USER PROFILE MATCHING IN SOCIAL NETWORKS

GROUP 13

FINAL REPORT

MLDM - 15 CSE 401

CB.EN.U4CSE17014	CH. SAI PHANI JASWANTH
CB.EN.U4CSE17043	NANDITHA MENON
CB.EN.U4CSE17049	RISHI VIGNESH
CB.EN.U4CSE17065	T. SAI SRIHITHA REDDY

INTRODUCTION

Social networking sites, especially online dating sites, maintain a user profile database to match a user with others who have similar preferences and tastes. The users specify specific criteria, based on which their profiles are matched with that of other users. We plan to use Clustering algorithms to find users that are most similar to each other and recommend dating profiles the same way Spotify or Netflix recommends different music and movies. Our focus on this project will be using unsupervised machine learning algorithms for Clustering. We will be clustering users based on their profile and showing them profiles that are most similar to their own. By using the clustering algorithm like k-means, we can reduce the number of profiles to be compared in the knn clustering algorithm.

Clustering is one of the most common exploratory data analysis techniques used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. Clustering analysis can be done on the basis of features where we try to find subgroups of samples based on features or on the basis of samples where we try to find subgroups of features based on samples. Clustering is considered an unsupervised learning method since we don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. We only want to try to investigate the structure of the data by grouping the data points into distinct subgroups.

ALGORITHMS USED

Kmeans is considered as one of the most used clustering algorithms due to its simplicity. K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different as possible.The less variation we have within clusters, the more homogeneous the data points are within the same cluster.

Hierarchical Clustering, also called Hierarchical cluster analysis or HCA is an unsupervised clustering algorithm which involves creating clusters that have predominant ordering from top to bottom. The algorithm groups similar objects into groups called clusters. The endpoint is a set of clusters or groups, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular unsupervised learning method utilized in model building and machine learning algorithms. It is used in machine learning to separate clusters of high density from clusters of low density. Given that DBSCAN is a density based clustering algorithm, it does a great job of seeking areas in the data that have a high density of observations, versus areas of the data that are not very dense with

observations. DBSCAN can sort data into clusters of varying shapes as well, another strong advantage.

KNN is a model that classifies data points based on the points that are most similar to it. It uses test data to make an "educated guess" on what an unclassified point should be classified as. KNN is an algorithm that is considered both non-parametric and an example of lazy learning. KNN is often used in simple recommendation systems, image recognition technology, and decision-making models.

PROBLEM STATEMENT

To recommend k similar profiles to a user based on the personal information provided through the survey.

OBJECTIVE

- To cluster the users into various groups
- To apply knn on each cluster to find k nearest profiles for a user
- To apply knn on the whole dataset to find k nearest profiles for a user

NOVELTY OF THE WORK

Clustering the users reduces the size of the dataset passed into the knn algorithm. This in turn leads to reduced execution time while finding k nearest neighbours of the current node. Clustering can also provide deeper insights into the users through clusters and related cluster analysis.

METHOD

1. Generating dataset
2. Analysing the data
3. Preprocessing the data
4. Applying clustering algorithms
5. Recommend top 5 similar profiles

ALGORITHM

1. Apply clustering to create clusters of users with similar profiles
2. Use knn clustering to find top k profiles identical to the user
3. Compare the results with when knn is performed on the whole dataset

DATA COLLECTION AND PREPARATION

Due to the privacy issues of the dating websites and social networking websites, the data has been collected from students by circulating a google form.
The responses of the google have been saved in the form of csv

Timestamp	Email Address	Name	Age	Zodiac	Gender	I'm an
9-8-2020 13:34:28	nandithankumar@gmail.co	Nanditha Menon	21	Leo	Female	Ambivert
9-8-2020 14:43:00	titusaishu@gmail.com	Aishwarya Titus	21	Aries	Female	Ambivert
9-8-2020 14:44:08	meghanab2000@gmail.cor	B Meghana	21	Pisces	Female	Introvert
9-8-2020 14:45:07	sayoojsanthosh21@gmail.(	Sayooj	20		Male	Ambivert
9-8-2020 14:45:54	titusaishu@gmail.com	Aishwarya Titus	21	Aries	Female	Ambivert

I prefer	Books or Movies(Movies i	Books or Movie Genre	Music. Yeah yeah, there a	Cook or Take out	Pets
Warm and cozy Indoors.	Both	Crime, Drama, Fantasy	Pop, Hip Hop, EDM	Both	Dogs
The great Outdoors	Both	Action Genre, Animation, C	Blues, Country, Pop, Hip H	Both	Dogs
Warm and cozy Indoors.	Movies	Crime, Historical, Romance	Hip Hop, Bollywood	Cook	No pets
The great Outdoors	Movies	Action Genre, Animation, C	Blues, Jazz, Rock Music, C	Both	Both
The great Outdoors	Both	Action Genre, Animation, C	Blues, Country, Pop, Hip H	Both	Dogs

Columns - Timestamp, Email Address, Name, Age, Zodiac, Gender, Character, Preference, TV Show Genres, Music Genres, Food, Pets

IMPLEMENTATION

DATA CLEANING

The dataset is cleaned by renaming the columns into

```
[ ]  df.columns=["timestamp","email","name","age","sunSign","gender","character","preferance","TV","genres","music","food","pets"]
```

And dropping duplicate entries based on the email column -

```
[ ]  df.drop_duplicates(subset=["email"],inplace=True)
```

