Sec_1_Homework_9

March 6, 2024

1 0.) Import and Clean data

[41]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')
```

[42]:
```
df = pd.read_csv("Country-data.csv", sep = ",")
```

[43]:
```
names = df[["country"]].copy()
X = df.drop("country",axis =1 )
```

[44]:
```
scaler = StandardScaler().fit(X)
X_scaled = scaler.transform(X)
```

2 1.) Fit a kmeans Model with any Number of Clusters

[45]:
```
kmeans= KMeans(n_clusters = 5).fit(X_scaled)
```

3 2.) Pick two features to visualize across

[46]:
```
X.columns
```

[46]:
```
Index(['child_mort', 'exports', 'health', 'imports', 'income', 'inflation',
       'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

[47]:
```
import matplotlib.pyplot as plt

x1_index = 0
x2_index = 1
```

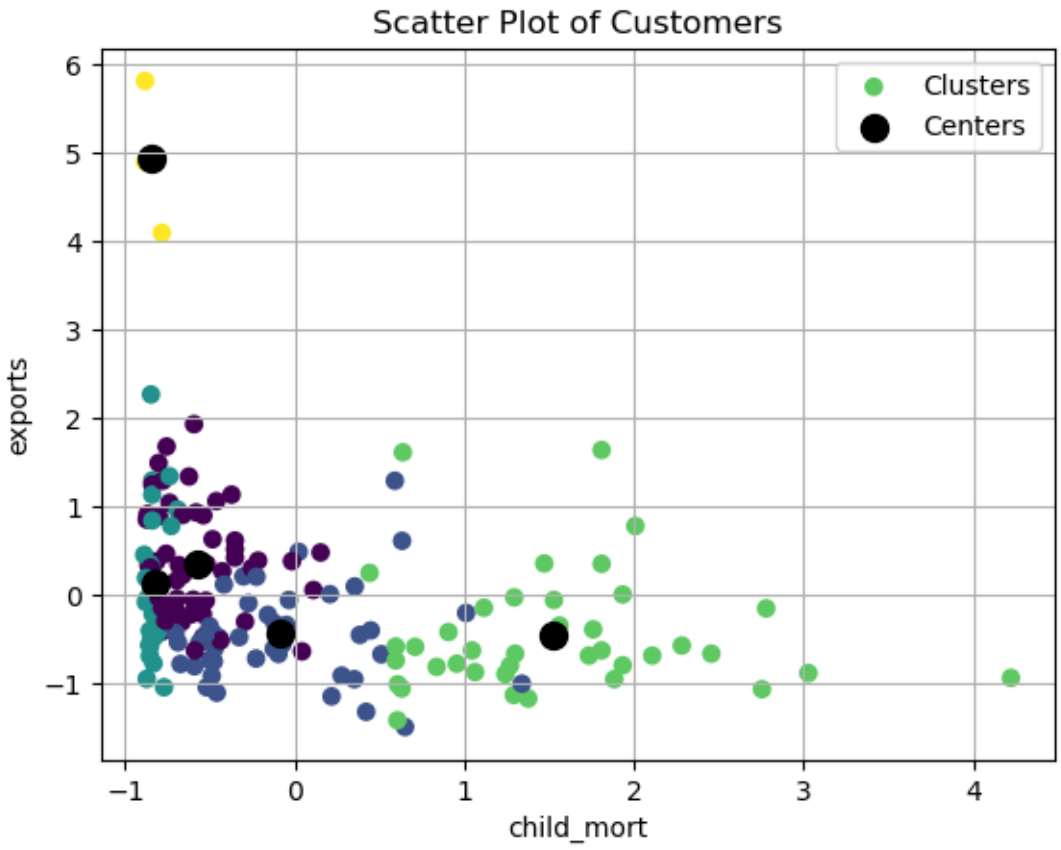
```
scatter = plt.scatter(X_scaled[:, x1_index], X_scaled[:, x2_index], c=kmeans.
 ↪labels_, cmap='viridis', label='Clusters')


centers = plt.scatter(kmeans.cluster_centers_[:, x1_index], kmeans.
 ↪cluster_centers_[:, x2_index], marker='o', color='black', s=100,␣
 ↪label='Centers')

plt.xlabel(X.columns[x1_index])
plt.ylabel(X.columns[x2_index])
plt.title('Scatter Plot of Customers')

# Generate legend
plt.legend()

plt.grid()
plt.show()
```



