Experiment 2 Implementation of Pca

Implementation of PCA

```
import pandas as pd
df = pd.read_csv("/content/IRIS_dataset (1).csv")
df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

```
# Part A: Compute Eigen Values and Vectors
```

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
import numpy as np
import matplotlib.pyplot as plt
X =df[['sepal_length','sepal_width','petal_length','petal_width']]
```

```
X.head()
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
Y = df[['species']]
Y.head()
```

	species
0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa

```
df.shape
```

```
(150, 5)
```

```
X.shape
```

```
(150, 4)
```

```
Y.shape
```

```
(150, 1)
```

```
# Compute the covariance matrix
features = X.T
features.shape
```

```
(4, 150)
```

```
covariance_matrix = np.cov(features)
covariance_matrix
```

```
array([[ 0.68569351, -0.03926846,  1.27368233,  0.5169038 ],
       [-0.03926846,  0.18800403, -0.32171275, -0.11798121],
       [ 1.27368233, -0.32171275,  3.11317942,  1.29638747],
       [ 0.5169038 , -0.11798121,  1.29638747,  0.58241432]])
```

```
eig_vals, eig_vecs = np.linalg.eig(covariance_matrix)
```

```
eig_vals
```

```
array([4.22484077, 0.24224357, 0.07852391, 0.02368303])
```

```
eig_vecs
```

```
array([[ 0.36158968, -0.65653988, -0.58099728,  0.31725455],
       [-0.08226889, -0.72971237,  0.59641809, -0.32409435],
       [ 0.85657211,  0.1757674 ,  0.07252408, -0.47971899],
       [ 0.35884393,  0.07470647,  0.54906091,  0.75112056]])
```

```
eig_vals[0]/sum(eig_vals)
```

```
0.9246162071742685
```

```
projected_X = X.dot(eig_vecs.T[0])
```

```
projected_X
```

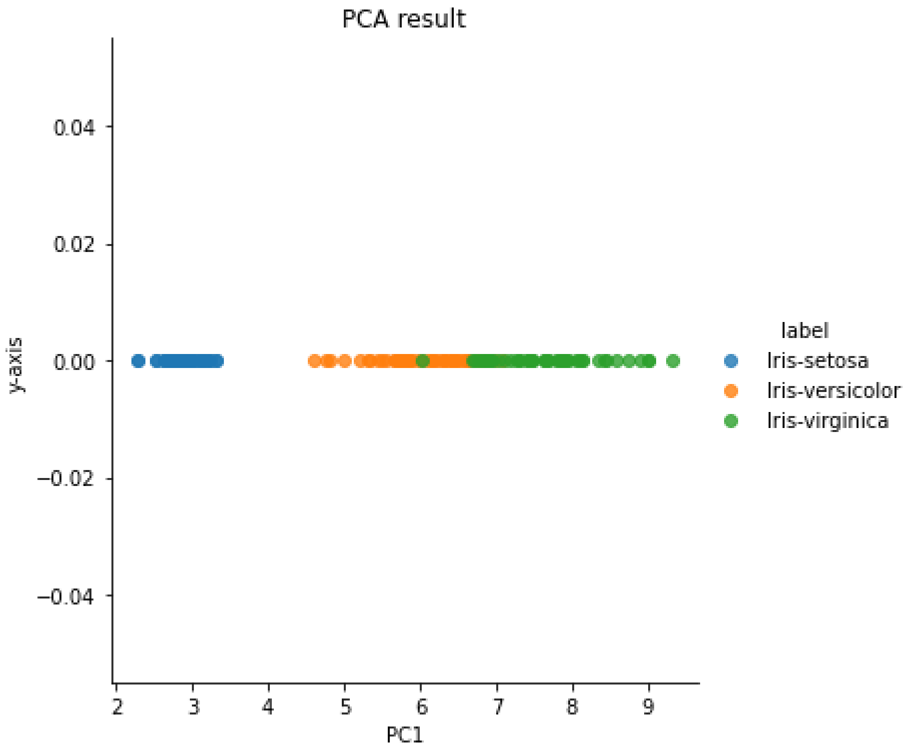
```
0      2.827136
1      2.795952
2      2.621524
3      2.764906
4      2.782750
         ...
145    7.455360
146    7.037007
147    7.275389
148    7.412972
149    6.901009
Length: 150, dtype: float64
```

```
result = pd.DataFrame(projected_X,columns=['PC1'])
result['y-axis']=0.0
result['label']= Y
result
```

	PC1	y-axis	label
0	2.827136	0.0	Iris-setosa
1	2.795952	0.0	Iris-setosa
2	2.621524	0.0	Iris-setosa
3	2.764906	0.0	Iris-setosa

```
import seaborn as sns
sns.lmplot('PC1', 'y-axis', data =result, fit_reg = False, hue = 'label')
plt.title("PCA result")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
  FutureWarning
Text(0.5, 1.0, 'PCA result')
```



