*Machine Learning Lab* Assignment 4

**Name – Soumalya Sahoo**

**Roll - 001811001055**

**Semester - 7**

**Year - 4**

**Department - Information Technology(IT)**

Github link:

# Partition based: K-means

**IRIS PLANT DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

kmeans = KMeans(init="random", n\_clusters=3, n\_init=10, max\_iter=300, random\_state=42)

y = kmeans.fit\_predict(x)

print("K-Means Cluster Centers")

print(kmeans.cluster\_centers\_)

print("Cluster Labels")

print(kmeans.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=kmeans.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

axes[1].set\_ylabel('Sepal width', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)

from sklearn.metrics import silhouette\_score

print("The silhouette score is :")

silhouette\_score(x, kmeans.labels\_)

from sklearn.metrics import calinski\_harabasz\_score

print("The calinski harabasz score is :")

calinski\_harabasz\_score(x, kmeans.labels\_)

from sklearn.metrics import davies\_bouldin\_score

print("The davies bouldin score is :")

davies\_bouldin\_score(x, kmeans.labels\_)

print("It is observed that TSS=SSE+SSB is a constant. Hence we will calculate the TSS ans substract SSE from it to get SSB")

print("The value of SSE is: ")

print(kmeans.inertia\_)

# Finding the overall centroid of the data points

centers = kmeans.cluster\_centers\_

center\_x = []

for center in centers:

  center\_x.append(center[0])

center\_x

overall\_center = sum(center\_x)/len(center\_x)

tss = 0

for i in range(len(df)):

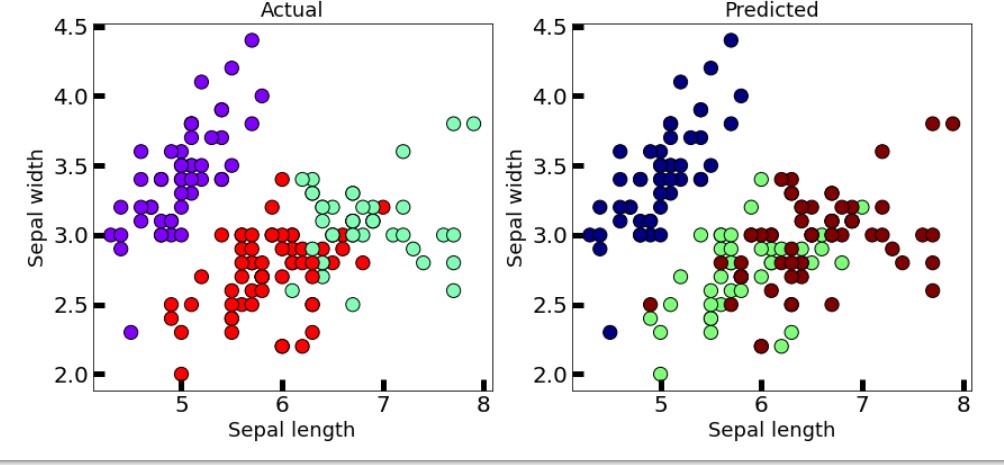
  a = df.iloc[i][0] - overall\_center

  b = pow(a,2)

  tss = tss+b

print("The value of SSB is: ")

print(tss - kmeans.inertia\_)



WINE DATASET

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

kmeans = KMeans(init="random", n\_clusters=3, n\_init=10, max\_iter=300, random\_state=42)

y = kmeans.fit\_predict(x)

print("K-Means Cluster Centers")

print(kmeans.cluster\_centers\_)

print("Cluster Labels")

print(kmeans.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=kmeans.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)

from sklearn.metrics import silhouette\_score

print("The silhouette score is :")

silhouette\_score(x, kmeans.labels\_)

from sklearn.metrics import calinski\_harabasz\_score

print("The calinski harabasz score is :")

calinski\_harabasz\_score(x, kmeans.labels\_)

from sklearn.metrics import davies\_bouldin\_score

print("The davies bouldin score is :")

davies\_bouldin\_score(x, kmeans.labels\_)

print("It is observed that TSS=SSE+SSB is a constant. Hence we will calculate the TSS ans substract SSE from it to get SSB")

print("The value of SSE is: ")

print(kmeans.inertia\_)

# Finding the overall centroid of the data points

centers = kmeans.cluster\_centers\_

center\_x = []

for center in centers:

  center\_x.append(center[0])

center\_x

overall\_center = sum(center\_x)/len(center\_x)

tss = 0

for i in range(len(df)):

