|  |  |
| --- | --- |
| **EX.NO 10 DATA & TEXT CLUSTERING USING GAUSSIAN MIXTURE MODEL** | |
| IN [1] | **import** os  **for** dirname, \_, filenames **in** os**.**walk('/kaggle/input'):  **for** filename **in** filenames:  print(os**.**path**.**join(dirname, filename)) |
| IN [2] | **import** numpy **as** np  **import** pandas **as** pd  **import** matplotlib.pyplot **as** plt  **from** pandas **import** DataFrame  **from** sklearn.preprocessing **import** StandardScaler, normalize  **from** sklearn.decomposition **import** PCA  **from** sklearn.mixture **import** GaussianMixture  **from** sklearn.metrics **import** silhouette\_score  **from** sklearn.model\_selection **import** train\_test\_split  **from** sklearn **import** metrics |
| IN [3] | raw\_df **=** pd**.**read\_csv('../input/ccdata/CC GENERAL.csv')  raw\_df **=** raw\_df**.**drop('CUST\_ID', axis **=** 1)  raw\_df**.**fillna(method **=**'ffill', inplace **=** **True**)  raw\_df**.**head(2) |
| IN [4] | *# Standardize data*  scaler **=** StandardScaler()  scaled\_df **=** scaler**.**fit\_transform(raw\_df)    *# Normalizing the Data*  normalized\_df **=** normalize(scaled\_df)    *# Converting the numpy array into a pandas DataFrame*  normalized\_df **=** pd**.**DataFrame(normalized\_df)    *# Reducing the dimensions of the data*  pca **=** PCA(n\_components **=** 2)  X\_principal **=** pca**.**fit\_transform(normalized\_df)  X\_principal **=** pd**.**DataFrame(X\_principal)  X\_principal**.**columns **=** ['P1', 'P2']    X\_principal**.**head(2) |
| IN [5] | gmm **=** GaussianMixture(n\_components **=** 3)  gmm**.**fit(X\_principal) |
| IN [6] | *# Visualizing the clustering*  plt**.**scatter(X\_principal['P1'], X\_principal['P2'],  c **=** GaussianMixture(n\_components **=** 3)**.**fit\_predict(X\_principal), cmap **=**plt**.**cm**.**winter, alpha **=** 0.6)  plt**.**show() |
| IN [7] | **def** SelBest(arr:list, X:int)**->**list:  '''  returns the set of X configurations with shorter distance  '''  dx**=**np**.**argsort(arr)[:X]  **return** arr[dx] |
| IN [8] | n\_clusters**=**np**.**arange(2, 8)  sils**=**[]  sils\_err**=**[]  iterations**=**20  **for** n **in** n\_clusters:  tmp\_sil**=**[]  **for** \_ **in** range(iterations):  gmm**=**GaussianMixture(n, n\_init**=**2)**.**fit(X\_principal)  labels**=**gmm**.**predict(X\_principal)  sil**=**metrics**.**silhouette\_score(X\_principal, labels, metric**=**'euclidean')  tmp\_sil**.**append(sil)  val**=**np**.**mean(SelBest(np**.**array(tmp\_sil), int(iterations**/**5)))  err**=**np**.**std(tmp\_sil)  sils**.**append(val)  sils\_err**.**append(err) |
| IN [9] | plt**.**errorbar(n\_clusters, sils, yerr**=**sils\_err)  plt**.**title("Silhouette Scores", fontsize**=**20)  plt**.**xticks(n\_clusters)  plt**.**xlabel("N. of clusters")  plt**.**ylabel("Score") |
| IN [10] | **def** gmm\_js(gmm\_p, gmm\_q, n\_samples**=**10**\*\***5):  X **=** gmm\_p**.**sample(n\_samples)[0]  log\_p\_X **=** gmm\_p**.**score\_samples(X)  log\_q\_X **=** gmm\_q**.**score\_samples(X)  log\_mix\_X **=** np**.**logaddexp(log\_p\_X, log\_q\_X)  Y **=** gmm\_q**.**sample(n\_samples)[0]  log\_p\_Y **=** gmm\_p**.**score\_samples(Y)  log\_q\_Y **=** gmm\_q**.**score\_samples(Y)  log\_mix\_Y **=** np**.**logaddexp(log\_p\_Y, log\_q\_Y)  **return** np**.**sqrt((log\_p\_X**.**mean() **-** (log\_mix\_X**.**mean() **-** np**.**log(2))  **+** log\_q\_Y**.**mean() **-** (log\_mix\_Y**.**mean() **-** np**.**log(2))) **/** 2)  ` |
| IN [11] | n\_clusters**=**np**.**arange(2, 8)  iterations**=**20  results**=**[]  res\_sigs**=**[]  **for** n **in** n\_clusters:  dist**=**[]    **for** iteration **in** range(iterations):  train, test**=**train\_test\_split(X\_principal, test\_size**=**0.5)    gmm\_train**=**GaussianMixture(n, n\_init**=**2)**.**fit(train)  gmm\_test**=**GaussianMixture(n, n\_init**=**2)**.**fit(test)  dist**.**append(gmm\_js(gmm\_train, gmm\_test))  selec**=**SelBest(np**.**array(dist), int(iterations**/**5))  result**=**np**.**mean(selec)  res\_sig**=**np**.**std(selec)  results**.**append(result)  res\_sigs**.**append(res\_sig) |
| IN [12] | plt**.**errorbar(n\_clusters, results, yerr**=**res\_sigs)  plt**.**title("Distance between Train and Test GMMs", fontsize**=**20)  plt**.**xticks(n\_clusters)  plt**.**xlabel("N. of clusters")  plt**.**ylabel("Distance")  plt**.**show() |
| IN [13] | n\_clusters**=**np**.**arange(2, 8)  bics**=**[]  bics\_err**=**[]  iterations**=**20  **for** n **in** n\_clusters:  tmp\_bic**=**[]  **for** \_ **in** range(iterations):  gmm**=**GaussianMixture(n, n\_init**=**2)**.**fit(X\_principal)    tmp\_bic**.**append(gmm**.**bic(X\_principal))  val**=**np**.**mean(SelBest(np**.**array(tmp\_bic), int(iterations**/**5)))  err**=**np**.**std(tmp\_bic)  bics**.**append(val)  bics\_err**.**append(err) |
| IN [14] | plt**.**errorbar(n\_clusters,bics, yerr**=**bics\_err, label**=**'BIC')  plt**.**title("BIC Scores", fontsize**=**20)  plt**.**xticks(n\_clusters)  plt**.**xlabel("N. of clusters")  plt**.**ylabel("Score")  plt**.**legend() |
| IN [15] | plt**.**errorbar(n\_clusters, np**.**gradient(bics), yerr**=**bics\_err, label**=**'BIC')  plt**.**title("Gradient of BIC Scores", fontsize**=**20)  plt**.**xticks(n\_clusters)  plt**.**xlabel("N. of clusters")  plt**.**ylabel("grad(BIC)")  plt**.**legend() |

**OUTPUT:**

|  |  |
| --- | --- |
| OUT [3] |  |
| OUT [4] |  |
| OUT [5] |  |
| OUT [6] |  |
| OUT [9] |  |
| OUT [12] |  |
| OUT [14] |  |
| OUT [15] |  |