```python
#
# pca_dataset.py
#

import cv2
import numpy as np
import argparse

from sklearn import decomposition
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import time

from sklearn import datasets


#one of three options only should be neq ""
#--dataset = mnist, wines, iris (built-in datasets)

def ParseArguments():
    parser = argparse.ArgumentParser(description="Project ")
    parser.add_argument('--dataset', default="wine", required=False, help='mnist, wines, iris  (default: %(default)s)')

    args = parser.parse_args()

    return args.dataset


# main progam
dataset = ParseArguments()


if(dataset!=""):
    print("Using built-in dataset: ", dataset)



if(dataset=="mnist" or dataset=="MNIST"):

    ile=1000
    our_data = datasets.load_digits()
    points = our_data.data[:ile]
    data_classes = our_data.target[:ile]
    classes  = our_data.target_names
```

```
45      classes_names= classes
46
47
48
49  if(dataset=="wine"):
50      our_data = datasets.load_wine()
51      points = our_data.data
52      data_classes = our_data.target
53      classes  = our_data.target_names
54      classes_names= classes
55
56
57  if(dataset=="iris"):
58      our_data = datasets.load_iris()
59      points = our_data.data
60      data_classes = our_data.target
61      classes  = our_data.target_names
62      classes_names= classes
63
64
65  print("Calculating PCA...", end="", flush=True)
66  start_time = time.time()
67  pca = decomposition.PCA(n_components=3)
68  pca.fit(points)
69  points_pca_reduced = pca.transform(points)
70  print("\t\t took %s seconds " % round((time.time() - start_time),5))
71
72
73  fig_pca = plt.figure(1)
74  ax_pca = fig_pca.add_subplot(111, projection='3d')
75
76  ax_pca.set_title(dataset + ": PCA")
77
78
79  for wt in range(0,data_classes.max()+1):
80      points_pca=points_pca_reduced[data_classes == wt];
81      ax_pca.scatter(points_pca[:,0], points_pca[:,1], points_pca[:,2],
        label=classes_names[wt])
82
83  ax_pca.legend()
84  plt.show();
85
```