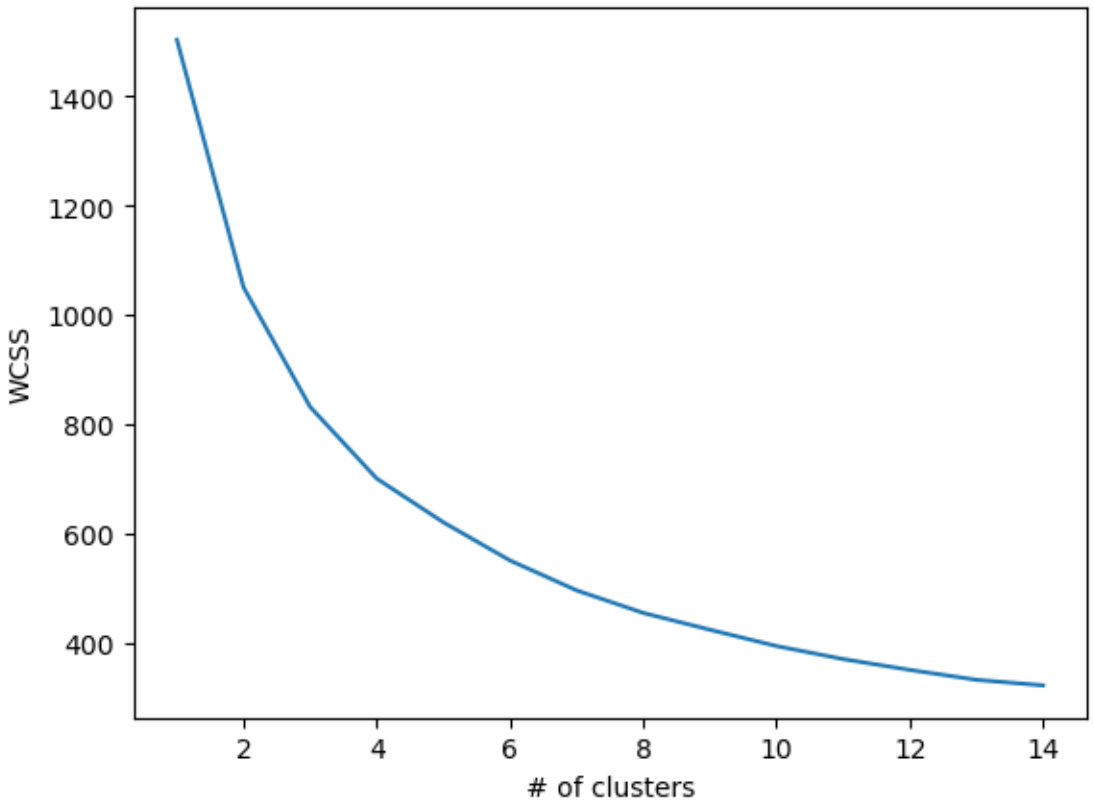
4 3.) Check a range of k-clusters and visualize to find the elbow. Test 30 different random starting places for the centroid means

[53]:
```
WCSSs = []
Ks = range(1,15)
for k in Ks:
    kmeans = KMeans(n_clusters = k, n_init = 30).fit(X_scaled)
    WCSSs.append(kmeans.inertia_)
```

[49]:
```
##OPTIONAL DO IN 1 LINE OF CODE
# WCSSs = [KMeans(n_clusters = 5,n_init = 30).fit(X_scaled).inertia_ for k in␣
↪range(1,15)]
```

5 4.) Use the above work and economic critical thinking to choose a number of clusters. Explain why you chose the number of clusters and fit a model accordingly.

