Pattern Processing using AI Practical File

COSCE60

Nikhil Gupta
2019UCO1719
COE Section 3

In [ ]:
```
import json
import requests
```

In [ ]:
```
def chatbot():
    print("Hi! I'm a weather prediction chatbot")
    api_key = "f37d0c3cb6c045218e1152633232504"
    while(1):
        print("Which city would you like the weather forcast for? Type 'exit' to qu
        city = input().lower()
        if(city=="exit"):
            break
        url = f"https://api.weatherapi.com/v1/forecast.json?key={api_key}&q={city}
        response = requests.get(url)
        data = json.loads(response.text)
        if len(data)>1:
            temp = data["current"]["temp_c"]
            description = data["current"]["condition"]["text"]
            humidity = data["current"]["humidity"]
            wind_speed = data["current"]["wind_kph"]
            print(f"The weather in {city.title()} is {description}, with a tempera
            print()
        else:
            print("City not found. Please try again.")
            print()
    print("Thank you for using the chatbot.")
```

In [ ]:
```
chatbot()
```

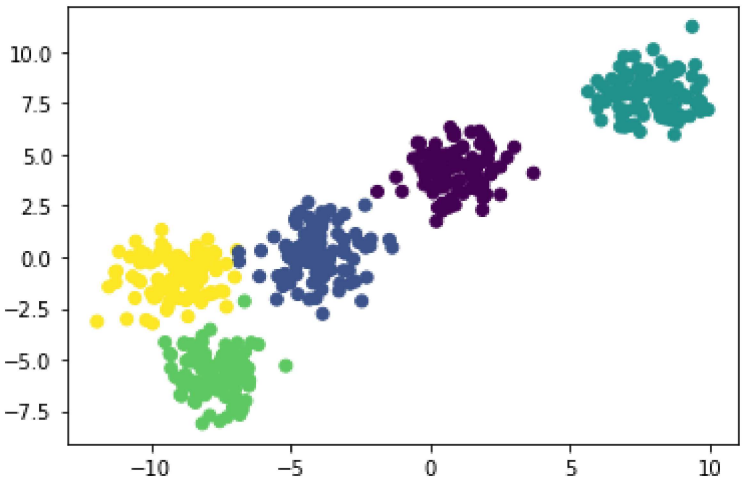
```
Hi! I'm a weather prediction chatbot
Which city would you like the weather forcast for? Type 'exit' to quit
The weather in New Delhi is Mist, with a temperature of 32.0°C , humidity of 26%,
and wind speed of 9.0 km/h.

Which city would you like the weather forcast for? Type 'exit' to quit
The weather in Mumbai is Overcast, with a temperature of 31.0°C , humidity of 59%,
and wind speed of 13.0 km/h.

Which city would you like the weather forcast for? Type 'exit' to quit
The weather in Landon is Partly cloudy, with a temperature of 5.0°C , humidity of
81%, and wind speed of 19.1 km/h.

Which city would you like the weather forcast for? Type 'exit' to quit
The weather in Dwarka is Sunny, with a temperature of 31.9°C , humidity of 59%, an
d wind speed of 16.9 km/h.

Which city would you like the weather forcast for? Type 'exit' to quit
City not found. Please try again.

Which city would you like the weather forcast for? Type 'exit' to quit
The weather in Mexico is Partly cloudy, with a temperature of 21.0°C , humidity of
23%, and wind speed of 13.0 km/h.

Which city would you like the weather forcast for? Type 'exit' to quit
Thank you for using the chatbot.
```

In [ ]:

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
```

In [ ]:
```
X,Y = make_blobs(n_samples=500,n_features=2,centers=5,random_state=3)
```

In [ ]:
```
plt.figure(0)
plt.scatter(X[:,0],X[:,1],c=Y)
plt.show()
```



In [ ]:
```
k = 5
color = ["green","red","yellow","blue","orange"]

clusters = {}

for i in range(k):
    center = 10*(2*np.random.random((X.shape[1],))-1)
    points = []
    cluster = {
        "center":center,
        "points":points,
        "color":color[i]
    }

    clusters[i] = cluster
```

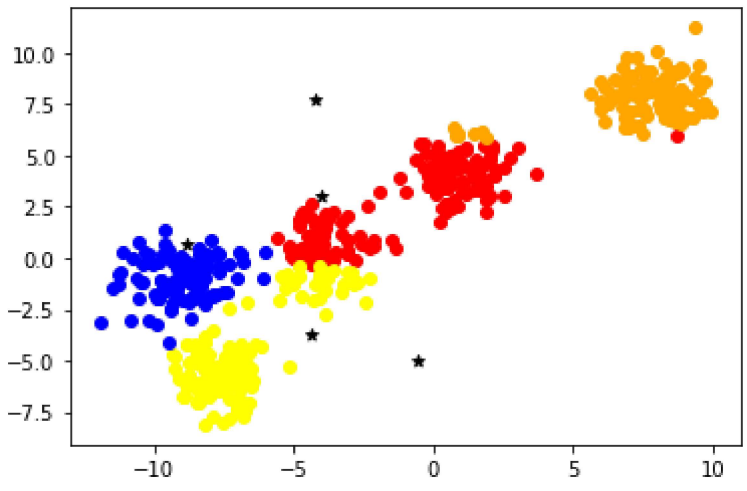
In [ ]:
```
print(clusters)
```

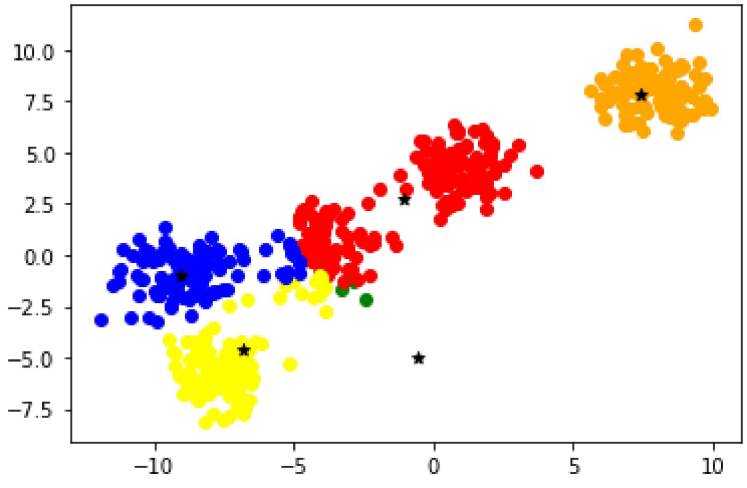
```
{0: {'center': array([-0.53948567, -4.99002492]), 'points': [], 'color': 'green'},
1: {'center': array([-4.0003705 ,  3.05827539]), 'points': [], 'color': 'red'}, 2:
{'center': array([-4.4061116, -3.7088346]), 'points': [], 'color': 'yellow'}, 3:
{'center': array([-8.86629643,  0.72809729]), 'points': [], 'color': 'blue'}, 4:
{'center': array([-4.2499521,  7.7633045]), 'points': [], 'color': 'orange'}}
```

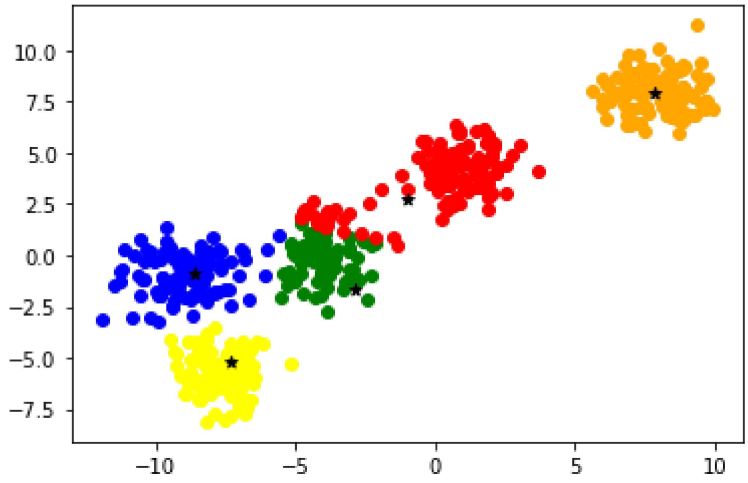
In [ ]:
```
def distance(x1,x2):
    return np.sqrt(np.sum((x1-x2)**2))