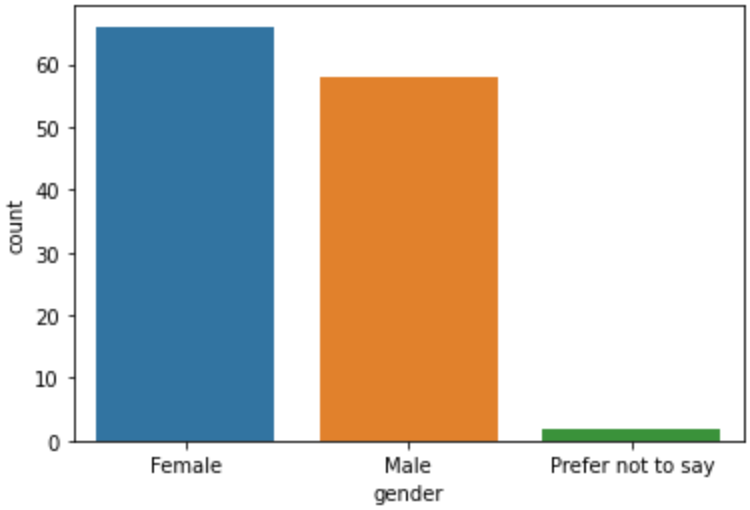
```
[ ]  df.head()
```

	email	name	age	sunSign	gender	character	preferance	TV	genres	music	food	pets
0	nandithankumar@gmail.com	Nanditha Menon	21	Leo	Female	Ambivert	Warm and cozy Indoors.	Both	Crime, Drama, Fantasy	Pop, Hip Hop, EDM	Both	Dogs
1	titusaishu@gmail.com	Aishwarya Titus	21	Aries	Female	Ambivert	The great Outdoors	Both	Action Genre, Animation, Comedy Genre, Crime, ...	Blues, Country, Pop, Hip Hop, EDM	Both	Dogs
2	meghanab2000@gmail.com	B Meghana	21	Pisces	Female	Introvert	Warm and cozy Indoors.	Movies	Crime, Historical, Romance, Science Fiction	Hip Hop, Bollywood	Cook	No pets
3	sayoojsanthosh21@gmail.com	Sayooj	20	NaN	Male	Ambivert	The great Outdoors	Movies	Action Genre, Animation, Comedy Genre, Crime, ...	Blues, Jazz, Rock Music, Country, Rap	Both	Both
4	vishnu21200@gmail.com	Vishnu Mahesh Pothuvath	20	Pisces	Male	Ambivert	Warm and cozy Indoors.	Both	Comedy Genre, Historical, Romance, Science Fic...	Jazz, Country, Pop, EDM	Both	Dogs