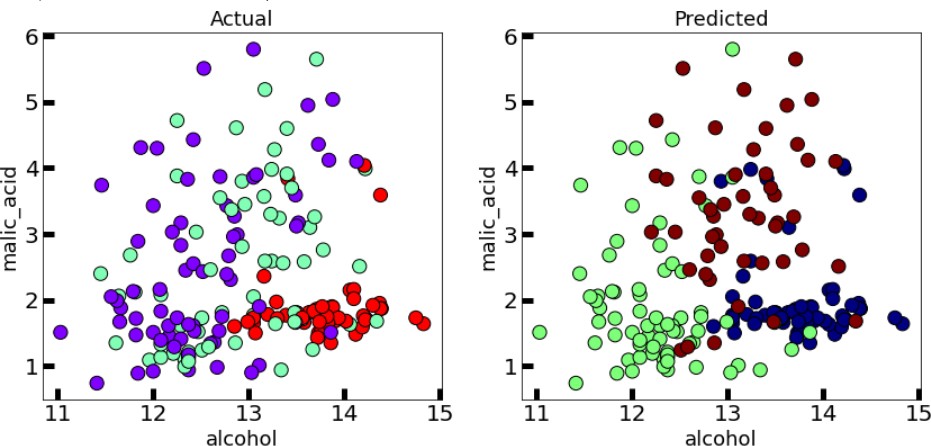
  a = df.iloc[i][0] - overall\_center

  b = pow(a,2)

  tss = tss+b

print("The value of SSB is: ")

print(tss - kmeans.inertia\_)



**This algorithm generalizes to clusters of different shapes and sizes, such as elliptical clusters. The**

**problem with it is that we need to manually choose the value of “k”.**

1. ***Partition based: K-medoids***
   1. **IRIS PLANT DATASET**

#importing libraries

!pip install scikit-learn-extra

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn\_extra.cluster import KMedoids

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

kmedoid = KMedoids(init="heuristic", n\_clusters=3, max\_iter=300, random\_state=42)

y = kmedoid.fit\_predict(x)

print("K-Medoids Cluster Centers")

print(kmedoid.cluster\_centers\_)

print("Cluster Labels")

print(kmedoid.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=kmedoid.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

axes[1].set\_ylabel('Sepal width', fontsize=18)

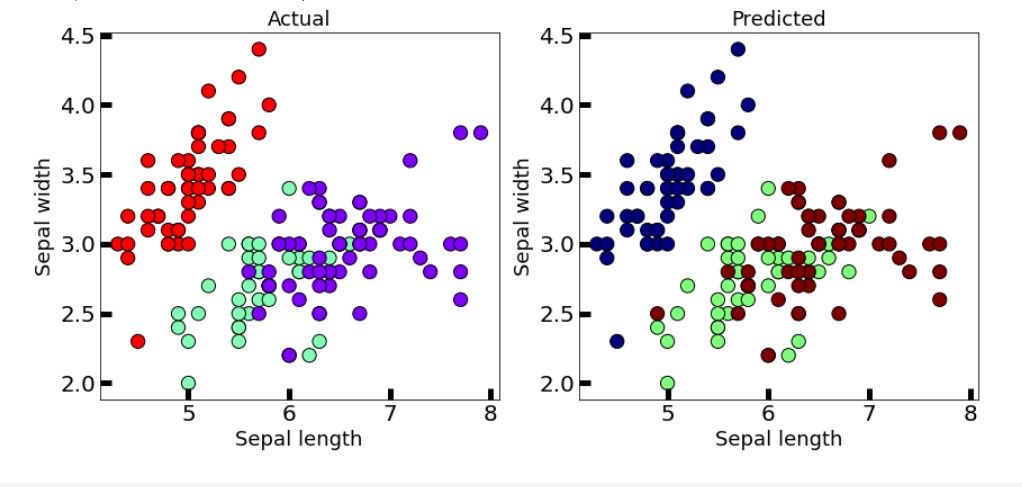
axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)

**Also added some scores code .**



* 1. **WINE DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn\_extra.cluster import KMedoids

from sklearn.datasets import load\_iris

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

kmedoid = KMedoids(init="heuristic", n\_clusters=3, max\_iter=300, random\_state=42)

y = kmedoid.fit\_predict(x)

print("K-Medoids Cluster Centers")

print(kmedoid.cluster\_centers\_)

print("Cluster Labels")

print(kmedoid.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=kmedoid.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

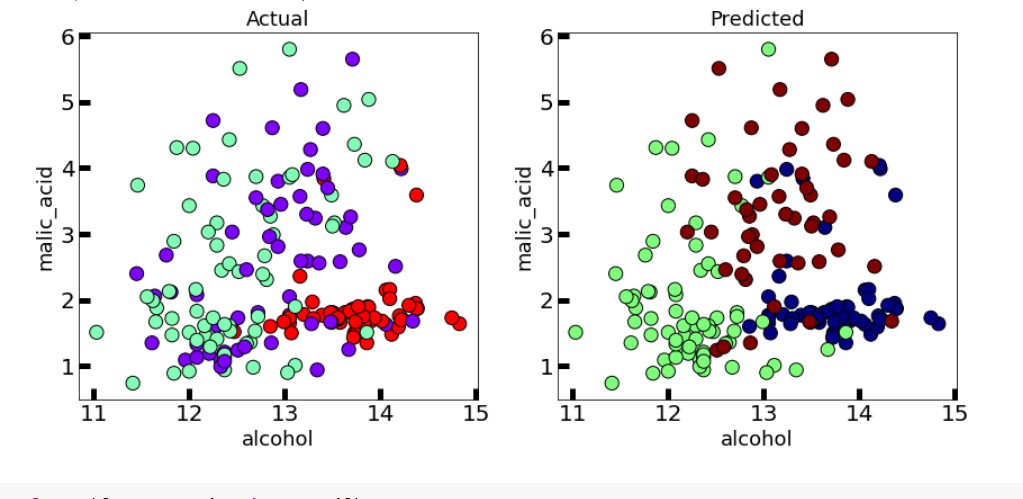
axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



This algorithm solves the problem with the K-means algorithm.