def assignPointsToCluster(clusters):
    for i in range(X.shape[0]):
        clust_x = X[i]
        dist = []
        for kx in range(k):
            d = distance(clust_x,clusters[kx]['center'])
            dist.append(d)
```

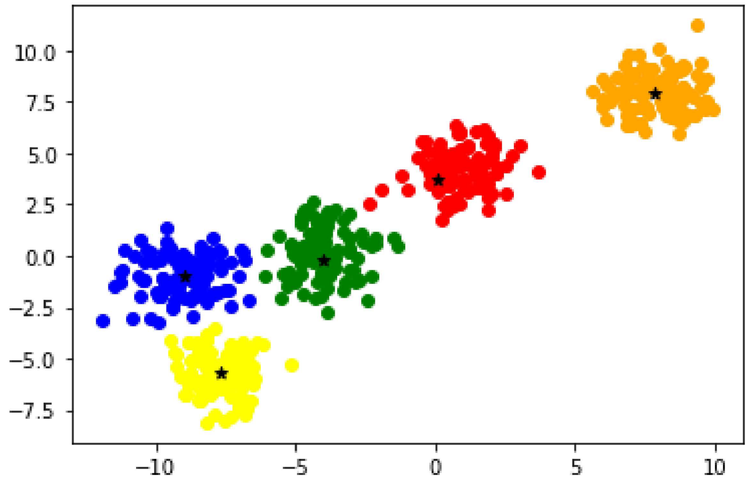
```
        idx = np.argmin(dist)
        clusters[idx]['points'].append(clust_x)

def updateCluster(clusters):
    for kx in range(k):
        pts = np.array(clusters[kx]['points'])

        if(pts.shape[0]>0):
            new_centers = np.mean(pts,axis=0)
            clusters[kx]['center'] = new_centers
            clusters[kx]['points'] = []

def plotClusters(clusters):

    plt.figure()
    for kx in range(k):
        pts = np.array(clusters[kx]['points'])

        try:
            plt.scatter(pts[:,0],pts[:,1],color=clusters[kx]['color'])
        except:
            pass

        cent = clusters[kx]['center']
        plt.scatter(cent[0],cent[1],color='black',marker="*")
```

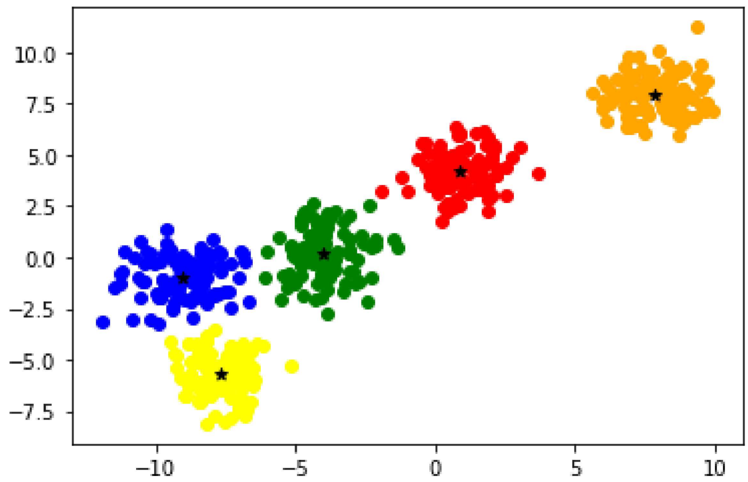
In [ ]:
```
epoch = 10
for i in range(epoch):
    assignPointsToCluster(clusters)
    plotClusters(clusters)
    updateCluster(clusters)
```

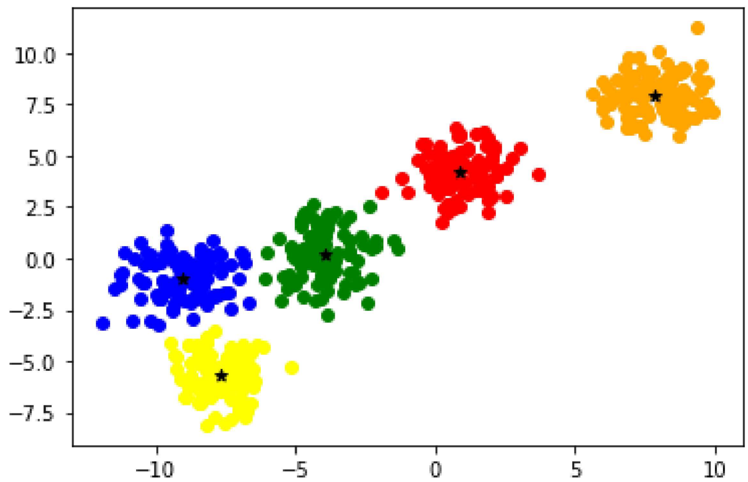


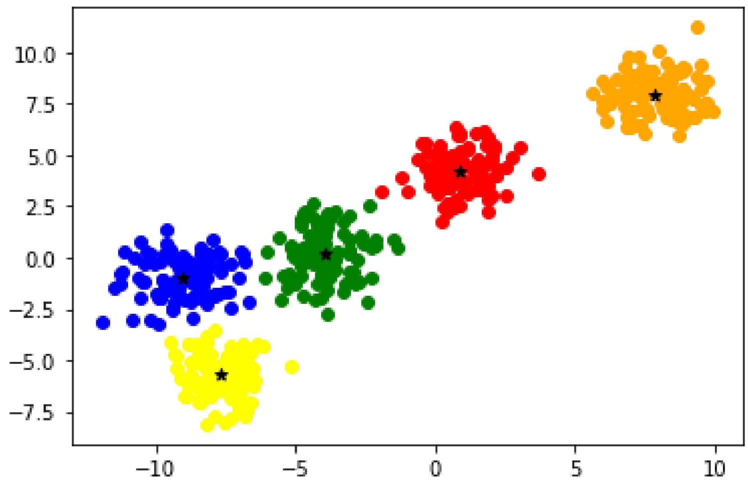


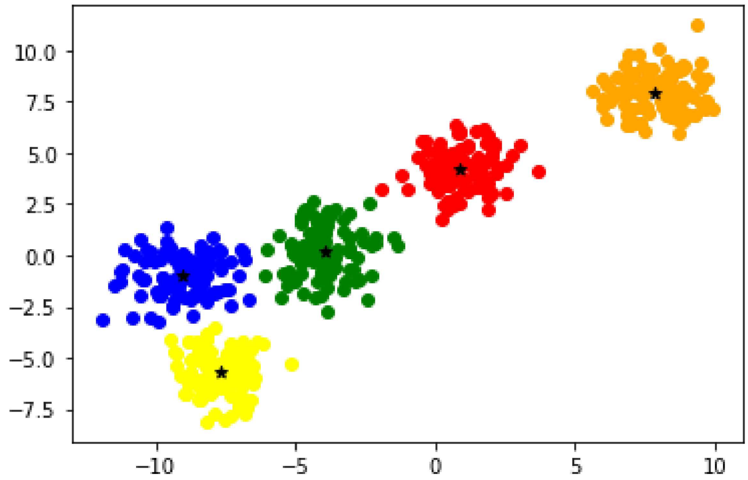

10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10

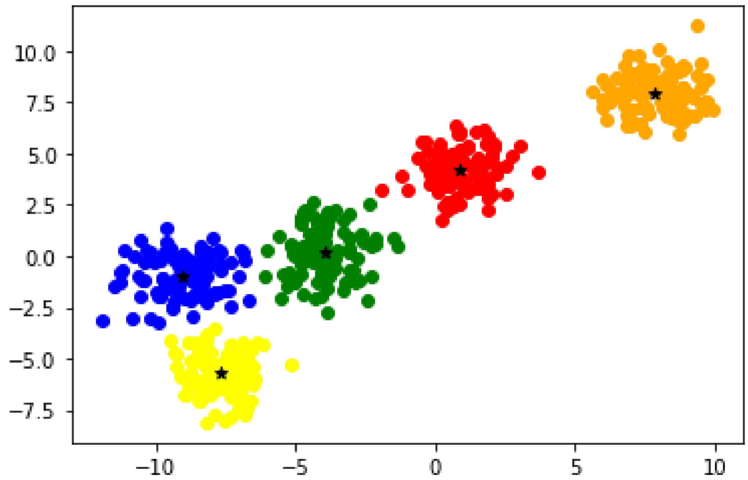

10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10

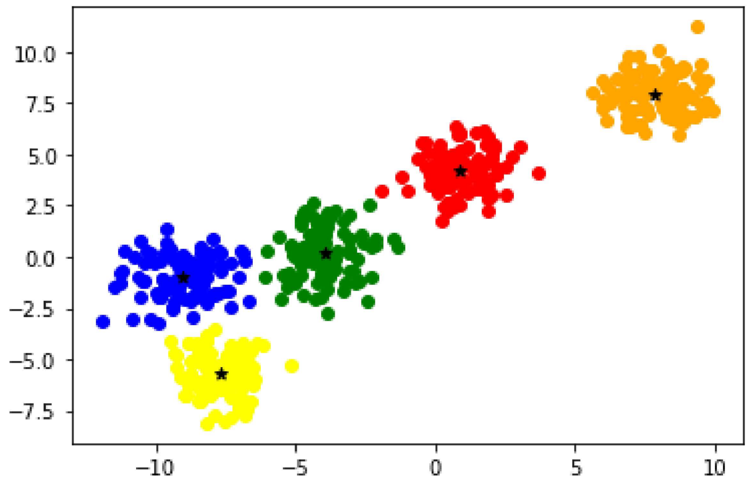

10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10


10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−5
0
5
10

In [ ]:

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
x_axis = np.arange(-100,100,0.1)
print(x_axis)
```

