Assignment 3

Clustering Modal

Human Development Index

Clustering
Using

K-Means
DBSCAN,
Gaussian Mixture Modal

By: Shristi Maskay

Introduction
The Human Development Index (HDI) is an index that measures key dimensions of human development that mainly depends on human life expectancy, literacy rate and standard of living. In this assignment we are basically using the life expectancy and literacy rate of all the districts in Nepal for the year 2013.

Objective:
Using the dataset : Life Expectancy Income of Nepal by District and Literacy rates of different district of Nepal and if needed any other data set figure out the similar districts forming a cluster to be able to determine HDI based on the observations

Models to be used:

K-means

Categorizes data items into clusters using the kMeans algorithm; an unsupervised learning algorithm. KMeans is used to categorize the items into k groups of similarity using euclidean distance as measurement.

Flow:

1. First initialize k points, called means, randomly.
2. Then categorize each item to its closest mean and update the mean's coordinates, which are the averages of the items categorized in that mean so far.
3. Then repeat the process for a given number of iterations and at the end,
4. Then at the end when we get the same centroid in iteration our cluster is formed.

Initialize k means with random values

For a given number of iterations:
  Iterate through items:
    Find the mean closest to the item
    Assign item to mean
    Update mean

To determine the optimal number of clusters we can use

Elbow Method : It uses Distortion (average of square distances from cluster center) and Inertia (sum of squared distances of samples to their closest cluster center) for each value of K in given range

Silhouette Algorithm : Assumes data to be clustered into k clusters by a clustering technique and then for each data point, we define the following:-

I. The cluster assigned to the ith data point
II. The number of data points in the cluster assigned to the ith data point
Iii. It gives a measure of how well assigned the ith data point is to it's cluster
iV. It is defined as the average dissimilarity to the closest cluster which is not it's cluster
V. Determine silhouette coefficient

The average silhouette for each value of k and for the value of k which has the maximum value of *silhouette coefficient* is considered the optimal number of clusters for the unsupervised learning algorithm.

DBSCAN

Clustering analysis basically an Unsupervised learning method that divides the data points into a number of specific groups, such that the data points in the same groups have similar properties and data points in different groups have different properties

Similar to K-Means which is distance between points, DBSCAN is distance between nearest points, i.e **Density-based spatial clustering of applications with noise. T**he ***DBSCAN algorithm*** is based on the concept of "clusters" and "noise". The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

DBSCAN algorithm requires two parameters –

eps : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered as neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and majority of the data points will be in the same clusters. One way to find the eps value is based on the *k-distance graph*.

MinPts: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

DBSCAN Clusters have 3 types of data points.
Core Point: A point is a core point if it has more than MinPts points within eps.
Border Point: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.
Noise or outlier: A point which is not a core point or border point.

Gaussian Mixture Model

A Gaussian mixture model finds a mixture of multi-dimensional Gaussian probability distributions that best model the input dataset.

Pipeline

1. Data exploration and analysis

Life-expectancy-income(set1) and literacy-rate(set2) data sets were loaded and was found that Life expectancy-income dataset consists of:

	District	Life expectancy(In Years)	Per Capita Income(In USD)	Unnamed: 3
58	Achham	67.14	536	NaN
36	Arghakhanchi	68.56	909	NaN
34	Baglung	68.83	868	NaN
32	Baitadi	68.88	573	NaN
66	Bajhang	65.22	487	NaN

Sample Data first 5 row

And literacy-rate dataset consists of:

	District	Total	Female	Male	Year
59	Achham	55.7	42.9	70.7	2013
15	Arghakhanchi	72.6	65.8	81.8	2013
18	Baglung	71.9	65.3	80.6	2013
44	Baitadi	63.0	49.2	79.0	2013
61	Bajhang	55.6	40.1	73.0	2013

Sample Data first 5 row

On generating descriptive statistics for the sets:

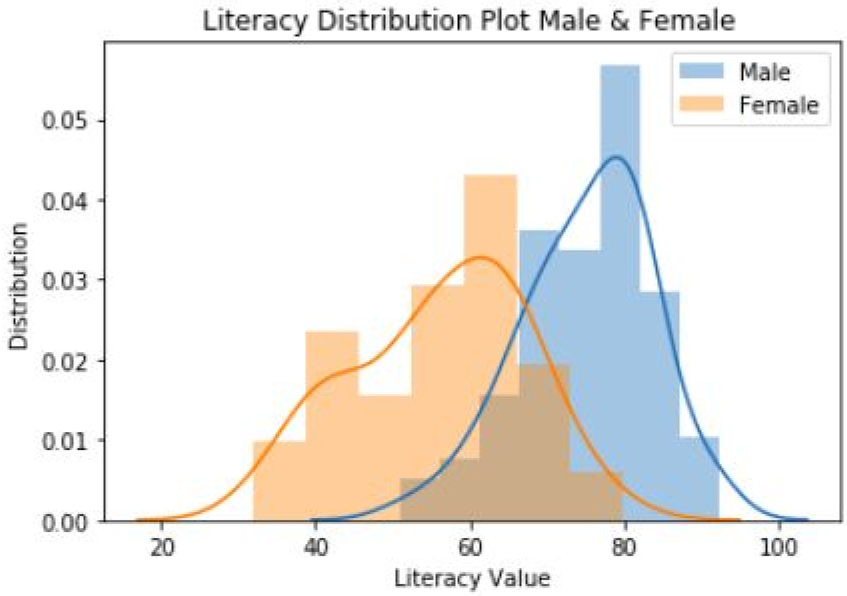
	Life expectancy(In Years)	Unnamed: 3
count	75.000000	0.0
mean	68.405333	NaN
std	2.251472	NaN
min	61.200000	NaN
25%	67.285000	NaN
50%	68.550000	NaN
75%	70.190000	NaN
max	72.900000	NaN

Stats for set1 data

	Total	Female	Male
count	75.0000	75.000000	75.000000
mean	65.1120	56.053333	75.016000
std	9.5331	11.199968	8.464617
min	41.7000	32.000000	50.900000
25%	57.0500	47.600000	69.300000
50%	66.2000	57.100000	76.200000
75%	71.9000	64.350000	80.900000
max	86.3000	79.800000	92.200000

Stats for set2 data

Plot showing Literacy distribution between Male and Female which shows that literacy value of make is higher than that of female



2. Feature Selection and preprocessing:

There are some features in both the datasets which is not required for further processing.

