Dimensionality Reduction Algorithm

Shubh Agarwal
AIML B2
22070126108

Synopsis of High Dimension Dataset

Dataset Characteristics

Multivariate

Subject Area

Health and Medicine

Associated Tasks

Classification

Feature Type

Real

Instances

569

# Features

30

Additional Information

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at http://www.cs.wisc.edu/~street/images/

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:
ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

THEORY OF PCA

A popular dimensionality reduction method in data analysis and machine learning is
principal component analysis or PCA. Its main objective is to create a new set of uncorrelated variables from a dataset's original attributes.
referred to as the main elements. The degree of variance in the data that each of these
principle components explains determines their ranking.
The iris dataset has extremely few characteristics, as we can see, so even with PCA, the
accuracy remained rather constant. In the second dataset with large dimensions, the PCA

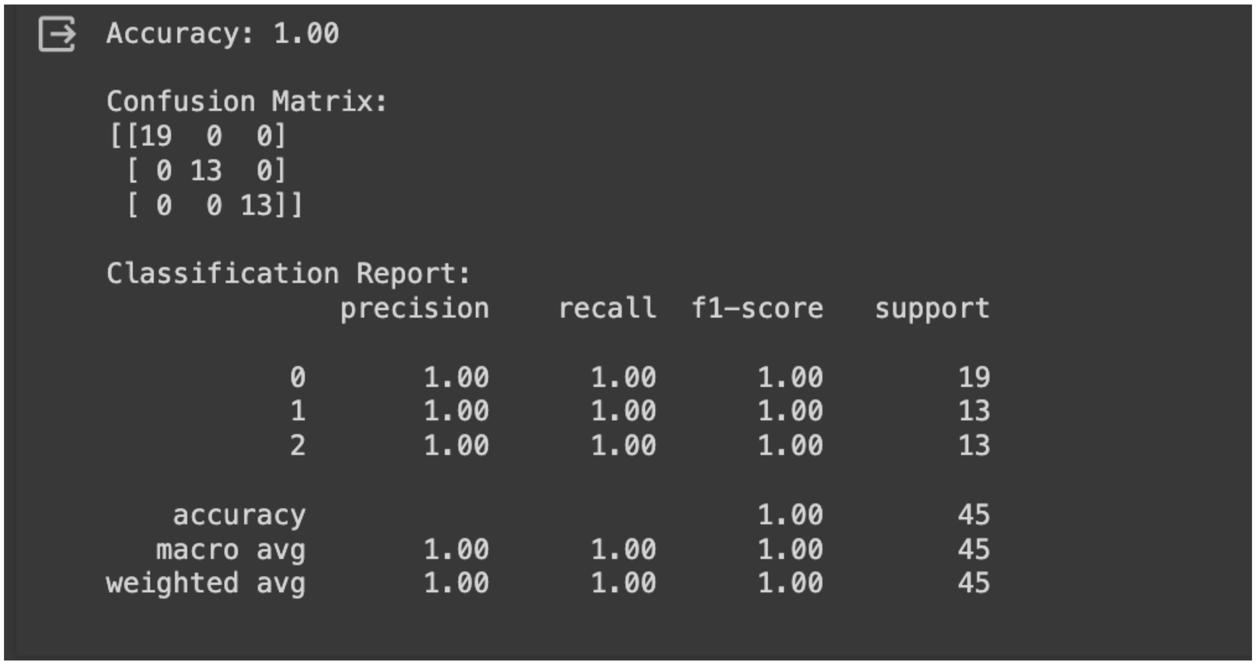
does not work well as it can be seen from the graph the classification is not that good .

IRIS DATASET

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler



from sklearn.datasets import load_iris

# Load the Iris dataset
iris_data = load_iris()

# The data is stored in the 'data' attribute
X_varibales = iris_data.data

# The target labels are stored in the 'target' attribute
y_variables = iris_data.target

sc = StandardScaler()
transformed_df = sc.fit_transform(X_variables)
```

Applying Logistic Regression Classifier

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score , confusion_matrix
, classification_report

x_train , x_test , y_train , y_test = train_test_split(X_vari
ables,y_variables,test_size = 0.3 , random_state = 42)

model = LogisticRegression()
model.fit(x_train , y_train)
y_pred = model.predict(x_test)

# Evaluating the model

accuracy = accuracy_score(y_test , y_pred)
conf_matrix = confusion_matrix(y_test,y_pred)
report = classification_report(y_test,y_pred)

print(f'Accuracy: {accuracy:.2f}')
print('\nConfusion Matrix:')
print(conf_matrix)
print('\nClassification Report:')
print(report)
```

