Clustering states of India based on COVID-19 vaccine deployment/inoculation status

A Report Submitted in Partial Fulfillment of the Requirements

for the

6th Semester B.Tech. Summer Internship

(Mode of Internship is ONLINE)

By

Your Name Scholar ID

RAMAVTAR PAREEK 1815071

SAMAY SINGH 1815072

MAHENDRA VERMA 1815073

DIVESH 1815078

Under the Guidance of

Dr. Malaya Dutta Borah

Assistant Professor

Computer Science and Engineering Department

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

Assam

May-June 2021

DECLARATION

We declare that the presented work represents largely our own ideas and work in our own words. Where others ideas or words have been included, we have adequately cited and listed in the reference materials. We have adhered to all principles of academic honesty and integrity.

RAMAVTAR PAREEK	1815071
SAMAY SINGH	1815072
MAHENDRA VERMA	1815073
DIVESH	1815078

Ramavtar Pareek samay Singh [illegible]

Divesh

Department of Computer Science and Engineering

National Institute Of Technology Silchar, Assam

ACKNOWLEDGEMENT

It is our great pleasure and privilege to thank NIT Silchar and the Computer Science and Engineering department for providing us this opportunity to do the project under Satyendra Nath Bose Summer Internship Program 2021. We express our utmost gratitude to Dr. Malaya Dutta Borah ma’am for her constant guidance and support throughout the project. We offer our sincere appreciation for the learning opportunity provided to us. We would also like to express our warm regards to Zakir Hussain sir for his cooperation, understanding and constant encouragement which were the sustaining factors in carrying out the work successfully.

We hope that this opportunity will add new skills to our skillset. Lastly we thank God and our family members who constantly encouraged us throughout this period

RAMAVTAR PAREEK	1815071
SAMAY SINGH	1815072
MAHENDRA VERMA	1815073
DIVESH	1815078

ABSTRACT

In the past few years Data Mining has emerged to be a popular field which uses the concept of core computer science and statistics. It is used in various area like education, healthcare etc. which involves a large amount of data. Unstructured data leads to errors and uncertainty. To resolve this problem, Data Mining comes to the rescue, its main purpose is to extract the useful information/ pattern from a huge bundle of data so that it can moulded into an understandable and structured format. It can later be used for estimation of maintenance cost, error detection etc. In this paper, we use Data Mining algorithms to perform the clustering of Indian states based on Covid-19 vaccine deployment. The increasing stress of covid cases and large requirement of vaccines makes it necessary to cluster the data into a recognizable structure for future study and reference. To carry out this task, we have implemented clustering algorithms namely K-Means, Agglomerative Hierarchical Clustering and DBSCAN and elaborated the working of these algorithms on the input dataset. We completed the implementation using Python language on Google Colaboratory. Finally we obtained the dataset in the desired clustered format.

1. INTRODUCTION

In this project, we focused our attention on reviewing and implementing the various clustering algorithms. We have used coivd-19 vaccine deployment dataset for the purpose. As the data is increasing each day with more vaccines delivered to the beneficiaries, sometimes the data becomes unstructured and errant. Hence, we propose to organize the available data using the above algorithms. This will help us in visualizing and correctly analyzing the data for future references.

1.1 <u>MOTIVATION</u>:-

- Data Explosion:

  A large amount of data is available and are being collected for various purposes.

  For example, Business Transactions, Healthcare etc.
- Difficulties:

  Volume of data is excessively large for humans to analyse and relational databases cannot compute transitive closure and answer some complex predictive questions.
- Data Mining Comes to Rescue:

  1) Provide better and structured insights into the data
  2) Provides enough visualization for hypothesis generation.

1.2 PROBLEM STATEMENT and OBJECTIVES:-

India is currently suffering from the second wave of covid-19 and the nationwide vaccination is a big challenge for the government. At present, two age groups are receiving vaccines : i) 18-45 and ii) above 45.

