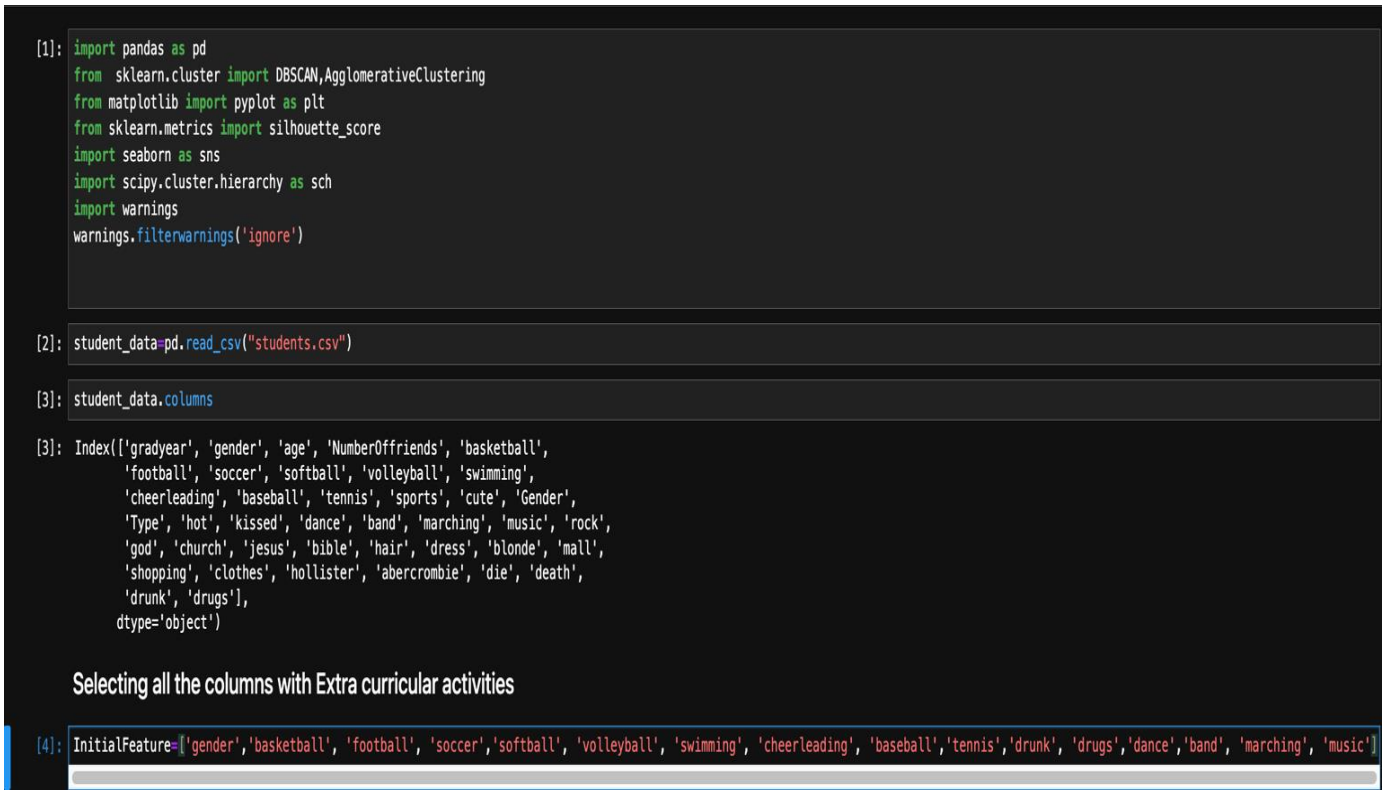
Machine Learning Assignment 2: -
DBSCAN and Hierarchical Clustering

Presented By: -
Ashutosh Samal (2403RES15)