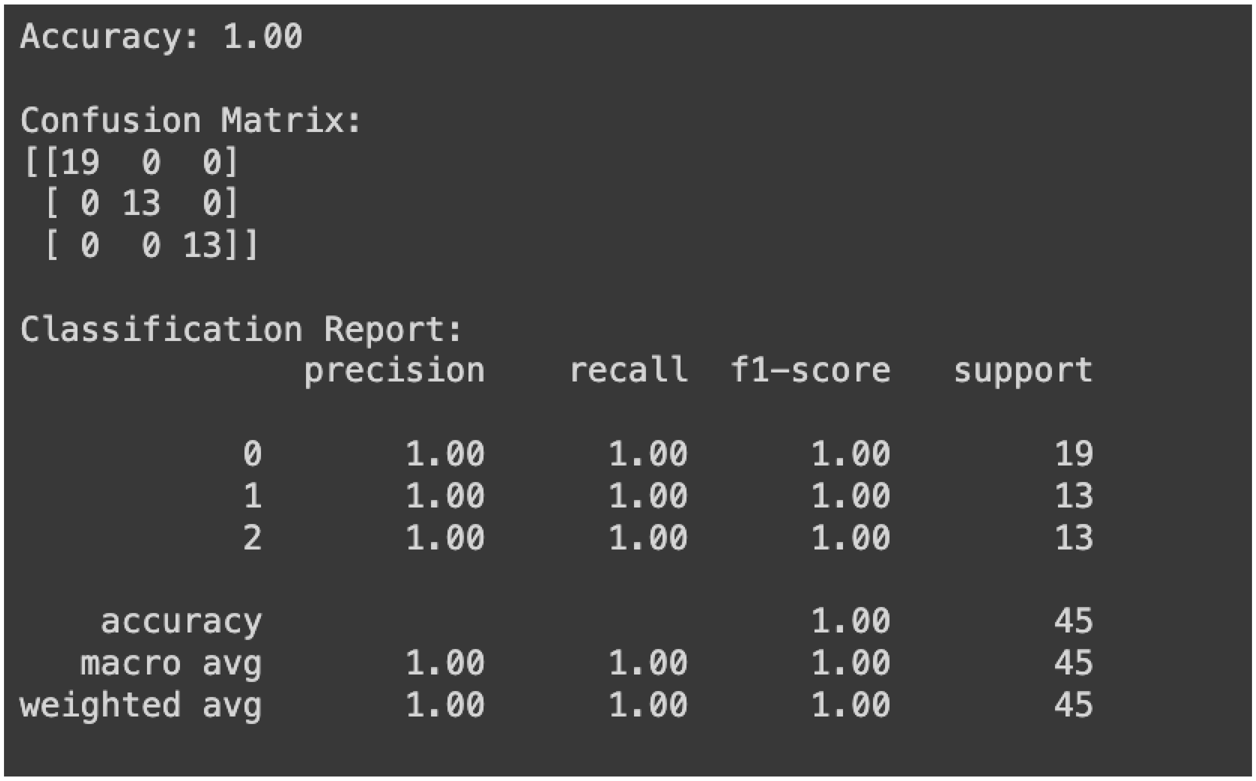


```
Accuracy: 1.00

Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Accuracy with Logistic Regression is 1.0

Applying Random Forest Classifier

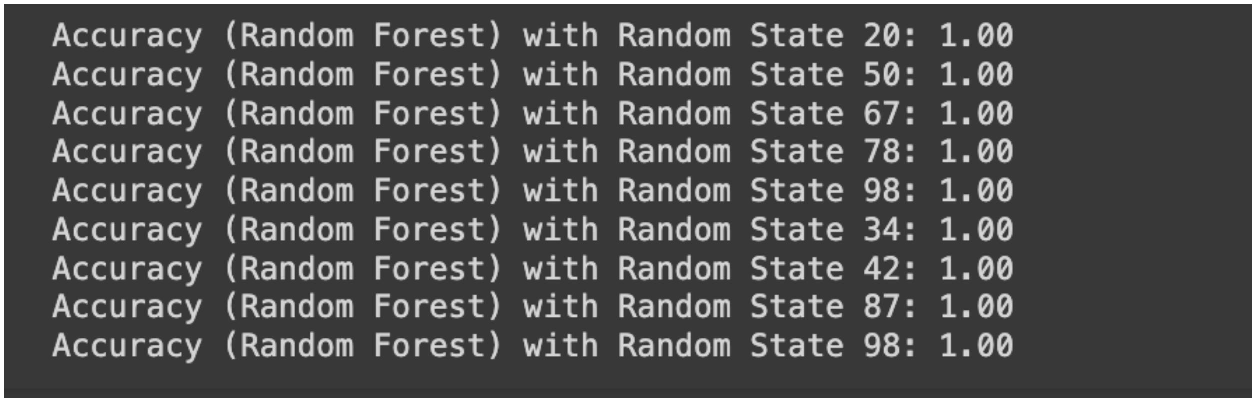
```
from sklearn.ensemble import RandomForestClassifier

model1 = RandomForestClassifier()
model1.fit(x_train , y_train)
y_pred1 = model1.predict(x_test)

accuracy = accuracy_score(y_test , y_pred1)
conf_matrix = confusion_matrix(y_test,y_pred1)
report = classification_report(y_test,y_pred1)

print(f'Accuracy: {accuracy:.2f}')
print('\nConfusion Matrix:')
print(conf_matrix)
print('\nClassification Report:')
print(report)
```



```
Accuracy: 1.00

Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Again the Accuracy with Random Forest Classifier is 1.0

```
random_states = [ 20,50,67,78,98,34, 42,87,98]

for random_state in random_states:

    rf_model = RandomForestClassifier(random_state=random_sta
te)


    rf_model.fit(x_train, y_train)


    y_pred_rf = rf_model.predict(x_test)


    accuracy_rf = accuracy_score(y_test, y_pred_rf)
    print(f'Accuracy (Random Forest) with Random State {rando
m_state}: {accuracy_rf:.2f}')
```



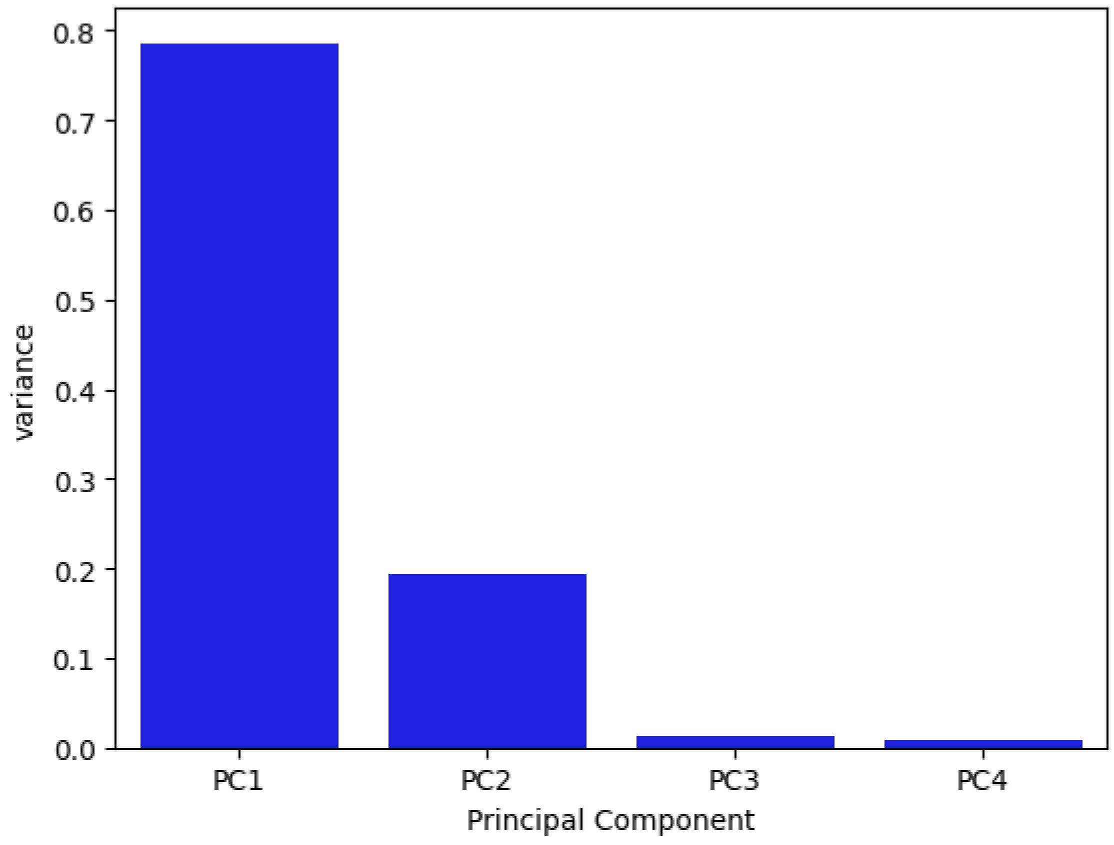
```
Accuracy (Random Forest) with Random State 20: 1.00
Accuracy (Random Forest) with Random State 50: 1.00
Accuracy (Random Forest) with Random State 67: 1.00
Accuracy (Random Forest) with Random State 78: 1.00
Accuracy (Random Forest) with Random State 98: 1.00
Accuracy (Random Forest) with Random State 34: 1.00
Accuracy (Random Forest) with Random State 42: 1.00
Accuracy (Random Forest) with Random State 87: 1.00
Accuracy (Random Forest) with Random State 98: 1.00
```