[54]:
```
plt.plot(Ks,WCSSs)
plt.xlabel("# of clusters")
plt.ylabel("WCSS")
plt.show()
```

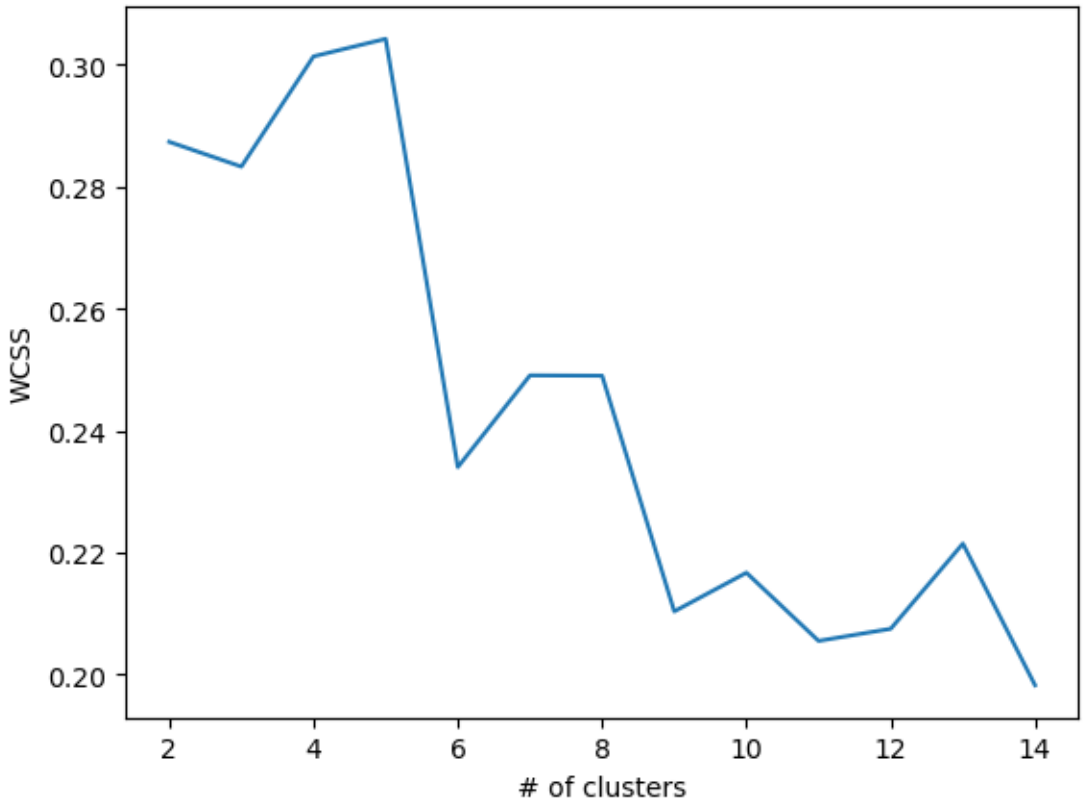


6 6.) Do the same for a silhoutte plot

[11]:
```
from sklearn.metrics import silhouette_score
```

[12]:
```
SSs = []
Ks = range(2,15)
for k in Ks:
    kmeans = KMeans(n_clusters = k, n_init = 30).fit(X_scaled)
    sil = silhouette_score(X_scaled,kmeans.labels_)
    SSs.append(sil)
```

[13]:
```
plt.plot(Ks,SSs)
plt.xlabel("# of clusters")
plt.ylabel("WCSS")
plt.show()
```



7 7.) Create a list of the countries that are in each cluster. Write interesting things you notice.

[14]:
```
kmeans = KMeans(n_clusters = 2, n_init = 30).fit(X_scaled)
```

[15]:
```
preds = pd.DataFrame(kmeans.labels_)
```

[16]:
```
output = pd.concat([preds,df],axis = 1)
```

[17]:
```
print("Cluster 1: ")
list(output.loc[output[0] == 0, "country"])
```

```
Cluster 1:
```

