Bahria University, Islamabad Campus
Department of Computer Sciences
Artificial Intelligence
Assignment-3
Class: BSCS-6A/6B

Deadline: 19th December 2021

Name	Enrollment
Tauheed Butt	01-134191-068
Mubeen Ahmed	01-134191-018

Objectives:

To understate the concept of dimensionality reduction by applying the Principal Component Analysis (PCA) Technique. PCA reduces high dimensional data to lower dimensions while capturing maximum variability of the dataset. The main steps of the PCA Algorithm are as follows:

1. Compute mean of each variable and subtract it from the dataset to centre data on the origin.
2. Calculate the Covariance Matrix.
3. Compute the Eigenvalues and the Eigenvectors.
4. Sort Eigenvalues in descending order.
5. Select a subset from the rearranged Eigenvalue matrix (i.e. number of components e.g. P=2).
6. Transform the data by computing a dot product between the **Transpose** of the Eigenvector subset and the **Transpose** of the mean-centered data. By transposing the outcome of the dot product, the result will be the data reduced to lower dimensions.

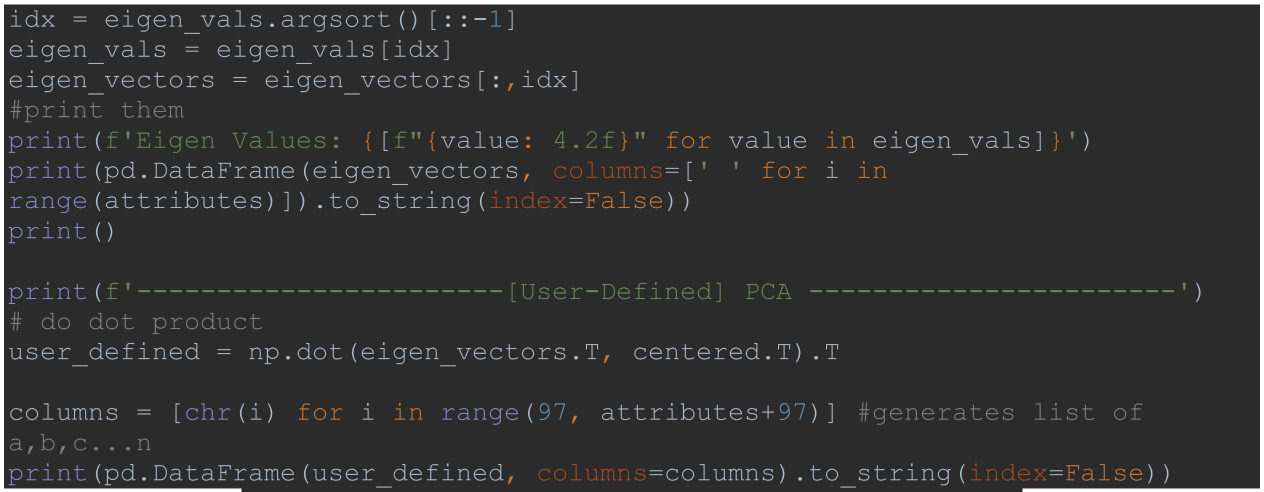
Task-1:

Implement each step of the PCA algorithm from scratch and apply it on either an existing dataset like “IRIS” or “MNIST” etc., or generate a dummy data by using the given command (it generates 6 variables for 10 samples, you can change as per your requirement).