- From Set1 data ‘Unnamed: 3’ is not needed as all the Data in that column has NaN Value so it can be dropped. Similarly from Set2 all the data are of year 2013 so that too can be dropped

We had 2 data set one showing life expectancy and other showing literacy rate so to get teh HDI we gonna merge the two datasheet so certain preprocessing needs to be done

```
exp_income_literacy = pd.merge(expectency_income, literacy_rate, on='District', how='inner')
exp_income_literacy = exp_income_literacy.sort_values(by=['District'])
exp_income_literacy.shape

```

```
(150, 6)
```

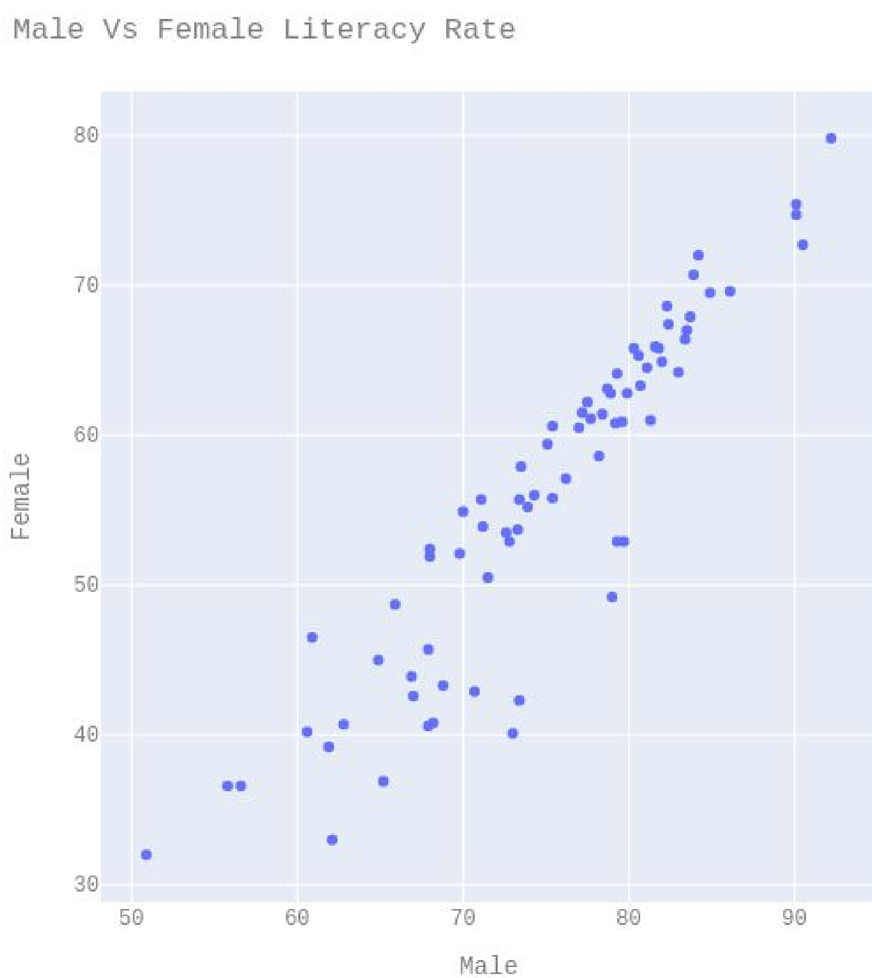
There seems to be something off with the data as we have 75 districts and the merged data is showing 150 districts so further data analysis and preprocessing is required

It is found that the District Name in both the sets do not match with each other as one of the set has space at the end of the name so after preprocessing the data the sets were merged and the final merged table obtained is as below:

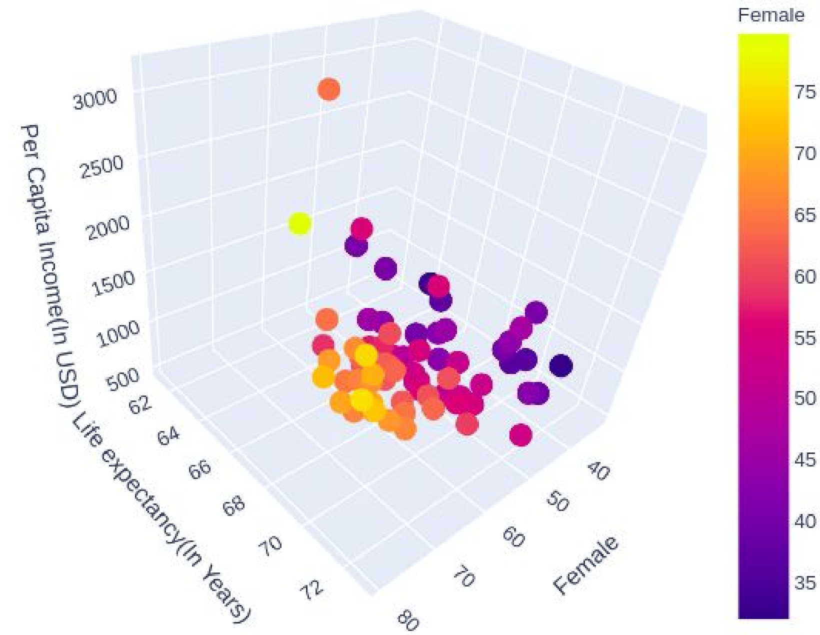
	District	Life expectancy(In Years)	Per Capita Income(In USD)	Total	Female	Male
0	Achham	67.14	536	55.7	42.9	70.7
1	Arghakhanchi	68.56	909	72.6	65.8	81.8
2	Baglung	68.83	868	71.9	65.3	80.6
3	Baitadi	68.88	573	63.0	49.2	79.0
4	Bajhang	65.22	487	55.6	40.1	73.0

3. Data Visualization

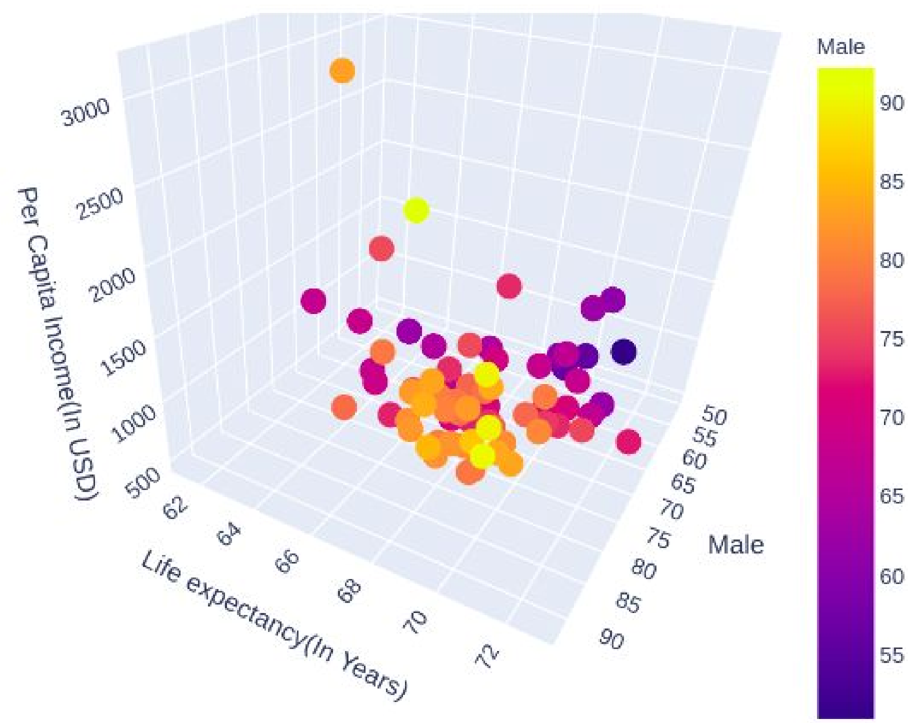
Scatter plot showing Male vs Female for 75 districts which showed Kathmandu to have highest Literacy Rate with Male 92.2 and Female 79.8 where as Rautahat to be least literate with Male 50.9 and Female 32