[17]:
```
['Albania',
 'Algeria',
 'Antigua and Barbuda',
 'Argentina',
 'Armenia',
 'Australia',
 'Austria',
 'Azerbaijan',
 'Bahamas',
 'Bahrain',
 'Barbados',
 'Belarus',
 'Belgium',
 'Belize',
 'Bhutan',
 'Bosnia and Herzegovina',
 'Brazil',
 'Brunei',
 'Bulgaria',
 'Canada',
 'Cape Verde',
 'Chile',
 'China',
 'Colombia',
 'Costa Rica',
 'Croatia',
 'Cyprus',
 'Czech Republic',
 'Denmark',
 'Dominican Republic',
 'Ecuador',
 'El Salvador',
 'Estonia',
```

```
'Fiji',
'Finland',
'France',
'Georgia',
'Germany',
'Greece',
'Grenada',
'Hungary',
'Iceland',
'Iran',
'Ireland',
'Israel',
'Italy',
'Jamaica',
'Japan',
'Jordan',
'Kazakhstan',
'Kuwait',
'Latvia',
'Lebanon',
'Libya',
'Lithuania',
'Luxembourg',
'Macedonia, FYR',
'Malaysia',
'Maldives',
'Malta',
'Mauritius',
'Moldova',
'Montenegro',
'Morocco',
'Netherlands',
'New Zealand',
'Norway',
'Oman',
'Panama',
'Paraguay',
'Peru',
'Poland',
'Portugal',
'Qatar',
'Romania',
'Russia',
'Saudi Arabia',
'Serbia',
'Seychelles',
'Singapore',
```

```
 'Slovak Republic',
 'Slovenia',
 'South Korea',
 'Spain',
 'Sri Lanka',
 'St. Vincent and the Grenadines',
 'Suriname',
 'Sweden',
 'Switzerland',
 'Thailand',
 'Tunisia',
 'Turkey',
 'Ukraine',
 'United Arab Emirates',
 'United Kingdom',
 'United States',
 'Uruguay',
 'Venezuela',
 'Vietnam']
```

[19]:
```
print("cluster 2: ")
list(output.loc[output[0] == 1, "country"])
```

```
cluster 2:
```

[19]:
```
['Afghanistan',
 'Angola',
 'Bangladesh',
 'Benin',
 'Bolivia',
 'Botswana',
 'Burkina Faso',
 'Burundi',
 'Cambodia',
 'Cameroon',
 'Central African Republic',
 'Chad',
 'Comoros',
 'Congo, Dem. Rep.',
 'Congo, Rep.',
 "Cote d'Ivoire",
 'Egypt',
 'Equatorial Guinea',
 'Eritrea',
 'Gabon',
 'Gambia',
 'Ghana',
 'Guatemala',
```

```
'Guinea',
'Guinea-Bissau',
'Guyana',
'Haiti',
'India',
'Indonesia',
'Iraq',
'Kenya',
'Kiribati',
'Kyrgyz Republic',
'Lao',
'Lesotho',
'Liberia',
'Madagascar',
'Malawi',
'Mali',
'Mauritania',
'Micronesia, Fed. Sts.',
'Mongolia',
'Mozambique',
'Myanmar',
'Namibia',
'Nepal',
'Niger',
'Nigeria',
'Pakistan',
'Philippines',
'Rwanda',
'Samoa',
'Senegal',
'Sierra Leone',
'Solomon Islands',
'South Africa',
'Sudan',
'Tajikistan',
'Tanzania',
'Timor-Leste',
'Togo',
'Tonga',
'Turkmenistan',
'Uganda',
'Uzbekistan',
'Vanuatu',
'Yemen',
'Zambia']
```

[18]:
```
#### Write an observation
```

8 8.) Create a table of Descriptive Statistics. Rows being the Cluster number and columns being all the features. Values being the mean of the centroid. Use the nonscaled X values for interprotation

[21]:
```
output.drop("country",axis = 1).groupby(0).mean()
```

[21]:

0	child_mort	exports	health	imports	income	inflation
0	12.161616	48.603030	7.314040	49.121212	26017.171717	5.503545
1	76.280882	30.198515	6.090147	43.642146	4227.397059	11.098750

0	life_expec	total_fer	gdpp
0	76.493939	1.941111	20507.979798
1	61.910294	4.413824	1981.235294

[22]:
```
output.drop("country",axis = 1).groupby(0).std()
```

[22]:

0	child_mort	exports	health	imports	income	inflation
0	8.523122	30.116032	2.716652	26.928785	20441.749847	6.957187
1	38.076068	18.201742	2.645319	19.323451	4890.581414	13.682630

0	life_expec	total_fer	gdpp
0	3.735757	0.486744	20578.727127
1	6.897418	1.285590	2528.509189

9 9.) Write an observation about the descriptive statistics.

[ ]:
```

```