```
#principal component analysis
#eigen values and eigen vectors
```

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

```
df=pd.read_csv('IRIS_dataset.csv')
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

```
df.shape
```

```
(150, 5)
```

```
x=df[['sepal_length','sepal_width','petal_length','petal_width']]
```

```
y=df[['species']]
```

```
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.head()
```

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

```
#determine the covariance matrix
features=x.T #transposing x
features.shape
features.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1
sepal_length	5.1	4.9	4.7	4.6	5.0	5.4	4.6	5.0	4.4	4.9	5.4	4.8	4.8	4.3	5.8	5.
sepal_width	3.5	3.0	3.2	3.1	3.6	3.9	3.4	3.4	2.9	3.1	3.7	3.4	3.0	3.0	4.0	4.
petal_length	1.4	1.4	1.3	1.5	1.4	1.7	1.4	1.5	1.4	1.5	1.5	1.6	1.4	1.1	1.2	1.
petal_width	0.2	0.2	0.2	0.2	0.2	0.4	0.3	0.2	0.2	0.1	0.2	0.2	0.1	0.1	0.2	0.

4 rows × 150 columns

```
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]])
```

```
eigen_vals,eigen_vecs=np.linalg.eig(covariance_matrix)
eigen_vals
```

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

```
eigen_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]])
```

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

```
0.9246162071742685
```

```
#projecting the data on the first eigen vector
projected_x=x.dot(eigen_vecs.T[0])
```

```
projected_x
```

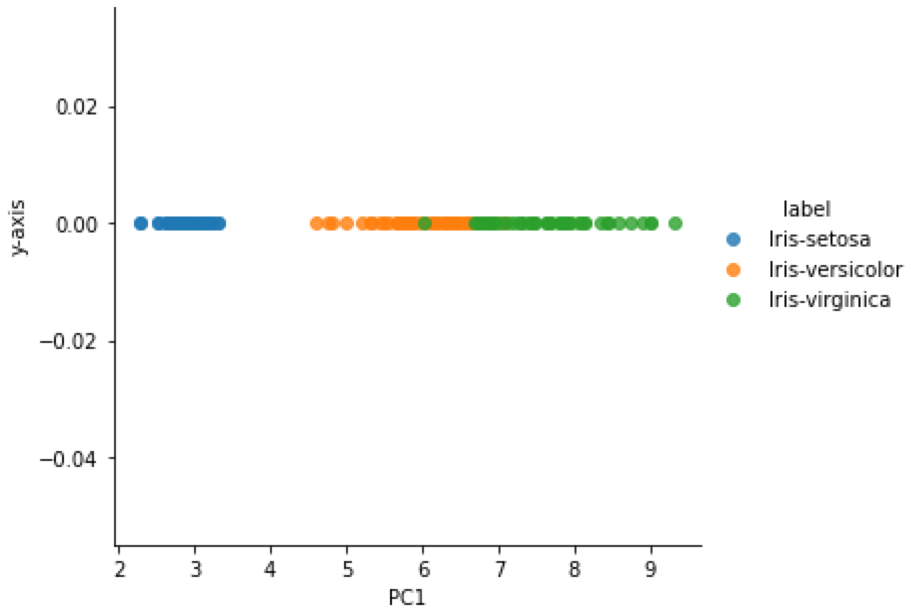
```
0      2.827136
1      2.795952
2      2.621524
3      2.764906
4      2.782750
         ...
145    7.455360
146    7.037007
147    7.275389
148    7.412972
149    6.901009
Length: 150, dtype: float64
```

```
#visualising the dataset
result=pd.DataFrame(projected_x,columns=['PC1'])
result['y-axis']=0.0
result['label']=y
result.head()
```

	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
4	2.782750	0.0	Iris-setosa

```
#plotting this transformed dataset
import seaborn as sns
sns.lmplot('PC1','y-axis',data=result,fit_reg=False,hue='label')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnir
  FutureWarning
<seaborn.axisgrid.FacetGrid at 0x7f968630ed90>
```



```
#Face Recognition using PCA
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_lfw_people
```

```
#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
```

```
#split the dataset into train and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