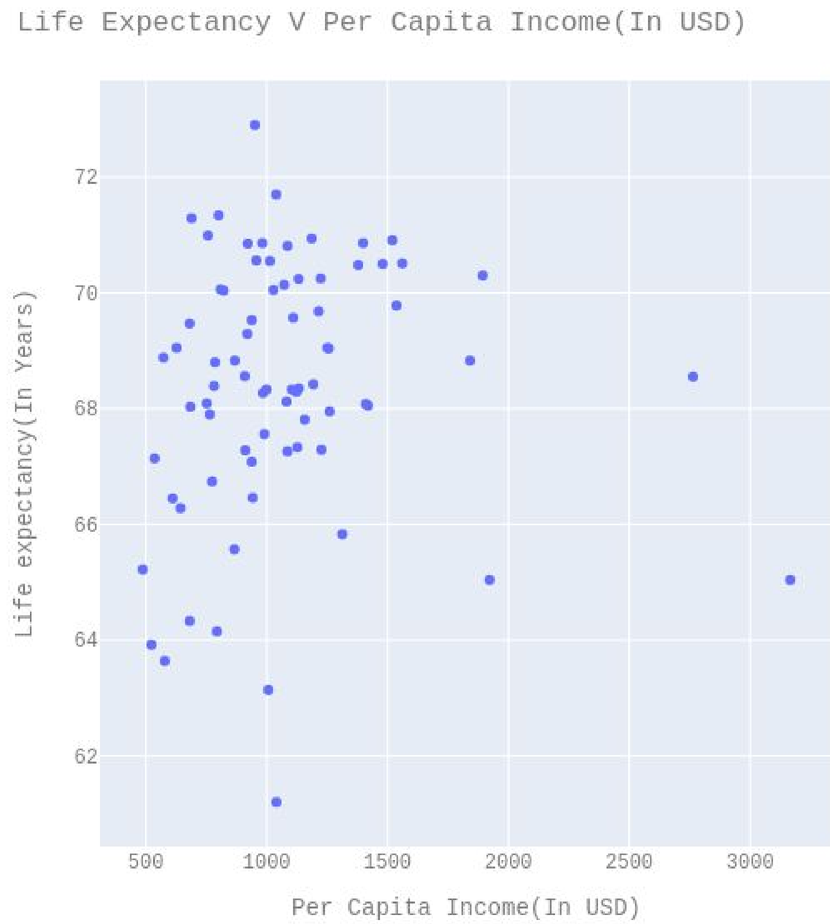


Scatter Plot showing Per capita Income for Female and Life Expectancy in Years



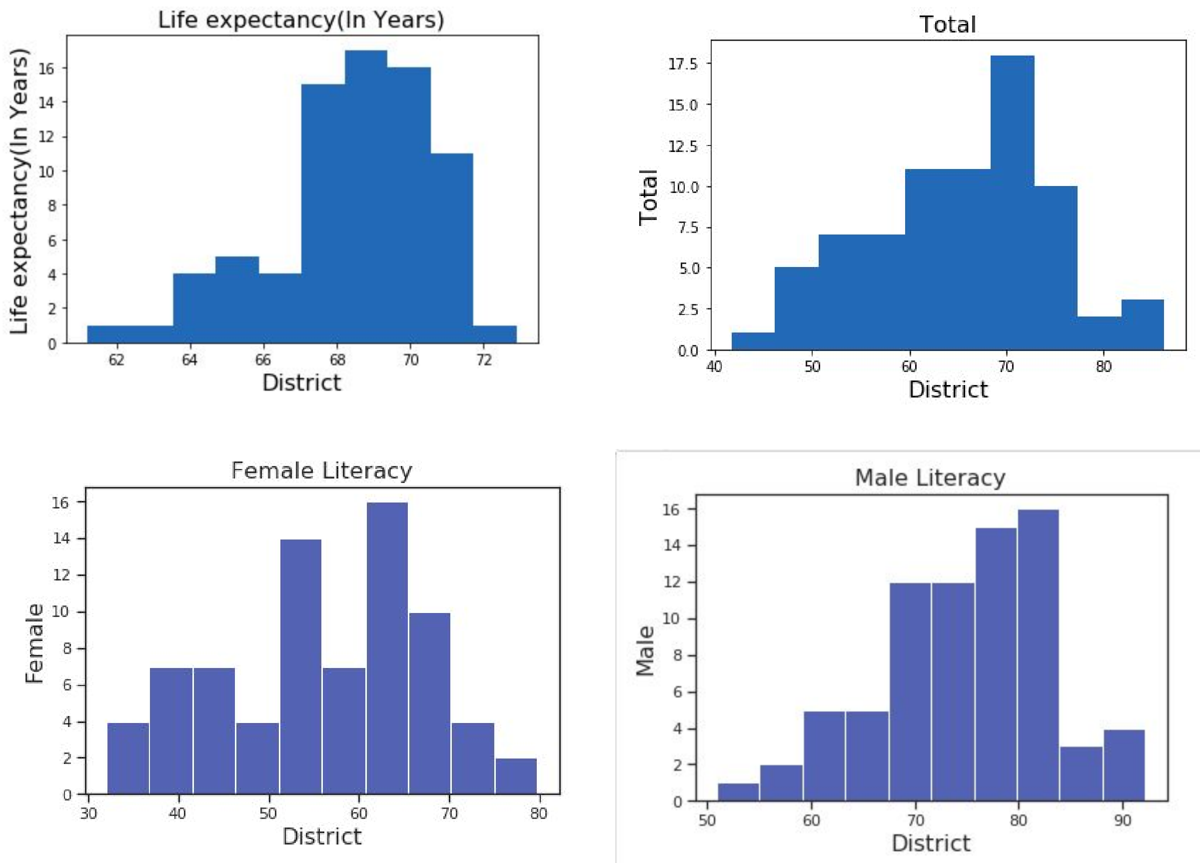
Scatter Plot showing Per capita Income for male and Life Expectancy in Years