There are currently three covid vaccines have been approved : a) Covaxin b) CoviShield and c) Sputnik V.

Therefore it becomes necessary to monitor the various prospects of vaccination based on vaccine deployment data.

To overcome this challenge, our task is to perform **Clustering states of India based on COVID-19 vaccine deployment/inoculation status.**

As the data on vaccination keeps on piling, it is becoming even harder for human beings to read, analyse and derive a meaningful info out this this data. So, there must be an alternative approach dealing with this huge data which is bound to increase further. The project aims to solve this issue.

The data is categorised and grouped using various techniques thus reducing the size of data without compromising the entirety .This will increase the probability of better understanding in comprehensive way. The main objective of the project is that it will increase readability, time efficiency and will require lesser human efforts in deep analysis of the data.

2. LITERATURE SURVEY

Cluster analysis could be divided into hierarchical clustering and non-hierarchical clustering techniques. Examples of hierarchical techniques are single linkage, complete linkage, average linkage, median, and Ward. Non-hierarchical techniques include k-means, adaptive k-means, k-medoids, and fuzzy clustering.

The research funded by Covenant University Center for Research and Development and led by Oyelade (O. J), Oladipupo (O.O.) and Obagbuwa
(I. C.) aimed to find out the Academic progress of students of private institutions in Nigeria. This was to help Academic planners to take effective decisions in improving the Academic knowledge.

During this research a set of 79 students were selected with 9 courses during the semester. Students were grouped using K-means clustering algorithm and Euclidean distance.

The average marks of students were taken into account and grouped in varied number of clusters and further divided them according to the performance. For instance like students group with average of 70 and above termed excellent,60-69 very good and so on.

The overall performance is evaluated by applying deterministic model, where the group assessment in each of the cluster size is evaluated by summing the average of the individual scores in each cluster.

$$\frac{1}{N}\left(\sum_{j=1}^{N}\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)\right)$$

Where

N = the total number of students in a cluster and

n = the dimension of the data

This way the K-means algorithm combined with Euclidean distance proved effective in helping the planners to have better understanding of academic level of students.

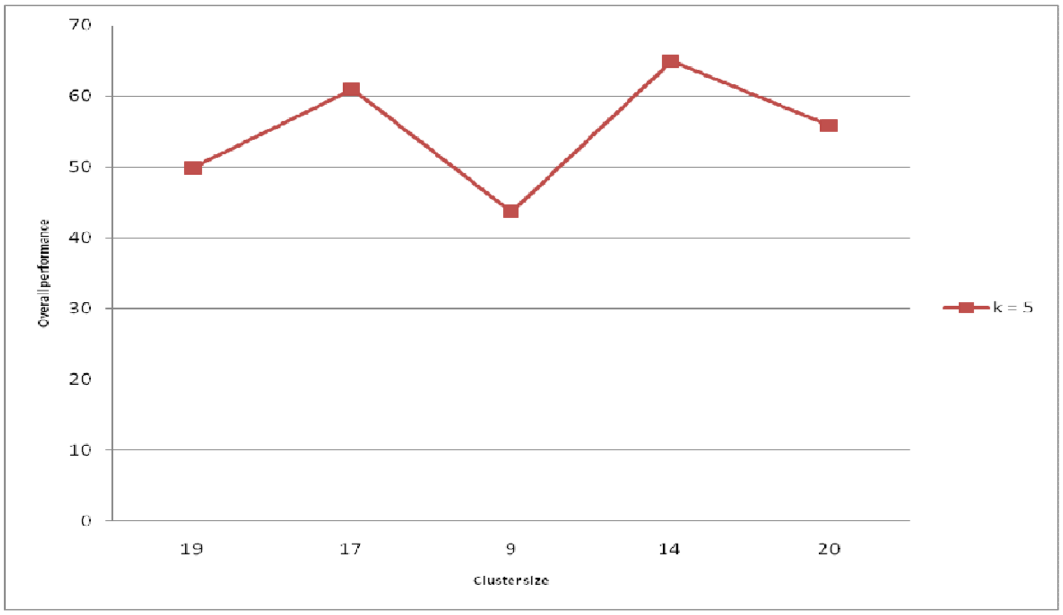


Fig. : Overall Performance versus cluster size (# of students) k = 5

By studying this paper we understood the implementation of k-Means algorithm using Euclidean Distance for a simple dataset. On similar lines we implemented the same algorithm using our required dataset.