Code Flow: -



```
[1]: import pandas as pd
     from  sklearn.cluster import DBSCAN,AgglomerativeClustering
     from matplotlib import pyplot as plt
     from sklearn.metrics import silhouette_score
     import seaborn as sns
     import scipy.cluster.hierarchy as sch
     import warnings
     warnings.filterwarnings('ignore')

[2]: student_data=pd.read_csv("students.csv")

[3]: student_data.columns

[3]: Index(['gradyear', 'gender', 'age', 'NumberOffriends', 'basketball',
            'football', 'soccer', 'softball', 'volleyball', 'swimming',
            'cheerleading', 'baseball', 'tennis', 'sports', 'cute', 'Gender',
            'Type', 'hot', 'kissed', 'dance', 'band', 'marching', 'music', 'rock',
            'god', 'church', 'jesus', 'bible', 'hair', 'dress', 'blonde', 'mall',
            'shopping', 'clothes', 'hollister', 'abercrombie', 'die', 'death',
            'drunk', 'drugs'],
           dtype='object')

     Selecting all the columns with Extra curricular activities

[4]: InitialFeature=['gender','basketball', 'football', 'soccer','softball', 'volleyball', 'swimming', 'cheerleading', 'baseball','tennis','drunk', 'drugs','dance','band', 'marching', 'music']
```

Initially selecting all the features which are related to extracurricular activities along with a categorical column gender.

Which are: -
InitialFeature=['gender','basketball', 'football', 'soccer','softball', 'volleyball', 'swimming', 'cheerleading', 'baseball','tennis','drunk', 'drugs','dance','band', 'marching', 'music']

```
[5]: student_data=student_data[InitialFeature]
     student_data.isnull().sum()
```

```
[5]: gender          1337
     basketball         0
     football           0
     soccer             0
     softball           0
     volleyball         0
     swimming           0
     cheerleading       0
     baseball           0
     tennis             0
     drunk              0
     drugs              0
     dance              0
     band               0
     marching           0
     music              0
     dtype: int64
```

As the gender column have null values i am droping those rows

```
[6]: student_data=student_data[InitialFeature]
     student_data=student_data.dropna()
```

```
[7]: student_data
```

[7]:

	gender	basketball	football	soccer	softball	volleyball	swimming	cheerleading	baseball	tennis	drunk	drugs	dance	band	marching	music
1	F	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0
2	F	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1
3	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	F	0	0	0	0	0	1	0	0	3	0	0	0	0	0	1
5	M	0	5	0	0	0	0	0	0	0	0	0	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14995	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14996	F	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
14997	F	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
14998	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

As gender column have 1337 null values I am dropping all the rows with null, so after that we are left with 13663 rows for further process.

As the Gender is a categorical feature i am doing one hot encoding to have two more column gender_M and gender_F which are boolean.

```
[8]: student_data=pd.get_dummies(student_data,columns=["gender"])  ## One hot encoding
```

```
[9]: student_data
```