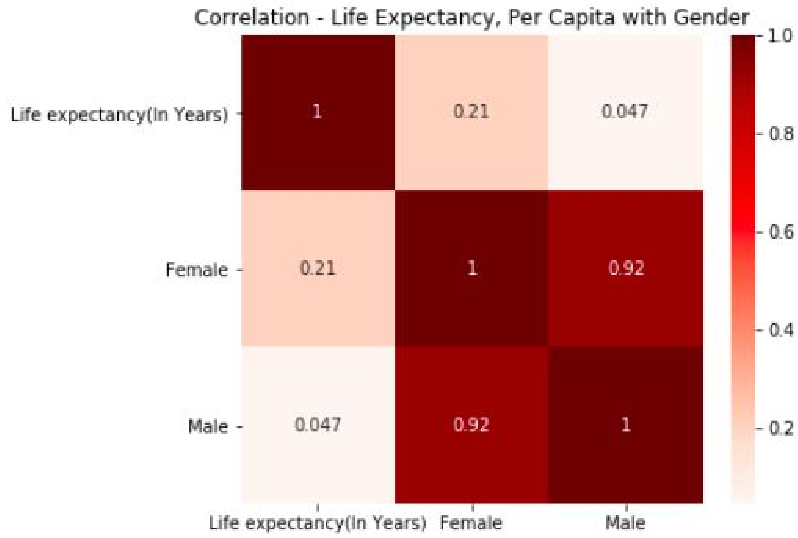
The above plot shows Manag district to have Per capita of 3166 USD with life Expectancy of 65.04yrs
Kathmandu : Per capita of 2764 USD with life Expectancy of 68.55yrs
Mustang : Per capita of 1922 USD with life Expectancy of 65.04 yrs

Rameshap has highest life expectancy of 72.9 with 951 per capita income
Lowest per capita income Bhajyang : Per capita of 487 USD with life Expectancy of 65.22 yrs

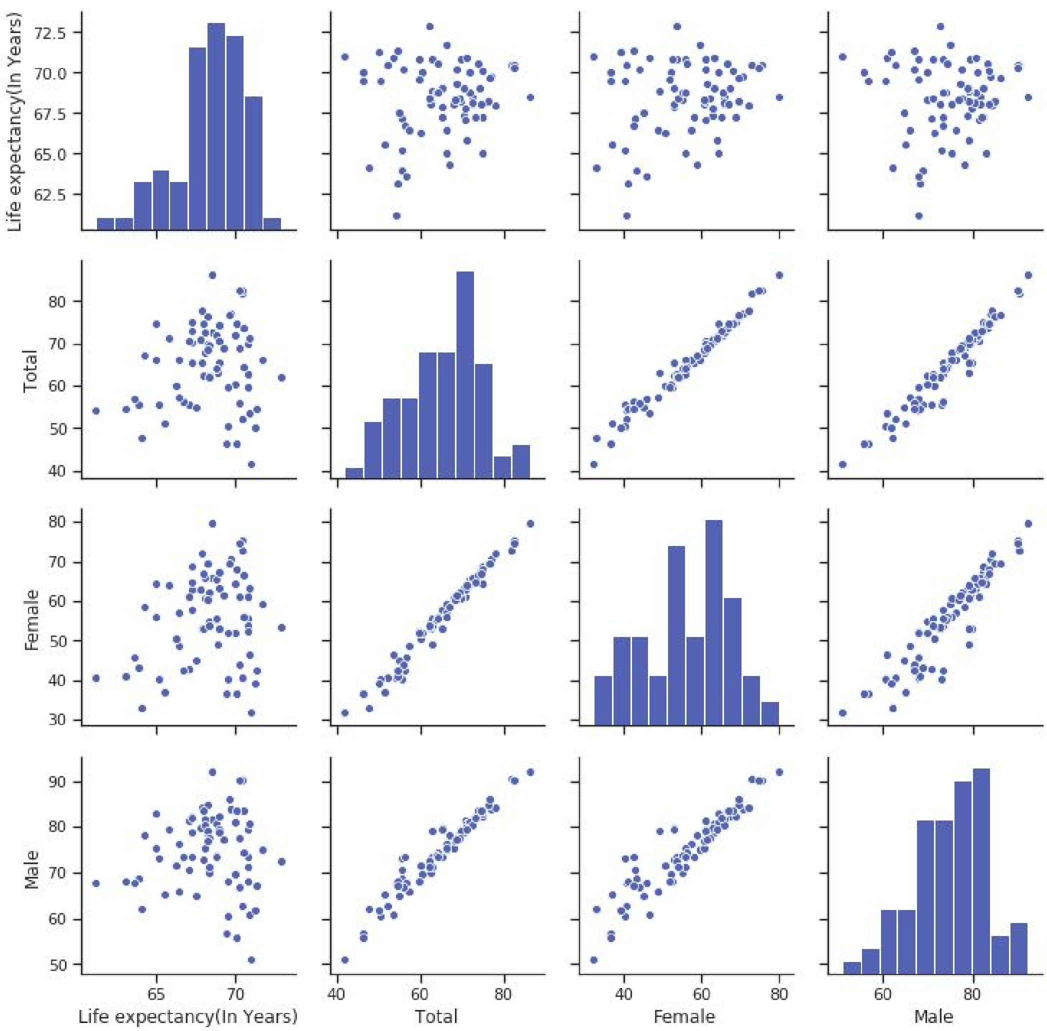


Histogram plot for each attributes based on district

	Life expectancy(In Years)	Per Capita Income(In USD)	Total	Female	Male
Life expectancy(In Years)	1.000000	0.070954	0.134884	0.209683	0.046707
Per Capita Income(In USD)	0.070954	1.000000	0.506040	0.498649	0.419661
Total	0.134884	0.506040	1.000000	0.986817	0.971105
Female	0.209683	0.498649	0.986817	1.000000	0.924431
Male	0.046707	0.419661	0.971105	0.924431	1.000000



Determining the correlation and visualizing it using pairplot



4. Standardization

Now Need to normalize the data to scale the data to 0 and 1 range
Library used: f
rom sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import StandardScaler

```
array([[-0.56578717, -1.23572093, -0.99394547, -1.1823167 , -0.51332084],
       [ 0.06915839, -0.38816428,  0.79076324,  0.87610087,  0.80685092],
       [ 0.18988747, -0.48132734,  0.7168404 ,  0.83115725,  0.66412965],
       [ 0.21224471, -1.15164695, -0.22303579, -0.61602715,  0.47383462],
       [-1.42430511, -1.34706215, -1.00450587, -1.43400094, -0.23977174]])
```