Accuracy with different Random States is also 1.0

Applying PCA

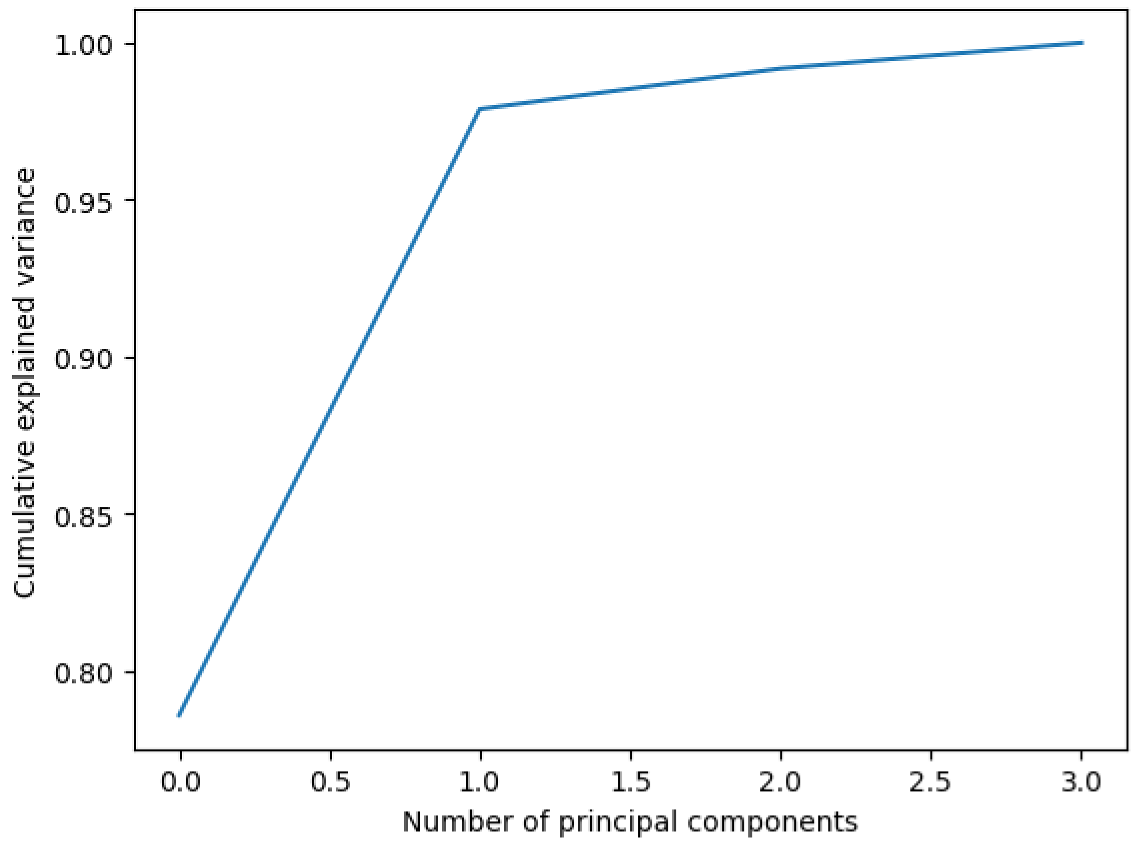
By calculating the covariance matrix, PCA can identify the directions of maximum variance in your data, making it easier to capture the essential features while reducing the dimensionality.

```
covariance_matrix = np.cov(transformed_df.T)
covariance_matrix
eigen_values, eigen_vectors = np.linalg.eig(covariance_matri
x)
eigen_pairs = [(np.abs(eigen_values[i]), eigen_vectors[:,i])
for i in range(len(eigen_values))]
print('Eigenvalues arranged in descending order:')
for i in eigen_pairs:
    print(i[0])
pca = PCA()
pca = pca.fit(transformed_df)
explained_variance = pca.explained_variance_ratio_
explained_variance
dataframe = pd.DataFrame({'variance':pca.explained_variance_r
atio_,
             'Principal Component':['PC1','PC2','PC3','PC
4']]})
sns.barplot(x='Principal Component',y="variance",
           data=dataframe, color="b");
```