K-means attempts to minimize the total squared error, while k-medoids minimizes the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster..

# Hierarchical: Dendrogram

* 1. **IRIS PLANT DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

from scipy.cluster.hierarchy import dendrogram, linkage

linked = linkage(x, 'single')

plt.figure(figsize=(10,7))

dendrogram(linked,

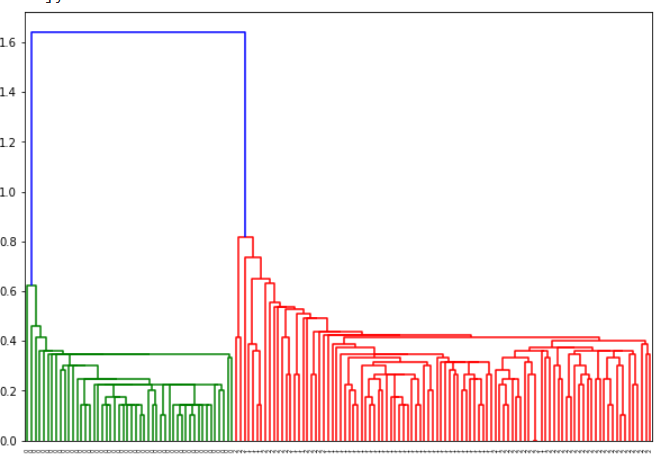
           orientation='top',

           labels=iris.target,

           distance\_sort='descending',

           show\_leaf\_counts=True)

plt.show()



* 1. **WINE DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

from scipy.cluster.hierarchy import dendrogram, linkage

linked = linkage(x, 'single')

plt.figure(figsize=(10,7))

dendrogram(linked,

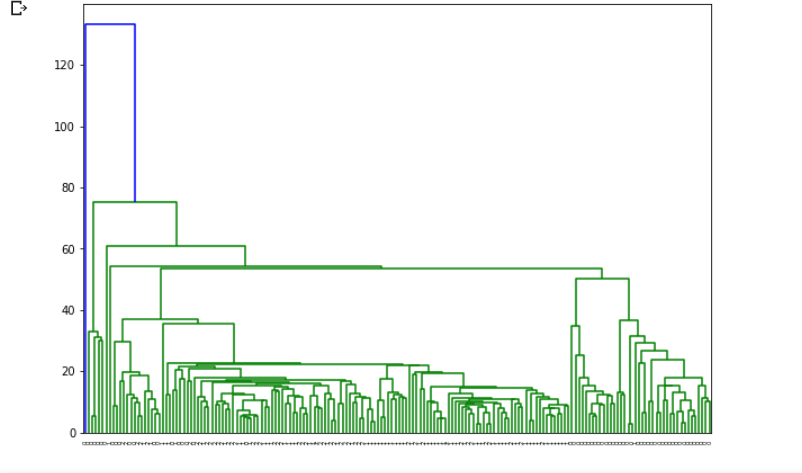
           orientation='top',

           labels=wine.target,

           distance\_sort='descending',

           show\_leaf\_counts=True)

plt.show()



A dendrogram is a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from hierarchical clustering. The main use of a dendrogram is to work out the best way to allocate objects to clusters.

# Hierarchical: AGNES

* 1. **IRIS PLANT DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')

y = cluster.fit\_predict(x)

print("Cluster labels:")

print(cluster.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=cluster.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

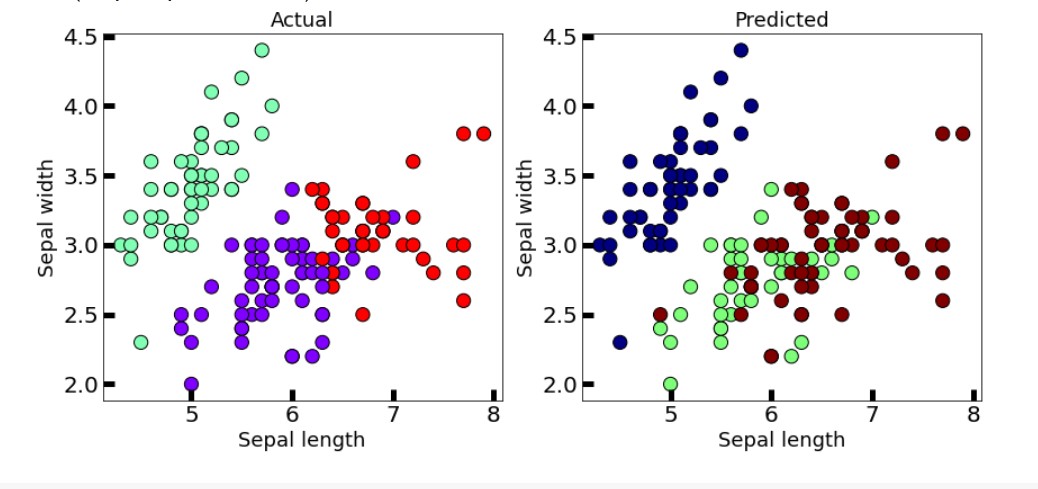
axes[1].set\_ylabel('Sepal width', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



