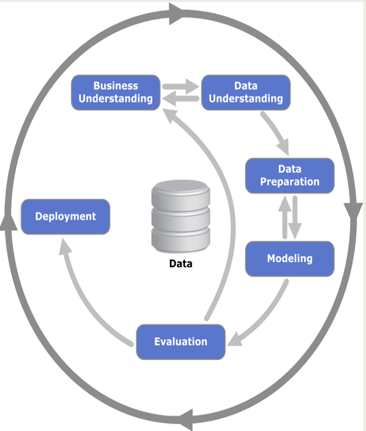
Introduction To Data Mining – Challenge 3

Muhammad Sumair – 22995



**CRISP-DM Approach:**

1) Business Understanding

2) Data Understanding

3) Data preparation

4) Modeling

5) Evaluation

6) Deployment

* **Business Understanding:-**

This is the initial stage of the CRISP-DM approach, where we must analyze the data and understand the problems and requirements of the project with respect to business.

The dataset we have for our challenge is about analysis of Clustering Algorithms on Disease Diagnosis. With this dataset, we must obtain clusters with the given data and each cluster will represent a single disease or a diagnosis. And to obtain these different clusters we will use different models and try to attain the highest accuracy possible.

* **Data Understanding:-**

The second stage of CRISP-DM approach involves understanding and requirements of the dataset that we are given. The data contains 2783 rows and 133 columns. These columns represent the symptoms that a patient experiences. After understanding the data we must predict the optimum number of clusters which will be put into the models then the dataset will be divided into those clusters.

* **Data Preparation:-**

The third stage of CRISP-DM approach involves data pre-processing where we will clean the data so that all the irrelevant data is removed. Firstly, when I was on the second stage I checked if there exist the null values which are to be removed, but since there we no null values, I did not remove any. Secondly, I applied the low variance filter, where I removed around 45 columns who had 0 variance. And after that I removed the columns with correlation greater than 0.95 which removed 10 columns more. And at the end I removed Row ID so that I can run the models.

* **Modeling:-**

The fourth stage is about the models used for prediction After all the pre-processing, I used two algorithms that are, Kmeans Clustering algorithm and Agglomerative Clustering Algorithm. I first run these algorithms with trial-and-error method And after that I used all the measures to get the optimum number of clusters which I then applied on my models.

The following measures were used to predict the optimal number of clusters.

1. Silhouette Coefficient
2. Dendogram
3. Elbow Curve & Knee Locator
4. **Silhouette Coefficient:**

I calculate silhouette coefficient for all the clusters and then I checked where the silhouette score was the greatest. The number of clusters where the silhouette score was the highest, I choose those clusters for example if I calculated my silhouette score for 10 clusters and the greatest score was on the 5th cluster, so I put in 5 clusters in k-means clustering algorithm to predict my accuracy. The range of clusters shown by silhouette coefficient in the data were 22-23 clusters.

1. **Dendogram:**

Dendogram was used for hierarchical clustering which was Agglomerative Clustering. But it did not benefit me because there were so many features so it was almost impossible to read.

1. **Elbow Curve & Knee Locator:-**

This measuring method works with Kmeans only. In this method, the inertia or sse(sum of squared errors) were plotted against number of clusters. The optimum number of clusters I got was 21-22 from this method.

* **Evaluation:-**

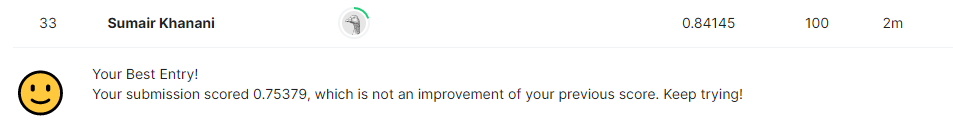
This is the fifth stage of CRISP-DM approach. The evaluation metric for this competition is [Adjusted Rand Index - ARI](https://en.wikipedia.org/wiki/Rand_index#:~:text=The%20adjusted%20Rand%20index%20is,specified%20by%20a%20random%20model.). In general, an ARI value lies between 0 and 1. The index value is equal to 1 only if a partition (cluster) is completely identical to the intrinsic structure (actual classes) and close to 0 for a random partition.