Above sample scaled data shows that the data has been normalized

5.1. K-Means | Approach taken and findings

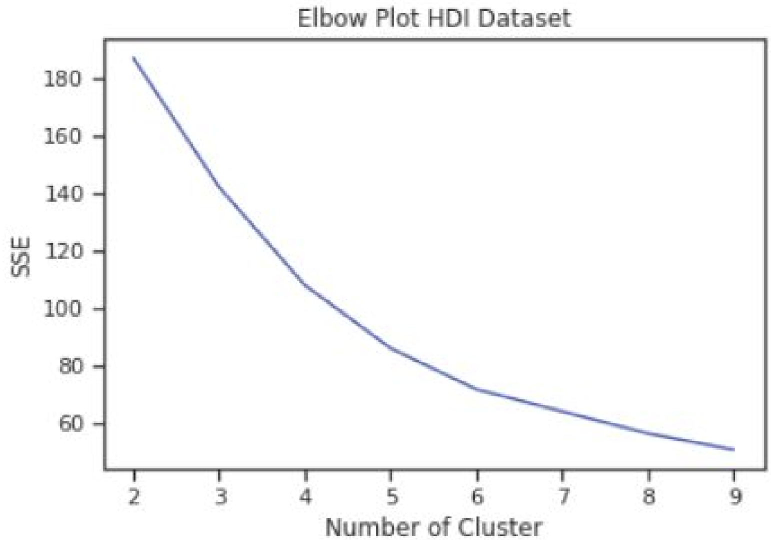
Library used:

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.neighbors import NearestNeighbors

To determine the optimal cluster:

```
Sum_of_squared_distances=[]
Silhouette=[]
k_values=[]
for k in range(2,10):
    kmeans = KMeans(n_clusters=k, random_state=1)
    kmeans.fit(LitInc_scaled_data)
    k_values.append(k)
    Sum_of_squared_distances.append(kmeans.inertia_)
    b = silhouette_score(LitInc_scaled_data, kmeans.labels_)
    Silhouette.append(b)
    print("cluster: ", k,"Inertia: ",kmeans.inertia_,
          'Silhouette:', silhouette_score(LitInc_scaled_data, kmeans.labels_))
```

```
cluster:  2 Inertia:  186.81253280380463 Silhouette: 0.33335065697828753
cluster:  3 Inertia:  141.99332751118027 Silhouette: 0.3529050467966264
cluster:  4 Inertia:  107.94490604958527 Silhouette: 0.37343379977758046
cluster:  5 Inertia:  86.02249178273615 Silhouette: 0.31140721221020023
cluster:  6 Inertia:  71.77133611443935 Silhouette: 0.3031290179482093
cluster:  7 Inertia:  64.06760158780207 Silhouette: 0.2928047292857756
cluster:  8 Inertia:  56.44832718440356 Silhouette: 0.2839187446989456
cluster:  9 Inertia:  50.78539911813376 Silhouette: 0.29174722909194034
```



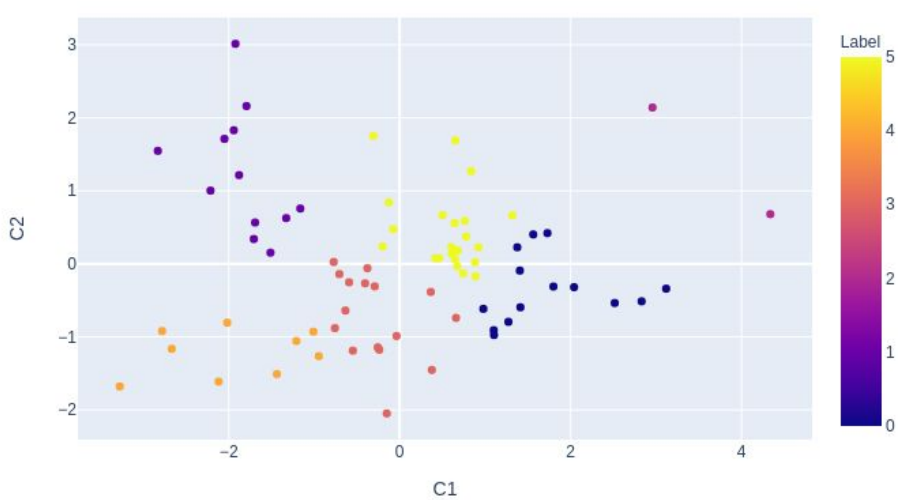
Elbow Method

From the above plot we can state that no of cluster will be 6

K-means prediction

```
y_kmeans = kmeans.predict(LitInc_scaled_data)
y_kmeans
```

```
array([1, 5, 5, 3, 1, 1, 3, 4, 5, 0, 5, 0, 5, 3, 5, 3, 3, 0, 4, 3, 1,
1,
       3, 5, 1, 0, 1, 5, 1, 5, 1, 5, 1, 0, 2, 0, 3, 0, 0, 4, 5, 2, 5,
1,
       5, 0, 5, 3, 3, 0, 5, 0, 4, 5, 3, 4, 4, 1, 3, 5, 3, 5, 4, 4, 3,
3,
       4, 5, 5, 5, 0, 0, 5, 0, 3], dtype=int32)
```



Model 1: Simple KMeans

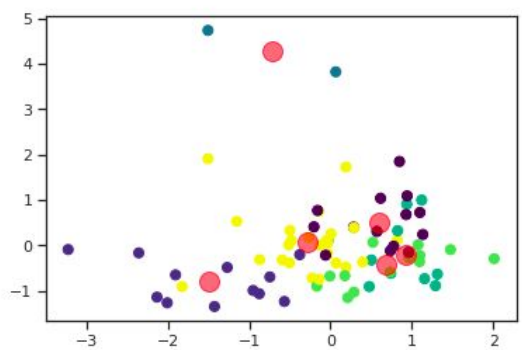
```
kmeans=KMeans(n_clusters=6, random_state=1)
kmeans.fit(LitInc_scaled_data)
kmeans_df_clustered = exp_income_literacy.copy()
kmeans_df_clustered['KMeans Label'] = kmeans.labels_
```

```
pd.set_option('max_rows', 100)
kmeans_df_clustered.sort_values(by='KMeans Label')
```

```
y_kmeans = kmeans.predict(LitInc_scaled_data)
y_kmeans
```

```
array([1, 5, 5, 4, 1, 1, 4, 3, 5, 0, 5, 0, 5, 4, 5, 4, 4, 0, 3, 4, 1, 1,
       4, 5, 1, 0, 1, 5, 1, 5, 1, 5, 1, 0, 2, 0, 5, 0, 0, 3, 5, 2, 5, 1,
       5, 0, 5, 4, 4, 0, 5, 0, 3, 5, 4, 3, 3, 1, 4, 5, 4, 5, 3, 3, 4, 4,
       3, 5, 5, 5, 0, 0, 5, 0, 5], dtype=int32)
```