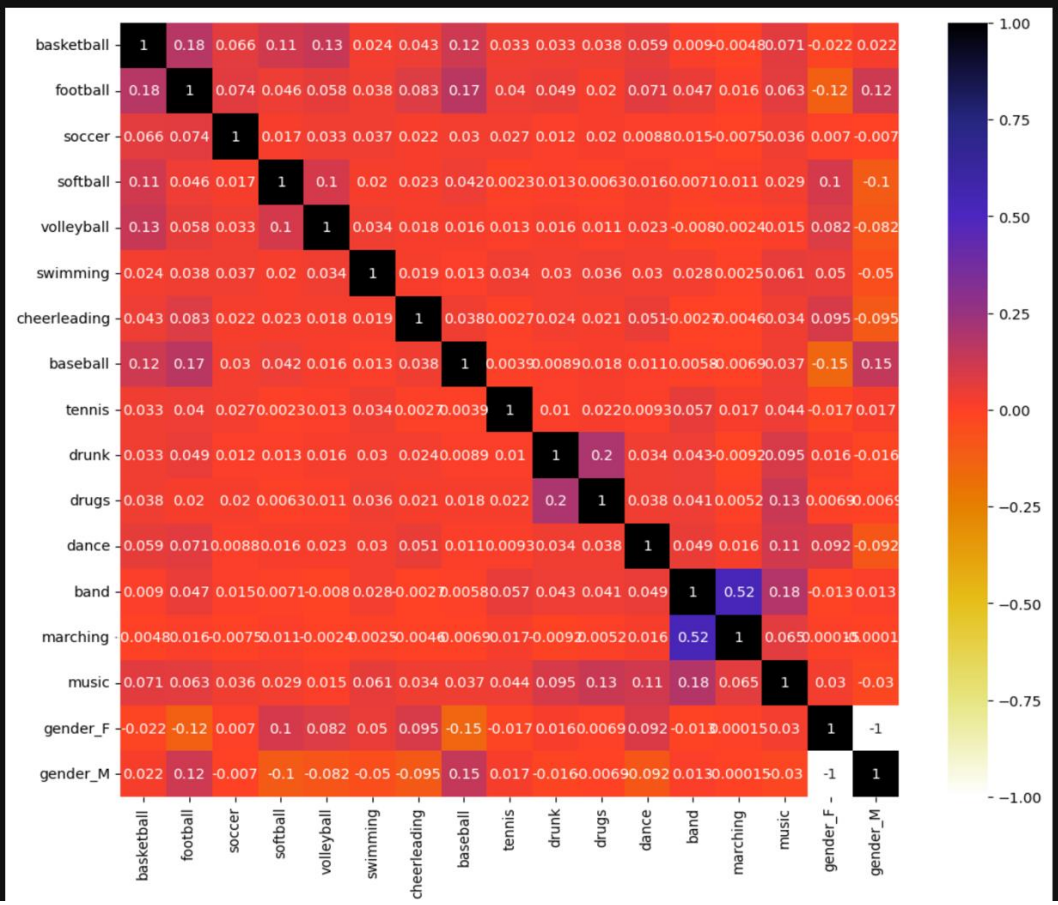
[9]:

	basketball	football	soccer	softball	volleyball	swimming	cheerleading	baseball	tennis	drunk	drugs	dance	band	marching	music	gender_F	gender_M
1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	True	False
2	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	True	False
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	True	False
4	0	0	0	0	0	1	0	0	3	0	0	0	0	0	1	True	False
5	0	5	0	0	0	0	0	0	0	0	0	0	1	0	1	False	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14995	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	True	False
14996	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	True	False
14997	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	True	False
14998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	True	False
14999	0	0	7	0	0	0	0	0	0	0	1	0	4	0	2	True	False

13663 rows × 17 columns

One hot encoding for gender column so we have 2 more columns gender_M and gender_F which are binary /boolean.

```
[11]: plt.figure(figsize=(12,10))
      corre=student_data.corr()
      sns.heatmap(corre,annot=True,cmap=plt.cm.CMRmap_r)

[11]: <Axes: >
```



Plotted a heat map for correlation to see if we have any similarity between columns so that we can eliminate columns based on this.

Here 2 features ‘Marching’ and ‘band’ have high correlations so I will only keep one with low standard deviation (Which I have done in later part).

```
[12]: def NumberOfOutliers(columns):
          count=0
          lower_range=columns.mean()-columns.std()
          upper_range=columns.mean()+columns.std()

          count+=(columns>=upper_range).sum()
          count+=(columns<=lower_range).sum()

          return (count/columns.count())*100
```

```
[13]: outlier_count=student_data.apply(lambda x:NumberOfOutliers(x))
```

```
[14]: outlier_count.sort_values()
```

```
[14]: marching         2.986167
      drugs            4.501208
      soccer           4.764693
      tennis           5.108688
      basketball       6.067482
      band             6.111396
      cheerleading     6.199224
      drunk            6.667643
      baseball         6.792066
      volleyball       8.233916
      softball         8.270512
      dance            9.375686
      swimming        10.261290
      music           16.658128
      footbal         17.236332
      gender_F        19.073410
      gender_M        19.073410
      dtype: float64
```

Here i will take top few feature as my final features for model with very less outliers

Here I have defined any point to be outlier if its value comes outside of the range mean+-standard_deviation i.e. if it's one standard deviation away from the mean.