WINE DATASET

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')

y = cluster.fit\_predict(x)

print("Cluster Labels")

print(cluster.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=cluster.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

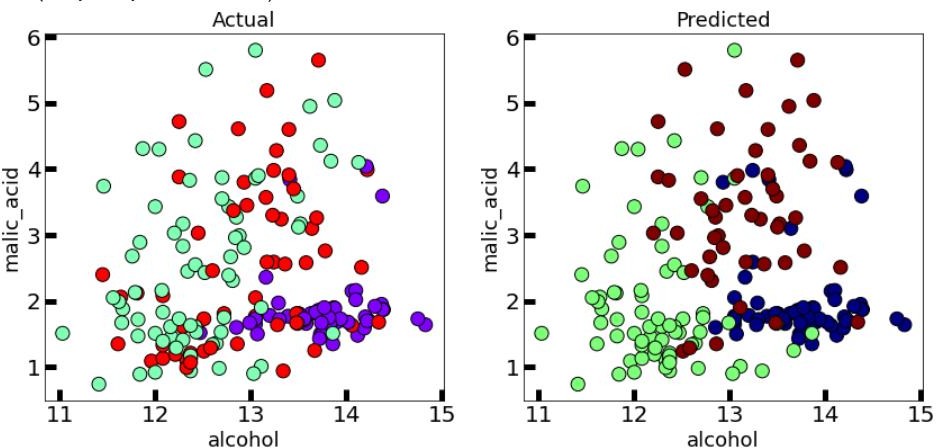
axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



The agglomerative clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It’s also known as AGNES (Agglomerative Nesting).

The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects.

The result is a tree-based representation of the objects, named dendrogram.

# Hierarchical: BIRCH

* 1. **IRIS PLANT DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

from sklearn.cluster import Birch

birch = Birch(n\_clusters=3, compute\_labels=True, branching\_factor=50)

y = birch.fit\_predict(x)

print("Cluster Labels")

print(cluster.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=birch.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

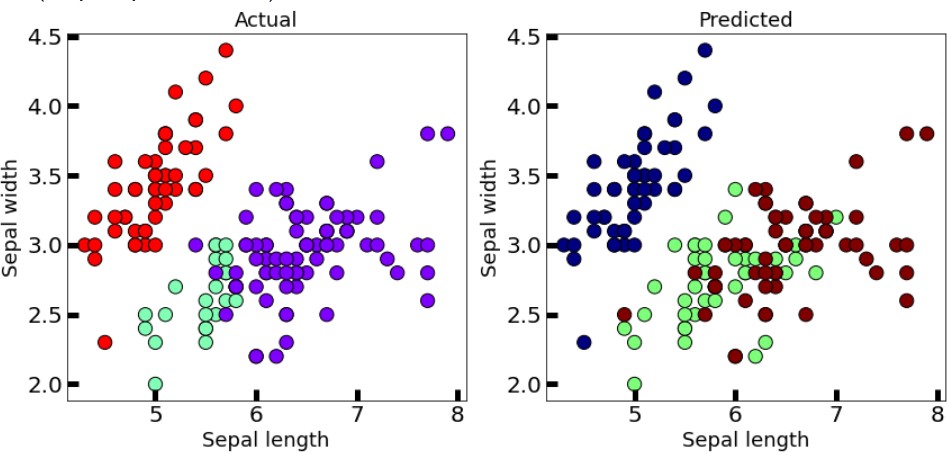
axes[1].set\_ylabel('Sepal width', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



* 1. **WINE DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

from sklearn.cluster import Birch

birch = Birch(n\_clusters=3, compute\_labels=True, branching\_factor=50)

y = birch.fit\_predict(x)

print("Cluster Labels")

print(cluster.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=birch.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

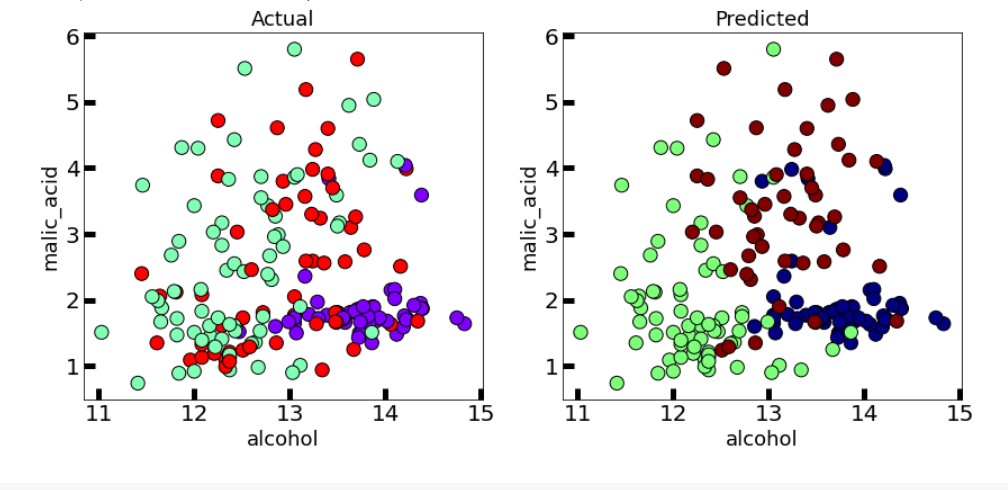
axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a clustering algorithm that can cluster large datasets by first generating a small and compact summary of the large dataset that retains as much information as possible.

