**<https://www.geeksforgeeks.org/ml-spectral-clustering/>**

**Step 1: Importing the required libraries**

|  |
| --- |
| **import** pandas as pd  **import** matplotlib.pyplot as plt  **from** sklearn.cluster **import** SpectralClustering  **from** sklearn.preprocessing **import** StandardScaler, normalize  **from** sklearn.decomposition **import** PCA  **from** sklearn.metrics **import** silhouette\_score |

**Step 2: Loading and Cleaning the Data**

|  |
| --- |
| # Changing the working location to the location of the data  cd "C:\Users\Dev\Desktop\Kaggle\Credit\_Card"    # Loading the data  X **=** pd.read\_csv('CC\_GENERAL.csv')    # Dropping the CUST\_ID column from the data  X **=** X.drop('CUST\_ID', axis **=** 1)    # Handling the missing values if any  X.fillna(method **=**'ffill', inplace **=** True)    X.head() |

Table

Description automatically generated

**Step 3: Preprocessing the data to make the data visualizable**

|  |
| --- |
| # Preprocessing the data to make it visualizable    # Scaling the Data  scaler **=** StandardScaler()  X\_scaled **=** scaler.fit\_transform(X)    # Normalizing the Data  X\_normalized **=** normalize(X\_scaled)    # Converting the numpy array into a pandas DataFrame  X\_normalized **=** pd.DataFrame(X\_normalized)    # Reducing the dimensions of the data  pca **=** PCA(n\_components **=** 2)  X\_principal **=** pca.fit\_transform(X\_normalized)  X\_principal **=** pd.DataFrame(X\_principal)  X\_principal.columns **=** ['P1', 'P2']    X\_principal.head() |

Table

Description automatically generated

**Step 4: Building the Clustering models and Visualizing the clustering**

In the below steps, two different Spectral Clustering models with different values for the parameter ‘affinity’. You can read about the documentation of the Spectral Clustering class [here](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html).

a) **affinity = ‘rbf’**

|  |
| --- |
| # Building the clustering model  spectral\_model\_rbf **=** SpectralClustering(n\_clusters **=** 2, affinity **=**'rbf')    # Training the model and Storing the predicted cluster labels  labels\_rbf **=** spectral\_model\_rbf.fit\_predict(X\_principal) |
| # Building the label to colour mapping  colours **=** {}  colours[0] **=** 'b'  colours[1] **=** 'y'    # Building the colour vector for each data point  cvec **=** [colours[label] **for** label **in** labels\_rbf]    # Plotting the clustered scatter plot    b **=** plt.scatter(X\_principal['P1'], X\_principal['P2'], color **=**'b');  y **=** plt.scatter(X\_principal['P1'], X\_principal['P2'], color **=**'y');    plt.figure(figsize **=**(9, 9))  plt.scatter(X\_principal['P1'], X\_principal['P2'], c **=** cvec)  plt.legend((b, y), ('Label 0', 'Label 1'))  plt.show() |

Chart, scatter chart

Description automatically generated

b) **affinity = ‘nearest\_neighbors’**

|  |
| --- |
| # Building the clustering model  spectral\_model\_nn **=** SpectralClustering(n\_clusters **=** 2, affinity **=**'nearest\_neighbors')    # Training the model and Storing the predicted cluster labels  labels\_nn **=** spectral\_model\_nn.fit\_predict(X\_principal) |

Chart, scatter chart

Description automatically generated

**Step 5: Evaluating the performances**

|  |
| --- |
| # List of different values of affinity  affinity **=** ['rbf', 'nearest-neighbours']    # List of Silhouette Scores  s\_scores **=** []    # Evaluating the performance  s\_scores.append(silhouette\_score(X, labels\_rbf))  s\_scores.append(silhouette\_score(X, labels\_nn))    print(s\_scores) |



**Step 6: Comparing the performances**

|  |
| --- |
| # Plotting a Bar Graph to compare the models  plt.bar(affinity, s\_scores)  plt.xlabel('Affinity')  plt.ylabel('Silhouette Score')  plt.title('Comparison of different Clustering Models')  plt.show() |

Chart, bar chart

Description automatically generated