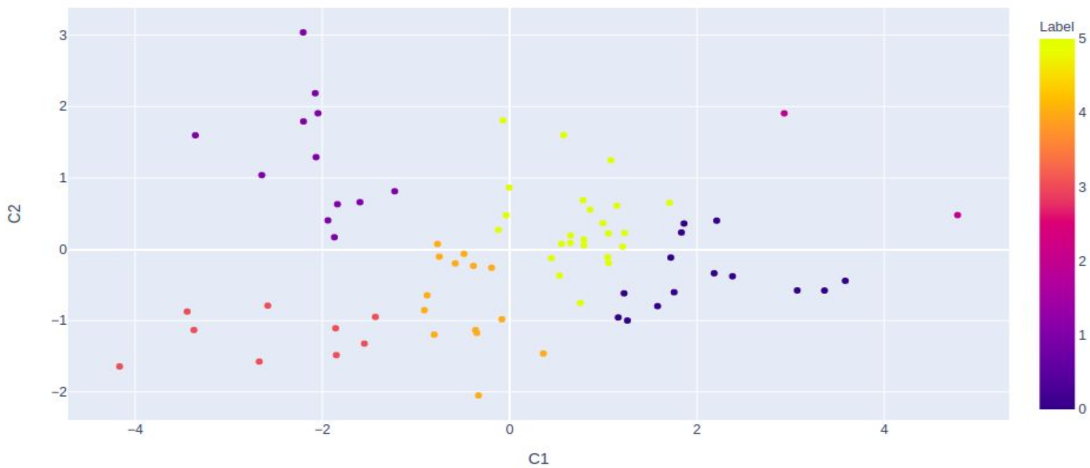
```
plt.scatter(LitInc_scaled_data[:, 0], LitInc_scaled_data[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.5);

```



Simple scatter plot for the data with Centroid

Using PCA to plot the Kmeans plot

```python
pca = PCA(n_components=2, random_state = 1)
pca = pca.fit(LitInc_scaled_data)
scaled_transformed = pca.transform(LitInc_scaled_data)
plot_data = pd.DataFrame(scaled_transformed, columns=['C1', 'C2'])
plot_data['Label'] = kmeans.labels_
plot_data['District'] = exp_income_literacy['District'].tolist()
fig = px.scatter(plot_data, x="C1", y="C2", color='Label', hover_data=['District'])
fig.show()
```



Model 2: KMeans after outlier detection

```python
kmeans_outlier = KMeans(n_clusters=1)
kmeans_outlier.fit(LitInc_scaled_data)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=1, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```python
# identify the 5 closest points
distances = kmeans_outlier.transform(LitInc_scaled_data)
```

```python
# argsort returns an array of indexes which will sort the array
# in ascending order. Reverse it with [::-1]
sorted_idx = np.argsort(distances.ravel())[::-1][:5]
```

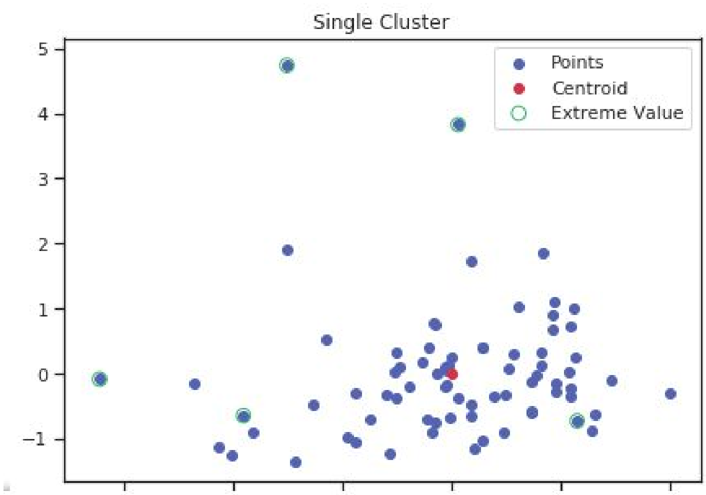
```python
print(LitInc_scaled_data[sorted_idx][:, 0])
print(LitInc_scaled_data[sorted_idx][:, 1])
```

```
[ 0.06468694 -1.50479117  1.15572016 -3.22182705 -1.90275    ]
[ 3.82689626  4.74034873 -0.7335493  -0.09049693 -0.64947531]
```

FOund that 5 point can be taken as an outlier

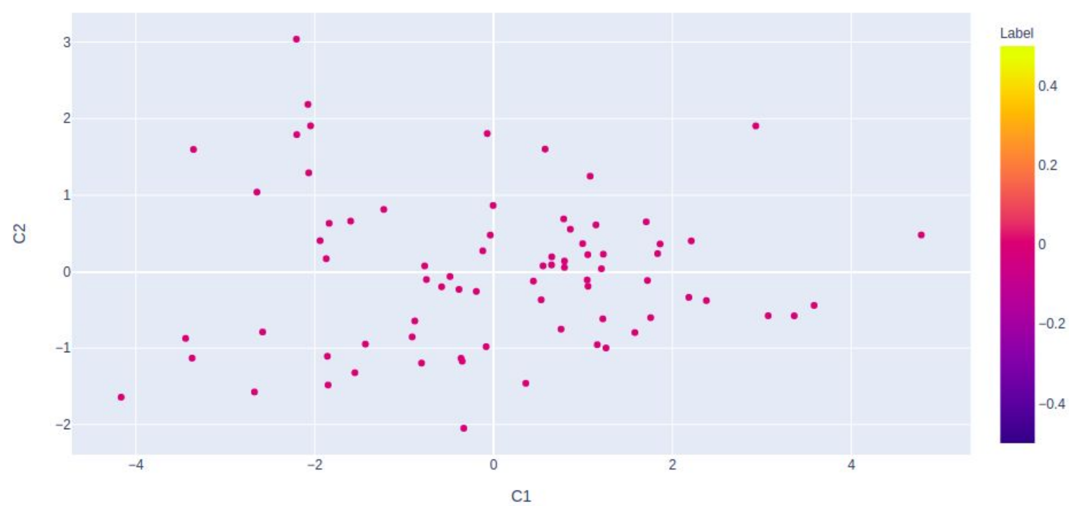
```python
f, ax = plt.subplots(figsize=(7,5))
ax.set_title('Single Cluster')
ax.scatter(LitInc_scaled_data[:, 0], LitInc_scaled_data[:, 1], label='Points')
ax.scatter(kmeans_outlier.cluster_centers_[:, 0],
           kmeans_outlier.cluster_centers_[:, 1],
           label='Centroid', color='r')
ax.scatter(LitInc_scaled_data[sorted_idx][:, 0],
           LitInc_scaled_data[sorted_idx][:, 1],
           label='Extreme Value', edgecolors='g',
           facecolors='none', s=75)
ax.legend(loc='best')
```

```
<matplotlib.legend.Legend at 0x7f890870ed68>
```