This smaller summary is then clustered instead of clustering the larger dataset

# Density based: DBSCAN

* 1. **IRIS PLANT DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, algorithm='auto', metric='euclidean')

y = dbscan.fit\_predict(x)

print("Cluster Labels")

print(dbscan.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=dbscan.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

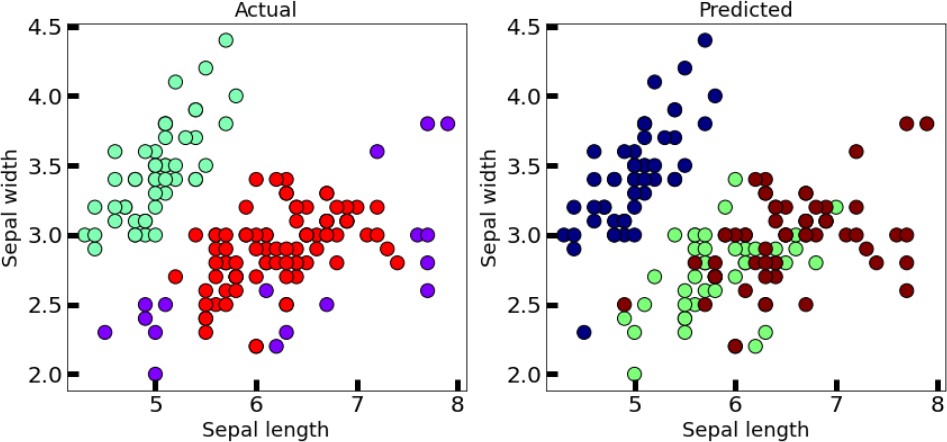
axes[1].set\_ylabel('Sepal width', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



* 1. **WINE DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=35, algorithm='auto', metric='euclidean')

y = dbscan.fit\_predict(x)

print("Cluster Labels")

print(dbscan.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=dbscan.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

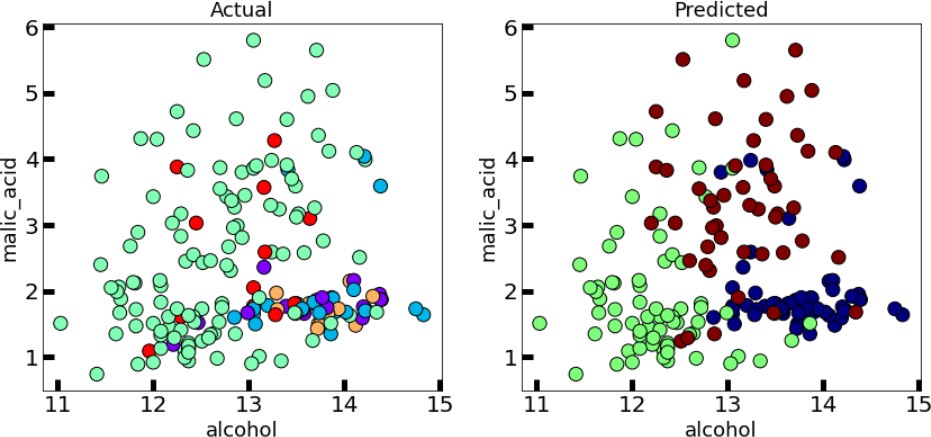
axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



Clusters are dense regions in the data space, separated by regions of the lower density of points. The DBSCAN algorithm is based on this intuitive notion 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.

# Density based: OPTICS

# IRIS PLANT DATASET

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, algorithm='auto', metric='euclidean')

y = dbscan.fit\_predict(x)

print("Cluster Labels")

print(dbscan.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=dbscan.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

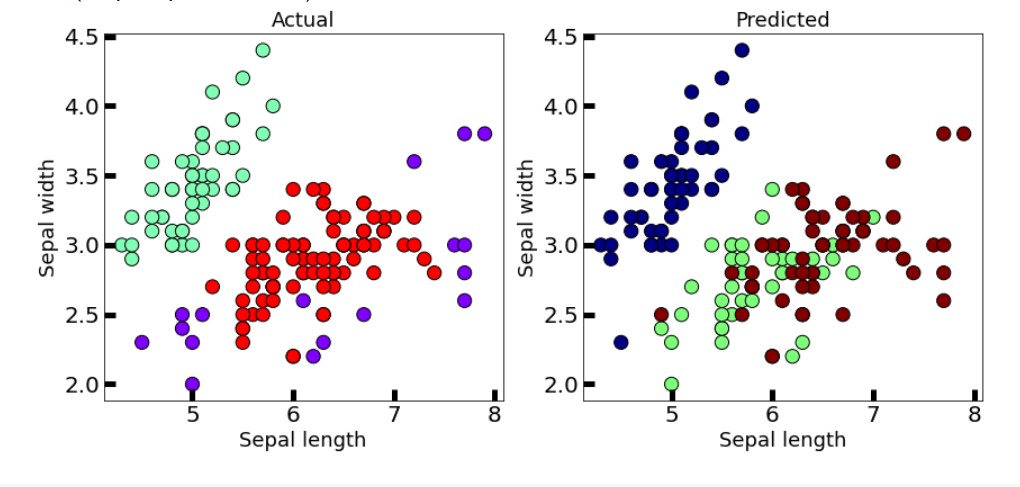
axes[1].set\_ylabel('Sepal width', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



