**K-Means Clustering in Python:**

**INSTALL STEPS:**

(base) $ conda install matplotlib numpy pandas seaborn scikit-learn ipython

(base) $ conda install -c conda-forge kneed

**SOURCE CODE:**

In [1]: import matplotlib.pyplot as plt

...: from kneedimpor tKneeLocator

...: from sklearn.datasets import make\_blobs

...: from sklearn.cluster import KMeans

...: from sklearn.metrics import silhouette\_score

...: from sklearn.preprocessing import StandardScaler

In [2]: features,true\_labels=make\_blobs(

...: n\_samples=200,

...: centers=3,

...: cluster\_std=2.75,

...: random\_state=42

...: )

In [3]: features[:5]

Out[3]:

array([[ 9.77075874, 3.27621022],

[ -9.71349666, 11.27451802],

[ -6.91330582, -9.34755911],

[-10.86185913, -10.75063497],

[ -8.50038027, -4.54370383]])

In [4]: true\_labels[:5]

Out[4]: array([1, 0, 2, 2, 2])

In [5]: scaler=StandardScaler()

...: scaled\_features=scaler.fit\_transform(features)

In [6]: scaled\_features[:5]

Out[6]:

array([[ 2.13082109, 0.25604351],

[-1.52698523, 1.41036744],

[-1.00130152, -1.56583175],

[-1.74256891, -1.76832509],

[-1.29924521, -0.87253446]])

In [7]: kmeans=KMeans(

...: init="random",

...: n\_clusters=3,

...: n\_init=10,

...: max\_iter=300,

...: random\_state=42

...: )

In [8]: kmeans.fit(scaled\_features)

Out[8]:

KMeans(init='random', n\_clusters=3, random\_state=42)

In [9]: # The lowest SSE value

...: kmeans.inertia\_

Out[9]: 74.57960106819854

In [10]: # Final locations of the centroid

...: kmeans.cluster\_centers\_

Out[10]:

array([[ 1.19539276, 0.13158148],

[-0.25813925, 1.05589975],

[-0.91941183, -1.18551732]])

In [11]: # The number of iterations required to converge

...: kmeans.n\_iter\_

Out[11]: 6

In [12]: kmeans.labels\_[:5]

Out[12]: array([0, 1, 2, 2, 2], dtype=int32)

In [13]: kmeans\_kwargs={

...: "init":"random",

...: "n\_init":10,

...: "max\_iter":300,

...: "random\_state":42,

...: }

...:

...: # A list holds the SSE values for each k

...: sse=[]

...: forkinrange(1,11):

...: kmeans=KMeans(n\_clusters=k,\*\*kmeans\_kwargs)

...: kmeans.fit(scaled\_features)

...: sse.append(kmeans.inertia\_)

In [14]: plt.style.use("fivethirtyeight")

...: plt.plot(range(1,11),sse)

...: plt.xticks(range(1,11))

...: plt.xlabel("Number of Clusters")

...: plt.ylabel("SSE")

...: plt.show()

In [15]: kl=KneeLocator(

...: range(1,11),sse,curve="convex",direction="decreasing"

...: )

In [16]: kl.elbow

Out[16]: 3

In [17]: # A list holds the silhouette coefficients for each k

...: silhouette\_coefficients=[]

...:

...: # Notice you start at 2 clusters for silhouette coefficient

...: forkinrange(2,11):

...: kmeans=KMeans(n\_clusters=k,\*\*kmeans\_kwargs)

...: kmeans.fit(scaled\_features)

...: score=silhouette\_score(scaled\_features,kmeans.labels\_)

...: silhouette\_coefficients.append(score)

In [18]: plt.style.use("fivethirtyeight")

...: plt.plot(range(2,11),silhouette\_coefficients)

...: plt.xticks(range(2,11))

...: plt.xlabel("Number of Clusters")

...: plt.ylabel("Silhouette Coefficient")

...: plt.show()

