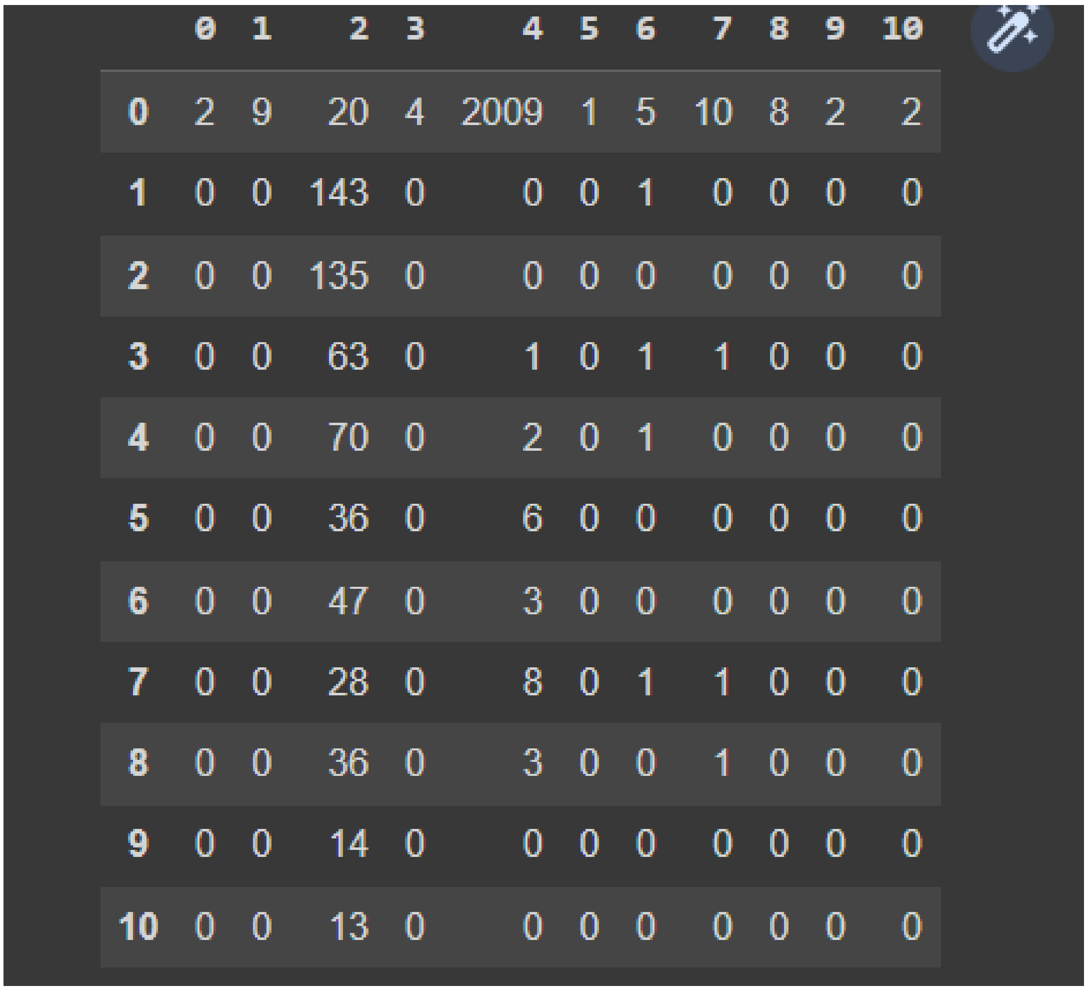
Implementation:

Inputs : similarity matrix resulting from RBF kernel (gamma = 0.1) and number of required clusters(K = 11 in this case).
Output: resulting clusters
We’re using the normalised cut in this assignment.
Degree matrix is computed from the similarity matrix

Then, the Laplacian matrix is computed then La matrix
After that, eigen values and vectors are computed for the La matrix
And U matrix is computed according to the value of K
U matrix is normalised and sent to Kmeans.