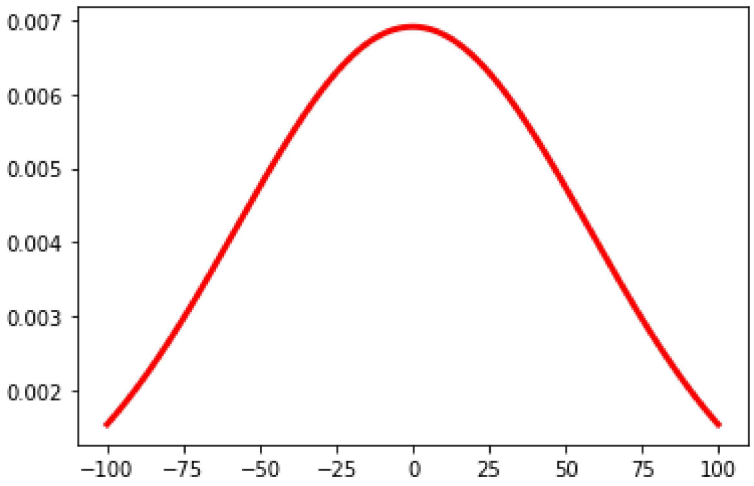
```
[-100.   -99.9  -99.8 ...   99.7   99.8   99.9]
```

In [ ]:
```
mean = np.mean(x_axis)
std = np.std(x_axis)
print(mean,std)
```

```
-0.05000000000567525 57.73501970208048
```

In [ ]:
```
y_axis = 1/(std * np.sqrt(2 * np.pi)) * np.exp( - (x_axis - mean)**2 / (2 * std**2
```

In [ ]:
```
plt.plot(x_axis,y_axis,linewidth=3, color='r')
plt.show()
```



In [ ]:
```
```

In [ ]:
```
!pip install graphviz
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
s/public/simple/
Requirement already satisfied: graphviz in /usr/local/lib/python3.9/dist-packages
(0.20.1)
```

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import export_graphviz
import graphviz
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:
```
iris = datasets.load_iris()
```

In [ ]:
```
X = iris.data
y = iris.target
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0, test_s
```

In [ ]:
```
tree = DecisionTreeClassifier(max_depth = 5)
tree.fit(X_train, y_train)
tree_predictions = tree.predict(X_test)
```

In [ ]:
```
cm = confusion_matrix(y_test, tree_predictions)
print(cm)
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

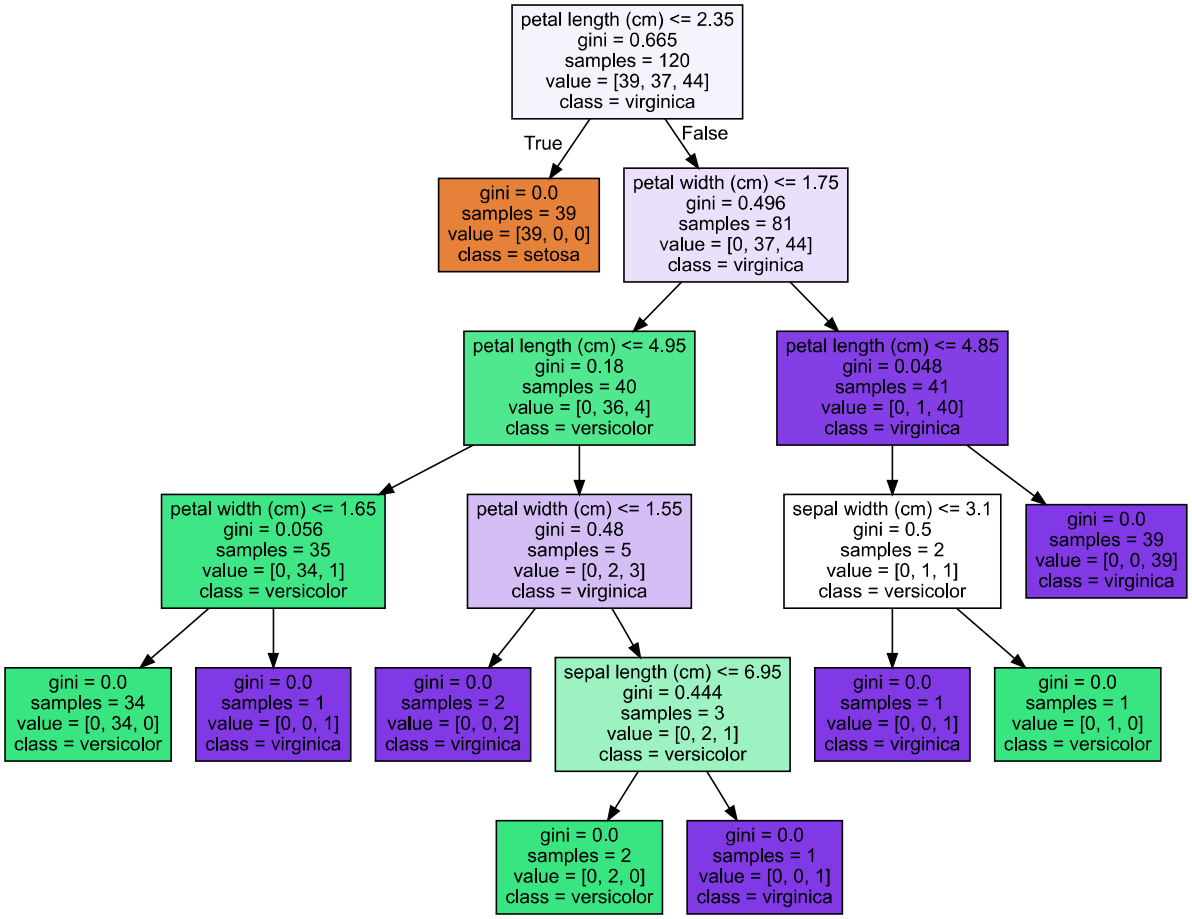
In [ ]:
```
score =accuracy_score(tree_predictions, y_test)
print(score)
```