* 1. **WINE DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=35, algorithm='auto', metric='euclidean')

y = dbscan.fit\_predict(x)

print("Cluster Labels")

print(dbscan.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=dbscan.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

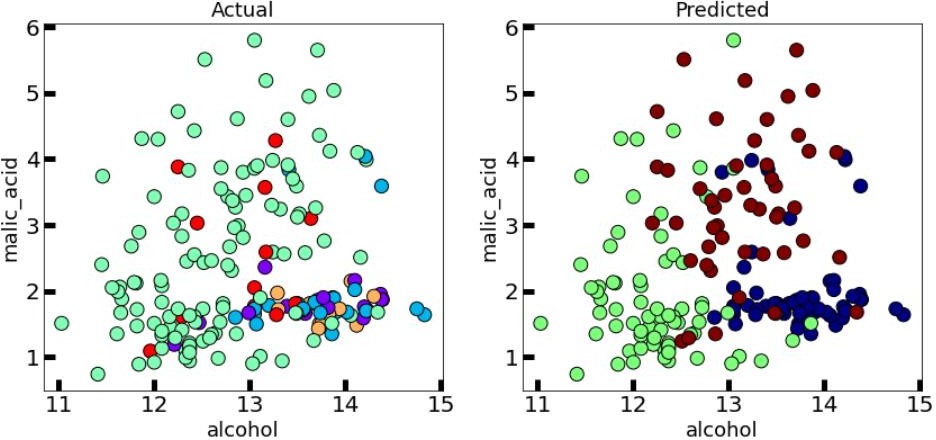
axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



This clustering technique is different from other clustering techniques in the sense that this technique does not explicitly segment the data into clusters.

Instead, it produces a visualization of Reachability distances and uses this visualization to cluster the data.

# K-means++

* 1. **IRIS PLANT DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

kmeans = KMeans(init='k-means++', n\_clusters=3, n\_init=10, max\_iter=300, random\_state=42)

y = kmeans.fit\_predict(x)

print("K-Means Cluster Centers")

print(kmeans.cluster\_centers\_)

print("Cluster Labels")

print(kmeans.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=kmeans.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

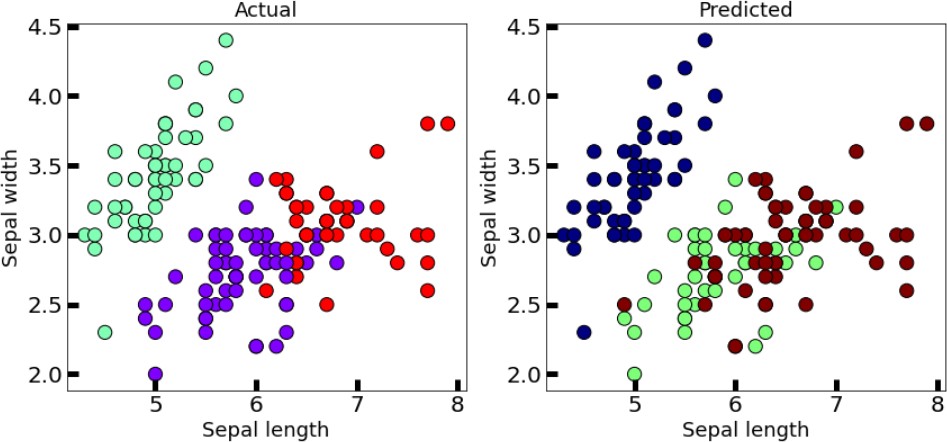
axes[1].set\_ylabel('Sepal width', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



* 1. **WINE DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

kmeans = KMeans(init='k-means++', n\_clusters=3, n\_init=10, max\_iter=300, random\_state=42)

y = kmeans.fit\_predict(x)

print("K-Means Cluster Centers")

print(kmeans.cluster\_centers\_)

print("Cluster Labels")

print(kmeans.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=kmeans.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=y, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