From the above plot got that there are 5 outliers denoted by Extreme Values and Point in red is centroid

```python
plot_data = pd.DataFrame(scaled_transformed, columns=['C1', 'C2'])
plot_data['Label'] = kmeans_outlier.labels_
plot_data['District'] = exp_income_literacy['District'].tolist()
fig = px.scatter(plot_data, x="C1", y="C2", color='Label', hover_data=['District'])
fig.show()
```



Model 3: Kmeans using PCA fit

```
n_samples, n_features = LitInc_scaled_data.shape
n_digits = len(np.unique(y_kmeans))
labels = y_kmeans
n_noise_k_means = kmeans.labels_
sample_size = 75
```

```
pca = PCA(n_components=2, random_state = 1)
pca = pca.fit(LitInc_scaled_data)
scaled_transformed = pca.transform(LitInc_scaled_data)
n_digits = 4
# labels = outlier_y_kmeans

```

```
# pca = PCA(n_components= 4).fit(LitInc_scaled_data)
print('----------PCA-ANALYSIS----------')
print("Noise:", n_noise_k_means)
evaluate_kmeans(KMeans(init='k-means++', n_clusters=n_digits, n_init=10),
               name="PCA-based", data=scaled_transformed)
```

On Evaluation of the KMeans model:

Model 1:

noise: 0
Model Evaluation: k-means++
****
Estimated number of clusters: 6
Homogeneity: 0.847
Completeness: 0.846
V-measure: 0.846
Adjusted Rand Index: 0.814
Adjusted Mutual Information: 0.827
silhouette_score: 0.29703249610999305

Model 2:

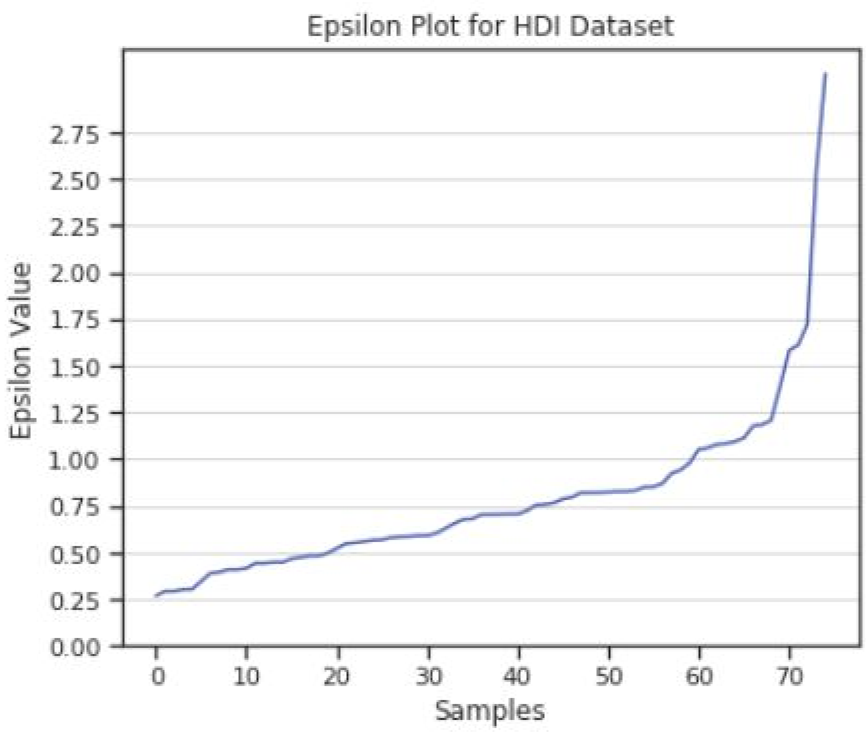
noise: 5
Model Evaluation: k-means with Outlier detection
****
Estimated number of clusters: 6
Homogeneity: 0.737
Completeness: 0.708
V-measure: 0.722
Adjusted Rand Index: 0.568
Adjusted Mutual Information: 0.682
silhouette_score: 0.29766792857176755

Model 3:

----------PCA-ANALYSIS----------
Model Evaluation: PCA-based
****
Estimated number of clusters: 4
Homogeneity: 0.564
Completeness: 0.761
V-measure: 0.648
Adjusted Rand Index: 0.501
Adjusted Mutual Information: 0.618
silhouette_score: 0.4236209710541152

4. DBSCAN | Approach taken and findings

Epsilon Plot for HDI dataset



Using eps=0.8 and min_sample =3

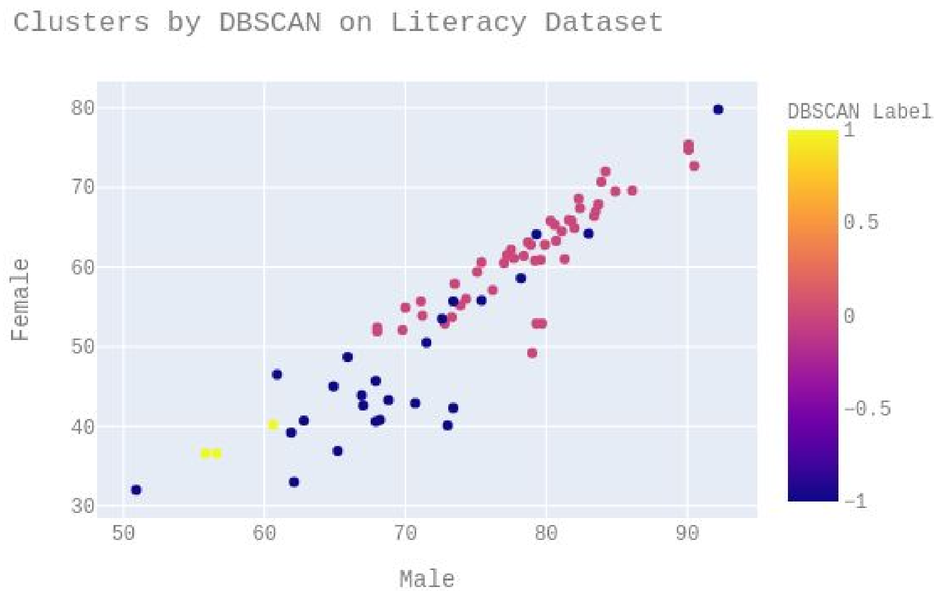
```
min_samples = 3
eps = 0.8

# dbscan = DBSCAN(eps=2., min_samples=8)
dbscan = DBSCAN(eps=0.8, min_samples=3)

y_dbscan = dbscan.fit_predict(LitInc_scaled_data)
exp_income_literacy['DBSCAN Label'] = dbscan.labels_
```

```
str(exp_income_literacy['DBSCAN Label'].tolist())