3. PROPOSED WORK

3.1 METHODOLOGY:

3.1.1 K-MEANS ALGORITHM:

K-means is a non-hierarchical clustering technique. Its purpose is to partition n observation or data points in K clusters in which each data point belongs to one cluster with the nearest mean (centroid).

Suppose we have a dataset of n points $x_1, x_2, \dots\dots\dots . x_n$. The problem is to minimize:

$$\frac{1}{n}\sum_{i=1}^{n}\left[min; d2\left(x_i, m_j\right)\right]$$

Where $d(x_i, m_j)$ is the Euclidean distance between x_i, m_j, where j=1, 2, 3,…..K are known as cluster centroids.

The best part of K-Means clustering is that it is highly scalable and it can be generalized to cluster of different sizes. However, when the number of dimensions of data points increases, the similarity measure converges to a constant value.

Algorithm steps:

Step 1: Accept the number of clusters to group data into and the dataset to cluster as input values

Step 2: Initialize the first K clusters - Take first k instances or - Take Random sampling of k elements

Step 3: Calculate the arithmetic means of each cluster formed in the dataset.
Step 4: K-means assigns each record in the dataset to only one of the initial clusters - Each record is assigned to the nearest cluster using a measure of distance (e.g Euclidean distance).
Step 5: K-means re-assigns each record in the dataset to the most similar cluster and re-calculates the arithmetic mean of all the clusters in the dataset.

3.1.2 DBSCAN - Density-Based Spatial Clustering of Applications with Noise:

Density-Based Clustering refers to unsupervised learning methods that identify distinctive clusters in the data, based on the idea that a cluster in data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for density-based clustering. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.

The DBSCAN algorithm uses two parameters:

- **minPts:** The minimum number of points (a threshold) clustered together for a region to be considered dense.
- **eps (ε):** A distance measure that will be used to locate the points in the neighbourhood of any point.

We may need a Density-Based clustering algorithm like DBSCAN in spite of having K-means clustering because:

K-Means clustering may cluster loosely related observations together. Every observation becomes a part of some cluster eventually, even if the observations are scattered far away in the vector space. Since clusters depend on the mean value of cluster elements, each data point plays a role in forming the clusters. A slight change in data points *might* affect the clustering outcome. This problem is greatly reduced in DBSCAN due to the way clusters are formed.

Another challenge with *k*-means is that we need to specify the number of clusters 'K' in order to use it. Much of the time, we won't know what a reasonable 'K' value is a priori.

Algorithm steps:

a) It starts with a random unvisited starting data point. All points within a distance 'Epsilon – Ɛ classify as neighbourhood points.

b) You need a minimum number of points within the neighborhood to start the clustering process. Under such circumstances, the current data point becomes the first point in the cluster. Otherwise, the point gets labeled as 'Noise.' In either case, the current point becomes a visited point.

c) All points within the distance Ɛ become part of the same cluster. Repeat the procedure for all the new points added to the cluster group.

d) Continue with the process until you visit and label each point within the Ɛ neighborhood of the cluster.

e) On completion of the process, start again with a new unvisited point thereby leading to the discovery of more clusters or noise. At the end of the process, you ensure that you mark each point as either cluster or noise.

3.1.3 AGGLOMERATIVE HIERARCHICAL CLUSTERING:

Agglomerative Hierarchical clustering is the common type of hierarchical clustering. It begins with treating each data point as a singleton cluster. Then they are successively merged into a big-single cluster. The result of the merging is a tree based representation called Dendogram.