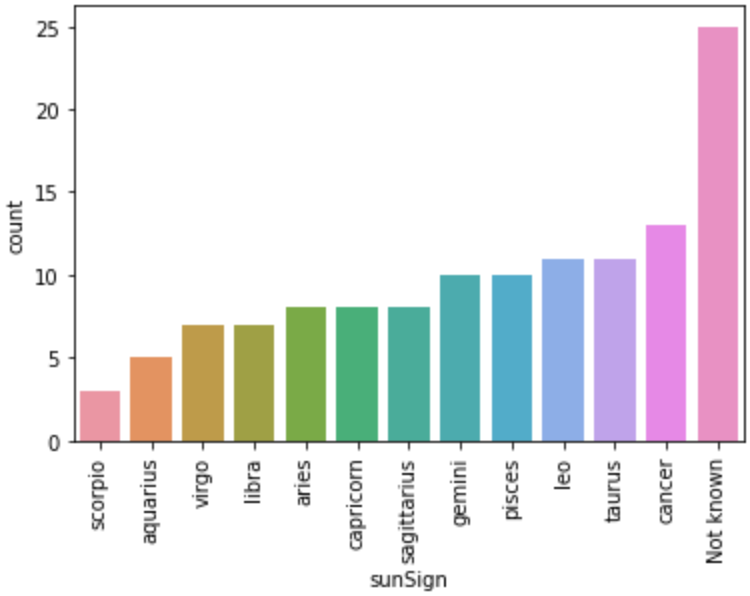
SNIPPET OF DATASET

EXPLORATORY DATA ANALYSIS

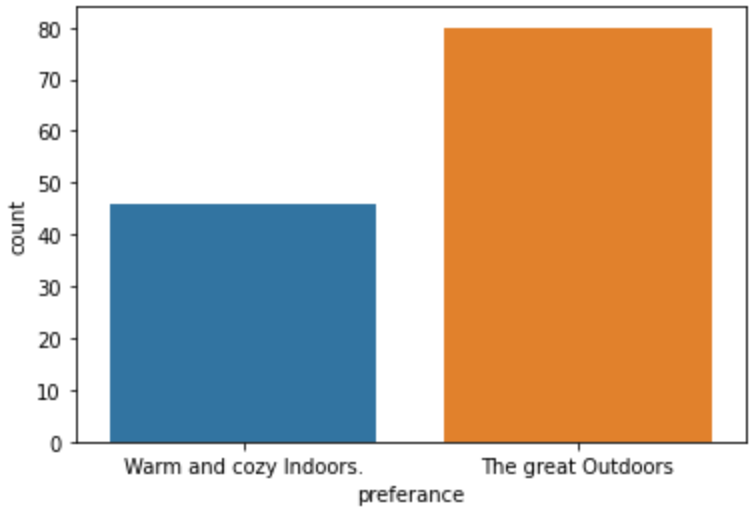
COUNT PLOT ON GENDER



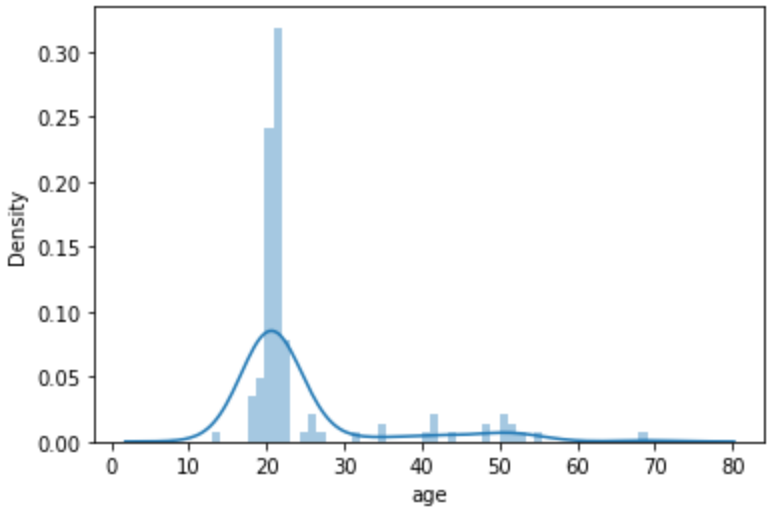
COUNT PLOT ON ZODIAC SIGNS



COUNT PLOT ON OUTDOOR VS INDOOR PREFERENCE

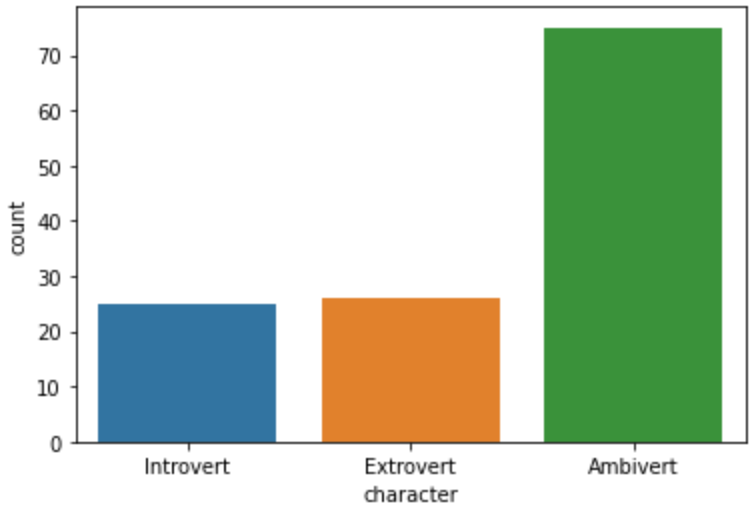


AGE DISTPLOT

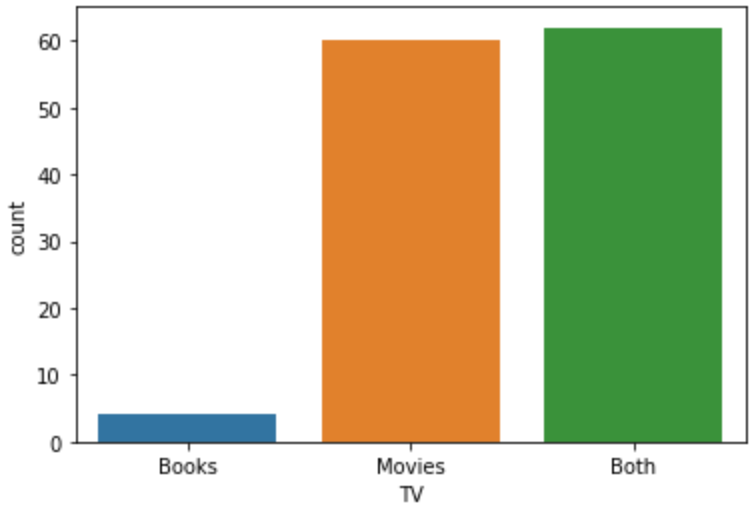


The above distplot is justified because the data mostly has been gathered from the college students whose age ranges form 18-22.