This function will calculate the number of such records and returns the percentage of it, and I will be selecting top few columns with less deviation along with gender column.

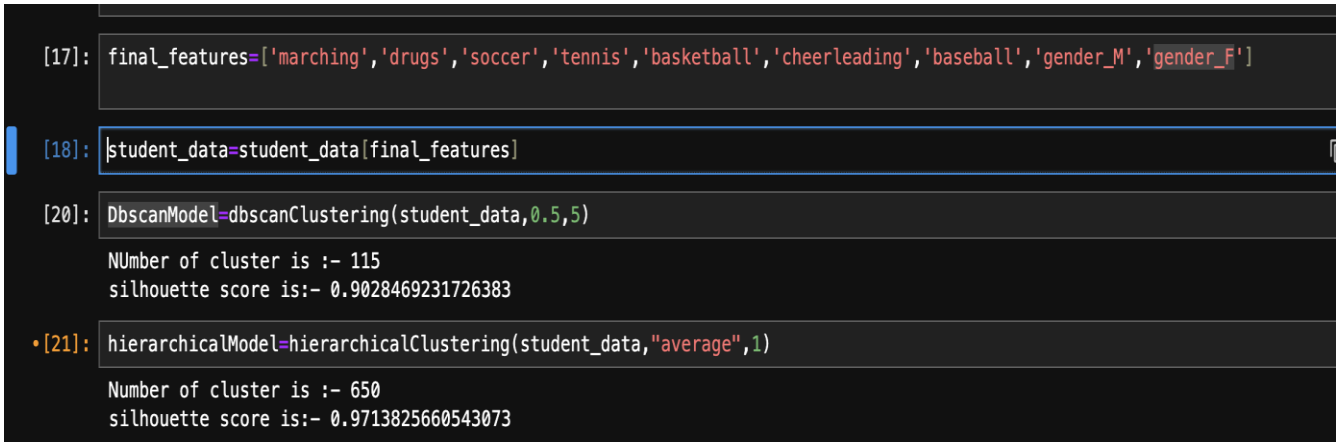
```
[15]: def dbscanClustering(data,eps,min_samples):
          clustering=DBSCAN(eps=eps,min_samples=min_samples)
          result=clustering.fit(student_data)
          labels=result.labels_
          score=silhouette_score(data, labels)
          print("NUmber of cluster is :-",len(set(labels)))
          print("silhouette score is:-",score)
          return result

[16]: def hierarchicalClustering(data,linkage,distance_threshold):
          clusters=AgglomerativeClustering(n_clusters=None,linkage=linkage,distance_threshold=distance_threshold)
          result=clusters.fit(data)
          lables=result.labels_
          score=silhouette_score(data,lables)
          print("Number of cluster is :-",len(set(lables)))
          print("silhouette score is:-",score)
          return result

[47]: def numberPointsIneachCluster(data,model):
          labels=model.labels_

          unique, counts = np.unique(labels, return_counts=True)

          count_map=np.asarray((unique, counts)).T

          for i in count_map:
              print("Number of records in ",i[0],"label is :-",i[1])
```

Here 1st function will run DBSCAN using sklearn library . It takes dataset, eps and min sample as input and returns the model and print number of clusters and silhouette score.

Here 2nd function will run Hierarchical clustering using sklearn library. It takes dataset, linkage type (possible value: - '***ward', 'complete', 'average', 'single***) and threshold distance as input and returns the model and print number of clusters and silhouette score.

3rd Function takes dataset and model to print number of records in each label.