Performance:

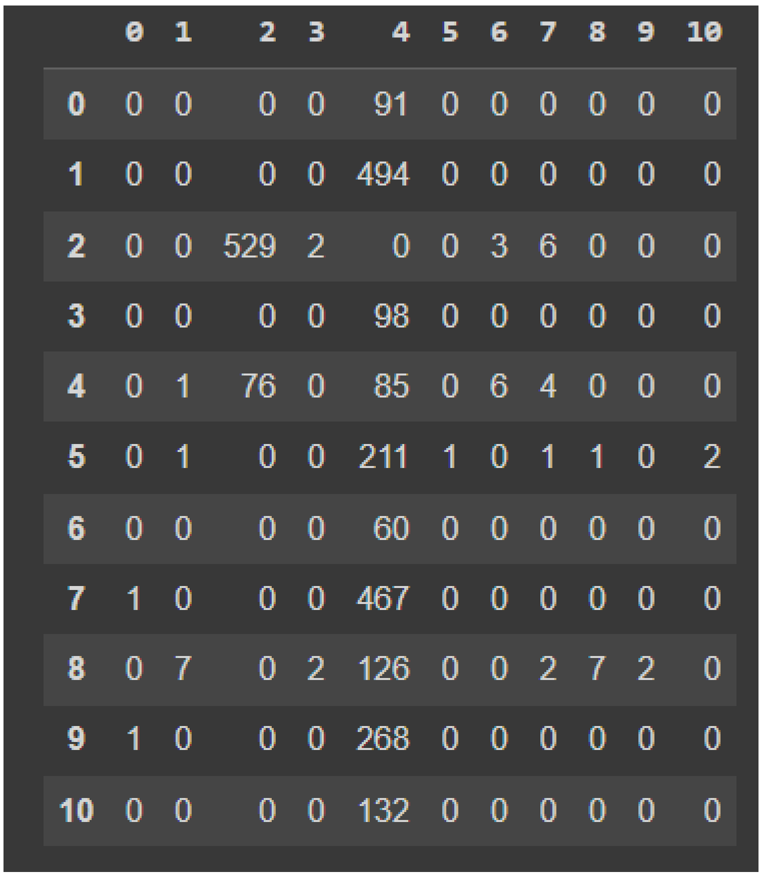
Contingency matrix resulting from spectral clustering(Rows →Clusters, Columns →Classes)



	0	1	2	3	4	5	6	7	8	9	10
0	2	9	20	4	2009	1	5	10	8	2	2
1	0	0	143	0	0	0	1	0	0	0	0
2	0	0	135	0	0	0	0	0	0	0	0
3	0	0	63	0	1	0	1	1	0	0	0
4	0	0	70	0	2	0	1	0	0	0	0
5	0	0	36	0	6	0	0	0	0	0	0
6	0	0	47	0	3	0	0	0	0	0	0
7	0	0	28	0	8	0	1	1	0	0	0
8	0	0	36	0	3	0	0	1	0	0	0
9	0	0	14	0	0	0	0	0	0	0	0
10	0	0	13	0	0	0	0	0	0	0	0

```
Conditional entropy = 0.2759862350568628
Precision = [0.9695945945945946, 0.9930555555555556, 1.0,
0.9545454545454546, 0.958904109589041, 0.8571428571428571, 0.94,
0.7368421052631579, 0.9, 1.0, 1.0]
Recall = [0.9886811023622047, 0.23636363636363636, 0.2231404958677686,
0.10413223140495868, 0.11570247933884298, 0.02975206611570248,
0.07768595041322314, 0.04628099173553719, 0.02975206611570248,
0.023140495867768594, 0.021487603305785124]
F1 score = 0.2320943551061556
```

Using K-means to cluster at K = 11



	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	91	0	0	0	0	0	0
1	0	0	0	0	494	0	0	0	0	0	0
2	0	0	529	2	0	0	3	6	0	0	0
3	0	0	0	0	98	0	0	0	0	0	0
4	0	1	76	0	85	0	6	4	0	0	0
5	0	1	0	0	211	1	0	1	1	0	2
6	0	0	0	0	60	0	0	0	0	0	0
7	1	0	0	0	467	0	0	0	0	0	0
8	0	7	0	2	126	0	0	2	7	2	0
9	1	0	0	0	268	0	0	0	0	0	0
10	0	0	0	0	132	0	0	0	0	0	0

Number of detected anomalies in case of Spectral clustering = 615 and all belong to Neptune class.

Number of detected anomalies in case of K-means clustering = 712 and all belong to Neptune class.

Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering algorithm that begins with each data point as its own cluster and iteratively merges the two closest clusters until a stopping criterion is met. In this algorithm, the distance between clusters is defined as the distance between their closest points (i.e., single linkage), their furthest points (i.e., complete linkage), or the average distance between their points (i.e., average linkage).

At each iteration, the two closest clusters are merged into a single cluster, and the distance matrix between clusters is updated accordingly. This process continues until all points are in a single cluster or until a predefined number of clusters is reached.

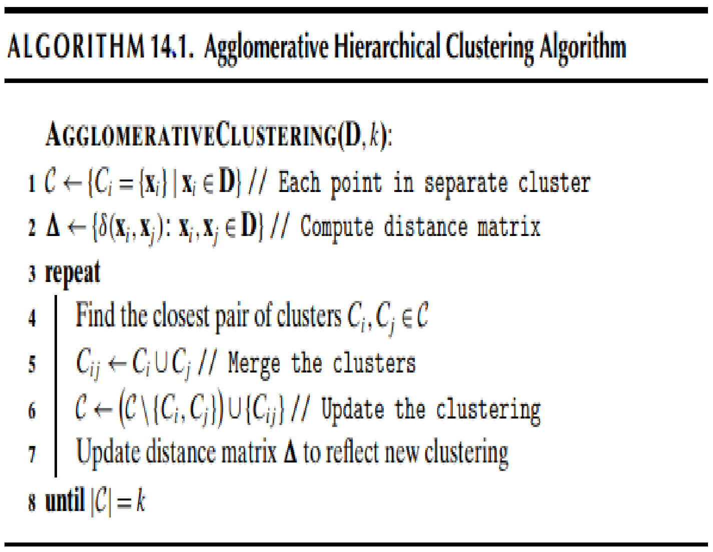
One **advantage** of agglomerative clustering is that it does not require a predefined number of clusters, as the algorithm determines the number of clusters based on the merging process. Additionally, the hierarchical structure of the resulting clusters can provide insight into the relationships between data points.

One **disadvantage** of agglomerative clustering is that it can be computationally expensive for large datasets. Since the algorithm requires the computation of pairwise distances between all data points at each iteration, the time complexity can be O(n^3) or higher for large datasets, where n is the number of data points.

Another potential disadvantage of agglomerative clustering is that it is sensitive to the choice of linkage criterion. Different linkage criteria can result in different cluster structures, and the choice of linkage criteria may depend on the nature of the data and the goals of the analysis. Moreover, agglomerative clustering is a greedy algorithm, and once a decision is made to merge two clusters, it cannot be undone, which can lead to suboptimal clustering in some cases.

Finally, agglomerative clustering can also suffer from the "**chaining**" effect, where small clusters may be merged into larger clusters too quickly, resulting in suboptimal clusters. This effect can be mitigated by using a stopping criterion that takes into account the size and density of the clusters, This is what happened with Neptune and Normal Classes

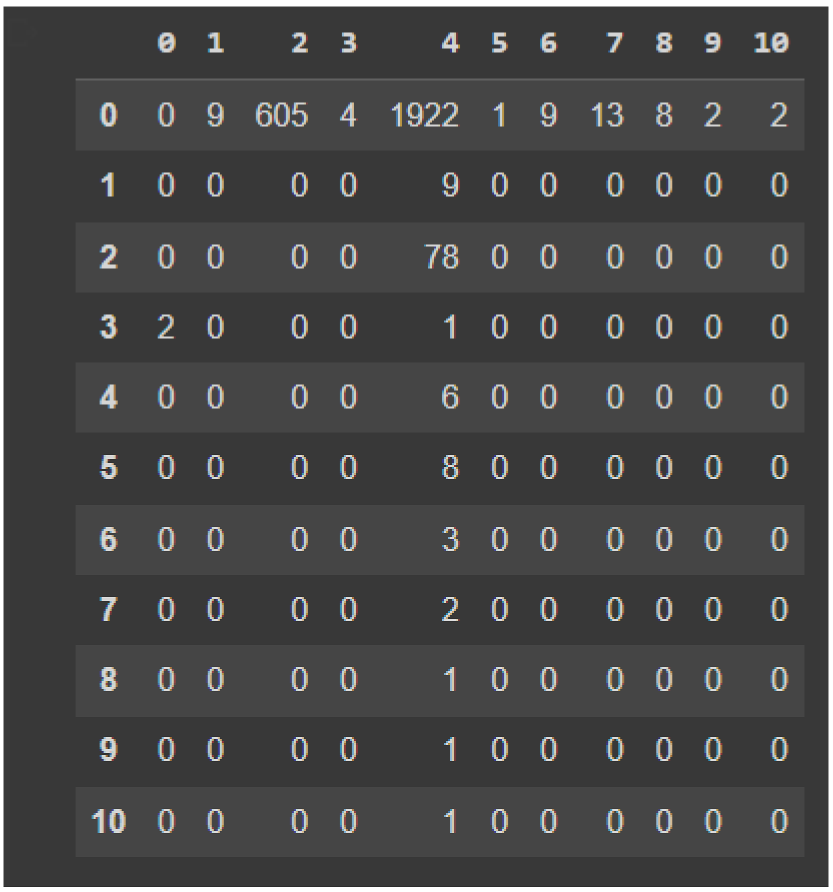
Algorithm:



ALGORITHM 14.1. Agglomerative Hierarchical Clustering Algorithm

```
AGGLOMERATIVECLUSTERING($\mathbf{D}, k$):
1 $\mathcal{C} \leftarrow \{C_i = \{\mathbf{x}_i\} \mid \mathbf{x}_i \in \mathbf{D}\}$ // Each point in separate cluster
2 $\boldsymbol{\Delta} \leftarrow \{\delta(\mathbf{x}_i, \mathbf{x}_j) \colon \mathbf{x}_i, \mathbf{x}_j \in \mathbf{D}\}$ // Compute distance matrix
3 repeat
4     Find the closest pair of clusters $C_i, C_j \in \mathcal{C}$
5     $C_{ij} \leftarrow C_i \cup C_j$ // Merge the clusters
6     $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_i, C_j\}) \cup \{C_{ij}\}$ // Update the clustering
7     Update distance matrix $\boldsymbol{\Delta}$ to reflect new clustering
8 until $|\mathcal{C}| = k$
```

Contingency matrix resulting from agglomerative clustering with k = 11



	0	1	2	3	4	5	6	7	8	9	10
0	0	9	605	4	1922	1	9	13	8	2	2
1	0	0	0	0	9	0	0	0	0	0	0
2	0	0	0	0	78	0	0	0	0	0	0
3	2	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	6	0	0	0	0	0	0
5	0	0	0	0	8	0	0	0	0	0	0
6	0	0	0	0	3	0	0	0	0	0	0
7	0	0	0	0	2	0	0	0	0	0	0
8	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	1	0	0	0	0	0	0

```
Conditional entropy = 0.9232042032831927
Precision = [0.7464077669902912, 1.0, 1.0, 0.6666666666666666, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0]
Recall = [0.9458661417322834, 0.004390243902439025,
0.038385826771653545, 0.0009813542688910696, 0.002952755905511811,
0.00392156862745098, 0.0014763779527559055, 0.0009813542688910696,
0.00012301636117603641, 0.00012301636117603641, 0.00012301636117603641]
F1 score = 0.08530598749559327
```