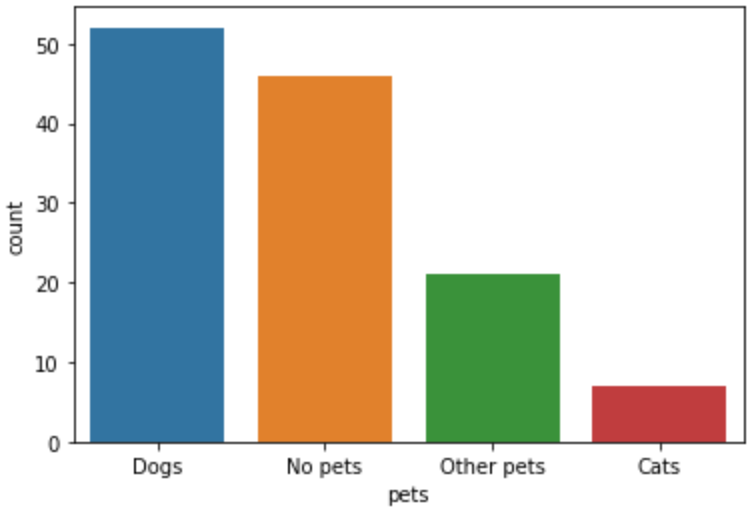
CHARACTER COUNT PLOT



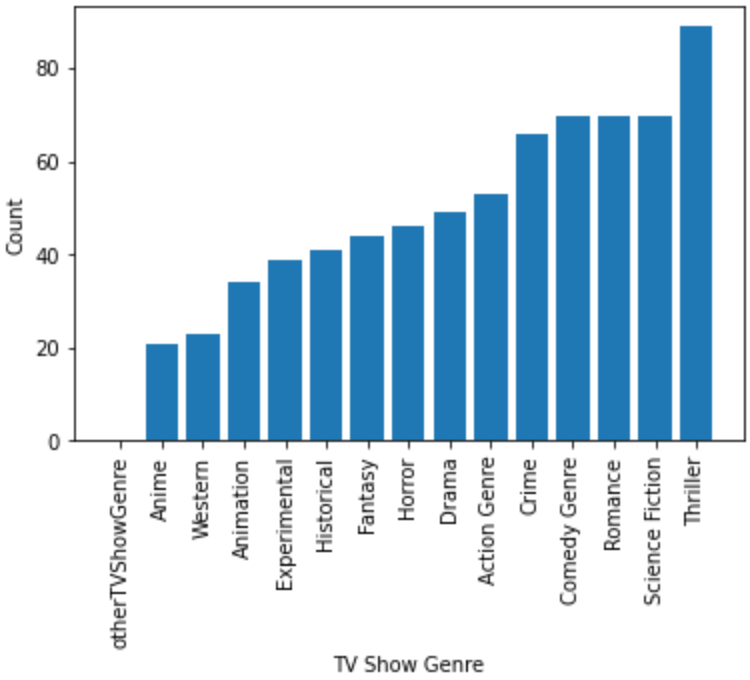
TV PREFERENCE COUNT PLOT



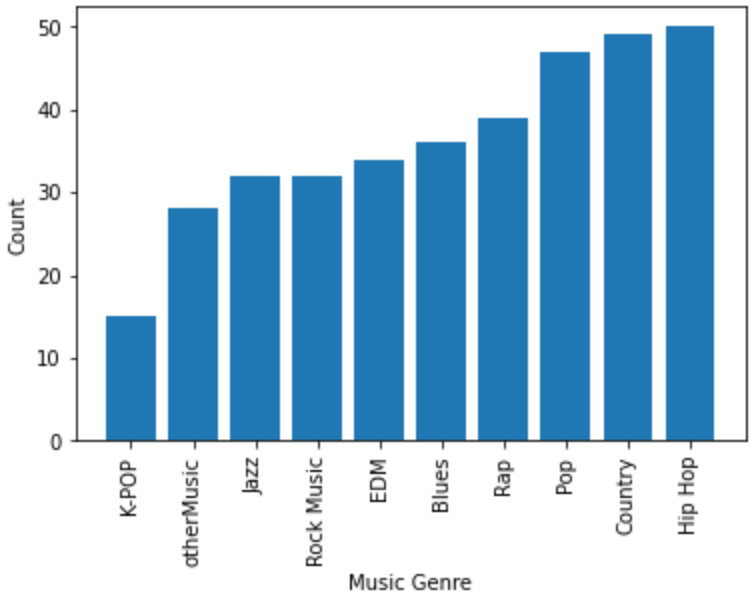
PET PREFERENCE



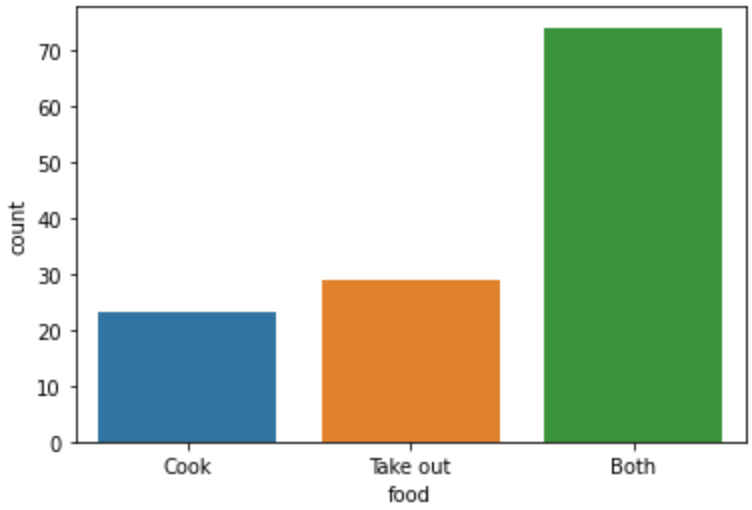
TV SHOW GENRE PREFERENCE



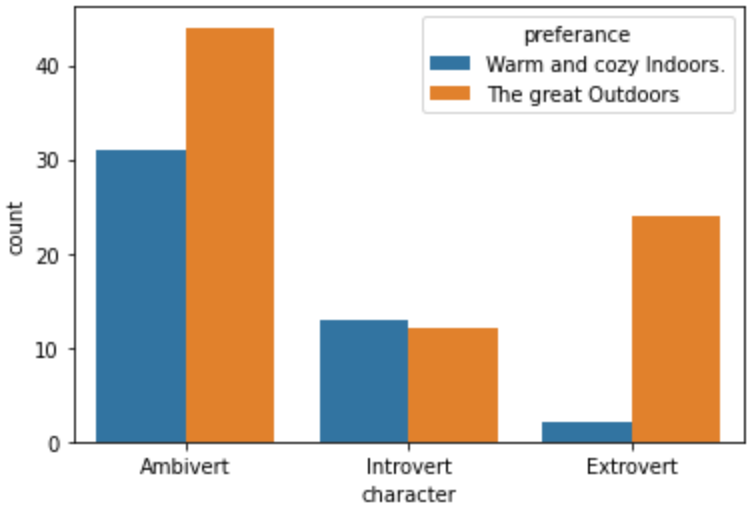
MUSIC GENRE PREFERENCE



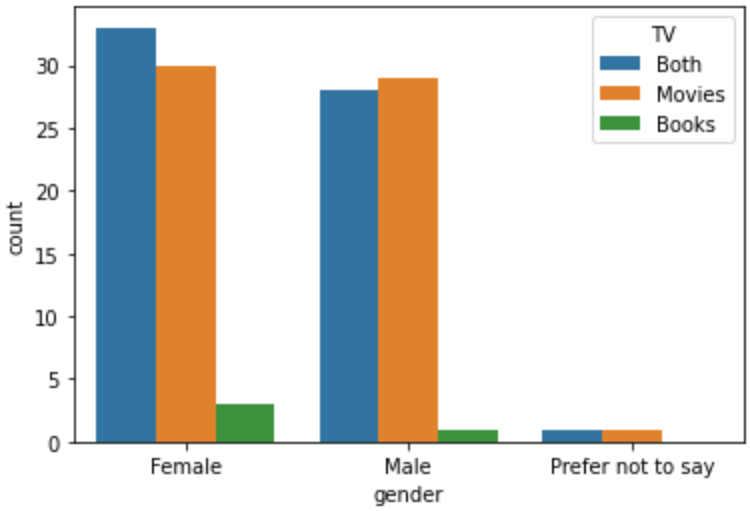
FOOD PREFERENCE COUNT PLOT



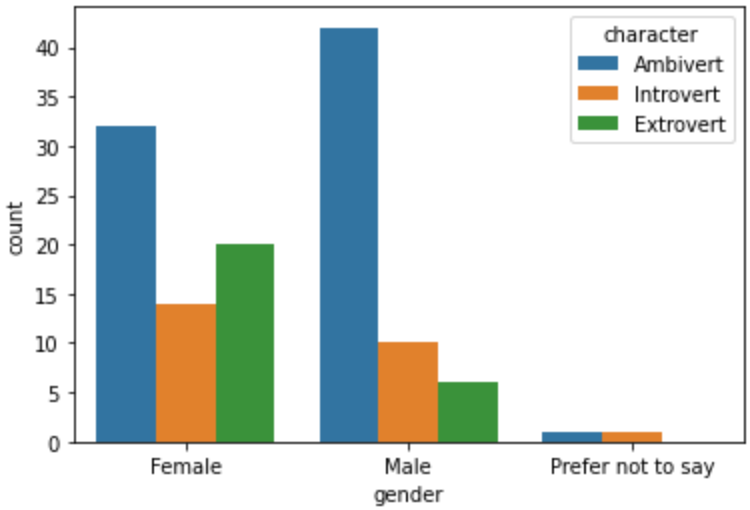
PLOTTING CHARACTER AGAINST OUTDOOR VS INDOOR PREFERENCE



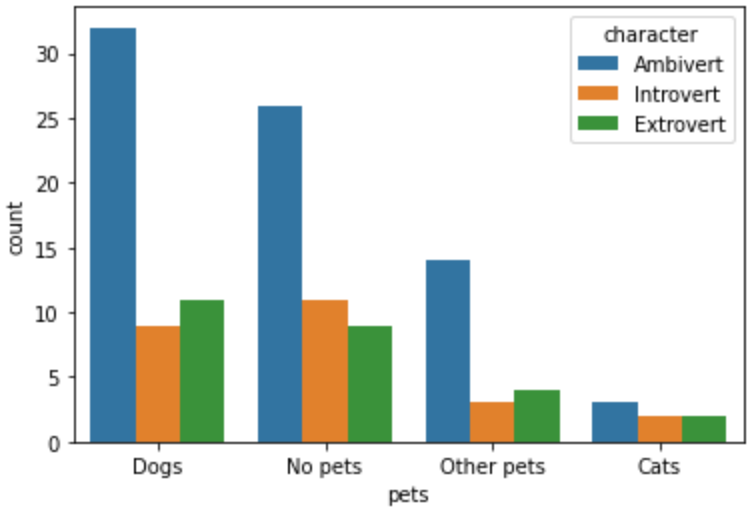
PLOTTING GENDER AGAINST TV PREFERENCES



PLOTTING GENDER AGAINST CHARACTER



PLOTTING PETS AGAINST CHARACTER



DATA PREPROCESSING

Convert all the categorical values into binary value columns using get_dummies function and label_encoder from sklearn.preprocessing.
Store the name and email details in a separate data frame.
Normalize the age to between 0 and 1 using MinMaxScaler from sklearn.preprocessing.

```
details=df.iloc[:,:2]
details.head()
```

	email	name
0	nandithankumar@gmail.com	Nanditha Menon
1	titusaishu@gmail.com	Aishwarya Titus
2	meghanab2000@gmail.com	B Meghana
3	sayoojsanthosh21@gmail.com	Sayooj
4	vishnu21200@gmail.com	Vishnu Mahesh Pothuvath

	age_normalized	gender	preferance	Western	Experimental	Romance	Action Genre	Animation	Science Fiction	Horror	Historical	Fantasy	Drama	Thriller
0	0.142857	0	1	0	0	0	0	0	0	0	0	1	1	0
1	0.142857	0	0	0	0	0	1	1	1	0	0	1	0	1
2	0.142857	0	1	0	0	1	0	0	1	0	1	0	0	0
3	0.125000	1	0	0	1	0	1	1	1	1	1	0	0	1
4	0.125000	1	1	0	0	1	0	0	1	0	1	0	0	1

otherTVShowGenre	Crime	Comedy Genre	Anime	Pop	Blues	Hip Hop	Jazz	Country	EDM	otherMusic	K-POP	Rock Music	Rap	Not known	aquarius	aries	cancer	capricorn
0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	1	0	1	1	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0
0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0

gemini	leo	libra	pisces	sagittarius	scorpio	taurus	virgo	Both_Food	Cook	Take out	Books	Both	Movies	Cats	Dogs	No pets	Other pets	Ambivert
0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1
0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1

