```
"""
Created on Wed Sep 21 19:14:12 2022
@author: Dr Ambi,
"""

from sklearn.datasets import fetch_openml
import pickle
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA



#Sklearn Feature selection
#https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selec
#https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection

#https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.Varia



digitsDataFirst100=pickle.load(open( "digitsDataFirst100.p", "rb" ) )
targetFirst100=pickle.load(open( "targetFirst100.p", "rb" ) )


digitsData=digitsDataFirst100
target=targetFirst100
digitsData.head()
```

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixe
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 784 columns

```
desc=digitsData.describe()
print(desc)
```

```
       pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pixel8  pixel9
count   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0
mean      0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
std       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
min       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
```

```
25%       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
50%       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
75%       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
max       0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0

       pixel10  ...  pixel775  pixel776  pixel777  pixel778  pixel779  \
count    100.0  ...     100.0     100.0     100.0     100.0     100.0   
mean       0.0  ...       0.0       0.0       0.0       0.0       0.0   
std        0.0  ...       0.0       0.0       0.0       0.0       0.0   
min        0.0  ...       0.0       0.0       0.0       0.0       0.0   
25%        0.0  ...       0.0       0.0       0.0       0.0       0.0   
50%        0.0  ...       0.0       0.0       0.0       0.0       0.0   
75%        0.0  ...       0.0       0.0       0.0       0.0       0.0   
max        0.0  ...       0.0       0.0       0.0       0.0       0.0   

       pixel780  pixel781  pixel782  pixel783  pixel784  
count     100.0     100.0     100.0     100.0     100.0  
mean        0.0       0.0       0.0       0.0       0.0  
std         0.0       0.0       0.0       0.0       0.0  
min         0.0       0.0       0.0       0.0       0.0  
25%         0.0       0.0       0.0       0.0       0.0  
50%         0.0       0.0       0.0       0.0       0.0  
75%         0.0       0.0       0.0       0.0       0.0  
max         0.0       0.0       0.0       0.0       0.0  

[8 rows x 784 columns]
```

```
desc=digitsData.describe()

print(digitsData.shape)

import matplotlib.pyplot as plt



#Plot first image
i=0
print(digitsData.iloc[i].values.shape)
original_image = digitsData.iloc[i].values.reshape([28,28])
plt.imshow(original_image, cmap='gray_r')
plt.title("original_image: Digit " + target[i], fontsize=15, pad=15)
plt.savefig("original_image image.png")
```

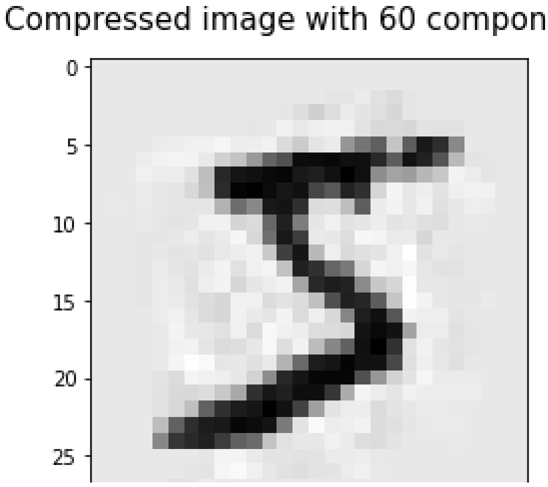
```
(100, 784)
(784,)
```

```
#No scaling required as all values in same scale
print(digitsData.iloc[1].min())
print(digitsData.iloc[1].max())
```

```
0.0
255.0
```

```
numComponents=60
pca = PCA(n_components=numComponents)
mnist_new_features = pca.fit_transform(digitsData)

#print(mnist_new_features.shape)
#print(mnist_new_features)
#print(type(mnist_new_features))
mnist_reduced_recovered_image = pca.inverse_transform(mnist_new_features)

image_reduced = mnist_reduced_recovered_image[i,:].reshape([28,28])
plt.figure(0)
plt.imshow(image_reduced, cmap='gray_r')
plt.title('Compressed image with ' + str(numComponents) +' components', fontsize=1
#plt.savefig("images/reduced_image_with_" + str(numComponents) + "_pca_components.
plt.savefig("reduced_image_with_" + str(numComponents) + "_pca_components.png")
np.cumsum(pca.explained_variance_ratio_ * 100)[-1]
cumulativevariance=np.cumsum(pca.explained_variance_ratio_*100)
plt.figure(1)
plt.plot(cumulativevariance)
plt.xlabel('number of components')
plt.ylabel('variance')
```

```
Text(0, 0.5, 'variance')
```