Single Linkage:

The shortest distance between two points in each cluster

$$L(r, s) = min\left(D\left(x_{r_i}, x_{s_j}\right)\right)$$

Complete Linkage:

The longest distance between two points in each cluster

$$L(r, s) = max\left(D\left(x_{r_i}, x_{s_j}\right)\right)$$

The best thing about this algorithm is that we don't have to assume a particular value of K. However, sometimes this algorithm is too slow for large datasets and time complexity exceeds $O(n^2 logn)$.

Algorithm steps:

a) Consider each data point as an individual cluster. The second step is to select a distance metric to measure the distance between the two groups. Use the average linkage method where the distance between two clusters is the average distance between the data points in one cluster and the data points in the other.

b) At each iteration, we merge two clusters with the smallest average linkage into one.

c) Repeat the above step until we have one large cluster containing all the data points.

3.2 DATASET PREPARATION / USED

The dataset used for the project is the data of the Indian states based on Covid-19 vaccine deployment status. It is downloaded from https://www.kaggle.com/datasets in '.csv' format. Before using the dataset for implementation, it is necessary to do preprocessing and cleaning to make the data clear and error free. It is performed using Microsoft Excel. It involves two steps:-

Step 1: Deleting blank or empty rows

Step 2: Deleting incomplete and irrelevant columns

4. RESULTS AND DISCUSSION

We have obtained the following results after implementing the above mentioned three algorithms:

k-Means:

```
import numpy as
np import
pandas as pd
from matplotlib import pyplot as plt
from sklearn.datasets.samples_generator import
make_blobs
from sklearn.cluster import KMeans
import seaborn as
sns import
warnings

    /usr/local/lib/python3.7/dist-
    packages/sklearn/utils/deprecation.py:144: Futur
    warnings.warn(message, FutureWarning)


from google.colab import
files uploaded =
files.upload()
```

Browse... covid_vaccine_statewises.csv **covid_vaccine_statewises.csv**(text/csv)

- 342769 bytes, last modified: n/a - 100% done

```
    Saving covid_vaccine_statewises.csv to

covid_vaccine_statewises.csv data =

pd.read_csv("covid_vaccine_statewise.csv")


#basic information of data
data.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 4551 entries, 0 to 4550
    Data columns (total 13 columns):
     #   Column                          Non-Null Count  Dtype
```

```
---  ------                              --------------  -----
0Updated On                             4551 non-null   object
1State                                  4551 non-null   object
2Total Individuals Vaccinated           4551 non-null   int64
3Total Sessions Conducted               4551 non-null   int64
4Total Sites                            4551 non-null   int64
5First Dose Administered                4551 non-null   int64
6Second Dose Administered               4551 non-null   int64
7Male(Individuals Vaccinated)           4551 non-null   int64
8Female(Individuals Vaccinated)         4551 non-null   int64
```

Executing (21s) Cell?upload()?eval_js()?read_reply_from_input()

min	7.000000e+00	0.000000e+00	0.000000	7.000000e+00	0.000000e+00	0
25%	4.485650e+04	2.061000e+03	66.000000	4.341300e+04	3.400000e+02	2
50%	2.482800e+05	1.188600e+04	560.000000	2.415950e+05	3.674200e+04	1
75%	1.860385e+06	1.523290e+05	1825.000000	1.788438e+06	3.534980e+05	9
max	1.442702e+08	1.078696e+07	73933.000000	1.442702e+08	4.092140e+07	7

```
9Transgender(Individuals Vaccinated)  4551 non-null   int64
10     Total Covaxin Administered           4551 non-null   int64
11     Total CoviShield Administered        4551 non-null   int64  12
Total Doses Administered             4551 non-null   int64 dtypes:
int64(11), object(2) memory usage: 462.3+ KB
```

```
#statics of data
data.describe()
```

```
                 Total       Total
                                         Total    First Dose Second Dose
        Male(I Individuals    Sessions
```