Extrovert	Introvert
0	0
0	0
0	1
0	0
0	0

SNIPPET OF DATASET AFTER PREPROCESSING

CHECKING FOR COLLINEARITY

When variables used in clustering have strong correlation, some variables get a higher weight than others. If two variables are perfectly correlated, they effectively represent the same concept. But that concept is now represented twice in the data and hence gets twice the weight of all the other variables. The final solution is likely to be skewed in the direction of that concept, which could be a problem if it's not anticipated.

```
[ ]  corr = df.corr()
     c1 = corr.abs()
     upper = c1.where(np.triu(np.ones(c1.shape), k=1).astype(np.bool))
     # Find index of feature columns with correlation greater than 0.5
     to_drop = [column for column in upper.columns if any(upper[column] >= 0.5)]
     print(to_drop)
```

```
['Cook', 'Take out', 'Movies', 'No pets', 'Extrovert', 'Introvert']
```

The columns - `['Cook', 'Take out', 'Movies', 'No pets', 'Extrovert', 'Introvert']` have correlation coefficients more than 0.5, making them columns with strong collinearity.

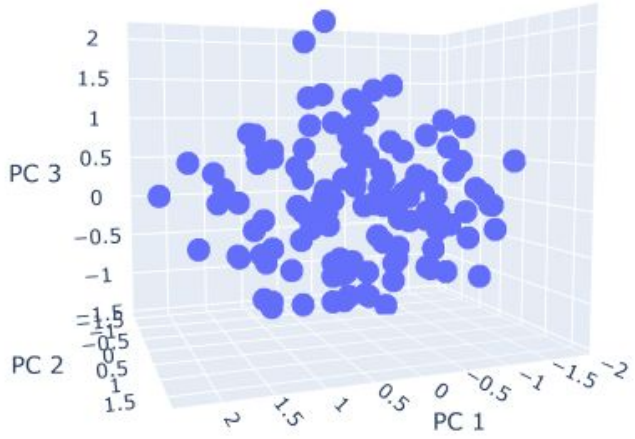
Drop the columns from the dataset.

```
df.drop(columns=['Cook', 'Take out', 'Movies', 'No pets', 'Extrovert', 'Introvert'],axis=1,inplace=True)
```

Since the dimensionality of the current dataset is still exceeding 50, PCA(Principal Component Analysis) is used to reduce the number of components of the dataset.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
newdf=pca.fit_transform(df)
```

3D PLOT OF THE REDUCED DATASET



CLUSTERING ALGORITHMS
HIERARCHICAL CLUSTERING

```
import scipy.cluster.hierarchy as hc
linkage_ward=hc.linkage(newdf, method='ward',optimal_ordering=True)
linkage_complete=hc.linkage(newdf, method='complete',optimal_ordering=True)
linkage_centroid=hc.linkage(newdf, method='centroid',optimal_ordering=True)
linkage_average=hc.linkage(newdf, method='average',optimal_ordering=True)
linkage_weighted=hc.linkage(newdf, method='weighted',optimal_ordering=True)
```