```

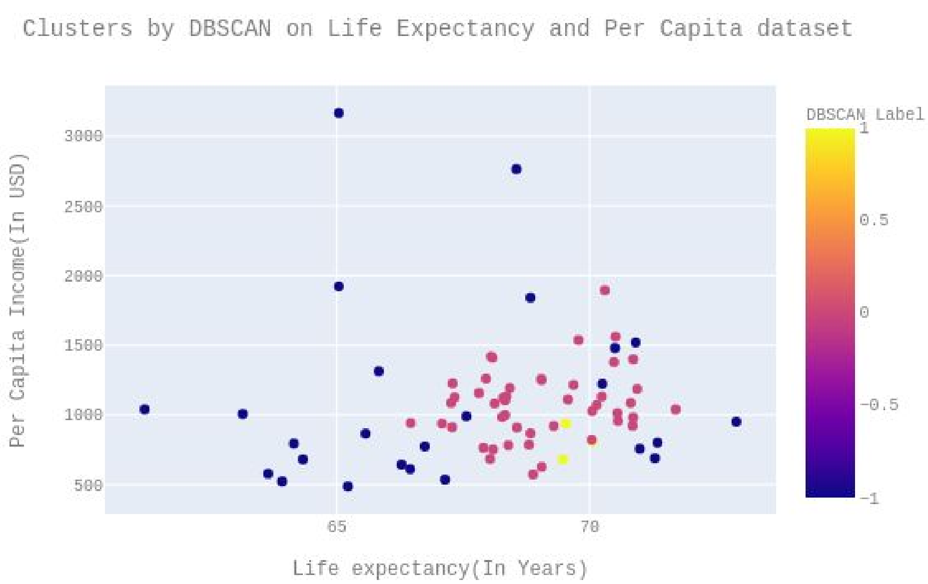
```
'[-1, 0, 0, 0, -1, -1, 0, -1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, -1, -1, 0, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1,
1, -1, 0, 0, 1, 0, -1, 0, -1, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, 0, 0, -
1, -1, 0, 0, 0, 0, -1, 0, 0]'
```

No of clusters = 2 obtained

Cluster of literacy rate based on Male and Female



Cluster of literacy rate based on Life Expectancy and Per capita



Model 4: DBSCAN

Estimated number of clusters: 2
Estimated number of noise points: 25
Homogeneity: 1.000
Completeness: 1.000
V-measure: 1.000
Adjusted Rand Index: 1.000

Adjusted Mutual Information: 1.000
Silhouette Coefficient: -0.165

5. GMM | Approach taken and findings

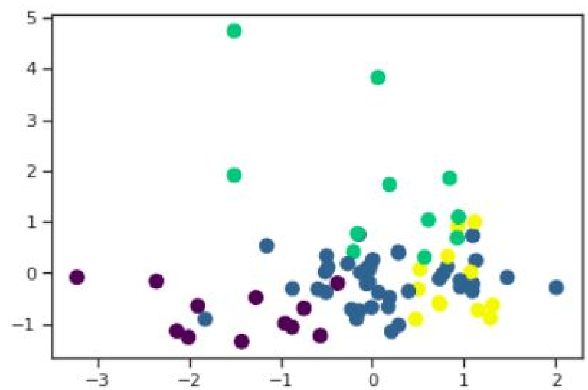
Library: from sklearn import mixture

Modal

```
mix_data = mixture.GaussianMixture(n_components=4, covariance_type='diag')
mix_data.fit(X)
```

```
GaussianMixture(covariance_type='diag', init_params='kmeans', max_iter=100,
                means_init=None, n_components=4, n_init=1, precisions_init=None,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose=0,
                verbose_interval=10, warm_start=False, weights_init=None)
```

No of cluster = 4



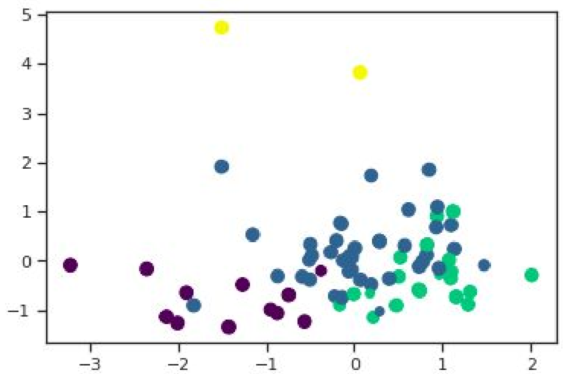
As Gaussian Mixture Models has a probabilistic model so finding a probabilistic cluster assignment.

```
probs = mix_data.predict_proba(X)

print(probs[:5].round(3))
```

```
[[0.989 0.    0.011 0.    ]
 [0.    1.    0.    0.    ]
 [0.    1.    0.    0.    ]
 [0.    0.097 0.903 0.    ]
 [1.    0.    0.    0.    ]]
```

Visualizing the probabilistiv cluster

```
size = 75 * probs.max(1) ** 2  # square emphasizes differences
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=size);
```



Comparison

	Model 1	Model 2	Model 3	Model 4
Estimated number of clusters:	**6**	**6**	**4**	**2**
Estimated number of noise points:	**0**	**5**		**25**
Homogeneity Score	**0.847**	**0.737**	**0.564**	**1.000**
Completeness	**0.846**	**0.708**	**0.761**	**1.000**
V-measure:	**0.846**	**0.722**	**0.648**	**1.000**
Adjusted Rand Index:	**0.814**	**0.568**	**0.501**	**1.000**
Adjusted Mutual Information:	**0.827**	**0.682**	**0.618**	**1.000**
Silhouette Coefficient:	**0.297**	**0.29766**	**0.4236**	**-0.165**

URL

https://github.com/Kristiee/HDI-clustering/blob/master/cluster-final-Copy1.ipynb

PCA Task:
https://github.com/Kristiee/HDI-clustering/blob/master/PCA.ipynb

Submitted by:
Shristi Maskay