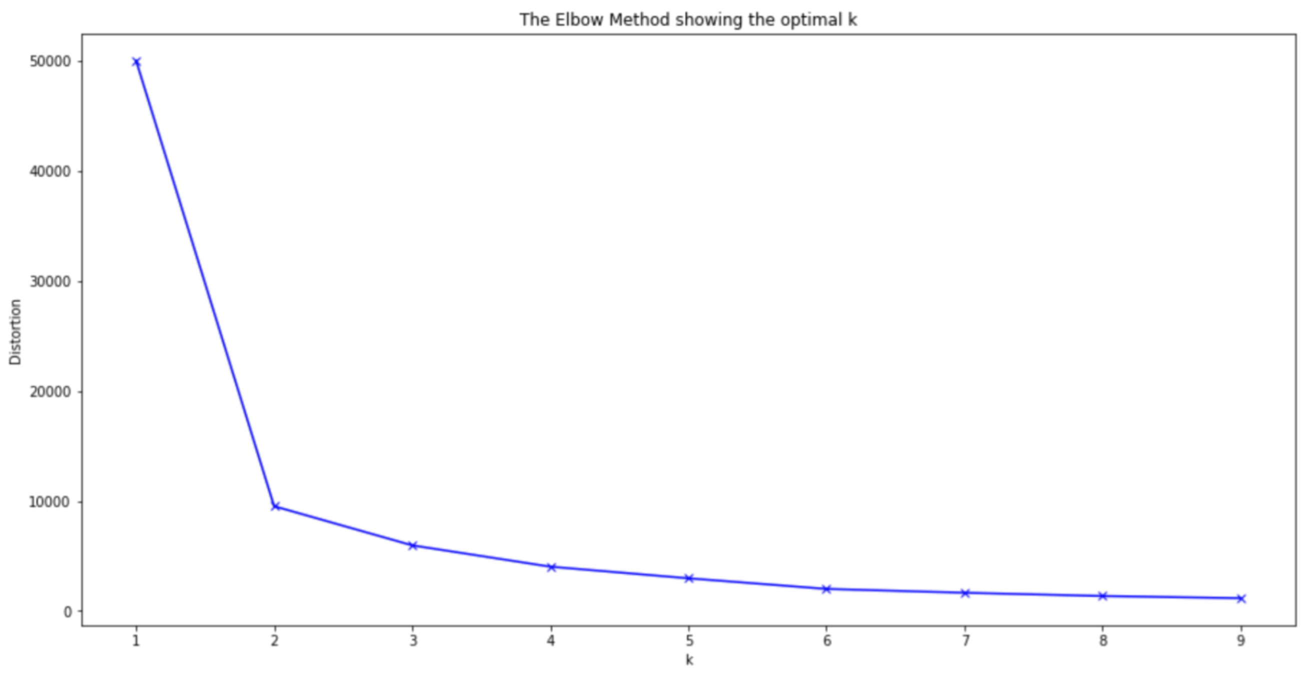
```
#standarizing the data
from sklearn.preprocessing import
StandardScaler scaler = StandardScaler()
data_scaled =
scaler.fit_transform(data[data.columns[2:]])
```

```
#changing scaled data  set type to pandas dataframes
df_scaled =
pd.DataFrame(data_scaled,columns=data.columns[2:])
```