Part B: Face Recognition using PCA

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_lfw_people
from sklearn.metrics import classification_report
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier
```

```
# Load the dataset
lfw_dataset = fetch_lfw_people(min_faces_per_person=100)
_, h, w = lfw_dataset.images.shape
X = lfw_dataset.data
```

```
Y = lfw_dataset.target
target_names = lfw_dataset.target_names
```

```
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
```

```
# Split the data into tarin and test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
```

```
X_train.shape
```

```
(912, 2914)
```

```
X.shape
```

```
(1140, 2914)
```

```
#compute PCA
n_components = 80
pca = PCA(n_components=n_components, whiten=True).fit(X_train)
```

```
#apply PCA transformation
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

```
X_train_pca.shape
```

```
(912, 80)
```

```
#Train a neural network on the transformed dataset
clf=MLPClassifier(hidden_layer_sizes=(512,),batch_size=256,verbose=True,early_stopping=Tru
```

```
Iteration 1, loss = 1.64811335
Validation score: 0.532609
Iteration 2, loss = 1.26218045
Validation score: 0.641304
Iteration 3, loss = 1.04980538
Validation score: 0.673913
Iteration 4, loss = 0.89809513
Validation score: 0.717391
Iteration 5, loss = 0.76291759
Validation score: 0.760870
Iteration 6, loss = 0.64634407
Validation score: 0.771739
Iteration 7, loss = 0.55256239
Validation score: 0.815217
Iteration 8, loss = 0.47807918
```

```
Validation score: 0.815217
Iteration 9, loss = 0.41929237
Validation score: 0.836957
Iteration 10, loss = 0.37042709
Validation score: 0.891304
Iteration 11, loss = 0.32853155
Validation score: 0.891304
Iteration 12, loss = 0.29322237
Validation score: 0.891304
Iteration 13, loss = 0.26232305
Validation score: 0.891304
Iteration 14, loss = 0.23548858
Validation score: 0.880435
Iteration 15, loss = 0.21268101
Validation score: 0.891304
Iteration 16, loss = 0.19316507
Validation score: 0.902174
Iteration 17, loss = 0.17557182
Validation score: 0.902174
Iteration 18, loss = 0.15969277
Validation score: 0.913043
Iteration 19, loss = 0.14602423
Validation score: 0.913043
Iteration 20, loss = 0.13392562
Validation score: 0.913043
Iteration 21, loss = 0.12321120
Validation score: 0.913043
Iteration 22, loss = 0.11375854
Validation score: 0.913043
Iteration 23, loss = 0.10502117
Validation score: 0.913043
Iteration 24, loss = 0.09716044
Validation score: 0.913043
Iteration 25, loss = 0.09044732
Validation score: 0.913043
Iteration 26, loss = 0.08438528
Validation score: 0.913043
Iteration 27, loss = 0.07876971
Validation score: 0.913043
Iteration 28, loss = 0.07334918
Validation score: 0.902174
Iteration 29, loss = 0.06859759
Validation score: 0.902174
```

```
Y_pred = clf.predict(X_test_pca)
print(classification_report(Y_test, Y_pred, target_names=target_names))
```

```
                   precision    recall  f1-score   support

     Colin Powell       0.92      0.88      0.90        52
  Donald Rumsfeld       0.68      0.76      0.72        25
    George W Bush       0.88      0.90      0.89        93
Gerhard Schroeder       0.92      0.76      0.83        29
       Tony Blair       0.77      0.79      0.78        29

         accuracy                           0.85       228
        macro avg       0.83      0.82      0.82       228
     weighted avg       0.86      0.85      0.85       228
```

```
#Visualization
import matplotlib.pyplot as plt

def plot_gallery(images, titles, h, w, rows=3, cols=4):
  plt.figure(figsize=(10,10))
  for i in range(rows*cols):
    plt.subplot(rows,cols,i+1)
    plt.imshow(images[i].reshape((h,w)),cmap=plt.cm.gray)
    plt.title(titles[i])
    plt.xticks(())
    plt.yticks(())
def titles(Y_pred, Y_test, target_names):
  for i in range(Y_pred.shape[0]):
    pred_name = target_names[Y_pred[i]].split(' ')[-1]
    true_name = target_names[Y_test[i]].split(' ')[-1]
    yield 'predicted: {0}\ntrue: {1}'.format(pred_name, true_name)

prediction_titles = list(titles(Y_pred, Y_test, target_names))
plot_gallery(X_test, prediction_titles, h, w)
```