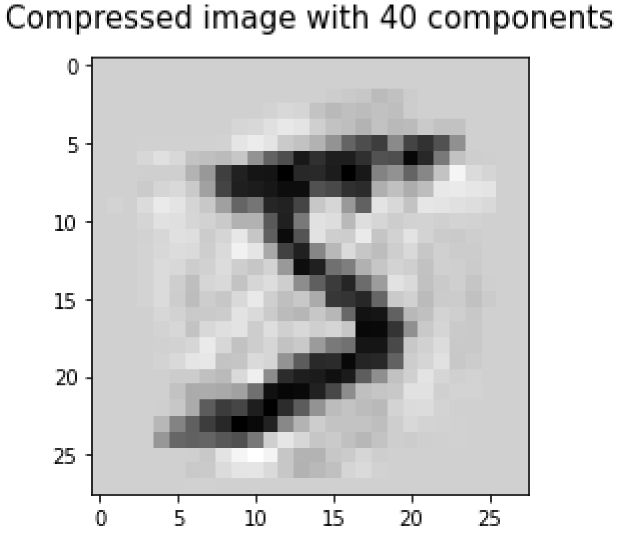
```
df1=pd.DataFrame(mnist_new_features)
df1.head()

desc=df1.describe()
print(desc)
```

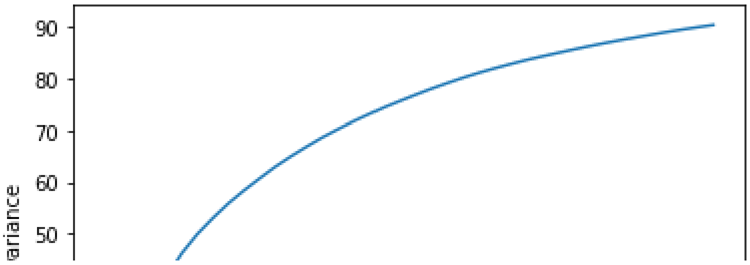
```
                  0             1             2             3             4
count  1.000000e+02  1.000000e+02  1.000000e+02  1.000000e+02  1.000000e+02
mean   7.617018e-14 -1.250555e-14 -1.324452e-13 -3.154810e-14  7.787548e-14
std    6.660491e+02  5.116718e+02  4.924304e+02  4.739857e+02  4.167841e+02
min   -9.088630e+02 -1.044248e+03 -9.125426e+02 -8.738673e+02 -8.436267e+02
25%   -5.636989e+02 -3.193785e+02 -3.585628e+02 -3.246417e+02 -3.024862e+02
50%   -1.350736e+02 -3.977772e+01 -5.245325e+01 -5.258952e+01 -2.122688e-01
75%    3.193674e+02  3.453945e+02  3.138463e+02  3.309972e+02  2.572756e+02
max    1.643322e+03  1.335523e+03  1.135010e+03  1.198639e+03  8.805235e+02

                  5             6             7             8             9
count  1.000000e+02  1.000000e+02  1.000000e+02  1.000000e+02  1.000000e+02
mean  -6.561862e-14 -1.477929e-14  4.177991e-14  9.094947e-15  1.790568e-14
std    3.960680e+02  3.495432e+02  3.189113e+02  3.059750e+02  2.859976e+02
min   -6.862783e+02 -7.376648e+02 -7.574006e+02 -6.440035e+02 -7.179005e+02
25%   -2.999328e+02 -2.601930e+02 -1.895441e+02 -2.330401e+02 -1.783052e+02
50%   -1.743466e+01 -2.478023e+00  3.185896e+01  7.396356e+01  1.647172e+01
75%    1.845557e+02  2.795184e+02  1.627047e+02  2.062653e+02  1.694890e+02
max    1.485748e+03  7.650856e+02  9.799049e+02  8.928392e+02  7.235824e+02

       ...            50            51            52            53  \
count  ...  1.000000e+02  1.000000e+02  1.000000e+02  1.000000e+02
mean   ...  3.410605e-15 -1.037392e-14  4.625633e-14 -1.492140e-15
std    ...  9.431079e+01  9.366248e+01  9.273455e+01  8.680868e+01
min    ... -2.092794e+02 -2.116437e+02 -2.255777e+02 -1.768802e+02
25%    ... -6.437777e+01 -6.336515e+01 -6.404706e+01 -6.281278e+01
50%    ... -7.346135e+00  5.524052e+00  1.809038e+00 -4.777214e+00
```

```
75%    ...  4.840948e+01  6.121614e+01  6.427238e+01  5.620991e+01
max    ...  3.031515e+02  2.381775e+02  2.387363e+02  2.599120e+02

                 54            55            56            57            58
count  1.000000e+02  1.000000e+02  1.000000e+02  1.000000e+02  1.000000e+02
mean  -2.242206e-14  7.354117e-15  1.016076e-14  2.984279e-15  1.827871e-14
std    8.548239e+01  8.386461e+01  8.171892e+01  8.137204e+01  7.952627e+01
min   -1.760079e+02 -1.887749e+02 -1.996700e+02 -1.872570e+02 -2.000621e+02
25%   -6.856838e+01 -6.479523e+01 -5.364471e+01 -5.058936e+01 -5.724038e+01
50%    5.080388e+00 -4.462696e+00 -3.640625e+00  4.399895e+00 -5.718475e+00
75%    5.598791e+01  5.391529e+01  4.632972e+01  4.923802e+01  5.472691e+01
max    2.336928e+02  2.356190e+02  2.467207e+02  1.947799e+02  2.170693e+02

                 59
count  1.000000e+02
mean  -9.201528e-15
std    7.861026e+01
min   -2.098779e+02
25%   -5.317492e+01
50%    5.437672e+00
75%    4.542992e+01
max    2.256590e+02

[8 rows x 60 columns]
```

```
numComponents=40
pca = PCA(n_components=numComponents)
mnist_new_features = pca.fit_transform(digitsData)

#print(mnist_new_features.shape)
#print(mnist_new_features)
#print(type(mnist_new_features))
mnist_reduced_recovered_image = pca.inverse_transform(mnist_new_features)

image_reduced = mnist_reduced_recovered_image[i,:].reshape([28,28])
plt.figure(0)
plt.imshow(image_reduced, cmap='gray_r')
plt.title('Compressed image with ' + str(numComponents) +' components', fontsize=1
#plt.savefig("images/reduced_image_with_" + str(numComponents) + "_pca_components.
plt.savefig("reduced_image_with_" + str(numComponents) + "_pca_components.png")
np.cumsum(pca.explained_variance_ratio_ * 100)[-1]
cumulativevariance=np.cumsum(pca.explained_variance_ratio_*100)
plt.figure(1)
plt.plot(cumulativevariance)
plt.xlabel('number of components')
plt.ylabel('variance')
```

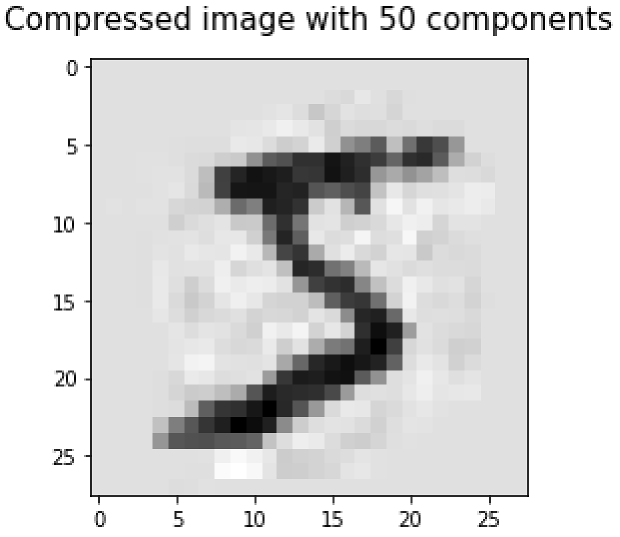
```
Text(0, 0.5, 'variance')
```

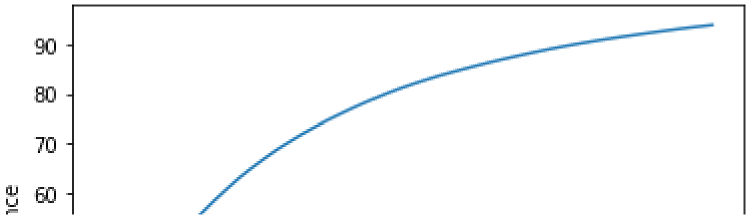




```
numComponents=50
pca = PCA(n_components=numComponents)
mnist_new_features = pca.fit_transform(digitsData)

#print(mnist_new_features.shape)
#print(mnist_new_features)
#print(type(mnist_new_features))
mnist_reduced_recovered_image = pca.inverse_transform(mnist_new_features)

image_reduced = mnist_reduced_recovered_image[i,:].reshape([28,28])
plt.figure(0)
plt.imshow(image_reduced, cmap='gray_r')
plt.title('Compressed image with ' + str(numComponents) +' components', fontsize=1
#plt.savefig("images/reduced_image_with_" + str(numComponents) + "_pca_components.
plt.savefig("reduced_image_with_" + str(numComponents) + "_pca_components.png")
np.cumsum(pca.explained_variance_ratio_ * 100)[-1]
cumulativevariance=np.cumsum(pca.explained_variance_ratio_*100)
plt.figure(1)
plt.plot(cumulativevariance)
plt.xlabel('number of components')
plt.ylabel('variance')
```

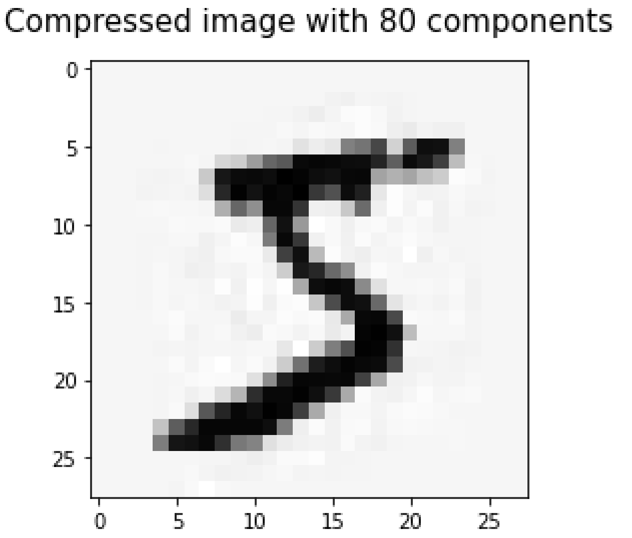
```
Text(0, 0.5, 'variance')
```

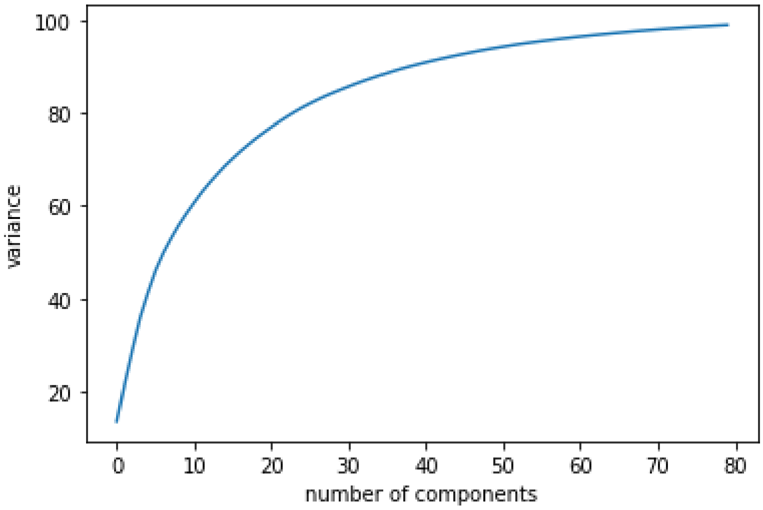




```
numComponents=80
pca = PCA(n_components=numComponents)
mnist_new_features = pca.fit_transform(digitsData)

#print(mnist_new_features.shape)
#print(mnist_new_features)
#print(type(mnist_new_features))
mnist_reduced_recovered_image = pca.inverse_transform(mnist_new_features)

image_reduced = mnist_reduced_recovered_image[i,:].reshape([28,28])
plt.figure(0)
plt.imshow(image_reduced, cmap='gray_r')
plt.title('Compressed image with ' + str(numComponents) +' components', fontsize=1
#plt.savefig("images/reduced_image_with_" + str(numComponents) + "_pca_components.
plt.savefig("reduced_image_with_" + str(numComponents) + "_pca_components.png")
np.cumsum(pca.explained_variance_ratio_ * 100)[-1]
cumulativevariance=np.cumsum(pca.explained_variance_ratio_*100)
plt.figure(1)
plt.plot(cumulativevariance)
plt.xlabel('number of components')
plt.ylabel('variance')
```

```
Text(0, 0.5, 'variance')
```





✓ 0s completed at 15:23