## Name:Mahima Yadav

Roll no:-08

Batch:-E1

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

1.to avoid the problem of overfitting,pca is used 2.reduce 100 attributes to 10 merge different attributes together which are highly corelated with each other 3.it is a feature extraction techniqe 4.used for dimensionality reduction 5.if the spread is more variance is more than variety of data is high,spread should be more

```
from sklearn import datasets
```

```
digit=datasets.load_digits()
```

```
digit.keys()
```

```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESCR'])
```

```
print(digit['DESCR'])
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 1797
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of
4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their
    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of
    Graduate Studies in Science and Engineering, Bogazici University.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.
    Linear dimensionalityreduction using relevance weighted LDA. School of
    Electrical and Electronic Engineering Nanyang Technological University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classification
    Algorithm. NIPS. 2000.
```

```
print(digit['target_names'])
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
print(digit['feature_names'])
```

```
['pixel_0_0', 'pixel_0_1', 'pixel_0_2', 'pixel_0_3', 'pixel_0_4', 'pixel_0_5', 'pixel_0_6', 'pixel_0_7', 'pixel_1_0', 'pixel_1_1', 'pixe
```

```
feature = digit.data
target = digit.target
```

```
from sklearn.model_selection import train_test_split
```

```
df=pd.DataFrame(digit['data'],columns=digit['feature_names'])
```

```
df.head()
```

| | pixel_0_0 | pixel_0_1 | pixel_0_2 | pixel_0_3 | pixel_0_4 | pixel_0_5 | pixel_0_6 | pixel_0_7 | pixel_1_0 | pixel_1_1 | ... | pixel_6_6 | pixel_6_7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 5.0 | 13.0 | 9.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.( |
| **1** | 0.0 | 0.0 | 0.0 | 12.0 | 13.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.( |
| **2** | 0.0 | 0.0 | 0.0 | 4.0 | 15.0 | 12.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 5.0 | 0.( |
| **3** | 0.0 | 0.0 | 7.0 | 15.0 | 13.0 | 1.0 | 0.0 | 0.0 | 0.0 | 8.0 | ... | 9.0 | 0.( |
| **4** | 0.0 | 0.0 | 0.0 | 1.0 | 11.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.( |

5 rows × 64 columns

```
df.tail()
```

| | pixel_0_0 | pixel_0_1 | pixel_0_2 | pixel_0_3 | pixel_0_4 | pixel_0_5 | pixel_0_6 | pixel_0_7 | pixel_1_0 | pixel_1_1 | ... | pixel_6_6 | pixel_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1792** | 0.0 | 0.0 | 4.0 | 10.0 | 13.0 | 6.0 | 0.0 | 0.0 | 0.0 | 1.0 | ... | 4.0 | |
| **1793** | 0.0 | 0.0 | 6.0 | 16.0 | 13.0 | 11.0 | 1.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | |
| **1794** | 0.0 | 0.0 | 1.0 | 11.0 | 15.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | |
| **1795** | 0.0 | 0.0 | 2.0 | 10.0 | 7.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 2.0 | |
| **1796** | 0.0 | 0.0 | 10.0 | 14.0 | 8.0 | 1.0 | 0.0 | 0.0 | 0.0 | 2.0 | ... | 8.0 | |

5 rows × 64 columns

```
X_train, X_test, y_train, y_test = train_test_split(df, target, train_size = 0.7, random_state = 3)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1257, 64)
(540, 64)
(1257,)
(540,)
```

```
from sklearn.linear_model import LogisticRegression
```

```
my_model = LogisticRegression()
```

```
my_model.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
preds = my_model.predict(X_test)
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
LogisticRegression()
```

```
print(accuracy_score(y_test, preds))
```

```
0.9537037037037037
```

```
from sklearn.decomposition import PCA
```

```
pca=PCA(n_components=30)
```

```
x_pca=pca.fit_transform(df)
```

```
df.shape
```

```
(1797, 64)
```

```
x_pca.shape
```

```
(1797, 30)
```

```
x_pca
```

```
array([[ -1.25946644,  21.27488341,  -9.46305465, ...,  -0.94057806,
         -1.14589241,   2.30963742],
       [  7.95761133, -20.76869889,   4.43950604, ...,  -0.61412919,
          2.43696391,   0.64573521],
       [  6.99192297,  -9.95598631,   2.95855808, ...,   2.14867056,
          0.86555209,  -0.44075962],
       ...,
       [ 10.80128371,  -6.96025226,   5.59955449, ...,   1.87382348,
          3.50644345,  -4.0235111 ],
       [ -4.8721001 ,  12.42395363, -10.17086641, ...,   0.98326887,
         -0.95572838,  -1.44957216],
       [ -0.34438961,   6.36554941,  10.77370858, ...,   1.15742648,
          2.74756365,  -6.67833307]])
```

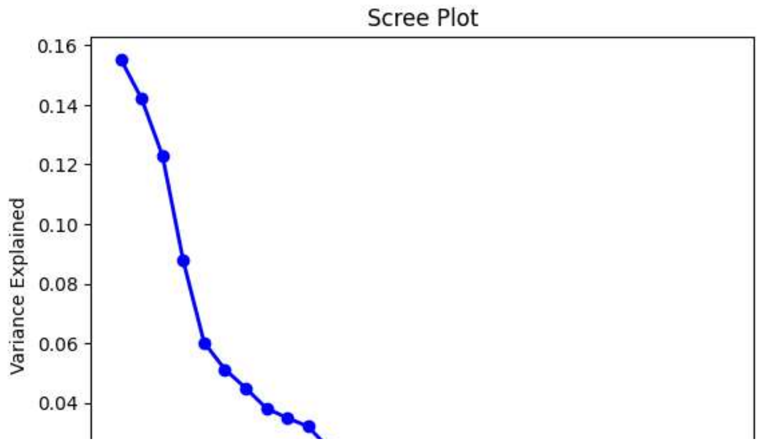
```
explained_variance = np.var(x_pca, axis=0)
print(explained_variance)
```

```
[178.90731578 163.62664073 141.70953623 101.04411456  69.47448269
  59.075632    51.85566624  43.99061301  40.28856291  36.99120196
  28.50317082  27.30596604  21.88930032  21.31248988  17.62690729
  16.93743145  15.84256866  14.99611043  12.22764288  10.88077315
  10.68761448   9.57725525   9.22125571   8.68546959   8.36092577
   7.16147816   6.91547153   6.1877996    5.88002415   5.14920005]
```

```
explained_variance_ratio = explained_variance / np.sum(explained_variance)
```

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

PC_values = np.arange(pca.n_components) + 1
plt.plot(PC_values, explained_variance_ratio, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```




last me jo accuracy aati hai use threshhold bolte hai

```
X_train, X_test, y_train, y_test = train_test_split(x_pca, target, train_size = 0.7, random_state = 10)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1257, 30)
(540, 30)
(1257,)
(540,)
```

```
my_model.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
preds = my_model.predict(X_test)
```

```
print(accuracy_score(y_test, preds))
```

```
0.95
```