### **Evaluating Clustering Performance Using Advanced Techniques::**

In [19]:from sklearn.clusterimportDBSCAN

...: from sklearn.datasetsimportmake\_moons

...: from sklearn.metricsimportadjusted\_rand\_score

In [20]: features,true\_labels=make\_moons(

...: n\_samples=250,noise=0.05,random\_state=42

...: )

...: scaled\_features=scaler.fit\_transform(features)

In [21]: # Instantiate k-means and dbscan algorithms

...: kmeans=KMeans(n\_clusters=2)

...: dbscan=DBSCAN(eps=0.3)

...:

...: # Fit the algorithms to the features

...: kmeans.fit(scaled\_features)

...: dbscan.fit(scaled\_features)

...:

...: # Compute the silhouette scores for each algorithm

...: kmeans\_silhouette=silhouette\_score(

...: scaled\_features,kmeans.labels\_

...: ).round(2)

...: dbscan\_silhouette=silhouette\_score(

...: scaled\_features,dbscan.labels\_

...: ).round(2)

In [22]: kmeans\_silhouette

Out[22]: 0.5

In [23]: dbscan\_silhouette

Out[23]: 0.38

In [25]: ari\_kmeans=adjusted\_rand\_score(true\_labels,kmeans.labels\_)

...: ari\_dbscan=adjusted\_rand\_score(true\_labels,dbscan.labels\_)

In [26]: round(ari\_kmeans,2)

Out[26]: 0.47

In [27]: round(ari\_dbscan,2)

Out[27]: 1.0

### **Building a K-Means Clustering Pipeline:**

In [1]: importtarfile

...: import urllib

...:

...: import numpy as np

...: import matplotlib.pyplot as plt

...: import pandas as pd

...: import seaborn as sns

...:

...: from sklearn.cluster import KMeans

...: from sklearn.decomposition import PCA

...: from sklearn.metrics import silhouette\_score, adjusted\_rand\_score

...: from sklearn.pipeline import Pipeline

...: from sklearn.preprocessing import LabelEncoder,

In [2]: uci\_tcga\_url="https://archive.ics.uci.edu/ml/machine-learning-databases/00401/"

...: archive\_name="TCGA-PANCAN-HiSeq-801x20531.tar.gz"

...: # Build the url

...: full\_download\_url=urllib.parse.urljoin(uci\_tcga\_url,archive\_name)

...:

...: # Download the file

...: r=urllib.request.urlretrieve(full\_download\_url,archive\_name)

...: # Extract the data from the archive

...: tar=tarfile.open(archive\_name,"r:gz")

...: tar.extractall()

...: tar.close()

In [3]: datafile="TCGA-PANCAN-HiSeq-801x20531/data.csv"

...: labels\_file="TCGA-PANCAN-HiSeq-801x20531/labels.csv"

...:

...: data=np.genfromtxt(

...: datafile,

...: delimiter=",",

...: usecols=range(1,20532),

...: skip\_header=1

...: )

...:

...: true\_label\_names=np.genfromtxt(

...: labels\_file,

...: delimiter=",",

...: usecols=(1,),

...: skip\_header=1,

...: dtype="str"

...: )

In [4]: data[:5,:3]

Out[4]:

array([[0. , 2.01720929, 3.26552691],

[0. , 0.59273209, 1.58842082],

[0. , 3.51175898, 4.32719872],

[0. , 3.66361787, 4.50764878],

[0. , 2.65574107, 2.82154696]])

In [5]: true\_label\_names[:5]

Out[5]: array(['PRAD', 'LUAD', 'PRAD', 'PRAD', 'BRCA'], dtype='<U4')

In [6]: label\_encoder=LabelEncoder()

In [7]: true\_labels=label\_encoder.fit\_transform(true\_label\_names)

In [8]: true\_labels[:5]

Out[8]: array([4, 3, 4, 4, 0])