```
from scipy.spatial.distance import pdist
for i in [linkage_ward,linkage_complete,linkage_centroid,linkage_average,linkage_weighted]:
    c, coph_dists = hc.cophenet(i, pdist(newdf))
    print(c)
```

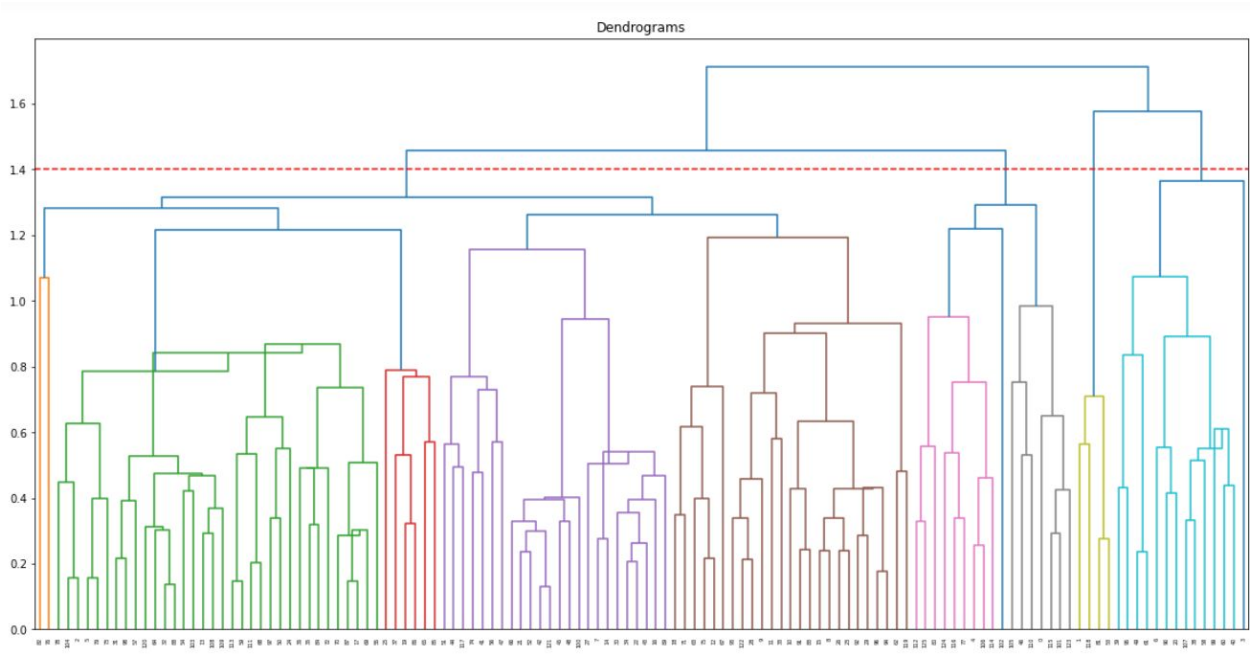
```
0.545819688776636
0.5125297813753943
0.6002306745588123
0.59978614921382
0.5494299660383987
```

Hierarchical clustering has many methods: ward, complete, centroid, average, weighted, etc. To find the best method for the current dataset we use cophenetic distance to compare these methods.

Cophenetic distance is simply correlation coefficient between distance matrix and cophenetic matrix. The higher the coefficient, the better the clustering fit. Centroid method linkage seems to provide the highest cophenetic distance. Using this method a dendrogram is drawn.

The scipy.cluster.hierarchy library has linkage and dendrogram functions which can be used to plot the clustering.

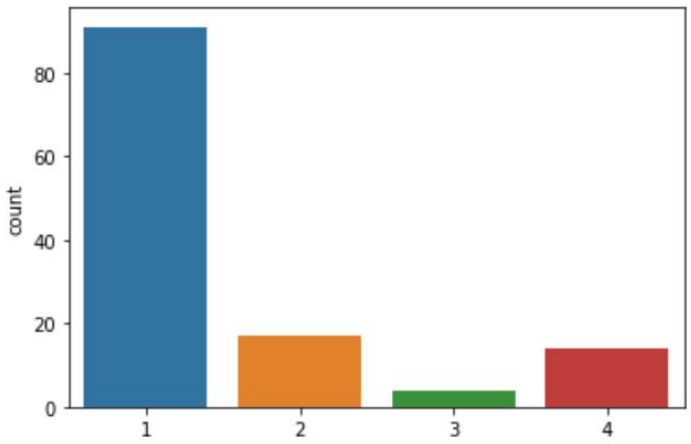
To choose the number of clusters, a horizontal line is drawn through 2 clusters which are the most farther from each other. In the below dendrogram, such a horizontal line passes through 4 vertical lines. So the apt number of clusters for the given data is 4!



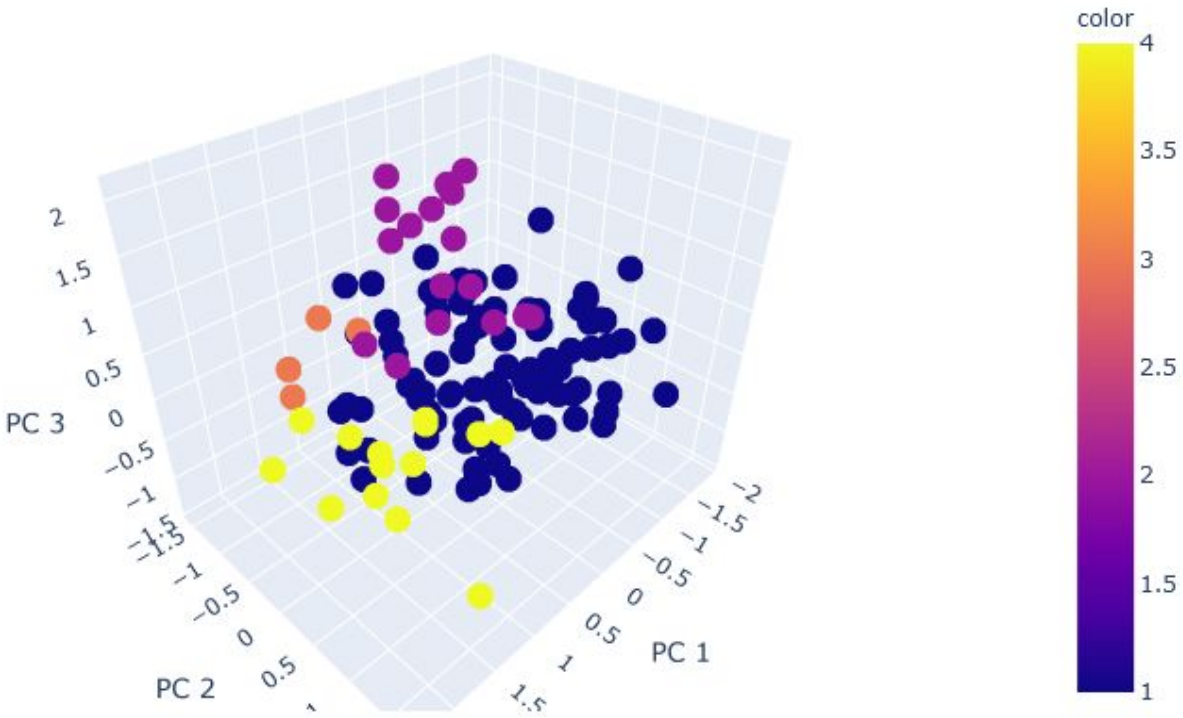

```
fl = hc.fcluster(linkage,4,criterion='maxclust')
```

fcluster from the same package can give the labels of the nodes in the dataset.

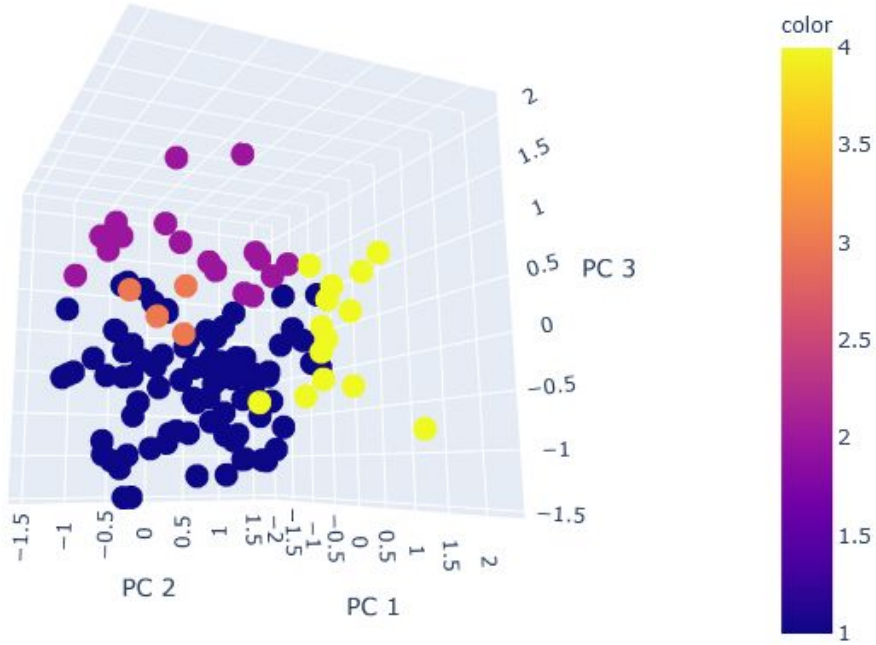
COUNT PLOT ON LABELS OF CLUSTERS



Using PCA to reduce the dimensions and plotting the nodes in a 3D map colored by the labels obtained gives -



VIEW 1



VIEW 2

K-MEANS

To find the appropriate number of clusters for k means we use silhouette score. Silhouette score is calculated using the mean intracluster distance ‘a’ and the mean nearest cluster distance ‘b’ for each sample and is calculated by (b - a) / max(a, b).

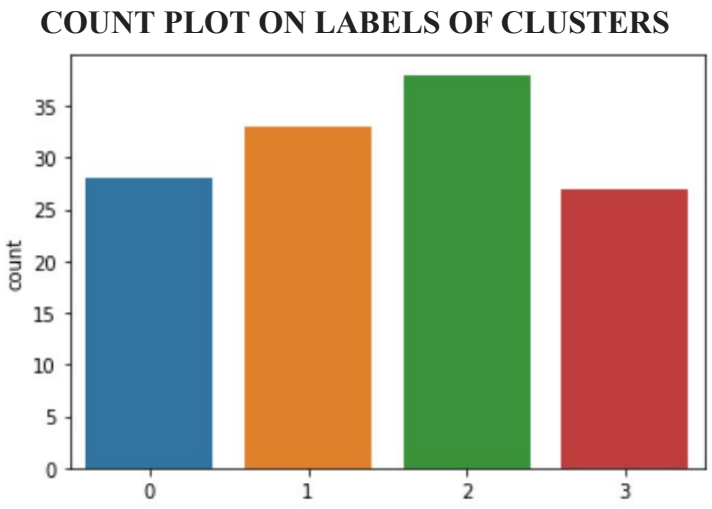
```
from sklearn import metrics
from sklearn.cluster import KMeans
for i in range(2,7):
    kmeans = KMeans(n_clusters=i, random_state=3).fit(newdf)
    y_kmeans=kmeans.labels_
    print(metrics.silhouette_score(newdf, y_kmeans, metric='euclidean'))
```