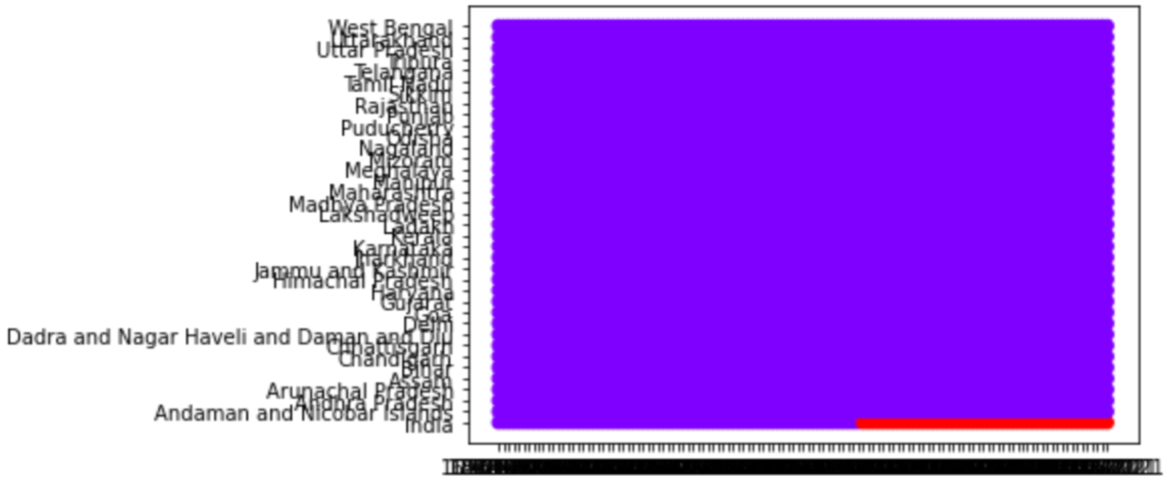
```
#view of first 5 data after scaling
df_scaled.head()
```

	Total Male(Individual Vaccinated	Total Individuals Vaccinated	Sessions Conducted	Total Sites	First Dose Administered	Second Dose Administered
0	-0.233317 0.23324	-0.240562	0.073418	-0.232202	-0.214308	-
1	-0.232462 0.23267	-0.235340	0.332679	-0.231346	-0.214308	-
2	-0.229080 0.23046	-0.230117	0.544165	-0.227962	-0.214308	-
3	-0.221126 0.22406	-0.225753	0.721766	-0.220001	-0.214308	-
4	-0.216510 0.22150	-0.217919	1.053210	-0.215382	-0.214308	-

```
kmeans = KMeans(n_clusters=2,init='k-means++')


kmeans.fit(df_scaled)

     KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
     n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
     (df_scaled)
    distortions.append(kmeanModel.inertia_)


plt.figure(figsize=(16,8))
plt.plot(K, distortions,
'bx-') plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



```
<matplotlib.collections.PathCollection at 0x7f506b91eed0>
```



Agglomerative Hierarchical:

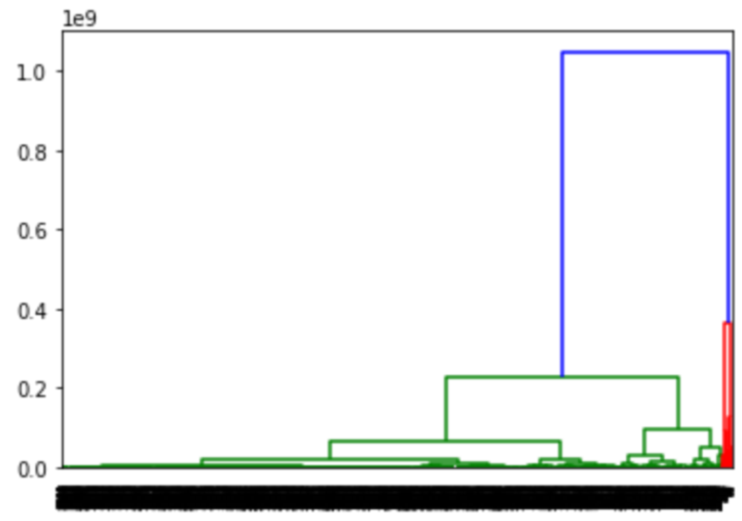
from google.colab import files uploaded = files.upload()

Browse... covid_vaccine_statewises.csv

covid_vaccine_statewises.csv(text/csv) - 342769 bytes, last modified: n/a - 100% done

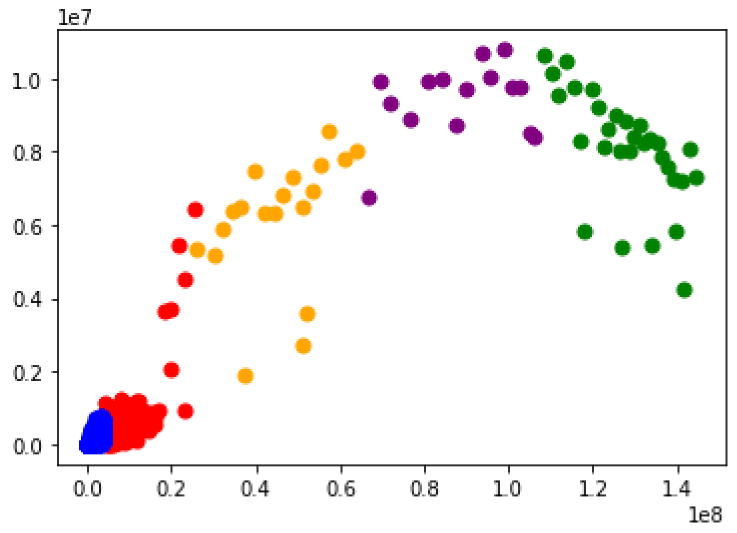
```
Saving covid_vaccine_statewises.csv to covid_vaccine_statewises (1).csv
```

