Experiment No. 6
Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:
Date of Submission:

Aim: Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model.

Objective: Able to perform various feature engineering tasks, perform dimetionality reduction on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

Theory:

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

Dataset:

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Code:

```
import pandas as pd
import numpy as np

data = pd.read_csv('/content/adult.csv')
print(data)

data = data.replace(["?"], np.nan)
print(data.isnull().sum())
data = data.drop(["fnlwgt","education"], axis = 1)

categorical_columns = ["workclass", "occupation", "native-country"]

for column in categorical_columns:
    mode_value = data[column].mode()[0]  # Calculate the mode
    data[column].fillna(mode_value, inplace=True)  # Fill missing values with mode

# Check again for missing values to ensure they are filled
print(data.isnull().sum())

colname = []

for i in data.columns:
    if(data[i].dtype == "object"):
        colname.append(i)

colname

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in colname:
    data[i] = le.fit_transform(data[i])
print(data.head())

X = data.values[:, :-1]
Y = data.values[:, -1]

print(X)
print(Y)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Assuming X and Y are already defined from your previous steps
# Split the dataset into training and testing sets (70% train, 30% test)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=10)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)  # Fit and transform on training data
X_test = scaler.transform(X_test)        # Only transform on testing data

# Apply PCA without reducing dimensions (to analyze variance)
pca = PCA(n_components=None)
X_train_pca_full = pca.fit_transform(X_train)  # Fit PCA and transform training data
X_test_pca_full = pca.transform(X_test)        # Transform testing data

# Print explained variance ratio for each principal component
explained_variance_full = pca.explained_variance_ratio_
print("\nExplained Variance (All Components):", explained_variance_full)

# Now apply PCA to retain 75% of the variance
pca = PCA(n_components=0.75)
X_train = pca.fit_transform(X_train)  # Fit PCA and transform training data
X_test = pca.transform(X_test)        # Transform testing data

# Print explained variance ratio for the selected components
explained_variance_reduced = pca.explained_variance_ratio_
print("\nExplained Variance (75% Components):", explained_variance_reduced)

# Number of components selected to explain 75% variance
n_components_selected = pca.n_components_
print("\nNumber of Components Selected:", n_components_selected)

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

print("Confusion Matrix = ")
print(confusion_matrix(Y_test, Y_pred), "\n")
print("Accuracy Score = ", accuracy_score(Y_test, Y_pred), "\n")
```

```
pca = PCA(n_components=0.8)
X_train = pca.fit_transform(X_train)  # Fit PCA and transform training data
X_test = pca.transform(X_test)
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

print("Confusion Matrix = ")
print(confusion_matrix(Y_test, Y_pred), "\n")
print("Accuracy Score = ", accuracy_score(Y_test, Y_pred), "\n")
```

Conclusion:

The impact of dimensionality reduction on the performance metrics of your logistic regression model can be summarized as follows:

1. Accuracy

Before PCA: The accuracy score was approximately **0.812**.

After PCA: The accuracy score slightly decreased to about **0.809**.

Observation: The model maintains a relatively high accuracy both before and after dimensionality reduction. However, the slight drop indicates that reducing dimensions may have removed some important information that could contribute to overall accuracy.

2. Precision

Class 0: Precision remained roughly the same (0.83).

Class 1: Precision decreased from **0.68** to **0.67**.

Observation: Precision for Class 1 slightly decreased, suggesting that while the model still predicts most of the true positives correctly for Class 1, it may have become slightly less reliable in distinguishing between Class 0 and Class 1 after PCA.

3. Recall

Class 0: Recall remained constant at **0.94**.

Class 1: Recall decreased from **0.41** to **0.40**.

Observation: The drop in recall for Class 1 indicates that the model missed slightly more true positives after PCA. This is crucial since low recall can be problematic in scenarios where missing a positive instance (like a medical diagnosis) is costly.

4. F1 Score

Class 0: F1 Score stayed at **0.88**.

Class 1: F1 Score decreased from **0.51** to **0.50**.

Observation: The F1 Score for Class 1 reflects the combined effect of precision and recall. The slight decrease signals a decline in the model's performance in identifying Class 1 instances, suggesting that dimensionality reduction might have caused some loss of critical features for this class.

5. Overall Metrics (Macro and Weighted Averages)
Macro Average:
Before PCA: Precision **0.76**, Recall **0.68**, F1 Score **0.70**
After PCA: Precision **0.75**, Recall **0.67**, F1 Score **0.69**
Weighted Average:
Before PCA: Precision **0.80**, Recall **0.81**, F1 Score **0.79**
After PCA: Precision **0.79**, Recall **0.81**, F1 Score **0.79**