Experiment No. 7
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:

Conclusion:

Original Data:

Accuracy: 85.37%

Precision: 72.78%

Recall: 60.70%

F1 Score: 66.19%

PCA-Reduced Data:

Accuracy: 85.32%

Precision: 74.60%

Recall: 57.32%

F1 Score: 64.83%

In conclusion, both the original data and the PCA-reduced data exhibit similar overall model performance. The PCA-reduced data, despite having fewer dimensions, achieves a relatively competitive level of performance with a slightly improved precision, although at the expense of recall. The choice between using the original or PCA-reduced data depends on your specific goals and the trade-offs between model complexity, interpretability, and performance.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,confusion_matrix
```

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

```
# Separate the target variable (income) from the features
X = data.drop('income', axis=1)
y = data['income']
```

```
categorical_columns = X.select_dtypes(include=['object']).columns
label_encoder = LabelEncoder()
X[categorical_columns] = X[categorical_columns].apply(label_encoder.fit_transform)
```

```
# Step 2: Apply PCA for dimensionality reduction
# You can adjust the number of components (n_components) as needed
n_components = 14  # You can change this value
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X)
```

```
# Step 3: Train classifiers on Original and Reduced-dimensioned data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_pca_train, X_pca_test, y_pca_train, y_pca_test = train_test_split(X_pca, y, test_size=0.2, random_state=
```

```
# Initialize a Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)
```

```
# Fit the classifier on the original data
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
```

```
# Calculate performance metrics for the original data
accuracy_original = accuracy_score(y_test, y_pred)
precision_original = precision_score(y_test, y_pred, pos_label='>50K')
recall_original = recall_score(y_test, y_pred, pos_label='>50K')
f1_score_original = f1_score(y_test, y_pred, pos_label='>50K')
```

```
# Fit the classifier on the reduced-dimensioned data
rf_classifier.fit(X_pca_train, y_pca_train)
y_pca_pred = rf_classifier.predict(X_pca_test)
```

```
# Calculate performance metrics for the reduced-dimensioned data
accuracy_pca = accuracy_score(y_pca_test, y_pca_pred)
precision_pca = precision_score(y_pca_test, y_pca_pred, pos_label='>50K')
recall_pca = recall_score(y_pca_test, y_pca_pred, pos_label='>50K')
f1_score_pca = f1_score(y_pca_test, y_pca_pred, pos_label='>50K')
```

```
confusion_matrix_original = confusion_matrix(y_test, y_pred)
print("Confusion Matrix for Original Data:")
print(confusion_matrix_original)
```

```
Confusion Matrix for Original Data:
[[4627  349]
 [ 604  933]]
```

```
# Print performance metrics for both original and PCA-reduced data
print("Performance Metrics for Original Data:")
print(f"Accuracy: {accuracy_original}")
```

```
print(f"Precision: {precision_original}")
print(f"Recall: {recall_original}")
print(f"F1 Score: {f1_score_original}")
```

```
Performance Metrics for Original Data:
Accuracy: 0.8536772608628896
Precision: 0.7277691107644306
Recall: 0.6070266753415745
F1 Score: 0.6619368570415041
```

```
print("\nPerformance Metrics for PCA-Reduced Data:")
print(f"Accuracy: {accuracy_pca}")
print(f"Precision: {precision_pca}")
print(f"Recall: {recall_pca}")
print(f"F1 Score: {f1_score_pca}")
```

```
Performance Metrics for PCA-Reduced Data:
Accuracy: 0.8532166436358053
Precision: 0.745977984758679
Recall: 0.5731945348080677
F1 Score: 0.648270787343635
```

```
# Calculate the confusion matrix for the PCA-reduced data
confusion_matrix_pca = confusion_matrix(y_pca_test, y_pca_pred)
print("\nConfusion Matrix for PCA-Reduced Data:")
print(confusion_matrix_pca)
```

```
Confusion Matrix for PCA-Reduced Data:
[[4676  300]
 [ 656  881]]
```