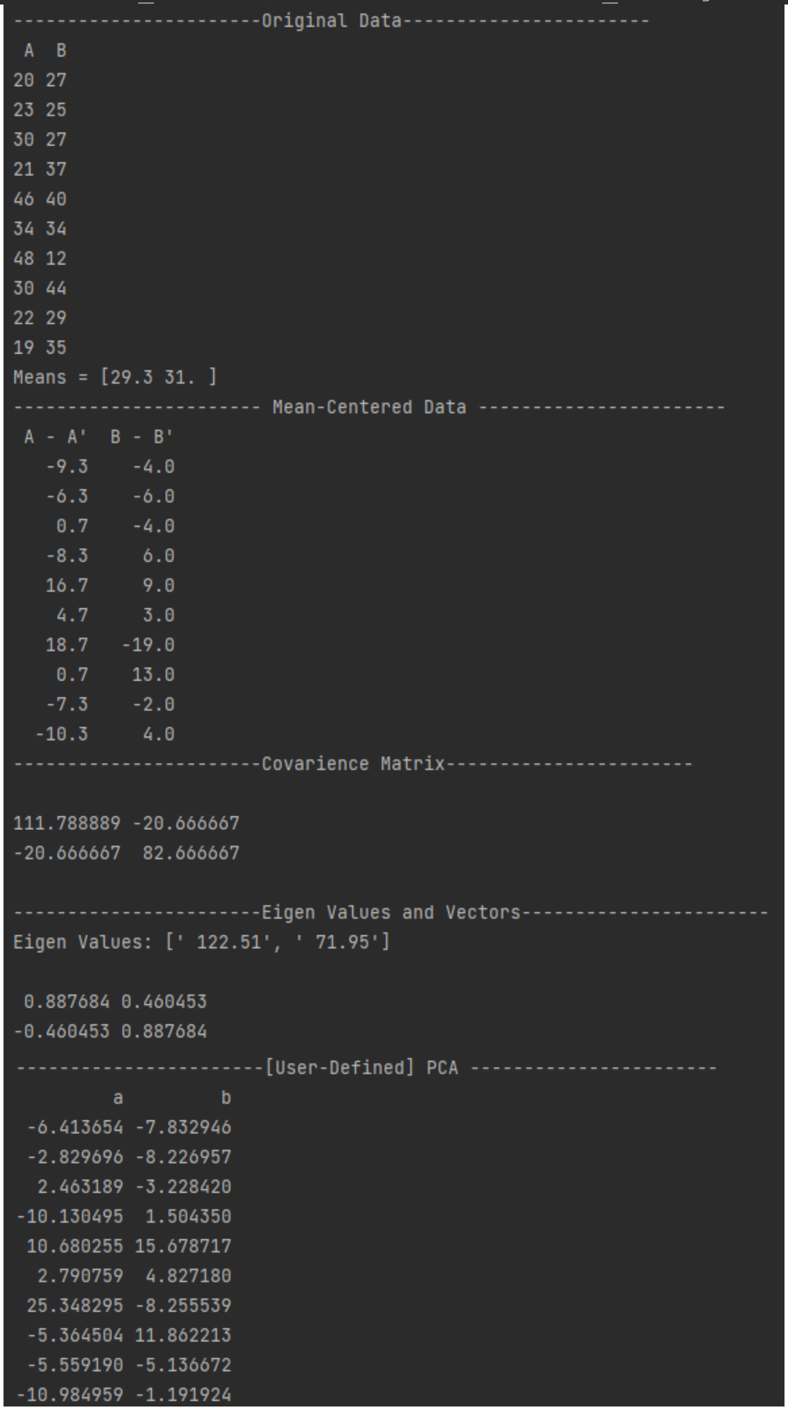
```
X = np.random.randint(10,50,100).reshape(10,6)
```

*NOTE: Do not use inbuilt function PCA() for this task, however, you can use inbuilt functions for implementing the steps. Show the output of each step as well.

```
from numpy import linalg as LA
import numpy as np
import pandas as pd
import random
from sklearn.datasets import load_iris


# attributes = int(input('Enter Total Attributes: '))
# samples = int(input('Enter Total Samples: '))

attributes = 2
samples = 10

# Load Iris dataset
# max attributes with iris = 4
# table = load_iris().data[:samples, :attributes]

# generating table
table = np.random.randint(10, 50, attributes*samples).reshape(samples,
attributes)

# calculate mean
means = np.mean(table, axis=0)

# subtract mean from each attribute
centered = table-means

print('-----------------------Original Data-----------------------')
columns = [chr(i+1) for i in range(64, (attributes-1)+65)] #generates list of
A,B,C...n
print(pd.DataFrame(table, columns=columns).to_string(index=False))
print(f'Means = {means}')


print('----------------------- Mean-Centered Data -----------------------')
columns = [f"{chr(i + 1)} - {chr(i + 1)}'" for i in range(64, (attributes-
1)+65)] #generates list of A-A',B-B',C-C'...n
print(pd.DataFrame(centered, columns=columns).to_string(index=False))

# calculate covariance matrix
print('-----------------------Covarience Matrix-----------------------')
cov_matrix = np.cov(centered.T)
print(pd.DataFrame(cov_matrix, columns=[' ' for i in
range(attributes)]).to_string(index=False))
print()

print('-----------------------Eigen Values and Vectors----------------------
')
# find eigen values and eigen vectors
eigen_vals, eigen_vectors = LA.eig(np.array(cov_matrix))
# sort eigen values and eigen vectors
```



```python
idx = eigen_vals.argsort()[::-1]
eigen_vals = eigen_vals[idx]
eigen_vectors = eigen_vectors[:,idx]
#print them
print(f'Eigen Values: {[f"{value: 4.2f}" for value in eigen_vals]}')
print(pd.DataFrame(eigen_vectors, columns=[' ' for i in
range(attributes)]).to_string(index=False))
print()

print(f'-----------------------[User-Defined] PCA -----------------------')
# do dot product
user_defined = np.dot(eigen_vectors.T, centered.T).T

columns = [chr(i) for i in range(97, attributes+97)] #generates list of
a,b,c...n
print(pd.DataFrame(user_defined, columns=columns).to_string(index=False))
```