```
1.0
```

In [ ]:
```
tree_formed = export_graphviz(tree, out_file = None, feature_names = iris.feature_
graph = graphviz.Source(tree_formed, format="png")
```

In [ ]:
```
graph
```

Out[ ]:



In [ ]:
```
print(type(graph))
```

```
<class 'graphviz.sources.Source'>
```

In [ ]:
```
graph.render(directory='doctest-output').replace('\\', '/')
```

Out[ ]:
```
'doctest-output/Source.gv.png'
```

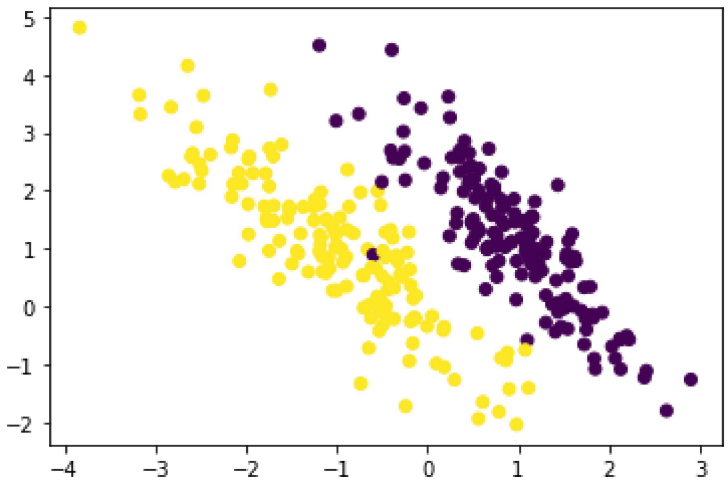
In [ ]:

Generate Dataset

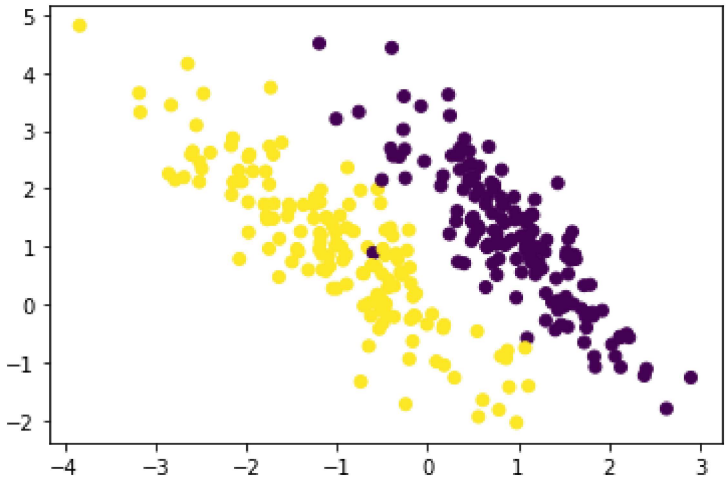
In [ ]:
```
from sklearn.datasets import  make_classification
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

In [ ]:
```
X,Y = make_classification(n_classes=2,n_samples=300,n_clusters_per_class=1,random_
```

In [ ]:
```
plt.scatter(X[:,0],X[:,1],c=Y)
plt.show()
```



In [ ]:
```
plt.scatter(X[:,0],X[:,1],c=Y)
plt.show()
```



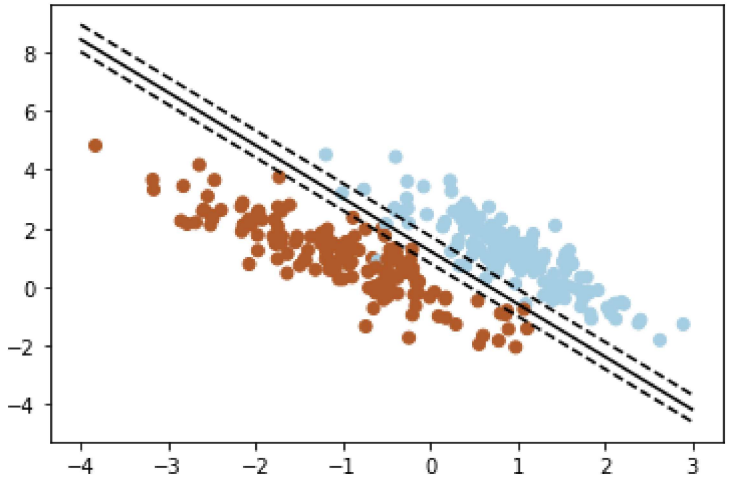
In [ ]:
```
from sklearn import svm
```

In [ ]:
```
svc = svm.SVC(kernel='linear')
svc.fit(X,Y)
print(svc.score(X,Y))
```

```
0.9966666666666667
```

In [ ]:
```
w = svc.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(-4, 3)
yy = a * xx - (svc.intercept_[0]) / w[1]
```

```
# plot the parallels to the separating hyperplane that pass through the
# support vectors
b = svc.support_vectors_[0]
yy_down = a * xx + (b[1] - a * b[0])
b = svc.support_vectors_[-1]
yy_up = a * xx + (b[1] - a * b[0])

# plot the line, the points, and the nearest vectors to the plane
plt.plot(xx, yy, 'k-')
plt.plot(xx, yy_down, 'k--')
plt.plot(xx, yy_up, 'k--')

plt.scatter(svc.support_vectors_[:, 0], svc.support_vectors_[:, 1],
            s=80, facecolors='none')
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)

plt.axis('tight')
plt.show()
```



In [ ]:

In [ ]:
```
import numpy as np
import math
from scipy.optimize import minimize
```

In [ ]:
```
mean = 10
std = 20
```

In [ ]:
```
s = np.random.normal(mean, std, 3000)
```

In [ ]:
```
def likelihood(mean, std, x):
    return (1 / math.sqrt(2 * math.pi * std**2)) * np.exp(-(x - mean)**2 / (2 * st

def log_likelihood(mean, std, data):
    return sum(np.log(likelihood(mean, std, x)) for x in data)
```

In [ ]:
```
neg_log_likelihood = lambda mean: -log_likelihood(mean, std, s)
```

In [ ]:
```
result = minimize(neg_log_likelihood, x0=0.0)
mean_mle = result.x[0]
print(mean_mle)
```

```
9.437499921115245
```

In [ ]:
```
print("Difference between original mean and new mean", mean-mean_mle)
```

```
Difference between original mean and new mean 0.5625000788847547
```

In [ ]:
```

```

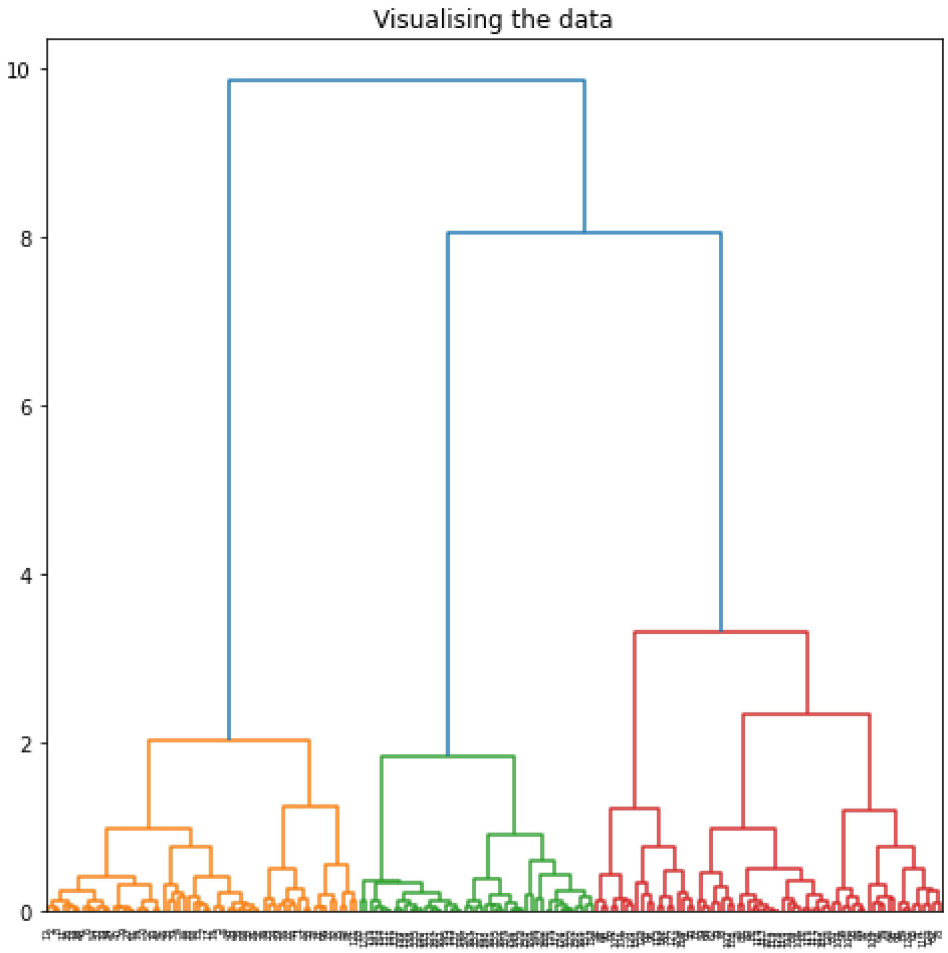
In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as shc
```

In [ ]:
```
X = pd.read_csv('wine-clustering.csv')