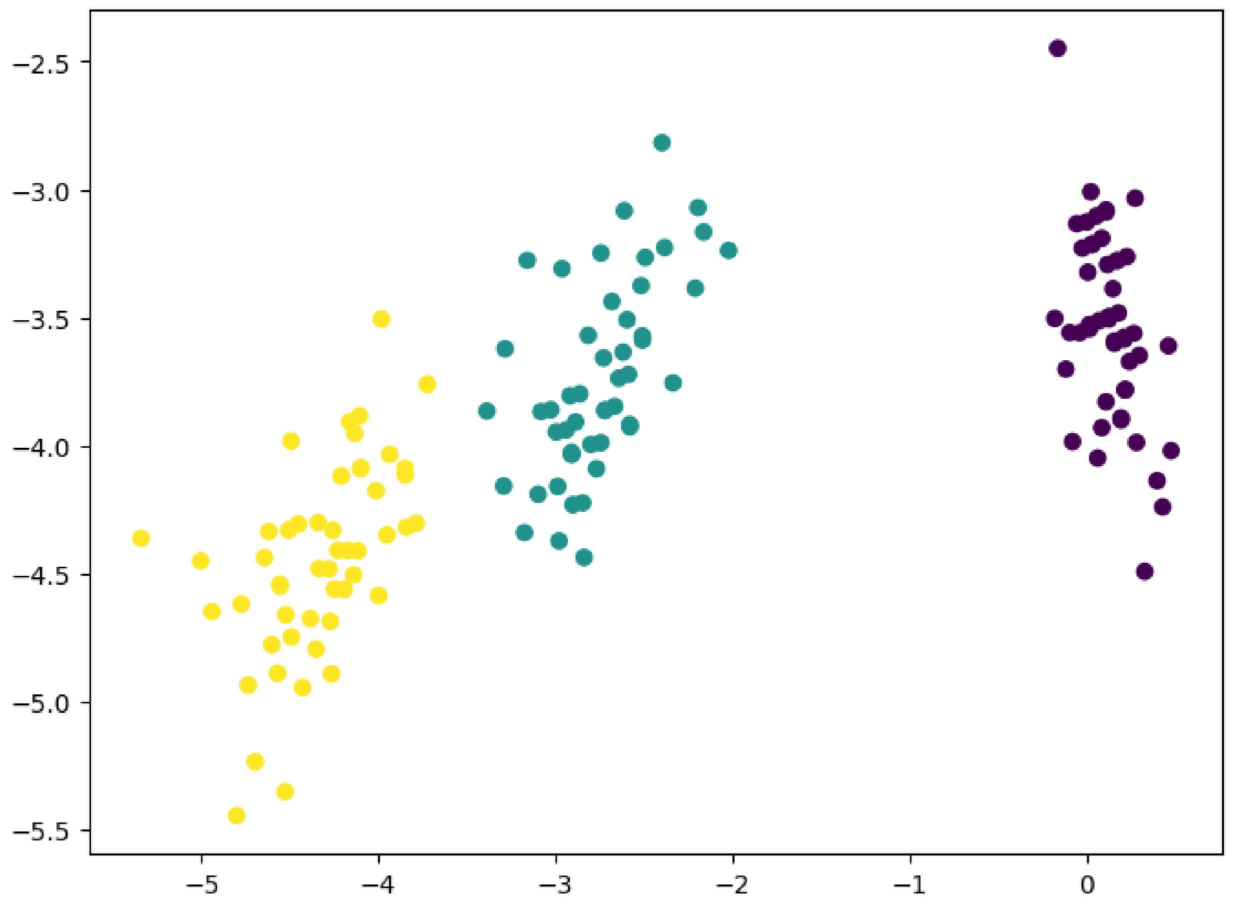


Shows the variance for different principal Components

```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of principal components')
plt.ylabel('Cumulative explained variance')
plt.show()
```

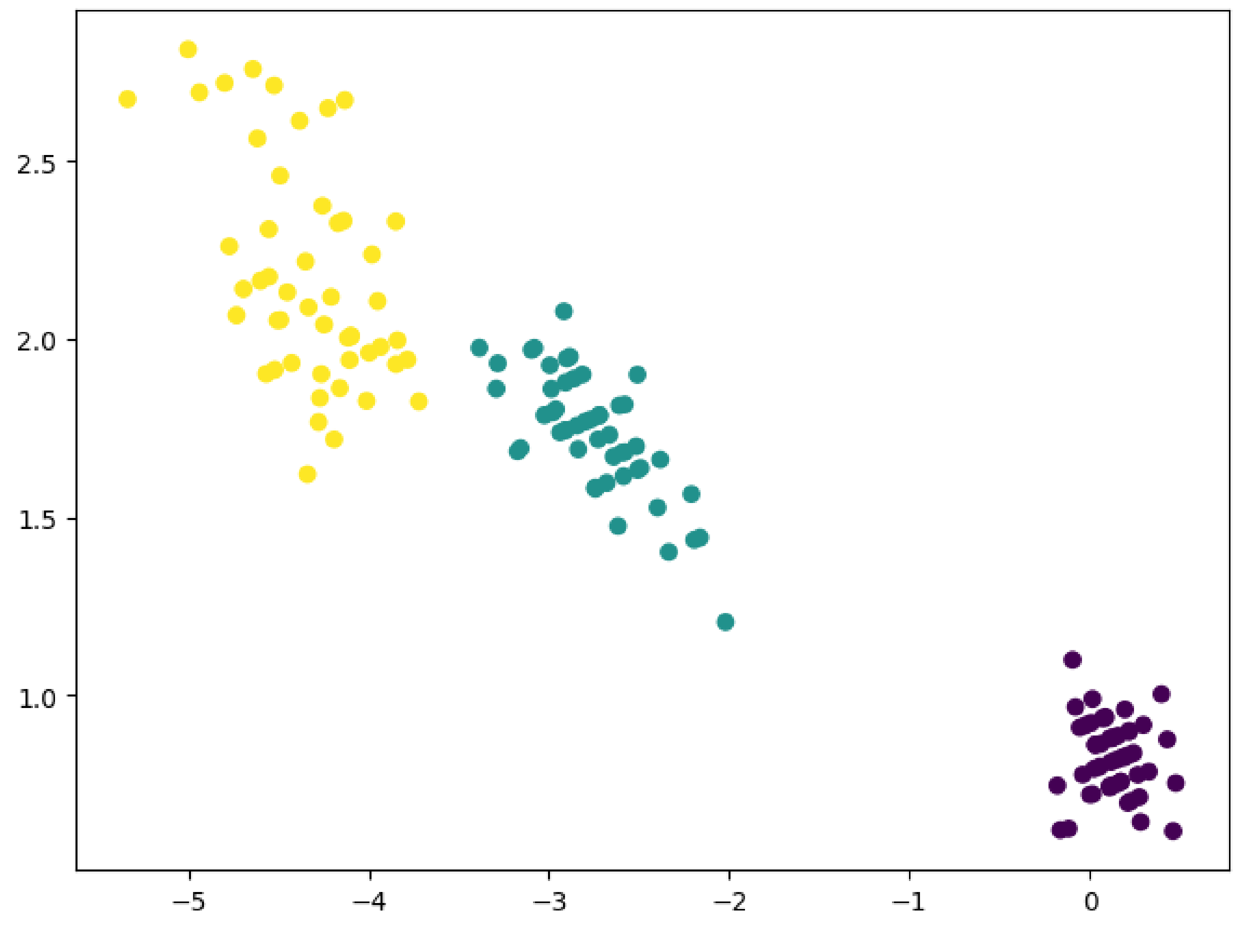


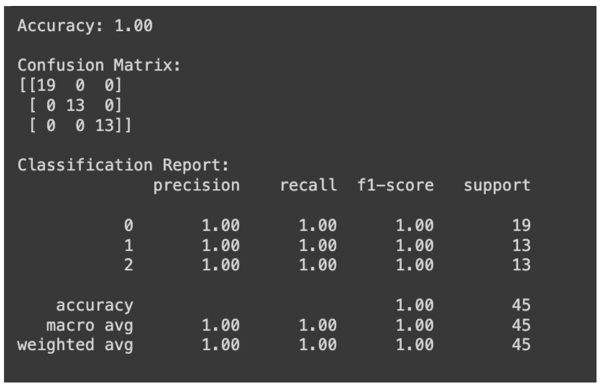
```
pca_2 = PCA(n_components =2 )
pca_2 = pca_2.fit(transformed_df)
pca_2d = pca_2.transform(X_variables)

plt.figure(figsize=(8,6))

plt.scatter(pca_2d[:,0], pca_2d[:,1],c=y_variables)
plt.show()
```