In [9]: label\_encoder.classes\_

Out[9]: array(['BRCA', 'COAD', 'KIRC', 'LUAD', 'PRAD'], dtype='<U4')

In [10]: n\_clusters=len(label\_encoder.classes\_)

In [11]: preprocessor=Pipeline(

...: [

...: ("scaler",MinMaxScaler()),

...: ("pca",PCA(n\_components=2,random\_state=42)),

...: ]

...: )

In [12]: clusterer=Pipeline(

...: [

...: (

...: "kmeans",

...: KMeans(

...: n\_clusters=n\_clusters,

...: init="k-means++",

...: n\_init=50,

...: max\_iter=500,

...: random\_state=42,

...: ),

...: ),

...: ]

...: )

In [13]: pipe=Pipeline(

...: [

...: ("preprocessor",preprocessor),

...: ("clusterer",clusterer)

...: ]

...: )

In [14]: pipe.fit(data)

Out[14]:

Pipeline(steps=[('preprocessor',

Pipeline(steps=[('scaler', MinMaxScaler()),

('pca',

PCA(n\_components=2, random\_state=42))])),

('clusterer',

Pipeline(steps=[('kmeans',

KMeans(max\_iter=500, n\_clusters=5, n\_init=50,

random\_state=42))]))])

In [15]: preprocessed\_data=pipe["preprocessor"].transform(data)

In [16]: predicted\_labels=pipe["clusterer"]["kmeans"].labels\_

In [17]: silhouette\_score(preprocessed\_data,predicted\_labels)

Out[17]: 0.5118775528450304

In [18]: adjusted\_rand\_score(true\_labels,predicted\_labels)

Out[18]: 0.722276752060253

In [19]: pcadf=pd.DataFrame(

...: pipe["preprocessor"].transform(data),

...: columns=["component\_1","component\_2"],

...: )

...:

...: pcadf["predicted\_cluster"]=pipe["clusterer"]["kmeans"].labels\_

...: pcadf["true\_label"]=label\_encoder.inverse\_transform(true\_labels)

In [20]: plt.style.use("fivethirtyeight")

...: plt.figure(figsize=(8,8))

...:

...: scat=sns.scatterplot(

...: "component\_1",

...: "component\_2",

...: s=50,

...: data=pcadf,

...: hue="predicted\_cluster",

...: style="true\_label",

...: palette="Set2",

...: )

...:

...: scat.set\_title(

...: "Clustering results from TCGA Pan-Cancer\nGene Expression Data"

...: )

...: plt.legend(bbox\_to\_anchor=(1.05,1),loc=2,borderaxespad=0.0)

...:

...: plt.show()

### **Tuning a K-Means Clustering Pipeline**

In [21]: # Empty lists to hold evaluation metrics

...: silhouette\_scores=[]

...: ari\_scores=[]

...: forninrange(2,11):

...: # This set the number of components for pca,

...: # but leaves other steps unchanged

...: pipe["preprocessor"]["pca"].n\_components=n

...: pipe.fit(data)

...:

...: silhouette\_coef=silhouette\_score(

...: pipe["preprocessor"].transform(data),

...: pipe["clusterer"]["kmeans"].labels\_,

...: )

...: ari=adjusted\_rand\_score(

...: true\_labels,

...: pipe["clusterer"]["kmeans"].labels\_,

...: )

...:

...: # Add metrics to their lists

...: silhouette\_scores.append(silhouette\_coef)

...: ari\_scores.append(ari)

In [22]: plt.style.use("fivethirtyeight")

...: plt.figure(figsize=(6,6))

...: plt.plot(

...: range(2,11),

...: silhouette\_scores,

...: c="#008fd5",

...: label="Silhouette Coefficient",

...: )

...: plt.plot(range(2,11),ari\_scores,c="#fc4f30",label="ARI")

...: plt.xlabel("n\_components")

...: plt.legend()

...: plt.title("Clustering Performance as a Function of n\_components")

...: plt.tight\_layout()

...: plt.show()