My final features and result of Dbscan and hierarchical clustering: -



Number of records in each cluster: -

Hierarchical Clustering: -

```
[48]: numberPointsIneachCluster(student_data,hierarchicalModel)

Number of records in  0 label is :- 4
Number of records in  1 label is :- 3
Number of records in  2 label is :- 5
Number of records in  3 label is :- 3
Number of records in  4 label is :- 3
Number of records in  5 label is :- 2
Number of records in  6 label is :- 6
Number of records in  7 label is :- 2
Number of records in  8 label is :- 2
Number of records in  9 label is :- 2
Number of records in  10 label is :- 11
Number of records in  11 label is :- 2
Number of records in  12 label is :- 6
Number of records in  13 label is :- 4
Number of records in  14 label is :- 6
Number of records in  15 label is :- 2
Number of records in  16 label is :- 8
Number of records in  17 label is :- 9
Number of records in  18 label is :- 2
Number of records in  19 label is :- 9
Number of records in  20 label is :- 3
Number of records in  21 label is :- 2
Number of records in  22 label is :- 2
Number of records in  23 label is :- 2
Number of records in  24 label is :- 2
Number of records in  25 label is :- 3
Number of records in  26 label is :- 2
Number of records in  27 label is :- 3
Number of records in  28 label is :- 2
Number of records in  29 label is :- 2
Number of records in  30 label is :- 20
Number of records in  31 label is :- 3
Number of records in  32 label is :- 2
Number of records in  33 label is :- 2
Number of records in  34 label is :- 3
Number of records in  35 label is :- 4
Number of records in  36 label is :- 2
Number of records in  37 label is :- 6
Number of records in  38 label is :- 2
Number of records in  39 label is :- 3
Number of records in  40 label is :- 2
Number of records in  41 label is :- 2
Number of records in  42 label is :- 3
```