This graph shows the classification when we take PCA component = 2 and the classification of three classes is pretty much clear

```
pca_4 = PCA(n_components =4 )
pca_4 = pca_4.fit(transformed_df)
pca_4d = pca_4.transform(X_variables)

plt.figure(figsize=(8,6))

plt.scatter(pca_4d[:,0], pca_4d[:,3],c=y_variables)
plt.show()
```





```
Accuracy: 1.00

Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Accuracy after PCA is same as 1.0

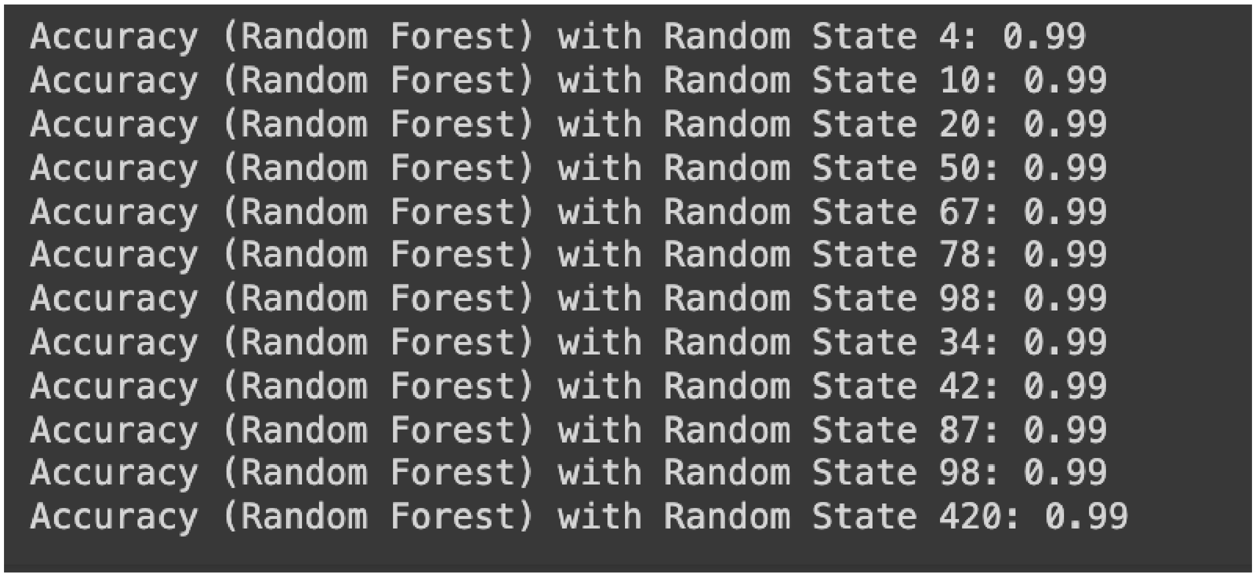
Overall, the goal is to see if the reduced-dimensional representation (using PCA) reveals patterns or separations between the different classes in the Iris dataset. It helps in visualizing how well the chosen components capture the variability in the data.

Dataset - 2 Breast Cancer Wisconsin (Diagnostic)

```
from sklearn.datasets import load_iris, load_breast_cancer

X_cancer, y_cancer = load_breast_cancer(return_X_y=True)

sc = StandardScaler()
X_cancer = sc.fit_transform(X_cancer)
```

Logistic Regreesion

```

random_states = [ 4,10,20,50,67,78,98,34, 42,87,98,420]

for random_state in random_states:

    clf = LogisticRegression(random_state=random_state)
    clf.fit(X_train_cancer, y_train_cancer)


    y_pred_rf = clf.predict(X_test_cancer)


    accuracy_rf = accuracy_score(y_test_cancer, y_pred_rf)

    print(f'Accuracy (Random Forest) with Random State {rando
m_state}: {accuracy_rf:.2f}')
```



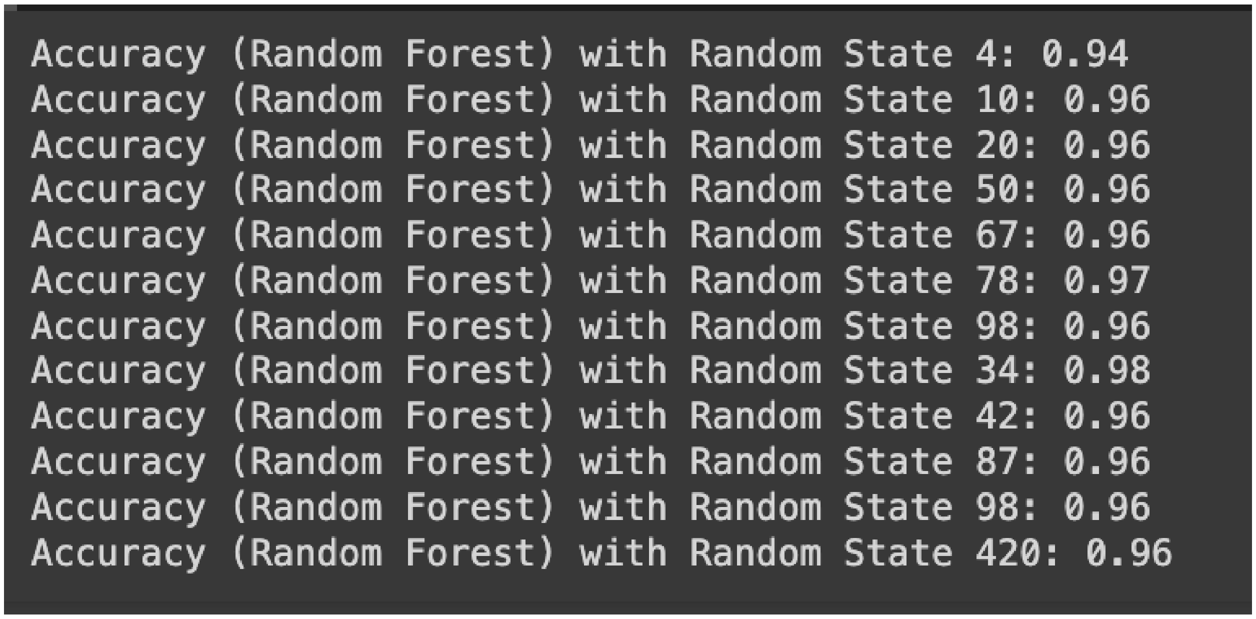
```
Accuracy (Random Forest) with Random State 4: 0.99
Accuracy (Random Forest) with Random State 10: 0.99
Accuracy (Random Forest) with Random State 20: 0.99
Accuracy (Random Forest) with Random State 50: 0.99
Accuracy (Random Forest) with Random State 67: 0.99
Accuracy (Random Forest) with Random State 78: 0.99
Accuracy (Random Forest) with Random State 98: 0.99
Accuracy (Random Forest) with Random State 34: 0.99
Accuracy (Random Forest) with Random State 42: 0.99
Accuracy (Random Forest) with Random State 87: 0.99
Accuracy (Random Forest) with Random State 98: 0.99
Accuracy (Random Forest) with Random State 420: 0.99
```