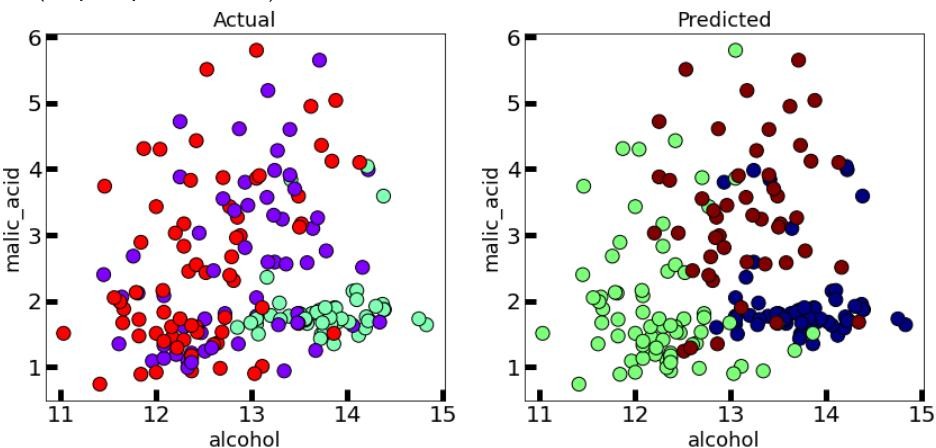
axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



In the case of K-Means clustering, we were using randomization. The initial k-centroids were picked randomly from the data points.

This randomization of picking k-centroids points results in the problem of initialization sensitivity. This problem tends to affect the final formed clusters. The final formed clusters depend on how initial centroids were picked.

K-Means++ solves the above problem.

# Bisecting K-means

* 1. **IRIS PLANT DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

iris=load\_iris()   #loading iris dataset from sklearn.datasets

iris

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df

x=iris.data

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=iris.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('Sepal Width', fontsize=18)

plt.ylabel('Sepal length', fontsize=18)

from sklearn.cluster import KMeans

import numpy as np

K = 2

current\_clusters = 1

split = 0

while current\_clusters != K:

    kmeans = KMeans(n\_clusters=2).fit(x)

    current\_clusters += 1

    split += 1

    cluster\_centers = kmeans.cluster\_centers\_

    sse = [0]\*2

    for point, label in zip(x, kmeans.labels\_):

        sse[label] += np.square(point-cluster\_centers[label]).sum()

    chosen\_cluster = np.argmax(sse, axis=0)

    chosen\_cluster\_data = x[kmeans.labels\_ == chosen\_cluster]

    x = chosen\_cluster\_data

print("K-Means Cluster Centers")

print(kmeans.cluster\_centers\_)

print("Cluster Labels")

print(kmeans.labels\_)

plt.scatter(x=df['sepal length'], y=df['sepal width'] ,c=kmeans.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['sepal length'], y=df['sepal width'], c=kmeans.labels\_, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['sepal length'], y=df['sepal width'], c=iris.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('Sepal length', fontsize=18)

axes[0].set\_ylabel('Sepal width', fontsize=18)

axes[1].set\_xlabel('Sepal length', fontsize=18)

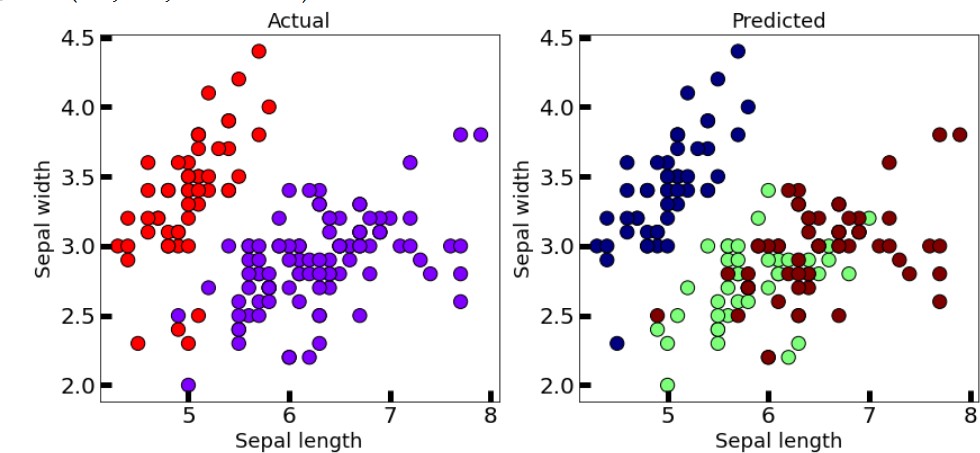
axes[1].set\_ylabel('Sepal width', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



* 1. **WINE DATASET**

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

wine

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

  'od280/od315\_of\_diluted\_wines',

  'proline'])

df

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)

from sklearn.cluster import KMeans

import numpy as np

K = 2

current\_clusters = 1

split = 0

while current\_clusters != K:

    kmeans = KMeans(n\_clusters=2).fit(x)

    current\_clusters += 1

    split += 1

    cluster\_centers = kmeans.cluster\_centers\_

    sse = [0]\*2

    for point, label in zip(x, kmeans.labels\_):

        sse[label] += np.square(point-cluster\_centers[label]).sum()

    chosen\_cluster = np.argmax(sse, axis=0)

    chosen\_cluster\_data = x[kmeans.labels\_ == chosen\_cluster]

    x = chosen\_cluster\_data

print("K-Means Cluster Centers")

print(kmeans.cluster\_centers\_)

print("Cluster Labels")

print(kmeans.labels\_)