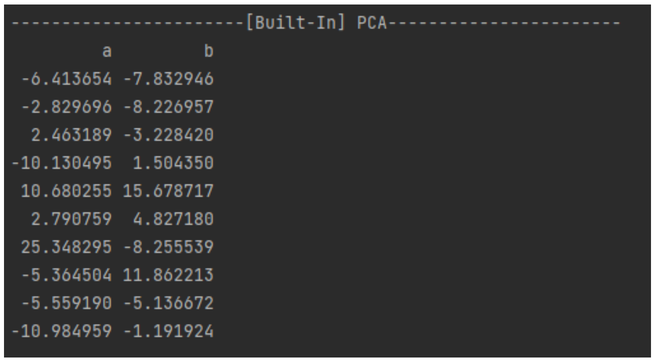


```
-----------------------Original Data-----------------------
 A  B
20 27
23 25
30 27
21 37
46 40
34 34
48 12
30 44
22 29
19 35
Means = [29.3 31. ]
----------------------- Mean-Centered Data -----------------------
 A - A'  B - B'
   -9.3    -4.0
   -6.3    -6.0
    0.7    -4.0
   -8.3     6.0
   16.7     9.0
    4.7     3.0
   18.7   -19.0
    0.7    13.0
   -7.3    -2.0
  -10.3     4.0
-----------------------Covarience Matrix-----------------------

111.788889 -20.666667
-20.666667  82.666667

-----------------------Eigen Values and Vectors-----------------------
Eigen Values: [' 122.51', ' 71.95']

 0.887684 0.460453
-0.460453 0.887684
-----------------------[User-Defined] PCA -----------------------
         a         b
 -6.413654 -7.832946
 -2.829696 -8.226957
  2.463189 -3.228420
-10.130495  1.504350
 10.680255 15.678717
  2.790759  4.827180
 25.348295 -8.255539
 -5.364504 11.862213
 -5.559190 -5.136672
-10.984959 -1.191924
```

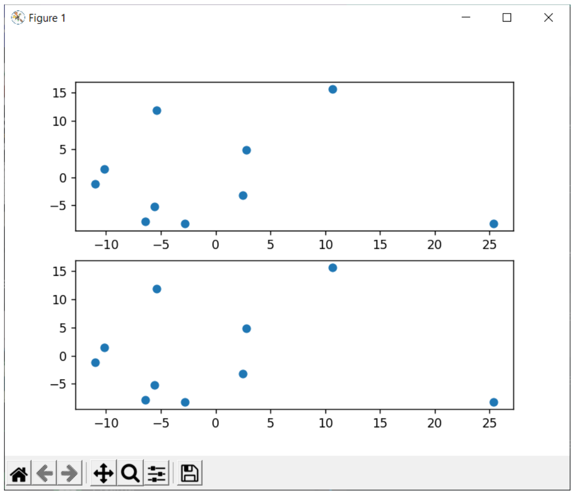
Task-2:

In the second task, use the same dataset employed in the previous task and apply the inbuilt function **PCA()** on it. Visualize the final output of both tasks to compare the working of the user-defined and inbuilt PCA functions.

```
import pandas as pd

from task_1 import *
import matplotlib.pyplot as plt

import numpy as np
from sklearn.decomposition import PCA


print(f'----------------------[Built-In] PCA----------------------')
pca = PCA(n_components=attributes)
built_in = pca.fit_transform(table)

print(pd.DataFrame(built_in, columns=columns).to_string(index=False))


# display graph
fig = plt.figure()
sub1 = fig.add_subplot(211)
sub2 = fig.add_subplot(212)

sub1.scatter(user_defined[:, 0], user_defined[:, 1])
sub2.scatter(built_in[:, 0], built_in[:, 1])

plt.show()
```



```
----------------------[Built-In] PCA----------------------
         a          b
 -6.413654 -7.832946
 -2.829696 -8.226957
  2.463189 -3.228420
-10.130495  1.504350
 10.680255 15.678717
  2.790759  4.827180
 25.348295 -8.255539
 -5.364504 11.862213
 -5.559190 -5.136672
-10.984959 -1.191924
```



Submission Requirements:

This is a group assignment, but no more than two members are allowed in a group. The same groups should be made as were in the previous assignment. **Python code** should be **executable** (should be copy-pasted from source file and not included as screen shot). **Screen shots of output** should be included. All by-hand work should be neat and comprehensible. Upload softcopies in LMS individually, however, group member names should be clearly mentioned in each assignment.

Submission Deadline:

19th December 2021