**Interpret the results of PCA analysis**

**To Interpret the results of PCA analysis**

**Aim:**

To Interpret the results of PCA analysis using iris dataset

**Algorithm:**

**Step 1** Load the Iris dataset and convert it into a DataFrame including feature values and target labels.

**Step 2**: Separate the feature matrix and target column from the DataFrame.

**Step 3**: Standardize the feature data to ensure equal contribution from all variables.

**Step 4**: Apply PCA to reduce the feature dimensions to two principal components and create a new DataFrame with results.

**Step 5**: Visualize the 2D PCA output using a scatter plot and print the explained variance ratio and total variance retained.

**Program:**

# Interpret the results of PCA analysis use iris dataset

import pandas as pd

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

# Load the Iris dataset

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['target'] = iris.target

# Separate features and target variable

X = df.drop('target', axis=1)

y = df['target']

# Standardize the features

x = StandardScaler().fit\_transform(X)

# Apply PCA with 2 components

pca = PCA(n\_components=2)

principalComponents = pca.fit\_transform(x)

principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])

principalDf['target'] = y

# Visualize the results

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

ax = fig.add\_subplot(1,1,1)

ax.set\_xlabel('Principal Component 1', fontsize = 15)

ax.set\_ylabel('Principal Component 2', fontsize = 15)

ax.set\_title('2 component PCA', fontsize = 20)

targets = [0, 1, 2]

colors = ['r', 'g', 'b']

for target, color in zip(targets,colors):

    indicesToKeep = principalDf['target'] == target

    ax.scatter(principalDf.loc[indicesToKeep, 'principal component 1']

               , principalDf.loc[indicesToKeep, 'principal component 2']

               , c = color

               , s = 50)

ax.legend(targets)

ax.grid()

plt.show()

# Explained variance ratio

print("Explained Variance Ratio:", pca.explained\_variance\_ratio\_)

print("Total Explained Variance:", sum(pca.explained\_variance\_ratio\_))

**OUTPUT**