```
X_train_cancer, X_test_cancer, y_train_cancer, y_test_cancer
= train_test_split(X_cancer, y_cancer, test_size=0.2, random_
state=82)

y_pred = clf.predict(X_test_cancer)

acc = accuracy_score(y_test_cancer, y_pred)
print("Logistic Regression model accuracy (in %):", acc*100)
```

Logistic Regression model accuracy (in %): 99.12280701754386

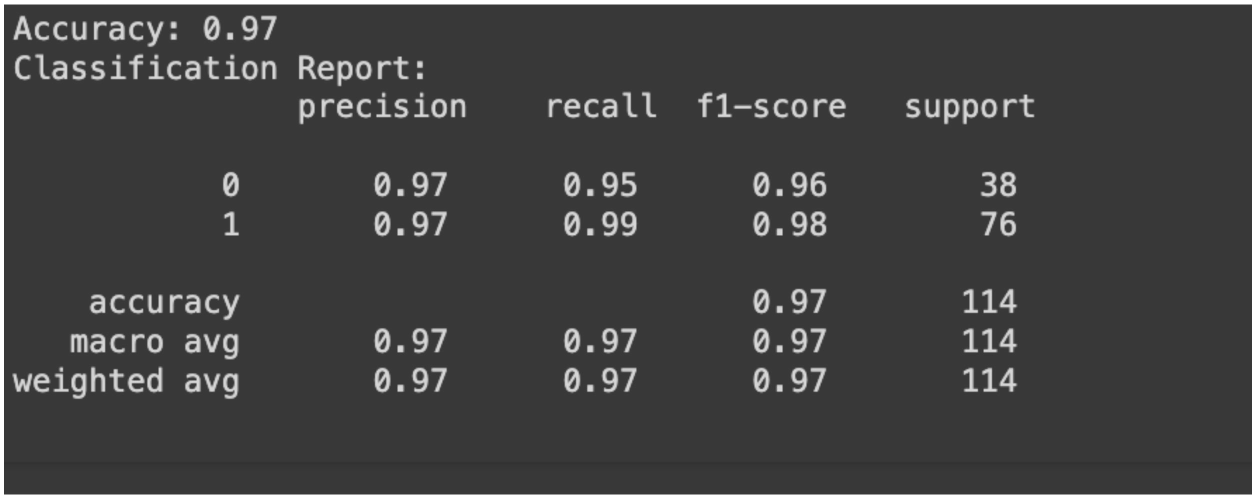
Random Forest Classifier

```
random_states = [ 4,10,20,50,67,78,98,34, 42,87,98,420]

for random_state in random_states:

    rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=random_state)



    rf_classifier.fit(X_train_cancer, y_train_cancer)
    y_pred_rf = rf_classifier.predict(X_test_cancer)


    accuracy_rf = accuracy_score(y_test_cancer, y_pred_rf)

    print(f'Accuracy (Random Forest) with Random State {rando
m_state}: {accuracy_rf:.2f}')
```



```
Accuracy (Random Forest) with Random State 4: 0.94
Accuracy (Random Forest) with Random State 10: 0.96
Accuracy (Random Forest) with Random State 20: 0.96
Accuracy (Random Forest) with Random State 50: 0.96
Accuracy (Random Forest) with Random State 67: 0.96
Accuracy (Random Forest) with Random State 78: 0.97
Accuracy (Random Forest) with Random State 98: 0.96
Accuracy (Random Forest) with Random State 34: 0.98
Accuracy (Random Forest) with Random State 42: 0.96
Accuracy (Random Forest) with Random State 87: 0.96
Accuracy (Random Forest) with Random State 98: 0.96
Accuracy (Random Forest) with Random State 420: 0.96
```

```
rf_classifier = RandomForestClassifier(n_estimators=100, rand
om_state=78)


rf_classifier.fit(X_train_cancer, y_train_cancer)

y_pred_rf = rf_classifier.predict(X_test_cancer)

accuracy = accuracy_score(y_test_cancer, y_pred_rf)
classification_rep = classification_report(y_test_cancer, y_p
red_rf)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_rep)
```

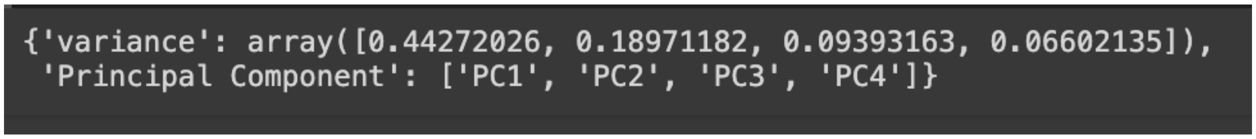


```
Accuracy: 0.97
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.95      0.96        38
           1       0.97      0.99      0.98        76