```
import io

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering import
scipy.cluster.hierarchy as sch data =
pd.read_csv(io.BytesIO(uploaded['covid_vaccine_statewises.csv'])) X
= data.iloc[:, range(0,2)].values

dendrogram = sch.dendrogram(sch.linkage(X, method='ward'))
```



```
model = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
model.fit (x)



labels = model.labels
```

```
plt.scatter(X[labels==0, 0], X[labels==0, 1], s=50, marker='o', color='red')
plt.scatter(X[labels==1, 0], X[labels==1, 1], s=50, marker='o', color='blue')
plt.scatter(X[labels==2, 0], X[labels==2, 1], s=50, marker='o', color='green')
plt.scatter(X[labels==3, 0], X[labels==3, 1], s=50, marker='o', color='purple')
plt.scatter(X[labels==4, 0], X[labels==4, 1], s=50, marker='o', color='orange') plt.show()
```



DBSCAN:

```
from google.colab import files uploaded = files.upload()
```

```
Browse... covid_vaccine_statewises.csv
covid_vaccine_statewises.csv(text/csv) - 342769
bytes, last modified: n/a - 100% done
Saving covid_vaccine_statewises.csv to
covid_vaccine_statewises.csv
```

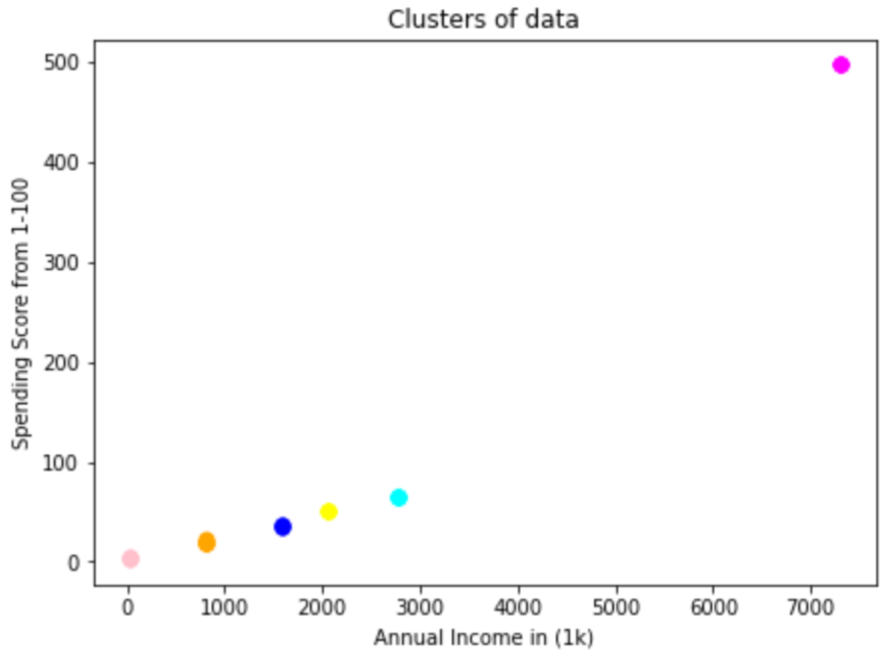
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

from sklearn.preprocessing import
StandardScaler from
sklearn.preprocessing import
normalize from
sklearn.decomposition import PCA

from sklearn import metrics

from sklearn.datasets import make_blobs


df =
pd.read_csv('covid_vaccine_statewises.csv')


x=df.iloc[:,range(0,11)].values


db=DBSCAN(eps=3,min_samples=4,metric='
euclidean')


model=db.fit(x)
label=model.la
bels_
```

```
#identifying the points which makes up our

core points

sample_cores=np.zeros_like(label,dtype=bo

ol)

sample_cores[db.core_sample_indices_]=Tr

ue #Calculating the number of clusters

n_clusters=len(set(label))- (1 if -1 in label
else 0) print('No of clusters:',n_clusters)
```

```
No of clusters: 6
```

```
y_means = db.fit_predict(x) plt.figure(figsize=(7,5))
plt.scatter(x[y_means == 0, 0], x[y_means == 0, 1], s = 50, c = 'pink')
plt.scatter(x[y_means == 1, 0], x[y_means == 1, 1], s = 50, c = 'yellow')
plt.scatter(x[y_means == 2, 0], x[y_means == 2, 1], s = 50, c = 'cyan')
plt.scatter(x[y_means == 3 0], x[y_means == 3 1], s = 50 c = 'magenta')
```

18s completed at 6:53 PM