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=kmeans.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()

fig, axes = plt.subplots(1, 2, figsize=(14,6))

axes[0].scatter(x=df['alcohol'], y=df['malic\_acid'], c=kmeans.labels\_, cmap='rainbow',edgecolor='k', s=150) #you can also try cmap='rainbow'

axes[1].scatter(x=df['alcohol'], y=df['malic\_acid'], c=wine.target, cmap='jet',edgecolor='k', s=150)

axes[0].set\_xlabel('alcohol', fontsize=18)

axes[0].set\_ylabel('malic\_acid', fontsize=18)

axes[1].set\_xlabel('alcohol', fontsize=18)

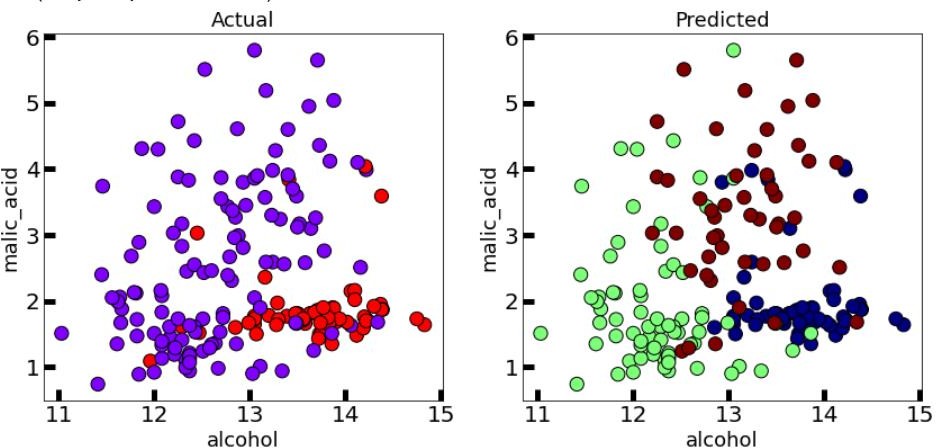
axes[1].set\_ylabel('malic\_acid', fontsize=18)

axes[0].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[1].tick\_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set\_title('Actual', fontsize=18)

axes[1].set\_title('Predicted', fontsize=18)



Bisecting K-means clustering technique is a little modification to the regular K-Means algorithm, wherein we can fix the procedure of dividing the data into clusters.

So, similar to K-means, we first initialize K centroids (You can either do this randomly or can have some prior).

After which we apply regular K-means with K=2 (that’s why the word bisecting). We keep repeating this bisection step until the desired number of clusters are reached.

For the Silhouette Score , Calinski Harabasz Score , Davies Bouldin Score , SSE and SSB I have written the code in only the first parts as it is same for all the comparisons and the comparison table is as shown :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Type of Algorithm** | **Algorithm** | **Dataset** | **Silhouette Score** | **Calinski Harabasz Score** | **Davies Bouldin Score** | **SSE** | **SSB** |
| **Partition Based** | K-means | IRIS PLANT DATASET | 0.5528190124 | 561.6277566 | 0.6619715465 | 78.85144143 | 24.18035247 |
|  | WINE DATASET | 0.5711381938 | 561.8156579 | 0.5342431775 | 2370689.687 | -2370571.805 |
| K-medoids | IRIS PLANT DATASET | 0.5201984013 | 521.5609065 | 0.668624441 | 98.86857318 | 5.11476015 |
|  | WINE DATASET | 0.5666480409 | 539.3792354 | 0.5292394126 | 16376.96932 | -16243.64278 |
| **Hierarchical** | Dendrogram | IRIS PLANT DATASET | - | - | - | - | - |
|  | WINE DATASET | - | - | - | - | - |
| AGNES | IRIS PLANT DATASET | 0.5543236611 | 558.0580408 | 558.0580408 | - | - |
|  | WINE DATASET | 0.5644796402 | 552.8517115 | 0.5357343074 | - | - |
| BIRCH | IRIS PLANT DATASET | 0.5019524848 | 458.4725106 | 0.6258305924 | - | - |
|  | WINE DATASET | 0.5644796402 | 552.8517115 | 0.5357343074 | - | - |
| **Density Based** | DBSCAN | IRIS PLANT DATASET | 0.486034197 | 220.297515 | 7.222448016 | - | - |
|  | WINE DATASET | 0.4413295945 | 208.9449396 | 7.812129203 | - | - |
| OPTICS | IRIS PLANT DATASET | 0.486034197 | 220.297515 | 7.222448016 | - | - |
|  | WINE DATASET | 0.4413295945 | 208.9449396 | 7.812129203 | - | - |
| **Additional** | K-means++ | IRIS PLANT DATASET | 0.5528190124 | 561.6277566 | 0.6619715465 | 78.85144143 | 24.18035247 |
|  | WINE DATASET | 0.5711381938 | 561.8156579 | 0.5342431775 | 2370689.687 | -2370571.805 |
| Bisecting K-means | IRIS PLANT DATASET | 0.3093066205 | 61.17725176 | 1.099971025 | 152.3479518 | -44.7653206 |
|  | WINE DATASET | 0.00384025695 | 1.06619667 | 9.045634695 | 4543749.615 | -4543626.929 |