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

Applying PCA

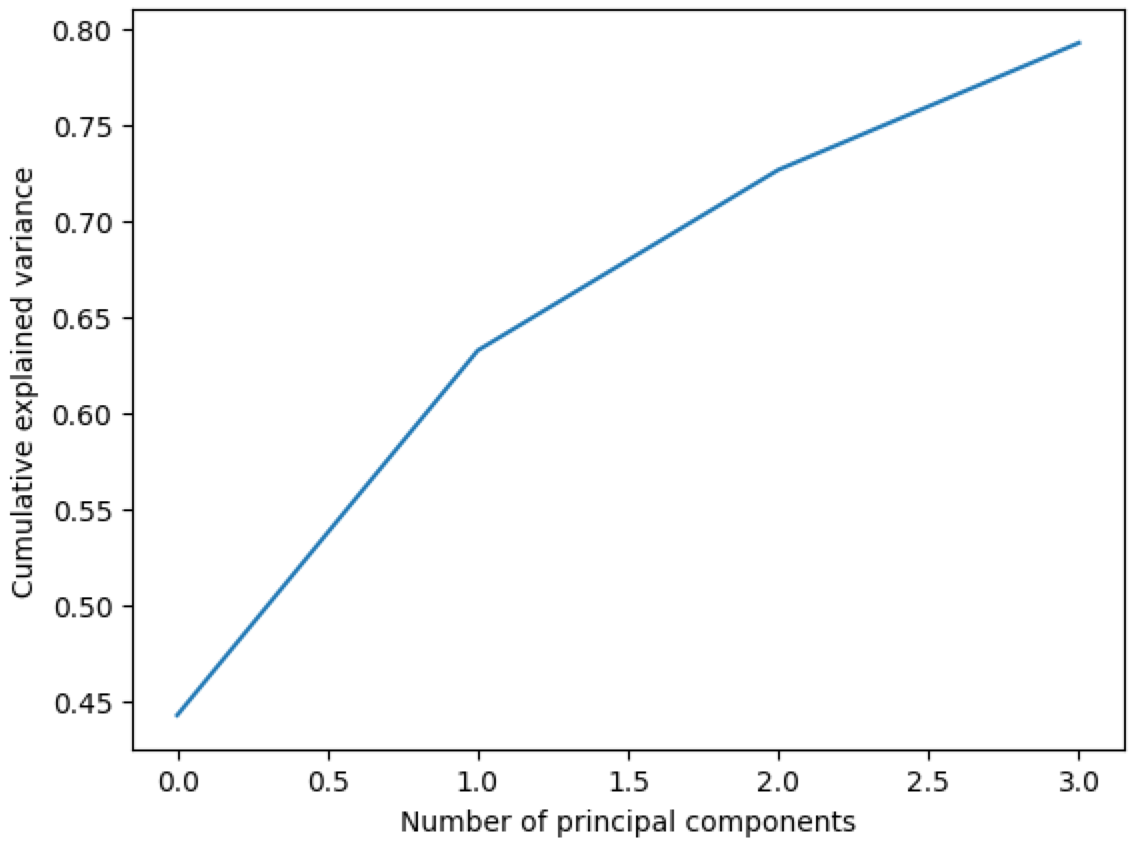
```
covariance_matrix = np.cov(X_cancer.T)
pca = PCA(n_components=4)
pca = pca.fit(X_cancer)
explained_variance = pca.explained_variance_ratio_

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming pca.explained_variance_ratio_ has 4 elements
pca_data = {
    'variance': pca.explained_variance_ratio_,
    'Principal Component': ['PC1', 'PC2', 'PC3', 'PC4'][:len
(pca.explained_variance_ratio_)]
}

dataframe = pd.DataFrame(pca_data)
```

```
sns.barplot(x='Principal Component', y="variance", data=dataf
rame, color="b")
plt.show()

pca_data = {
    'variance': pca.explained_variance_ratio_,
    'Principal Component': ['PC1', 'PC2', 'PC3', 'PC4'][:len
(pca.explained_variance_ratio_)]
}

pca_data
```

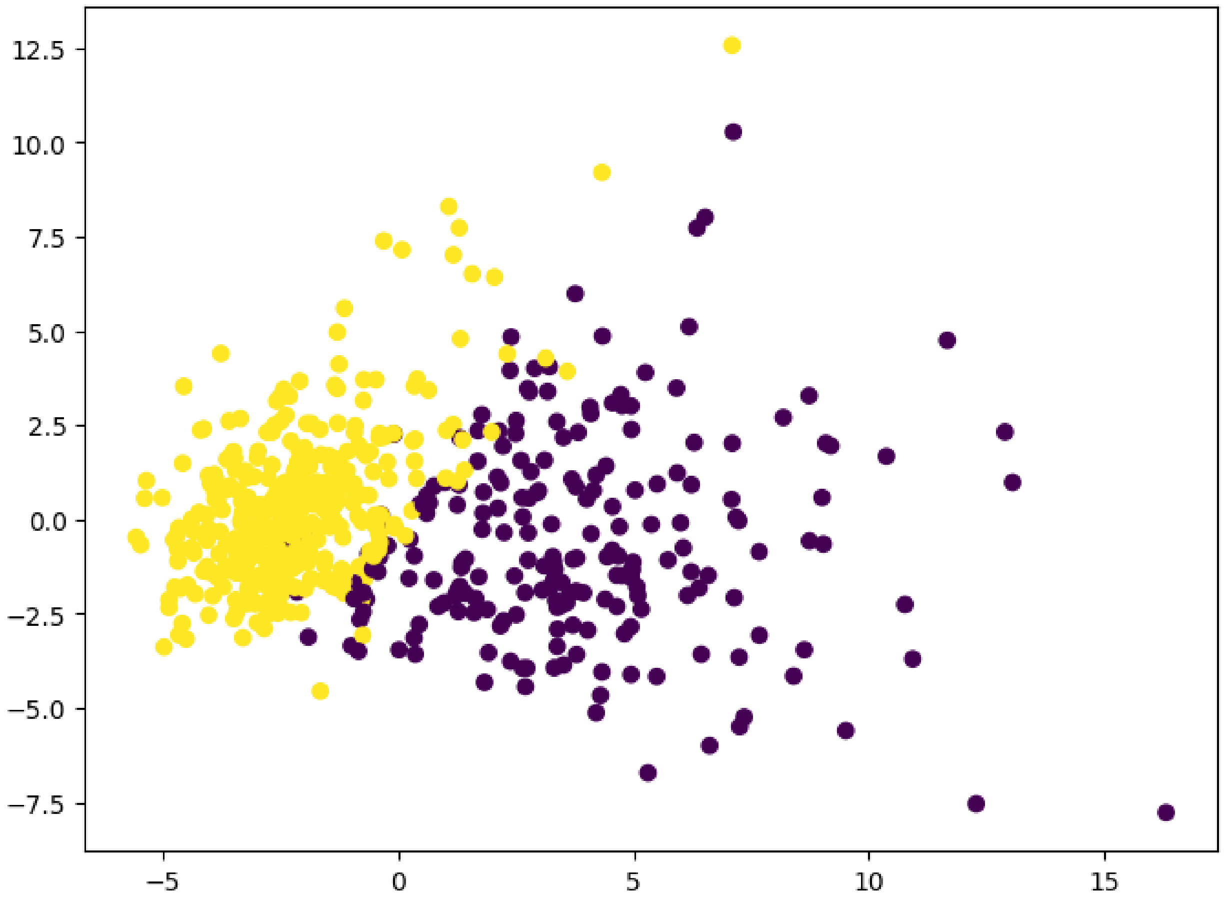


```
{'variance': array([0.44272026, 0.18971182, 0.09393163, 0.06602135]),
 'Principal Component': ['PC1', 'PC2', 'PC3', 'PC4']}
```