```
y.shape
```

```
(1140,)
```

```
lfw_dataset.images.shape
```

```
(1140, 62, 47)
```

```
X.shape
```

```
(1140, 62, 47)
```

```
X_train.shape
```

```
(912, 2914)
```

```
#compute the PCA components
n_components=80
from sklearn.decomposition import PCA
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)
```

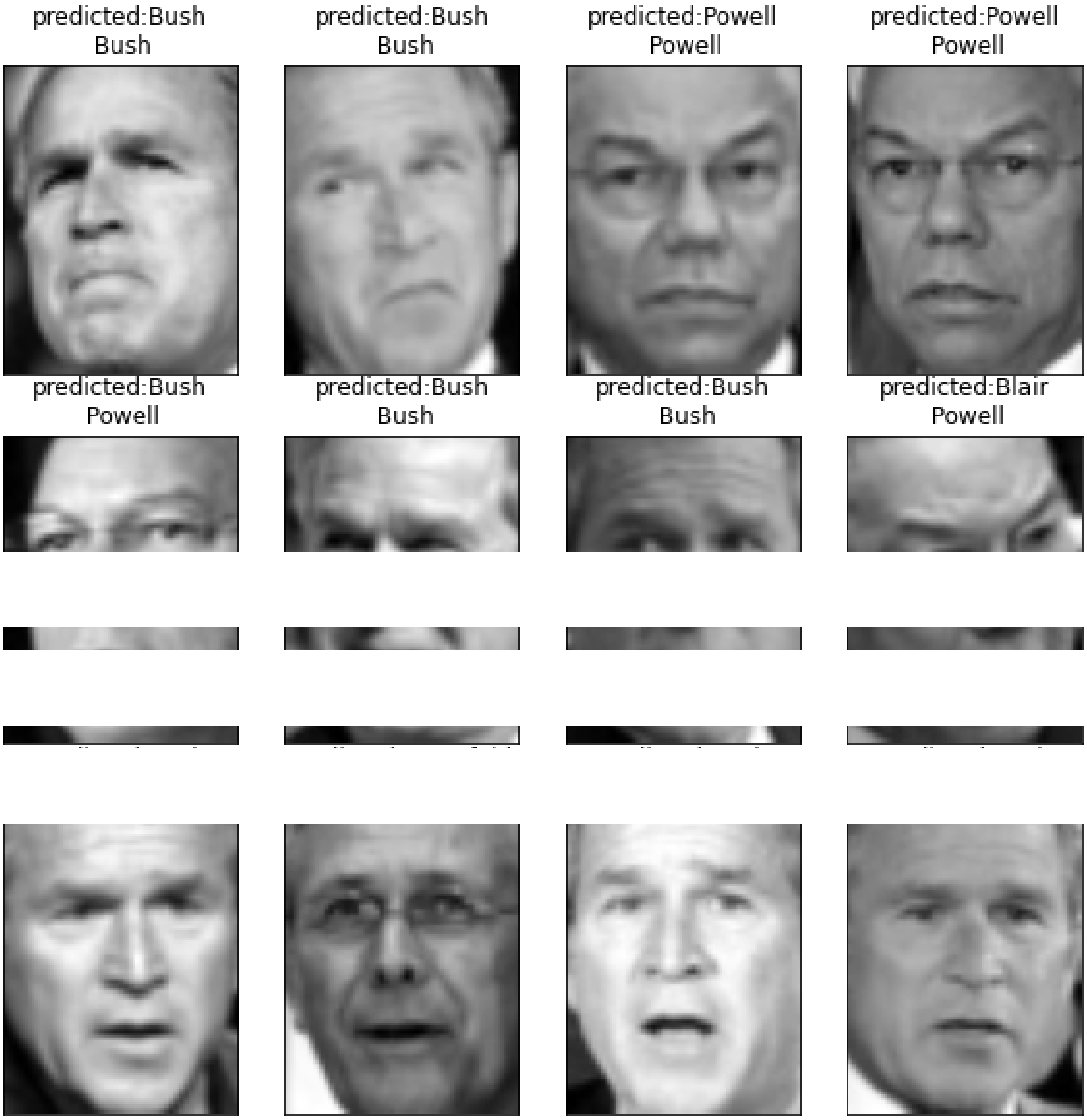
```
#training a classifier
from sklearn.neural_network import MLPClassifier
clf=MLPClassifier(hidden_layer_sizes=(1024,),batch_size=256,verbose=True,early_stop
```

```
Iteration 1, loss = 1.59978568
Validation score: 0.565217
Iteration 2, loss = 1.11860683
Validation score: 0.532609
Iteration 3, loss = 0.89484985
Validation score: 0.586957
Iteration 4, loss = 0.70665905
Validation score: 0.706522
Iteration 5, loss = 0.56258550
Validation score: 0.771739
Iteration 6, loss = 0.45942564
Validation score: 0.793478
Iteration 7, loss = 0.38563679
Validation score: 0.847826
Iteration 8, loss = 0.32862548
Validation score: 0.858696
Iteration 9, loss = 0.27949180
Validation score: 0.847826
Iteration 10, loss = 0.24033505
Validation score: 0.847826
Iteration 11, loss = 0.20936717
Validation score: 0.858696
Iteration 12, loss = 0.18423225
Validation score: 0.858696
Iteration 13, loss = 0.16214867
Validation score: 0.858696
Iteration 14, loss = 0.14410596
Validation score: 0.880435
Iteration 15, loss = 0.12864475
Validation score: 0.880435
Iteration 16, loss = 0.11521119
Validation score: 0.880435
Iteration 17, loss = 0.10378969
Validation score: 0.880435
Iteration 18, loss = 0.09293771
Validation score: 0.880435
Iteration 19, loss = 0.08409606
Validation score: 0.880435
Iteration 20, loss = 0.07676068
Validation score: 0.869565
Iteration 21, loss = 0.07012997
Validation score: 0.869565
Iteration 22, loss = 0.06437694
Validation score: 0.869565
Iteration 23, loss = 0.05923214
Validation score: 0.869565
Iteration 24, loss = 0.05440116
Validation score: 0.869565
Iteration 25, loss = 0.05038845
Validation score: 0.869565
Validation score did not improve more than tol=0.000100 for 10 consecutive epc
```

```
from sklearn.metrics import classification_report
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       0.90      0.82      0.86        57
  Donald Rumsfeld       0.67      0.80      0.73        25
    George W Bush       0.87      0.92      0.89       106
Gerhard Schroeder       0.89      0.53      0.67        15
       Tony Blair       0.81      0.84      0.82        25

         accuracy                           0.85       228
        macro avg       0.83      0.78      0.79       228
     weighted avg       0.85      0.85      0.85       228
```

```
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}\n {1}'.format(pred_name,true_name)

prediction_titles=list(titles(y_pred,y_test,target_names))
plot_gallery(X_test,prediction_titles,h,w)
```


predicted:Bush
Bush
predicted:Bush
Bush
predicted:Powell
Powell
predicted:Powell
Powell
predicted:Bush
Powell
predicted:Bush
Bush
predicted:Bush
Bush
predicted:Blair
Powell