X.fillna(method ='ffill', inplace = True)
```

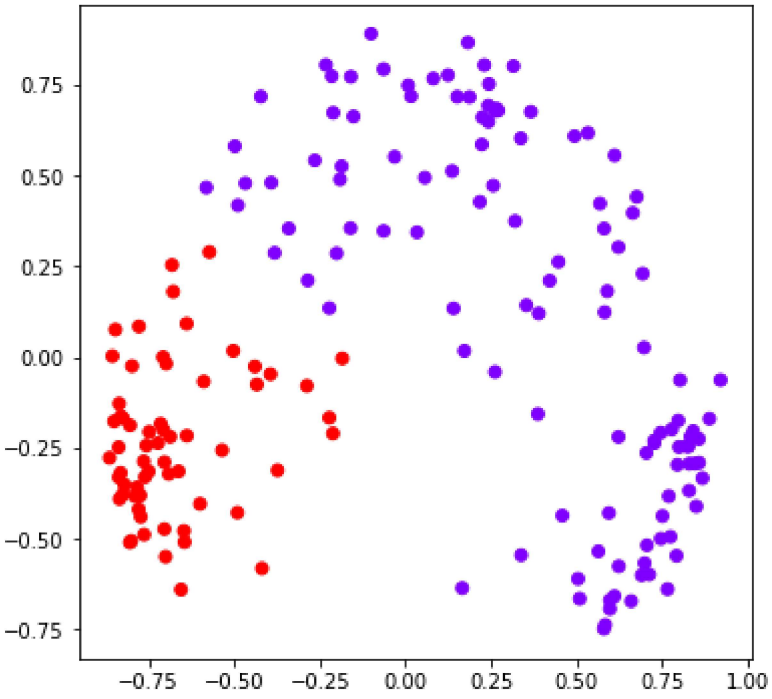
In [ ]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_normalized = normalize(X_scaled)

X_normalized = pd.DataFrame(X_normalized)
```

In [ ]:
```
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(X_normalized)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2']
```

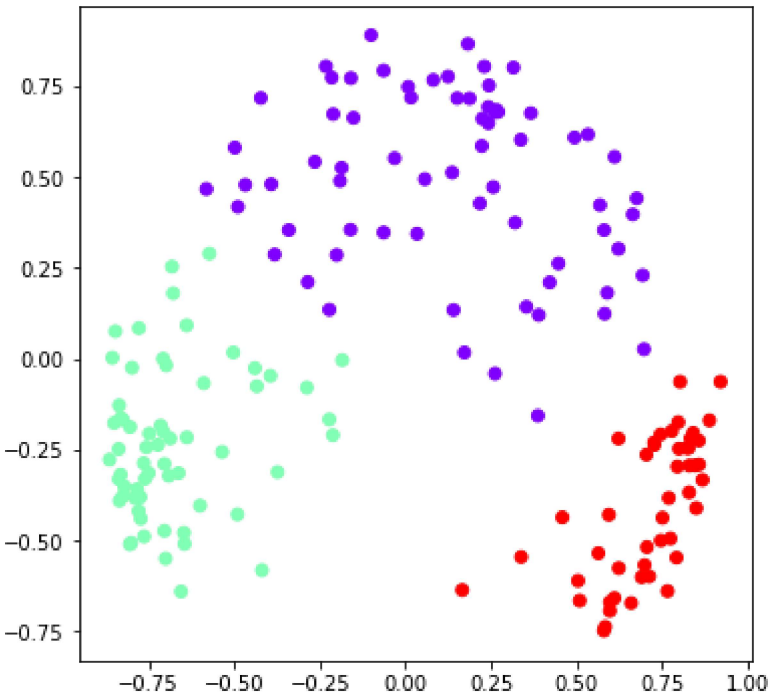
In [ ]:
```
plt.figure(figsize =(8, 8))
plt.title('Visualising the data')
Dendrogram = shc.dendrogram((shc.linkage(X_principal, method ='ward')))
```



In [ ]:
```
ac2 = AgglomerativeClustering(n_clusters = 2)

# Visualizing the clustering
plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
           c = ac2.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```

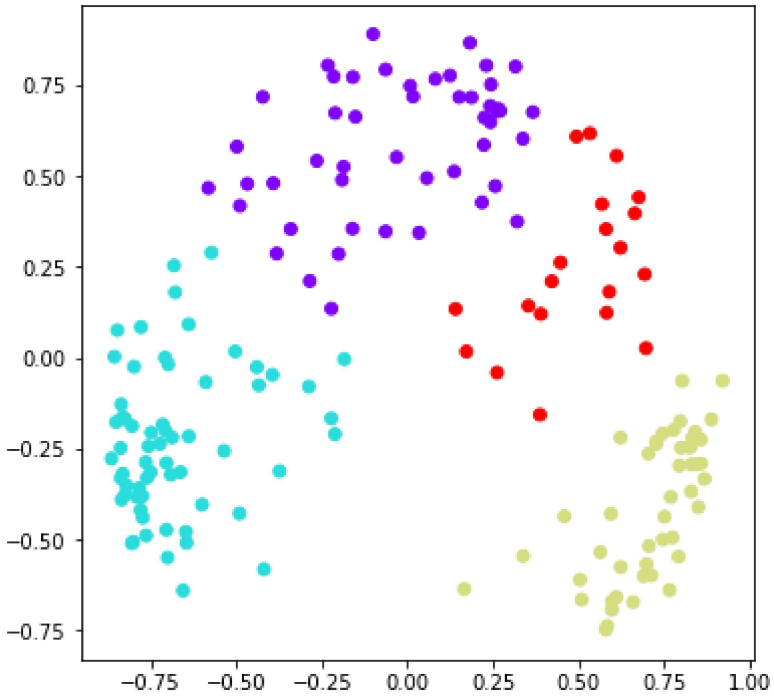


In [ ]:
```
ac3 = AgglomerativeClustering(n_clusters = 3)

plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
           c = ac3.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```