5. CONCLUSION

In our project, we performed clustering on the various input data sets using K-means clustering. In the K-means algorithm, the initial centroids of the cluster are chosen as random and it is required to specify the number of clusters. Using Elbow method we found that the optimum number of clusters for the dataset is 2. However, this algorithm is computationally expensive because of large number of comparisons being made. So this algorithm is not very reliable for very big and complex dataset with multiple dimensions. Also it only works with numerical values.

Further we performed Agglomerative Hierarchical Clustering algorithm on the same dataset. It works in a bottom-up manner by treating each data point as a single cluster and then successively merges the atomic cluster into a single big cluster till termination conditions are satisfied. The dendograms created in this algorithm allows us to see how different sub clusters are related to each other. However, it involves a lot of arbitrary decisions which cannot be undone. Also it works poorly with datasets containing multiple data types.

Lastly we also performed Density Based Spatial Clustering of Applications with Noise (DBSCAN). It is the 2nd most used clustering algorithm after K-Means. This algorithm makes use of the density of data inputs within a region to discover new cluster. In this algorithm our main aim is to simplify the process of optimizing epsilon and MinPts using DBSCAN function of 'sklearn' python library. Finally we got 6 clusters which includes core points that are neighbours and all the border points of these core points.

FUTURE WORK

This work can be extended to perform incremental way of clustering as new data is streamed in. Also, identifying appropriate number of clusters for a particular data set through some kind of evaluation instead of empirical analysis could also be a possible extension.

References:

- Oyelade O. J, Oladipupo O. O, Obagbuwa I. C, "Application of k-Means Clustering algorithm for prediction of Students' Academic Performance," in *International Journal of Computer Science and Information Security*, Vol. 7, No. 1. Ota, Nigeria: 2010.

- https://towardsdatascience.com
- Wikipedia
- Youtube
- https://www.geeksforgeeks.org