DBSCAN: -

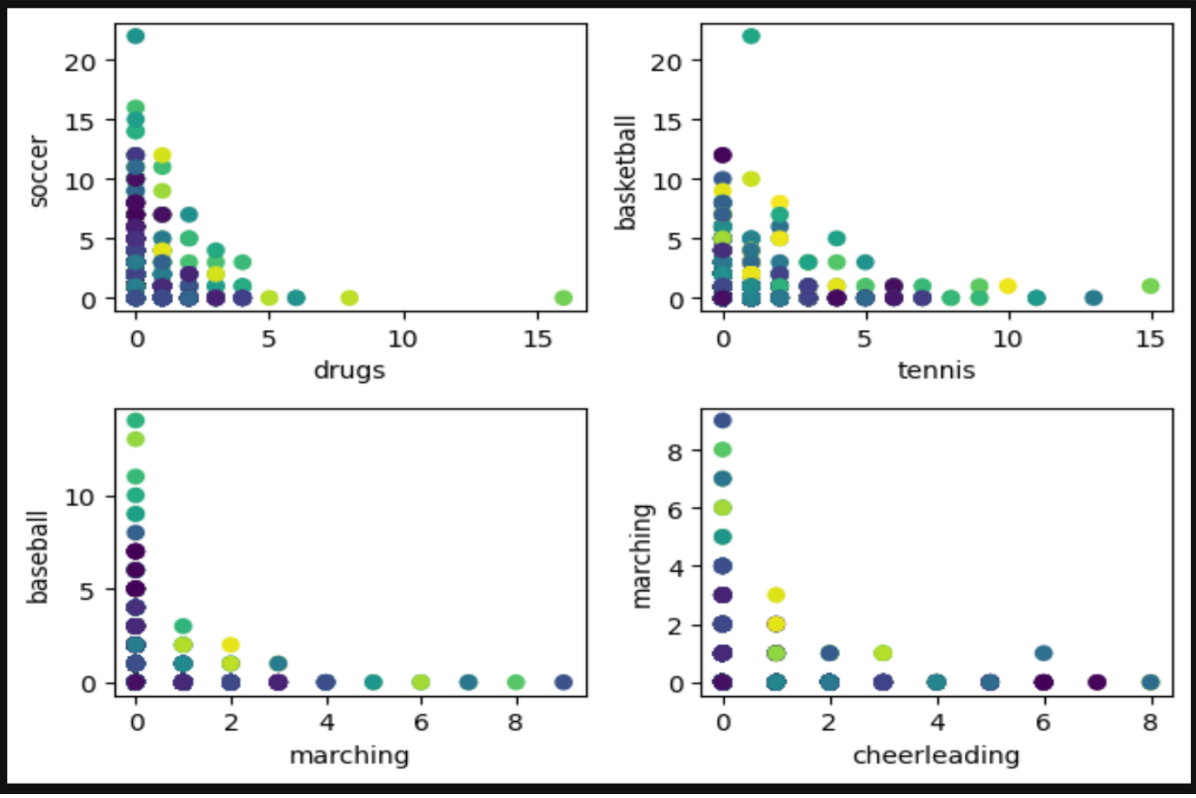
```
[49]: numberPointsIneachCluster(student_data,DbscanModel)

Number of records in  -1 label is :- 777
Number of records in  0 label is :- 329
Number of records in  1 label is :- 12
Number of records in  2 label is :- 6796
Number of records in  3 label is :- 23
Number of records in  4 label is :- 1561
Number of records in  5 label is :- 33
Number of records in  6 label is :- 150
Number of records in  7 label is :- 195
Number of records in  8 label is :- 639
Number of records in  9 label is :- 39
Number of records in  10 label is :- 71
Number of records in  11 label is :- 24
Number of records in  12 label is :- 276
Number of records in  13 label is :- 80
Number of records in  14 label is :- 24
Number of records in  15 label is :- 5
Number of records in  16 label is :- 183
Number of records in  17 label is :- 11
Number of records in  18 label is :- 215
Number of records in  19 label is :- 209
Number of records in  20 label is :- 5
Number of records in  21 label is :- 117
Number of records in  22 label is :- 23
Number of records in  23 label is :- 179
Number of records in  24 label is :- 37
Number of records in  25 label is :- 23
Number of records in  26 label is :- 7
Number of records in  27 label is :- 85
Number of records in  28 label is :- 37
Number of records in  29 label is :- 71
Number of records in  30 label is :- 104
Number of records in  31 label is :- 51
Number of records in  32 label is :- 57
Number of records in  33 label is :- 8
Number of records in  34 label is :- 9
Number of records in  35 label is :- 24
Number of records in  36 label is :- 6
Number of records in  37 label is :- 51
Number of records in  38 label is :- 44
Number of records in  39 label is :- 8
Number of records in  40 label is :- 9
Number of records in  41 label is :- 53
Number of records in  42 label is :- 16
```

This function takes dataset and model and print a figure with 4 graphs, I took features for plotting randomly.

```
2]: def plotGrapg(data,model):
        lables=model.labels_
        fig=plt.figure()

        plot1=fig.add_subplot(221)
        plot2=fig.add_subplot(222)
        plot3=fig.add_subplot(223)
        plot4=fig.add_subplot(224)

        plot1.scatter(data['drugs'],data['soccer'],c=lables)
        plot1.set_xlabel("drugs")
        plot1.set_ylabel("soccer")


        plot2.scatter(data["tennis"] ,data["basketball"] , c=lables)

        plot2.set_xlabel("tennis")
        plot2.set_ylabel("basketball")

        plot3.scatter(data["marching"] ,data["baseball"] , c=lables)

        plot3.set_xlabel("marching")
        plot3.set_ylabel("baseball")

        plot4.scatter(data["cheerleading"] ,data["marching"] , c=lables)

        plot4.set_xlabel("cheerleading")
        plot4.set_ylabel("marching")
        fig.tight_layout()


        fig.show()
```

Output of the function: -

```
: plotGrapg(student_data,hierarchicalModel)
```




```
[24]: plotGrapg(student_data,DbscanModel)
```