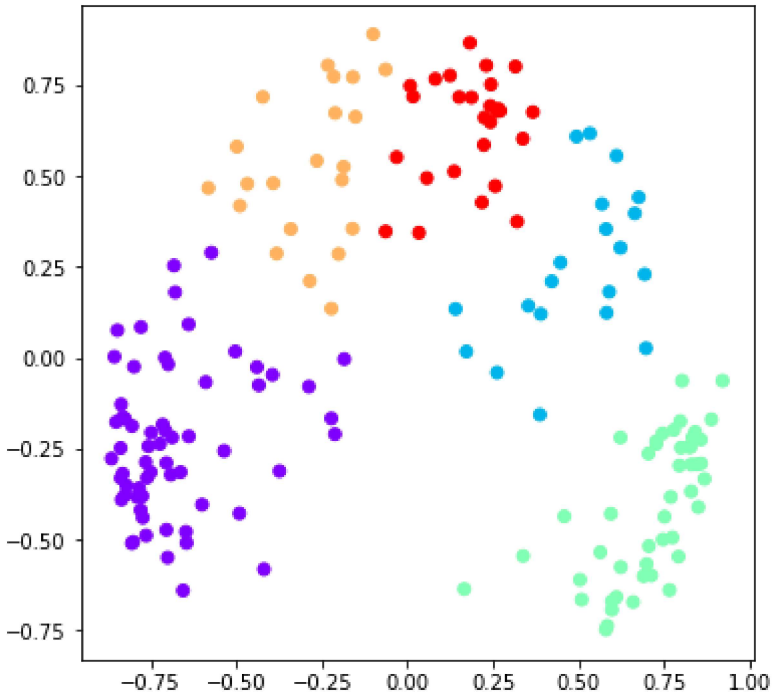


In [ ]:
```
ac4 = AgglomerativeClustering(n_clusters = 4)
```

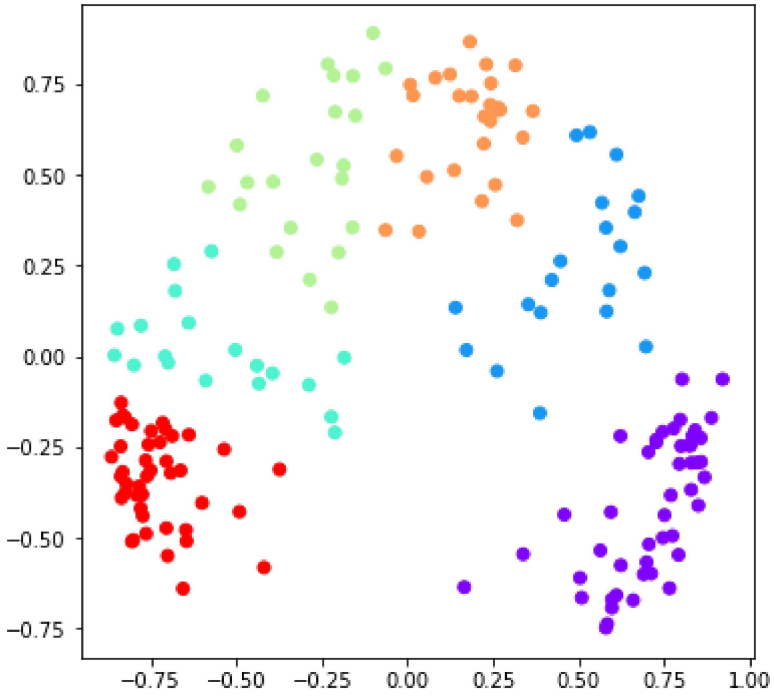
```
plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
            c = ac4.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```



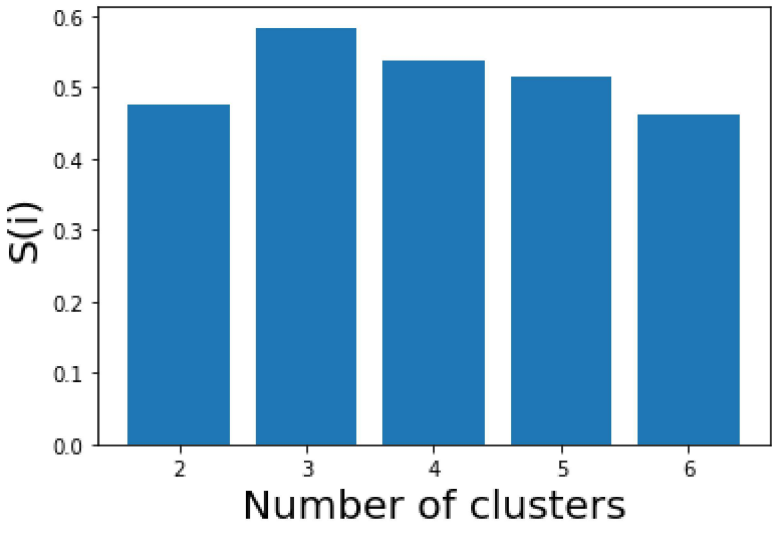
In [ ]:
```
ac5 = AgglomerativeClustering(n_clusters = 5)

plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
            c = ac5.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```



In [ ]:
```
ac6 = AgglomerativeClustering(n_clusters = 6)

plt.figure(figsize =(6, 6))
plt.scatter(X_principal['P1'], X_principal['P2'],
            c = ac6.fit_predict(X_principal), cmap ='rainbow')
plt.show()
```



In [ ]:
```
k = [2, 3, 4, 5, 6]

# Appending the silhouette scores of the different models to the list
silhouette_scores = []
silhouette_scores.append(
        silhouette_score(X_principal, ac2.fit_predict(X_principal)))
silhouette_scores.append(
        silhouette_score(X_principal, ac3.fit_predict(X_principal)))
silhouette_scores.append(
        silhouette_score(X_principal, ac4.fit_predict(X_principal)))
silhouette_scores.append(
        silhouette_score(X_principal, ac5.fit_predict(X_principal)))
silhouette_scores.append(
        silhouette_score(X_principal, ac6.fit_predict(X_principal)))

# Plotting a bar graph to compare the results
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 20)
plt.ylabel('S(i)', fontsize = 20)
plt.show()
```



In [ ]:

In [ ]:
```
!pip install clustimage
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
s/public/simple/
Collecting clustimage
  Downloading clustimage-1.5.14-py3-none-any.whl (37 kB)
Collecting ismember
  Downloading ismember-1.0.2-py3-none-any.whl (7.5 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (fr
om clustimage) (1.10.1)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.9/dist-packa
ges (from clustimage) (0.19.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packa
ges (from clustimage) (1.2.2)
Collecting distfit
  Downloading distfit-1.6.10-py3-none-any.whl (40 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.4/40.4 kB 2.1 MB/s eta 0:00:00
Collecting imagehash
  Downloading ImageHash-4.3.1-py2.py3-none-any.whl (296 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 296.5/296.5 kB 7.8 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (fr
om clustimage) (1.22.4)
Collecting pca
  Downloading pca-2.0.0-py3-none-any.whl (33 kB)
Collecting pypickle
  Downloading pypickle-1.1.0-py3-none-any.whl (5.1 kB)
Collecting umap-learn
  Downloading umap-learn-0.5.3.tar.gz (88 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 88.2/88.2 kB 6.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (fro
m clustimage) (4.65.0)
Collecting scatterd>=1.1.2
  Downloading scatterd-1.3.1-py3-none-any.whl (11 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages
(from clustimage) (2.27.1)
Collecting colourmap
  Downloading colourmap-1.1.11-py3-none-any.whl (8.1 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (f
rom clustimage) (1.5.3)
Collecting clusteval>=2.1.5
  Downloading clusteval-2.2.1-py3-none-any.whl (42 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.4/42.4 kB 2.1 MB/s eta 0:00:00
Requirement already satisfied: opencv-python in /usr/local/lib/python3.9/dist-pack
ages (from clustimage) (4.7.0.72)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-package
s (from clustimage) (3.7.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.9/dist-packages
(from clusteval>=2.1.5->clustimage) (0.12.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packages
(from distfit->clustimage) (23.1)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.9/dist-packag
es (from distfit->clustimage) (0.13.5)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packa
ges (from matplotlib->clustimage) (0.11.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-p
ackages (from matplotlib->clustimage) (3.0.9)
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python
3.9/dist-packages (from matplotlib->clustimage) (5.12.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-p
ackages (from matplotlib->clustimage) (1.0.7)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/di
st-packages (from matplotlib->clustimage) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-
packages (from matplotlib->clustimage) (4.39.3)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-pack
```

```
ages (from matplotlib->clustimage) (8.4.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-
packages (from matplotlib->clustimage) (1.4.4)
Requirement already satisfied: PyWavelets in /usr/local/lib/python3.9/dist-package
s (from imagehash->clustimage) (1.4.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packa
ges (from pandas->clustimage) (2022.7.1)
Collecting adjusttext
  Downloading adjustText-0.8-py3-none-any.whl (9.1 kB)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/d
ist-packages (from requests->clustimage) (1.26.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python
3.9/dist-packages (from requests->clustimage) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packa
ges (from requests->clustimage) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist
-packages (from requests->clustimage) (2022.12.7)
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.9/dist-pac
kages (from scikit-image->clustimage) (2.25.1)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.9/dis
t-packages (from scikit-image->clustimage) (2023.4.12)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.9/dist-pack
ages (from scikit-image->clustimage) (3.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/di
st-packages (from scikit-learn->clustimage) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-pack
ages (from scikit-learn->clustimage) (1.2.0)
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python3.9/dist-packag
es (from umap-learn->clustimage) (0.56.4)
Collecting pynndescent>=0.5
  Downloading pynndescent-0.5.10.tar.gz (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 31.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packag
es (from importlib-resources>=3.2.0->matplotlib->clustimage) (3.15.0)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python
3.9/dist-packages (from numba>=0.49->umap-learn->clustimage) (0.39.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-package
s (from numba>=0.49->umap-learn->clustimage) (67.7.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages
(from python-dateutil>=2.7->matplotlib->clustimage) (1.16.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.9/dist-packa
ges (from statsmodels->distfit->clustimage) (0.5.3)
Building wheels for collected packages: umap-learn, pynndescent
  Building wheel for umap-learn (setup.py) ... done
  Created wheel for umap-learn: filename=umap_learn-0.5.3-py3-none-any.whl size=82
830 sha256=b2389bf73a69af481f9e02b6da07ee19d1a9e044cb0bd2dae818d69db59eb32c
  Stored in directory: /root/.cache/pip/wheels/f4/3e/1c/596d0a463d17475af648688443
fa4846fef624d1390339e7e9
  Building wheel for pynndescent (setup.py) ... done
  Created wheel for pynndescent: filename=pynndescent-0.5.10-py3-none-any.whl size
=55640 sha256=fe9518719a7ebb2edd2d3af699d9e646daccf1ce87e6a8eccce46ceebce3b5fd
  Stored in directory: /root/.cache/pip/wheels/12/f9/4d/ec5ad1c823c710fcc4473669fd
cffc8891f4bc398c841af22e
Successfully built umap-learn pynndescent
Installing collected packages: pypickle, ismember, imagehash, pynndescent, colourm
ap, adjusttext, umap-learn, scatterd, distfit, pca, clusteval, clustimage
Successfully installed adjusttext-0.8 clusteval-2.2.1 clustimage-1.5.14 colourmap-
1.1.11 distfit-1.6.10 imagehash-4.3.1 ismember-1.0.2 pca-2.0.0 pynndescent-0.5.10
pypickle-1.1.0 scatterd-1.3.1 umap-learn-0.5.3
```

In [ ]:
```
from clustimage import Clustimage
```

In [ ]:
```
cl = Clustimage()
```

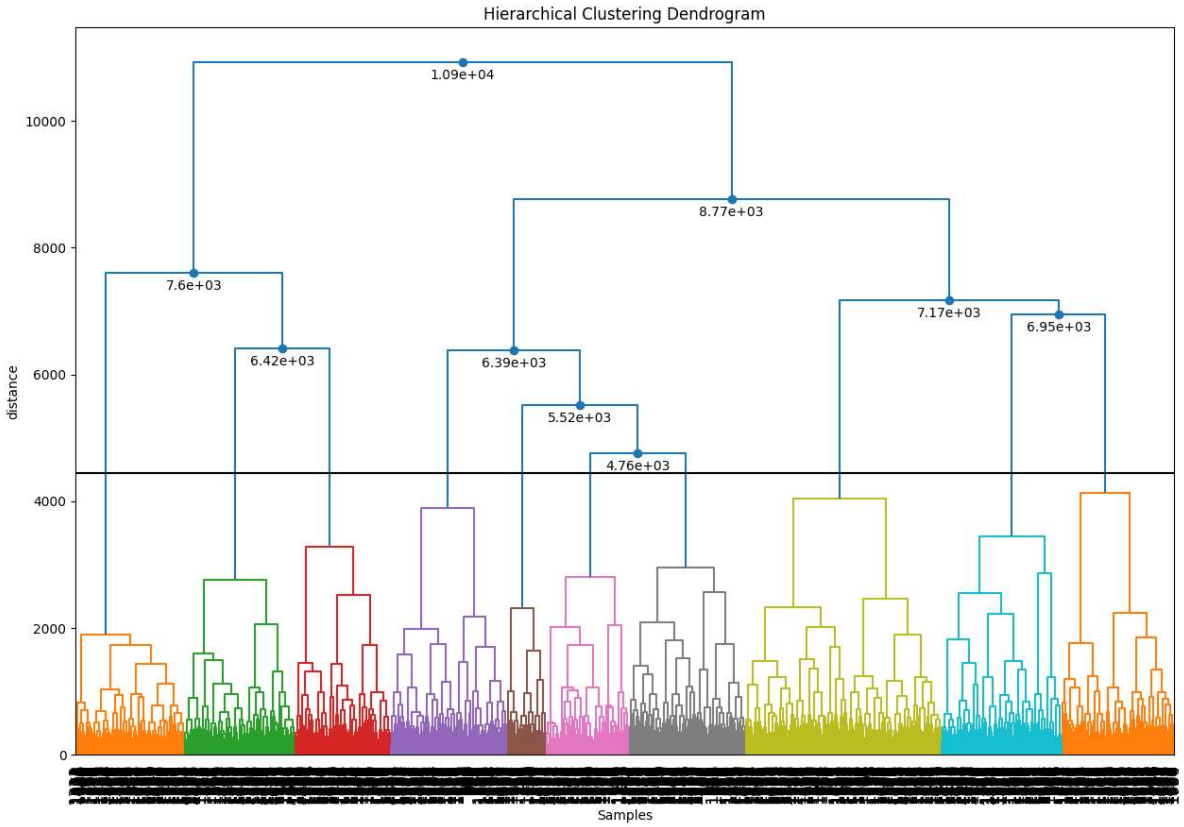
In [ ]:
```
X = cl.import_example(data='mnist')
```

In [ ]:
```
result = cl.fit_transform(X, cluster='agglomerative')
```

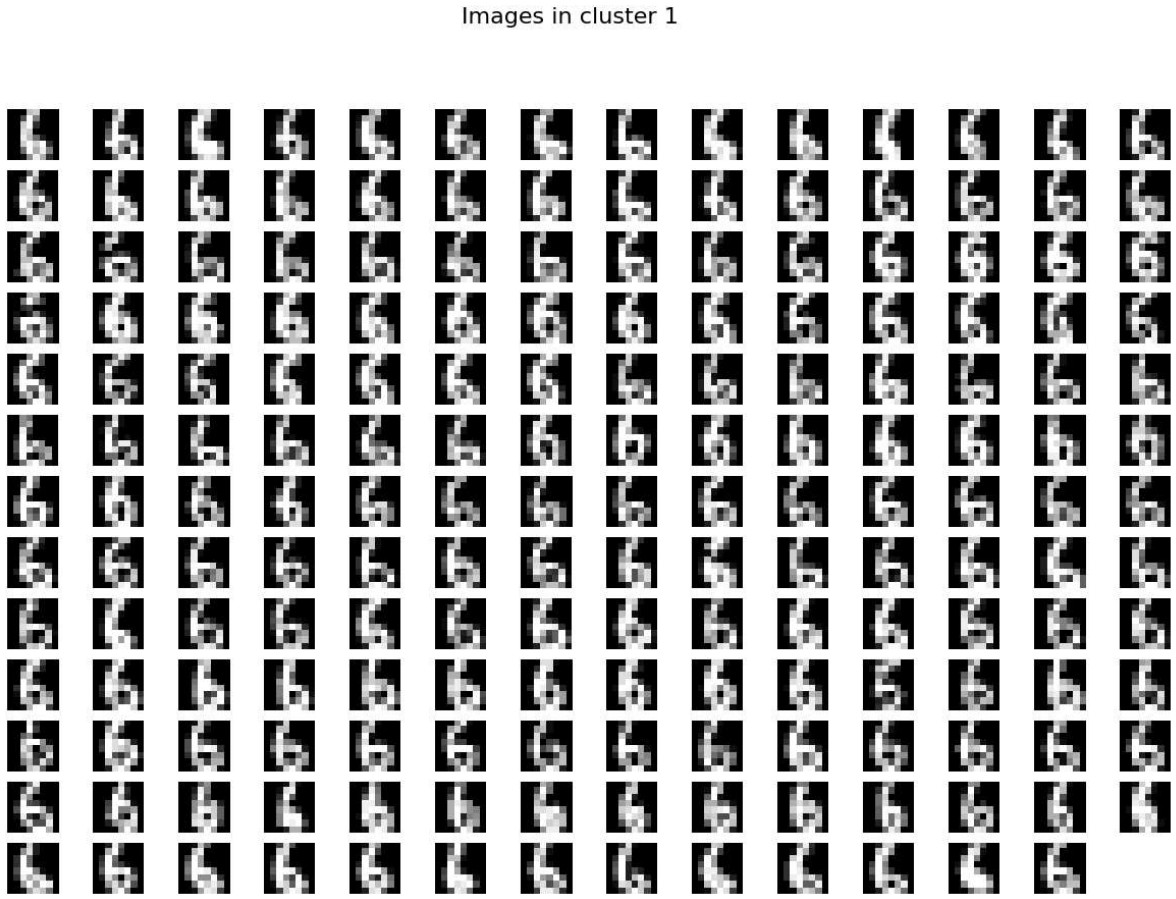
```
[clustimage] >INFO> Cleaning previous fitted model results
[clustimage] >INFO> Reading and checking images.
[clustimage] >INFO> Scaling images..
[clustimage] >INFO> Writing images to tempdir [/tmp/clustimage]
100%|██████████| 1797/1797 [00:00<00:00, 7590.95it/s]
[clustimage] >INFO> Extracting features using method: [pca]
[clustimage] >INFO> Extracted features using [pca]: samples=1797, features=29
[clustimage] >INFO> Compute [tsne] embedding
[pca] >Column labels are auto-completed.
[pca] >The PCA reduction is performed to capture [95.0%] explained variance using
the [64] columns of the input data.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[pca] >Compute explained variance.
[pca] >Number of components is [29] that covers the [95.00%] explained variance.
[pca] >The PCA reduction is performed on the [64] columns of the input dataframe.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[clustimage] >INFO> Cluster evaluation using the [high] feature space of the [pca]
features.
[clustimage] >INFO> Saving data in memory.
[clustimage] >INFO> Fit with method=[agglomerative], metric=[euclidean], linkage=
[ward]
[clustimage] >INFO> Evaluate using silhouette.
[clusteval] >INFO: 100%|██████████| 22/22 [00:02<00:00,  9.68it/s]
[clustimage] >DEBUG> [[ 1  2  2 ...  2  3  2]
 [ 1  3  3 ...  3  4  3]
 [ 1  3  3 ...  3  4  3]
 ...
 [ 1 11 13 ... 12 15 12]
 [ 1 11 13 ... 12 15 12]
 [ 1 11 13 ... 12 15 12]]
[clustimage] >DEBUG> Iloc: [ True  True  True  True  True  True  True  True  True
True  True  True
  True  True  True  True  True  True  True  True  True  True]
[clustimage] >DEBUG> silscores: [0.11822528 0.11539459 0.12517365 0.14390239 0.159
62764 0.17825663
 0.18850612 0.19173212 0.19102362 0.1875863  0.18588592 0.190128
 0.18680899 0.18696561 0.1871107  0.18639446 0.17413746 0.17526724
 0.17237909 0.17282404 0.16768274 0.16390601]
[clustimage] >DEBUG> sillclust: [ 3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 1
5. 16. 17. 18. 19. 20.
 21. 22. 23. 24.]
[clustimage] >DEBUG> clustlabx: [[ 1  2  2 ...  2  3  2]
 [ 1  3  3 ...  3  4  3]
 [ 1  3  3 ...  3  4  3]
 ...
 [ 1 11 13 ... 12 15 12]
 [ 1 11 13 ... 12 15 12]
 [ 1 11 13 ... 12 15 12]]
[clustimage] >INFO> Compute dendrogram threshold.
[clustimage] >INFO> Optimal number clusters detected: [10].
[clustimage] >INFO> Fin.
[clustimage] >INFO> Updating cluster-labels and cluster-model based on the (1797,
29) feature-space.
```

In [ ]:
```
cl.dendrogram()
```

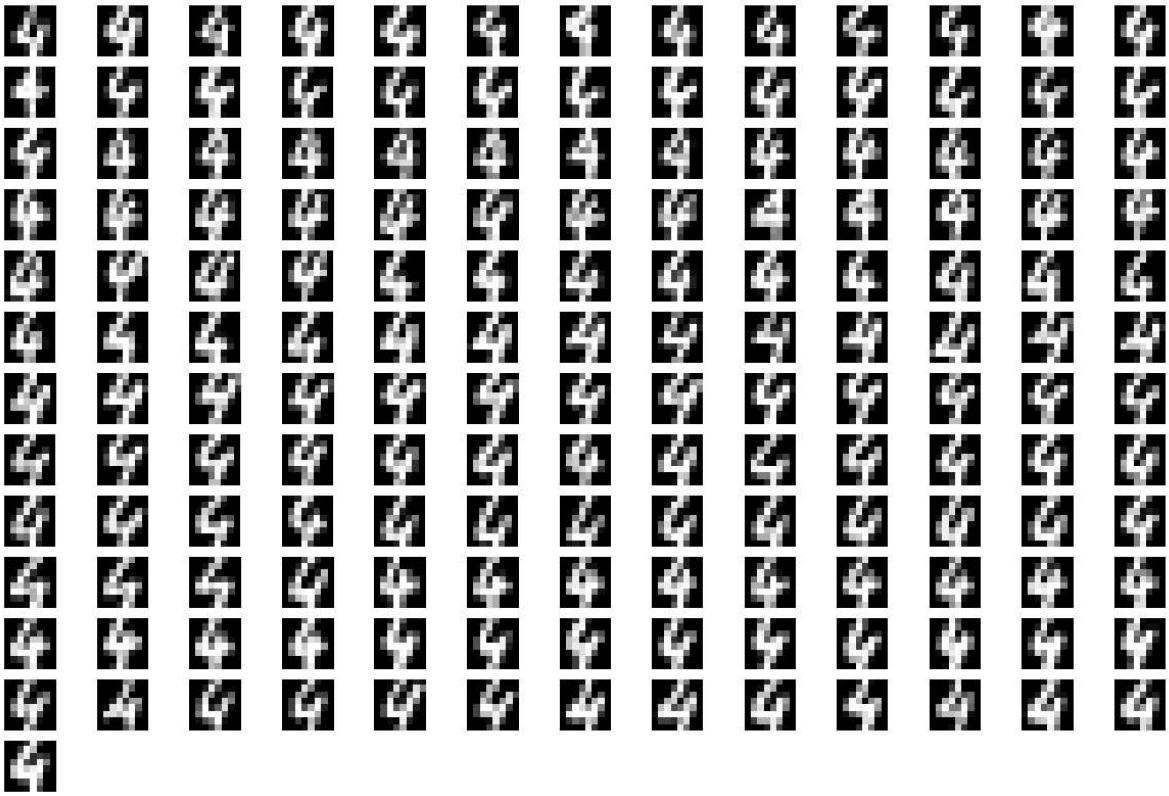
```
[clustimage] >INFO> Retrieving input data set.
[clustimage] >INFO> Plotting the dendrogram with optimized settings: metric=euclid
ean, linkage=ward, max_d=4445.959. Be patient now..
[clustimage] >INFO> Compute cluster labels.
```



In [ ]:
```
cl.plot(cmap='binary', labels=[1,2])
```



Images in cluster 2



In [ ]: