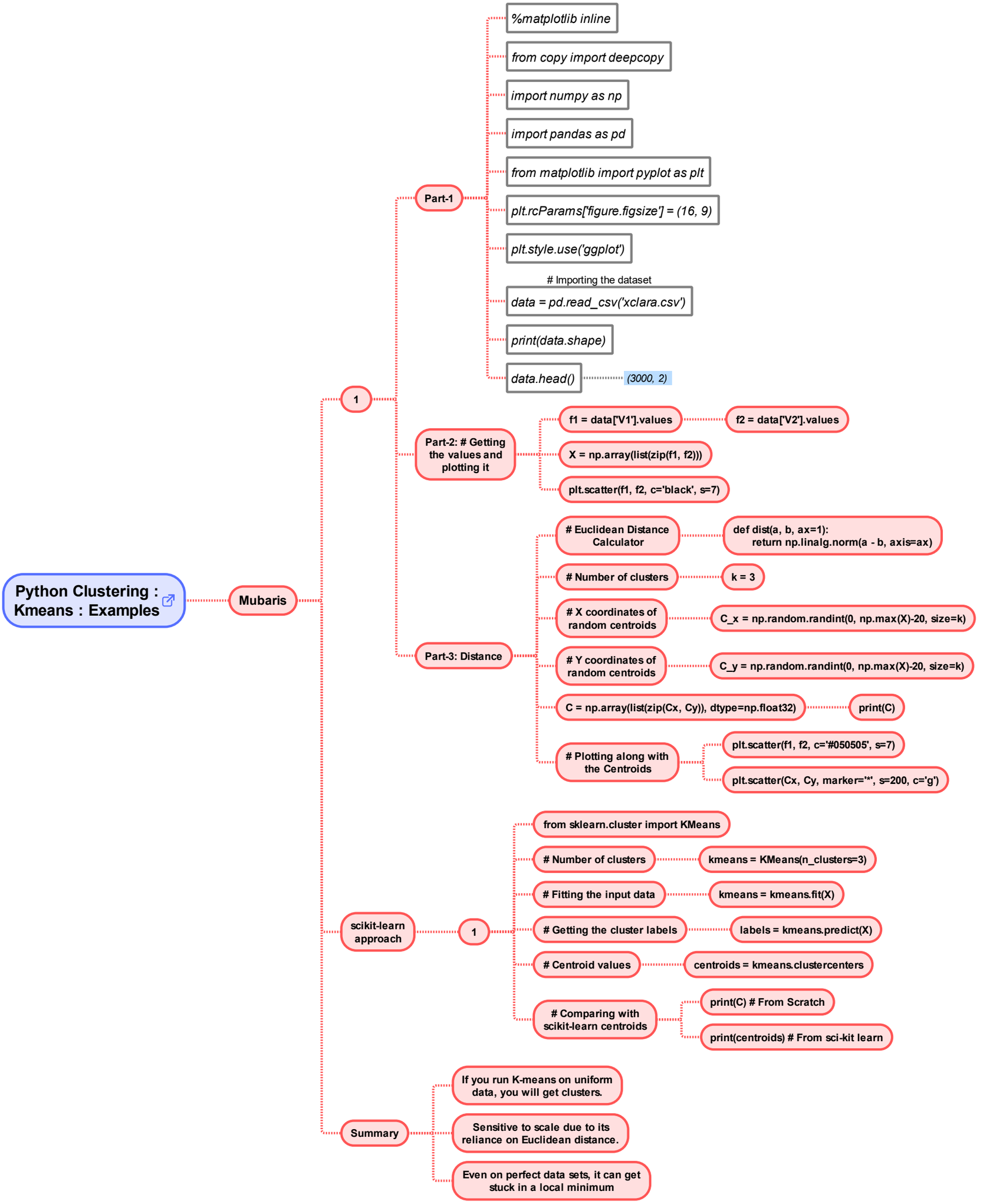
Python Clustering : Kmeans : Examples
Mubaris
1
Part-1
%matplotlib inline
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
# Importing the dataset
data = pd.read_csv('xclara.csv')
print(data.shape)
data.head()
(3000, 2)
Part-2: # Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)
Part-3: Distance
# Euclidean Distance Calculator
def dist(a, b, ax=1):
return np.linalg.norm(a - b, axis=ax)
# Number of clusters
k = 3
# X coordinates of random centroids
C_x = np.random.randint(0, np.max(X)-20, size=k)
# Y coordinates of random centroids
C_y = np.random.randint(0, np.max(X)-20, size=k)
C = np.array(list(zip(Cx, Cy)), dtype=np.float32)
print(C)
# Plotting along with the Centroids
plt.scatter(f1, f2, c='#050505', s=7)
plt.scatter(Cx, Cy, marker='*', s=200, c='g')
scikit-learn approach
1
from sklearn.cluster import KMeans
# Number of clusters
kmeans = KMeans(n_clusters=3)
# Fitting the input data
kmeans = kmeans.fit(X)
# Getting the cluster labels
labels = kmeans.predict(X)
# Centroid values
centroids = kmeans.clustercenters
# Comparing with scikit-learn centroids
print(C) # From Scratch
print(centroids) # From sci-kit learn
Summary
If you run K-means on uniform data, you will get clusters.
Sensitive to scale due to its reliance on Euclidean distance.
Even on perfect data sets, it can get stuck in a local minimum