```
0.26645400075471215
0.2797289374167322
0.28613549414415057
0.2855413777239677
0.27821498385175686
```

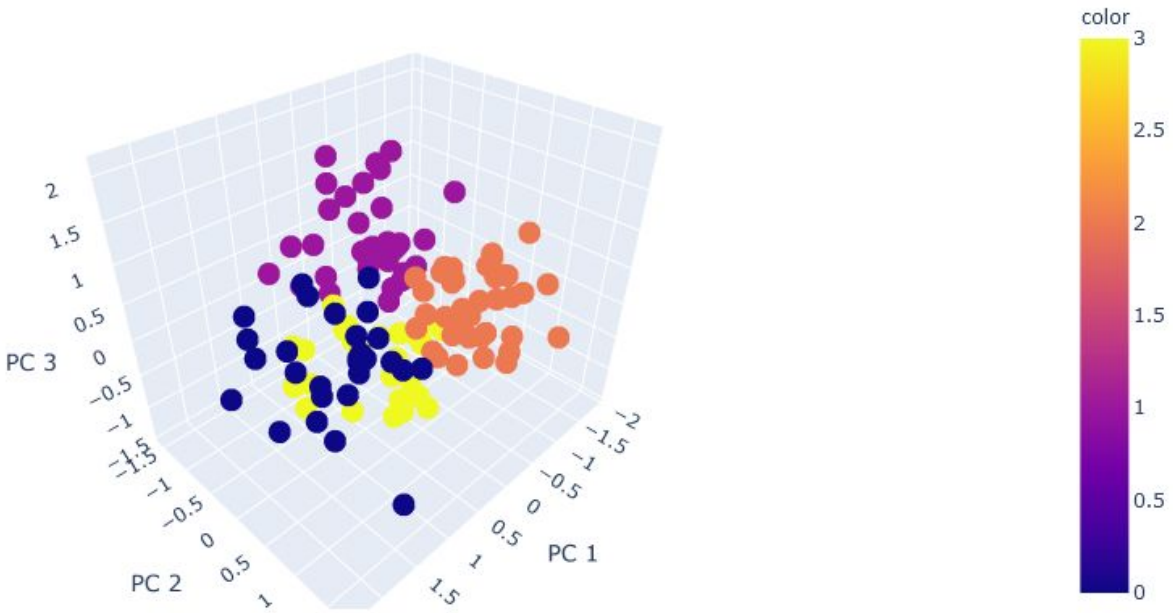
Applying K-Means on the dataset for clustering

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, random_state=3,n_init=50).fit(newdf)
```

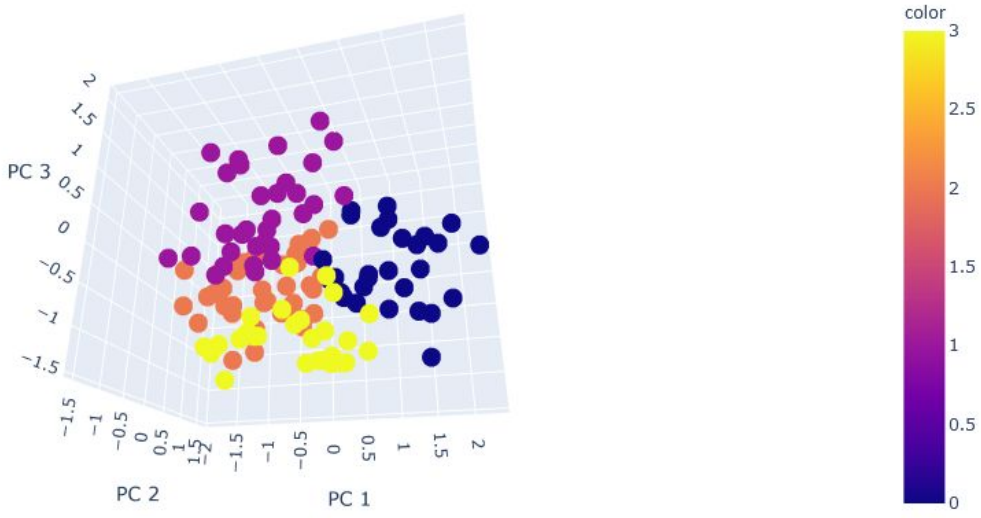
```
y_kmeans=kmeans.labels_
```



Coloring the 3D plot with k means predicted labels -



VIEW 1



VIEW 2

DBSCAN

In the DBSCAN algorithm, we have 2 attributes to be optimised: eps and min_samples.
If the distance between two points is below the threshold epsilon(eps), then the two points are considered neighbours. Min_samples is the minimum number of neighbours a given point should have in order to be classified as a care point.

Core point: A point with at least min_samples points whose distance with respect to the point is below the threshold defined by epsilon.

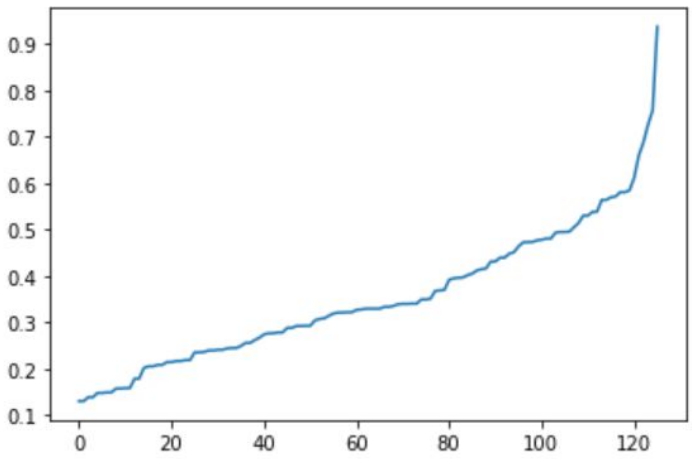
Border point: A point that isn't in close proximity to at least min_samples points but is close enough to one or more core points. Border points are included in the cluster of the closest core point.

Noise point: Points that aren't close enough to core points to be considered border points. Noise points are ignored. That is to say, they aren't part of any cluster.