* **Deployment:**

The last stage of CRISP-DM approach is about the deployment. This means that we will be deploying the code into the operating system. Once a code is deployed it is ready for new datasets and predicts the values of that data. So, it means that our code has been learned on the old data set and now it is smart enough to predict new datasets. Lastly and most importantly, our code representation must also contain the preparation of the data that led to the current modeling that we applied. This ensures that the model will treat new raw data in the same manner as during model development. But with this dataset we have not done deployment, we just run and obtained the accuracies. We have not further deployed the code into the operating system; hence it cannot be used to predict a new dataset.

**Results:-**

1. **Agglomerative Clustering – 0.84145 (Best)**
2. **Kmeans Clustering – 0.80009**

****

**My Insights:**

**Which algorithm worked best for the given dataset and why?**

Agglomerative Clustering worked out to be the best algorithm for me with clusters = 11, affinity = ‘Manhattan’, linkage = ‘average’. Firstly, I tried to get optimum number of clusters by sse, silhouette coefficient and elbow curve but those number of clusters did not helped me achieve this accuracy so I had a random approach and chose the clusters according to my previous results.

**What is the optimal number of clusters in the data as per your findings and why?**

The optimal number of clusters as per my findings ranged from 6 to 11 clusters. This was because my trial-and-error approach had good results on the number of clusters in this range. And the number of clusters that I got from silhouette coefficient did not help me seek any results or better accuracy.

**What were the overall challenges that you faced while improving the score, and so on?**

One major challenge that I faced was, when I achieved the accuracy of 0.80009 with Kmeans clustering, I could not get a greater accuracy. Although I tried all the possible parameters of both Kmeans and Agglomerative Clustering. But later on, I cleaned my data again, and changed the correlation and then removed those columns which helped me get my best score.

**Table of all the Entries:-**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Data Pre-processing** | **Model Details** | **Scores** |
| 1 | Checking the null values which do not exist and drop row ID | KMeans(20) | 0.45037 |
| 2 | Checking the null values which do not exist and drop row ID | KMeans(10) | 0.6634 |
| 3 | Checking the null values which do not exist and drop row ID | KMeans(2) | 0.16042 |
| 4 | Checking the null values which do not exist and drop row ID | KMeans(7) | 0.34851 |
| 5 | Checking the null values which do not exist and drop row ID | KMeans(14) | 0.704 |
| 6 | Checking the null values which do not exist and drop row ID | KMeans(init="random", n\_clusters=14, n\_init=10, max\_iter=1000, random\_state=42) | 0.51802 |
| 7 | Checking the null values which do not exist and drop row ID | KMeans(init="random", n\_clusters=14, n\_init=5, max\_iter=100, random\_state=0) | 0.59411 |
| 8 | Checking the null values which do not exist and drop row ID | KMeans(init="random", n\_clusters=14, n\_init=5, max\_iter=100, random\_state=2) | 0.59369 |
| 9 | Checking the null values which do not exist and drop row ID | KMeans(6) | 0.42358 |
| 10 | Checking the null values which do not exist and drop row ID | KMeans(init="k-means++", n\_clusters=14, n\_init=10, max\_iter=300) | 0.4967 |
| 11 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=7, affinity='euclidean', linkage='ward') | 0.54057 |
| 12 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=14, affinity='euclidean', linkage='complete') | 0.45375 |
| 13 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='euclidean', linkage='single') | 0.55748 |
| 14 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="k-means++", n\_clusters=14, n\_init=10, max\_iter=300) | 0.25944 |
| 15 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=14, affinity='euclidean', linkage='ward') | 0.54057 |
| 16 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=14, affinity='euclidean', linkage='complete') | 0.45375 |
| 17 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='euclidean', linkage='single') | 0.55748 |
| 18 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=7, affinity='euclidean', linkage='single') | 0.25944 |
| 19 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=14, affinity='euclidean', linkage='ward') | 0.54057 |
| 20 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=14, affinity='euclidean', linkage='average') | 0.66609 |
| 21 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=15, affinity='euclidean', linkage='average') | 0.64654 |
| 22 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='euclidean', linkage='average') | 0.704 |
| 23 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=11, affinity='euclidean', linkage='average') | 0.6328 |
| 24 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='cityblock', linkage='average') | 0.54639 |
| 25 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=13, affinity='cosine', linkage='average') | 0.76273 |
| 26 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='cosine', linkage='complete') | 0.61444 |
| 27 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='cosine', linkage='single') | 0.5452 |
| 28 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='cityblock', linkage='complete') | 0.40026 |
| 29 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='cityblock', linkage='single') | 0.55748 |
| 30 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=12, affinity='cosine', linkage='average') | 0.78729 |
| 31 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=11, affinity='cityblock', linkage='complete') | 0.40039 |
| 32 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="k-means++", n\_clusters=6, random\_state=0) | 0.80009 |
| 33 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=6, affinity='cosine', linkage='average') | 0.63913 |
| 34 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=6, affinity='cityblock', linkage='average') | 0.35528 |
| 35 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="k-means++", n\_clusters=6, random\_state=0, algorithm="elkan") | 0.57941 |
| 36 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="k-means++", n\_clusters=6, random\_state=0, algorithm="full") | 0.38554 |
| 37 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="k-means++", n\_clusters=6, random\_state=0, algorithm="auto") | 0.80009 |
| 38 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=6, random\_state=0, algorithm="elkan") | 0.40039 |
| 39 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=6, random\_state=0, algorithm="full") | 0.46508 |
| 40 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=6, random\_state=0, algorithm="auto") | 0.40039 |
| 41 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=7, random\_state=0, algorithm="full") | 0.46504 |
| 42 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=7, random\_state=0, algorithm="auto") | 0.40037 |
| 43 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=7, random\_state=0, algorithm="full") | 0.46509 |
| 44 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=6, random\_state=0, algorithm="auto") | 0.40039 |
| 45 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=6, random\_state=4, algorithm="full") | 0.46498 |
| 46 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | KMeans(init="random", n\_clusters=6, random\_state=2, algorithm="auto") | 0.40029 |
| 47 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=25, affinity='cosine', linkage='average') | 0.34478 |
| 48 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=25, affinity='cosine', linkage='single') | 0.3703 |
| 49 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=24, affinity='cosine', linkage='average') | 0.34715 |
| 50 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | AgglomerativeClustering(n\_clusters=20, affinity='cosine', linkage='complete') | 0.45037 |
| 51 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 | (init="k-means++", n\_clusters=8, random\_state=0, n\_init=100) | 0.62693 |
| 52 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | (init="k-means++", n\_clusters=8, random\_state=2, n\_init=100) | 0.6518 |
| 53 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | (init="k-means++", n\_clusters=8, random\_state=16, n\_init=100) | 0.62693 |
| 54 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(init="k-means++", n\_clusters=8, random\_state=2, n\_init=300) | 0.66589 |
| 55 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(init="k-means++", n\_clusters=8, random\_state=2, n\_init=1000) | 0.68636 |
| 56 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(init="k-means++", n\_clusters=8, random\_state=2, n\_init=3000) | 0.49422 |
| 57 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(init="k-means++", n\_clusters=8, n\_init=100) | 0.486 |
| 58 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(init="k-means++", n\_clusters=8, n\_init=100) | 0.452 |
| 59 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | AgglomerativeClustering(n\_clusters=12,linkage='single',compute\_full\_tree='bool') | 0.51168 |
| 60 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | AgglomerativeClustering(n\_clusters=12,affinity='cosine',linkage='single',compute\_full\_tree='bool') | 0.5452 |
| 61 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(n\_clusters=6,random\_state=0,max\_iter=300,n\_init =10,algorithm='elkan') | 0.80009 |
| 62 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(n\_clusters=6,random\_state=0,max\_iter=300,n\_init = 30) | 0.25335 |
| 63 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | KMeans(n\_clusters=3,random\_state=0,max\_iter=300) | 0.22922 |
| 64 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.85 | AgglomerativeClustering(n\_clusters=12,affinity='manhattan',linkage='average',compute\_full\_tree='auto') | 0.54639 |
| 65 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=24,affinity='manhattan',linkage='average',compute\_full\_tree='auto') | 0.34827 |
| 66 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=18,affinity='manhattan',linkage='average',compute\_full\_tree='auto') | 0.4974 |
| 67 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=12,random\_state=0,max\_iter=100) | 0.52663 |
| 68 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=5,affinity='cosine',linkage='average',compute\_full\_tree='auto') | 0.63913 |
| 69 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=6,affinity='cosine',linkage='average',connectivity=None, compute\_full\_tree='auto') | 0.6401 |
| 70 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=5,affinity='cosine',linkage='average',compute\_full\_tree='full') | 0.63937 |
| 71 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(init='k-means++',n\_clusters=12,random\_state=0,max\_iter=300) | 0.52663 |
| 72 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=15,affinity='cosine',linkage='average',connectivity=None, compute\_full\_tree='bool') | 0.75807 |
| 73 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=12,affinity='cosine',linkage='average',connectivity=None, compute\_full\_tree='bool') | 0.78729 |
| 74 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(init='k-means++',n\_clusters=25,random\_state=0) | 0.34492 |
| 75 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=30,affinity='cosine',linkage='single') | 0.36156 |
| 76 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=13,affinity='euclidean',linkage='average') | 0.68524 |
| 77 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | AgglomerativeClustering(n\_clusters=24,affinity='manhattan',linkage='average',compute\_full\_tree='auto') | 0.34827 |
| 78 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=3,random\_state=0,max\_iter=100) | 0.22922 |
| 79 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | Kmeans(13) | 0.61441 |
| 80 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | kmeans(n\_clusters=7,random state=0) | 0.77332 |
| 81 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=25, affinity='cosine', linkage='single') | 0.3703 |
| 82 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=11, affinity='manhattan', linkage='average') | 0.56577 |
| 83 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=11, affinity='manhattan', linkage='average') | 0.84145 |
| 84 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | KMeans(init="k-means++", n\_clusters=6, random\_state=0) | 0.66788 |
| 85 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=12, affinity='manhattan', linkage='average') | 0.704 |
| 86 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=11, affinity='cosine', linkage='average') | 0.75379 |
| 87 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=11, affinity='manhattan', linkage='average') | 0.56577 |
| 88 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=11, affinity='cosine', linkage='average') | 0.81704 |
| 89 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.97 | AgglomerativeClustering(n\_clusters=11, affinity='manhattan', linkage='average') | 0.84145 |
| 90 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.98 | AgglomerativeClustering(n\_clusters=11, affinity='manhattan', linkage='average') | 0.6328 |
| 91 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=16, random\_state=0) | 0.48587 |
| 92 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=13, random\_state=0) | 0.57937 |
| 93 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=13, linkage='average') | 0.45474 |
| 94 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=18, linkage='average') | 0.48585 |
| 95 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=15, linkage='average') | 0.50481 |
| 96 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=11, linkage='average') | 0.45891 |
| 97 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.90 | KMeans(n\_clusters=8, random\_state=0) | 0.42651 |
| 98 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | KMeans(n\_clusters=10 random\_state=0) | 0.62488 |
| 99 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=12, affinity='manhattan', linkage='average') | 0.704 |
| 100 | Checking the null values which do not exist and drop row ID and removing 45 columns with var=0 and correlation > 0.95 | AgglomerativeClustering(n\_clusters=11, affinity='cosine', linkage='average') | 0.75379 |