Lab Session 4

«Preprocessing, Principle Component Analysis (PCA), k-means and Gaussian Model»

1. Introduction

In this lab session, you will train and test your dataset using methodes namely : Preprocessing, Principle Component Analysis, k-means and Gaussian Model.

2. Learning outcome

On successful completion of this lab session you will be able to :

- Preprocessing
- Understand and implement Principle Component Analysis (PCA)
- Understand and implement k-means
- Understand and implement Gaussian Model

3. Ressources

Libraries Documentation

- Python : https://docs.python.org/3/
- NumPy : https://numpy.org/doc/
- SciPy : https://docs.scipy.org/doc/scipy/
- Matplotlib : https://matplotlib.org/3.5.1/
- Panda : https://pandas.pydata.org/docs/
- mglearn : https://libraries.io/pypi/mglearn
- sklearn : https://scikit-learn.org/stable/

**to proceed with the lab session, refer to this section*

1. Preprocessing

Import the following dataset :

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv('https://raw.githubusercontent.com/fivethirtyeight/data/master/airline-safety/airline-safety.csv')
data = df[['incidents_00_14', 'fatal_accidents_00_14']].to_numpy()
```

You can see visualize the dataset using this code :

```
def plot_data(data):
  fig, ax = plt.subplots(1,1)
  plt.grid()
  plt.scatter(data[:,0], data[:,1])
  ax.set_xlabel('Total number of incidents, 2000-2014')
  ax.set_ylabel('Total number of fatal accidents, 2000-2014')
  ax.set_title('Incidents and fatal accidents for differents airlines')
  plt.show()

plot_data(data)
```

Exercise :
Using the sklearn documentation :

1. Apply a standard scaling to the dataset.
   If you visualize the data, all the plot seams to be at the same place but if you look at the axis, you will see that they are scaled accordingly.
2. Apply a robust scaling to the dataset.
3. Apply a MinMax scaling to the dataset.
4. Normalize the dataset.

2. Principle Component Analysis (PCA)

in this tutorial we will use Sklearn breast cancer dataset. First, we import the essential libraries,load the dataset and seperate features from targets.

```
from sklearn.datasets import load_breast_cancer
import numpy as np
import pandas as pd

breastCancer=load_breast_cancer()

X=breastCancer.data
y=breastCancer.target
```

Then we standardize features by applying StanderScaler function. The goal here, is to normally redistribute the features over the origin.

```
from sklearn.preprocessing import StandardScaler

X=StandardScaler().fit_transform(X)
```

Now, we project the breast cancer dataset into a two-dimensional principal components. the n_components variable represent the targeted dimension in which we project our dataset.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

pca_breastcancer=pca.fit_transform(X)
```

Next, we create a Dataframe gathering PCA1, PCA2 and the labels of the targets, which will be replaced with their string representative to ease data visualisation

```
breastCancerDf = pd.DataFrame(data = pca_breastcancer, columns = ['principal component 1', 'principal component 2'])

breastCancerDf['label'] = breastCancer.target.tolist()
breastCancerDf['label'].replace(0, 'benign',inplace=True)
breastCancerDf['label'].replace(1, 'malignant',inplace=True)

breastCancerDf
```

Finally, we visualize the resulted dataframe

```
import matplotlib.pyplot as plt

plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Principal Component - 1',fontsize=20)
plt.ylabel('Principal Component - 2',fontsize=20)
plt.title("Principal Component Analysis of Breast Cancer Dataset",fontsize=20)
targets = breastCancer.target_names
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = breastCancerDf['label'] == target
    plt.scatter(breastCancerDf.loc[indicesToKeep, 'principal component 1']
               , breastCancerDf.loc[indicesToKeep, 'principal component 2'], c = color, s = 50)

plt.legend(targets,prop={'size': 15})
```

We can observe that the two classes can be linearly separable when projected to a two-dimensional space.

Exercice 1

Water potability is determined from various parameters such as ph, conductivity and chloramines. The following dataset contains records of potability test by analysing 9 parameters.Link : https://www.kaggle.com/datasets/adityakadiwal/water-potability

Are these parameters reliable to determine whether the water is potable or not ?

3. K-means clustering

In this tutorial we will identify different clusters of random blobs

standard imports

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()  # for plot styling
import numpy as np
```

Let's create 6 distinct blobs

```
from sklearn.datasets import make_blobs
X, y_true = make_blobs(n_samples=300, centers=6, cluster_std=0.50, random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50);
```

These clusters can be easily identified by eye. However, it is tricky to determine them automatically. We suppose that we want 6 clusters

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=6)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

our 6 clusters are visulized as follow

```
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```

In the majority of the cases, it is hard to predict the number of the cluster. However, we can determine the number of clusters using the elbow method.

```
from yellowbrick.cluster import KElbowVisualizer
kmeans=KMeans()
visualizer = KElbowVisualizer(kmeans, k=(4,12))
visualizer.fit(X)        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
```

Exercise 1

Using K-means clustering, divide and display each subset of the digits dataset. where do we have confusion ?

digit dataset : https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html

4. Gaussian model

Start by importing the dataset :

```
import pandas as pd
import matplotlib.pyplot as plt


df = pd.read_csv('https://raw.githubusercontent.com/BBUCHI/DS54/main/Clustering_gmm.csv')
```

```
plt.figure(figsize=(7,7))
plt.scatter(df["Weight"],df["Height"])
plt.xlabel('Weight')
plt.ylabel('Height')
plt.title('Data Distribution')
plt.show()
```

Exercise :

1. With the help of the sklearn documentation for GaussianMixture, fit a gaussian mixture model to the dataset.
2. Display the dataset once fitted.