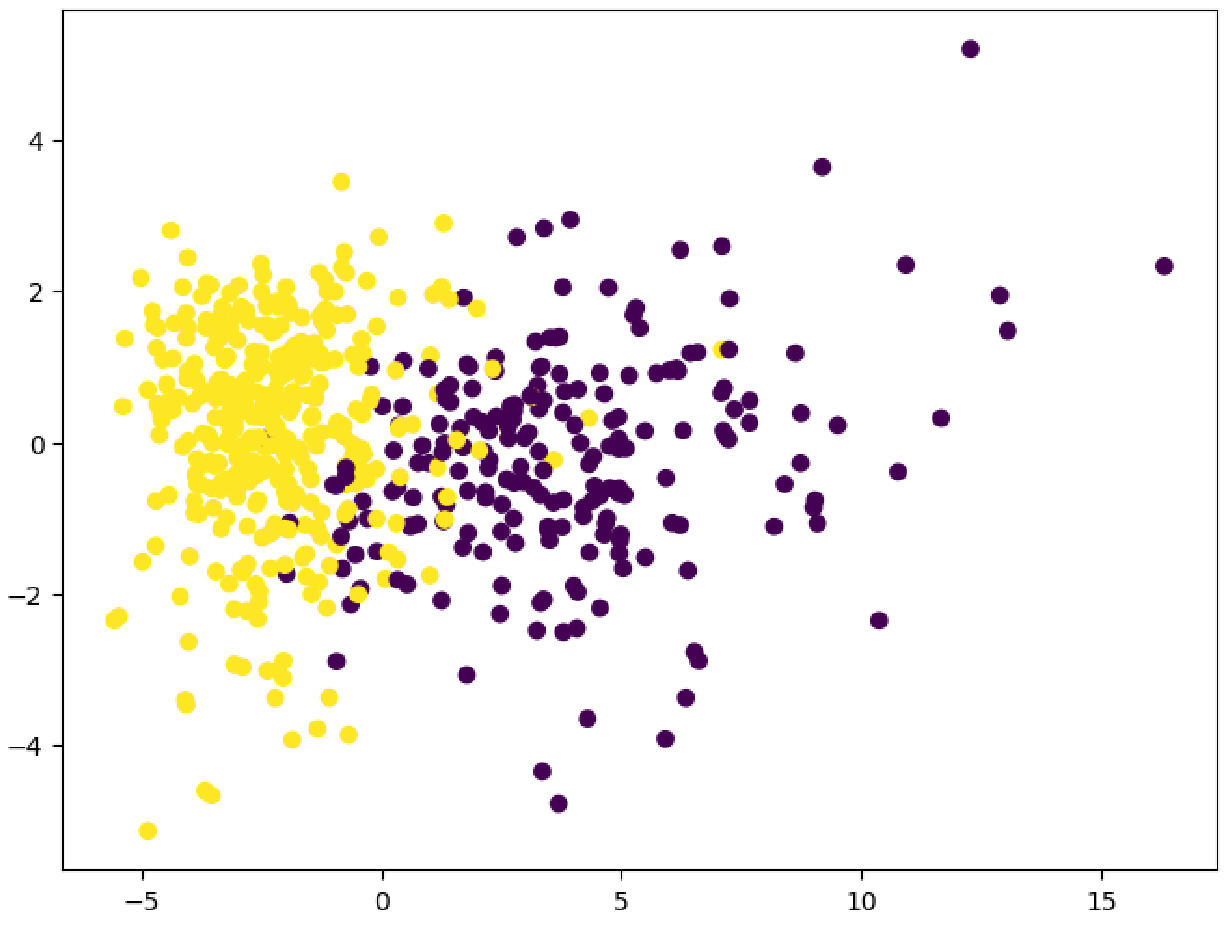
```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of principal components')
plt.ylabel('Cumulative explained variance')
plt.show()
```



```
pca_2 = PCA(n_components = 2 )
pca_2 = pca_2.fit(X_cancer)
pca_2d = pca_2.transform(X_cancer)

plt.figure(figsize=(8,6))

plt.scatter(pca_2d[:,0], pca_2d[:,1],c=y_cancer)
plt.show()

pca_4 = PCA(n_components =4 )
pca_4 = pca_4.fit(X_cancer)
pca_4d = pca_4.transform(X_cancer)
plt.figure(figsize=(8,6))
plt.scatter(pca_4d[:,0], pca_4d[:,3],c=y_cancer)
plt.show()
```

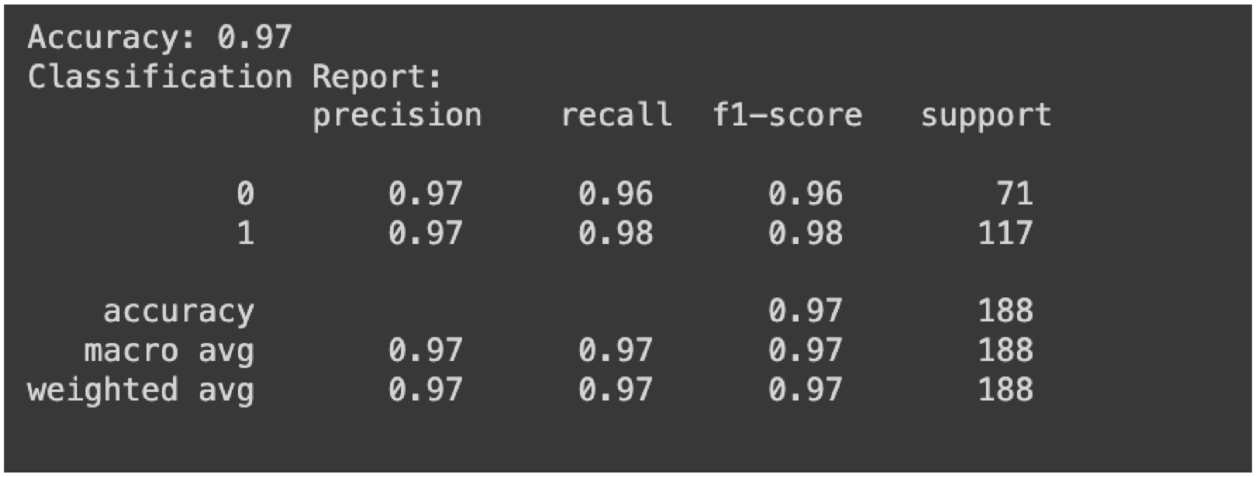


GRAPH WITH PCA COMPONENT = 2 it shows that classification is not so clear



GRAPH WITH PCA COMPONENT = 4 . it is also not clear so in short PCA should not be used with this dataset as classification is not so clear so accuracy will also be low .

```
# Accuracy after pca
accuracy = accuracy_score(y_test2, y_pred_rf1)
classification_rep = classification_report(y_test2, y_pred_rf
1)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_rep)
```



```
Accuracy: 0.97
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96        71
           1       0.97      0.98      0.98       117

    accuracy                           0.97       188
   macro avg       0.97      0.97      0.97       188
weighted avg       0.97      0.97      0.97       188
```

No change in Accuracy

Google Colaboratory

https://colab.research.google.com/drive/1PGWUyDMgy3YV
nGL7oz5Y-0T52h20RyMN#scrollTo=c7oZ-v8dMI49