We can find the suitable value for epsilon by calculating the distance to the nearest n points for each point, sorting and plotting the results. The optimal value for epsilon is found at the point of maximum curvature.

```
from sklearn.neighbors import NearestNeighbors
neigh = NearestNeighbors(n_neighbors=5)
nbrs = neigh.fit(newdf)
distances, indices = nbrs.kneighbors(newdf)
```

```
import numpy as np
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
```



The maximum curvature point can be seen at 0.6, so epsilon will be 0.6.
To find the appropriate min_samples value, we use silhouette score again.

```
from sklearn import metrics
from sklearn.cluster import DBSCAN
for i in range(10):
    clustering = DBSCAN(eps=0.6, min_samples=i).fit(newdf)
    db_labels=clustering.labels_
    print(metrics.silhouette_score(newdf, db_labels, metric='euclidean'))
```

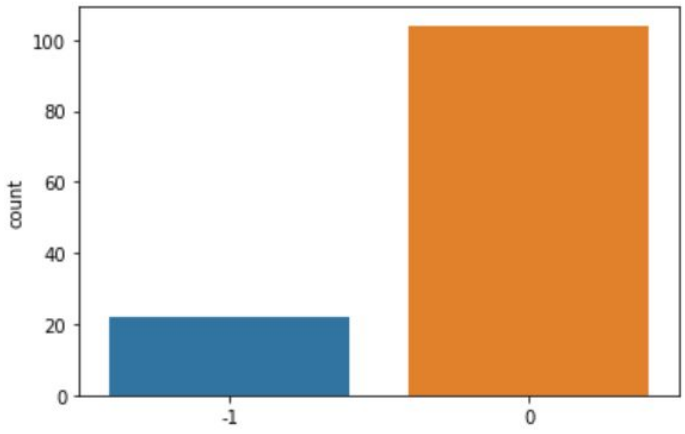
```
-0.17993776383069648
-0.17993776383069648
-0.0543675952136684
0.08028687398352281
0.18777399952922907
0.05860912212289217
0.1399903307848901
0.1391760268926779
0.13305687925081458
-0.007468698384947155
```

Applying DBSCAN for the given dataset with the optimised values for epsilon and min_samples

```
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps=0.6, min_samples=4).fit(newdf)
db_labels=clustering.labels_
```

```
sns.countplot(db_labels)
```

COUNT PLOT ON LABELS OF CLUSTERS



The above countplot indicates that more than 20 profiles have been classified as noise data. All the remaining nodes have been put into one cluster. This indicates that DBSCAN is a bad choice for clustering the dataset.

KNN ON WHOLE DATASET

```
import time

start=time.time()

from sklearn.neighbors import NearestNeighbors
import numpy as np
nbrs = NearestNeighbors(n_neighbors=5).fit(df)
distances, indices = nbrs.kneighbors(df)

end = time.time()

# total time taken
print(f"Runtime of the program is {end - start}")
```

```
Runtime of the program is 0.01296544075012207
```

NearestNeighbors class from sklearn.neighbors provides kneighbors function which can be used to predict 5 nearest nodes for the current node.

KNN ON CLUSTERS SEPARATELY

```
df['cluster_kmeans']=y_kmeans
```

```
set(df['cluster_kmeans'])
```

```
{0, 1, 2, 3}
```

```
cluster=[0]*4
for i in range(4):
    cluster[i]=df[df['cluster_kmeans']==i]
```

```
import time

start=time.time()

from sklearn.neighbors import NearestNeighbors
import numpy as np

distance=[0]*4
indices=[0]*4
for i in range(4):
    nbrs = NearestNeighbors(n_neighbors=5).fit(cluster[i])
    distance[i], indices[i] = nbrs.kneighbors(cluster[i])


end = time.time()

# total time taken
print(f"Runtime of the program is {end - start}")
```

```
Runtime of the program is 0.019947528839111328
```

KNN is performed on both the clusters separately again the same way it was performed on the whole dataset.

RESULTS

For whole dataset -

```
nearest_full['Vishnu']
```

```
['Vishnu', 'Jacob Mathew', 'Najiya', 'Dilshan P', 'Parvathy S']
```

```
nearest_full['Srihitha']
```

```
['Srihitha', 'Jaswanth', 'Seema', 'Sneha', 'Neshva Salim']
```

```
nearest_full['Rishi']
```

```
['Rishi', 'Muralidhar', 'Nitish', 'Divakar N', 'Vikram Chandrasekaran']
```

```
nearest_full['Angela ']
```

```
['Angela ',
 'Chalamalla Praveen Reddy',
 'Nivitha Varghese ',
 'Ananthu KS',
 'Bin laden']
```

Cluster wise results for same nodes -

```
nearest_full['Vishnu']
```

```
['Vishnu', 'Jacob Mathew', 'Najiya', 'Dilshan P', 'Parvathy S']
```

```
nearest_full['Srihitha']
```

```
['Srihitha', 'Jaswanth', 'Seema', 'Sneha', 'Neshva Salim']
```

```
nearest_full['Rishi']
```

```
['Rishi', 'Muralidhar', 'Nitish', 'Divakar N', 'Vikram Chandrasekaran']
```

```
nearest_full['Angela ']
```

```
['Angela ',
 'Chalamalla Praveen Reddy',
 'Nivitha Varghese ',
 'Ananthu KS',
 'Bin laden']
```

DISCUSSION

Time taken for KNN to be performed on whole dataset of appx. 130 rows - 0.013sec
Time taken for KNN on each cluster - 0.005sec on average
Once a node is classified into a cluster, KNN has to be re-applied for only that particular cluster and it takes less than half time for it to find k nearest nodes. But if we simply take the dataset as a whole, for each new node added, we will take an enormous amount of time re-applying KNN.

The downside of clustering the dataset is that - the nodes near the edge might suffer from variation in predictions if the clusters are not far enough.

PERFORMANCE EVALUATION

	silhoutte score	db score	ch score
hierarchical	0.165422	1.10353	23.9761
kmeans	0.285256	1.07656	54.0494
dbscan	0.187774	3.52279	6.43165

EVALUATION METRIC SCORES

DBSCAN, as mentioned earlier, is not an apt choice for clustering anymore.
K-means gives overall better evaluation scores. But they are in general low because of how random the data is.

CONCLUSION

The dataset acquired has been preprocessed and KMeans Clustering has been applied on it. KNN results of cluster-wise data frames and whole dataset have also been explored visually as well as through evaluation metrics.

FUTURE ENHANCEMENTS

The current dataset has only MCQ type responses. By adding columns with descriptive text sentences the users will be able to express themselves more. This might be able to provide more clear results in terms of clustering as well as analysis of the clusters.

REFERENCES

1. Survey of Clustering Data Mining Techniques
2. Online Dating Recommendations: Matching Markets and Learning Preferences
3. https://towardsdatascience.com/hierarchical-clustering-in-python-using-dendrogram-and-cophenetic-correlation-8d41a08f7eab
4. https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc
5. https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf