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 & OUTPUT:-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
url = "adult_dataset.csv"
columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',
        'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
        'hours-per-week', 'native-country', 'income']
data = pd.read_csv(url, names=columns, na_values=' ?')
data.dropna(inplace=True)

label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])

X = data.drop('income', axis=1)
y = data['income']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
accuracy_scores = []
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

```
y_pred = log_reg.predict(X_test)
accuracy_without_pca = accuracy_score(y_test, y_pred)
accuracy_scores.append(accuracy_without_pca)
print("Logistic Regression without PCA")
print(f"Accuracy: {accuracy_without_pca:.4f}")
print(classification_report(y_test, y_pred))
```

```
Logistic Regression without PCA
Accuracy: 0.7881
              precision    recall  f1-score   support

           0       0.81      0.94      0.87      7414
           1       0.62      0.32      0.42      2355

    accuracy                           0.79      9769
   macro avg       0.72      0.63      0.64      9769
weighted avg       0.77      0.79      0.76      9769
```

```
pca_all = PCA()
X_train_pca_all = pca_all.fit_transform(X_train)
X_test_pca_all = pca_all.transform(X_test)
log_reg_pca_all = LogisticRegression()
log_reg_pca_all.fit(X_train_pca_all, y_train)
y_pred_pca_all = log_reg_pca_all.predict(X_test_pca_all)

accuracy_pca_all = accuracy_score(y_test, y_pred_pca_all)
accuracy_scores.append(accuracy_pca_all)
print("\nLogistic Regression with PCA (whole dataset)")
print(f"Accuracy: {accuracy_pca_all:.4f}")
print(classification_report(y_test, y_pred_pca_all))
```

```
Logistic Regression with PCA (whole dataset)
Accuracy: 0.7881
              precision    recall  f1-score   support

           0       0.81      0.94      0.87      7414
           1       0.62      0.32      0.42      2355

    accuracy                           0.79      9769
   macro avg       0.72      0.63      0.64      9769
weighted avg       0.77      0.79      0.76      9769
```

```
pca_50 = PCA(0.5)  # Retain components that explain 50% of the variance
X_train_pca_50 = pca_50.fit_transform(X_train)
X_test_pca_50 = pca_50.transform(X_test)

log_reg_pca_50 = LogisticRegression()
log_reg_pca_50.fit(X_train_pca_50, y_train)
y_pred_pca_50 = log_reg_pca_50.predict(X_test_pca_50)

accuracy_pca_50 = accuracy_score(y_test, y_pred_pca_50)
accuracy_scores.append(accuracy_pca_50)

print("\nLogistic Regression with PCA (variance explained ≥ 0.5)")
print(f"Accuracy: {accuracy_pca_50:.4f}")
print(classification_report(y_test, y_pred_pca_50))
```

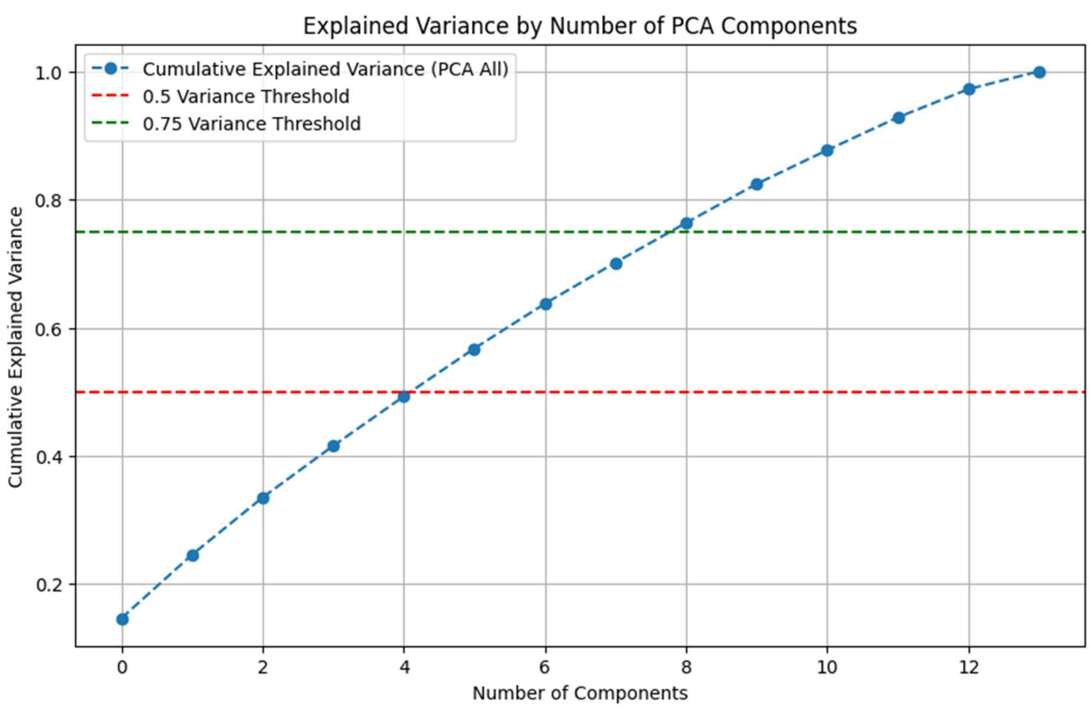
```
Logistic Regression with PCA (variance explained ≥ 0.5)
Accuracy: 0.7804
              precision    recall  f1-score   support

           0       0.81      0.94      0.87      7414
           1       0.59      0.29      0.39      2355

    accuracy                           0.78      9769
   macro avg       0.70      0.61      0.63      9769
weighted avg       0.75      0.78      0.75      9769
```

```
pca_75 = PCA(0.75)  # Retain components that explain 75% of the variance
X_train_pca_75 = pca_75.fit_transform(X_train)
X_test_pca_75 = pca_75.transform(X_test)
log_reg_pca_75 = LogisticRegression()
log_reg_pca_75.fit(X_train_pca_75, y_train)
y_pred_pca_75 = log_reg_pca_75.predict(X_test_pca_75)

accuracy_pca_75 = accuracy_score(y_test, y_pred_pca_75)
accuracy_scores.append(accuracy_pca_75)
print("\nLogistic Regression with PCA (variance explained ≥ 0.75)")
print(f"Accuracy: {accuracy_pca_75:.4f}")
print(classification_report(y_test, y_pred_pca_75))
```

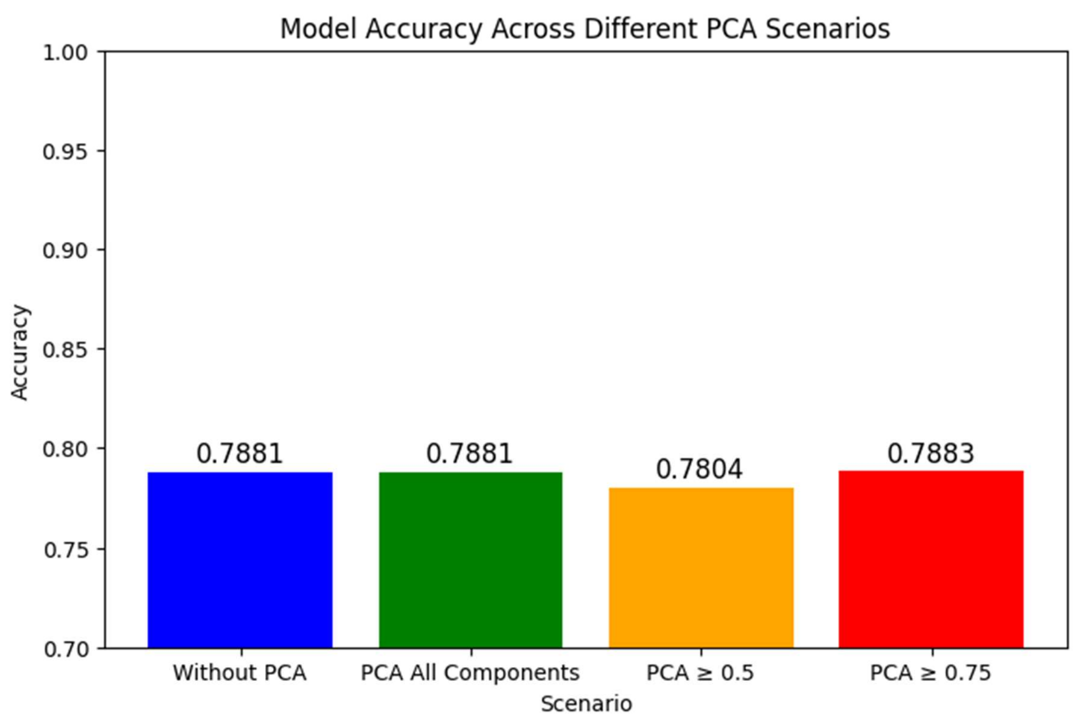
```
Logistic Regression with PCA (variance explained ≥ 0.75)
Accuracy: 0.7883
              precision    recall  f1-score   support

           0       0.81      0.94      0.87      7414
           1       0.63      0.30      0.41      2355

    accuracy                           0.79      9769
```

macro avg	0.72	0.62	0.64	9769
weighted avg	0.77	0.79	0.76	9769

```
# Plotting Explained Variance for Each Scenario
plt.figure(figsize=(10, 6))
explained_variance = np.cumsum(pca_all.explained_variance_ratio_)
plt.plot(explained_variance, marker='o', linestyle='--', label='Cumulative Explained Variance (PCA All)')
plt.axhline(y=0.5, color='r', linestyle='--', label='0.5 Variance Threshold')
plt.axhline(y=0.75, color='g', linestyle='--', label='0.75 Variance Threshold')
plt.title('Explained Variance by Number of PCA Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```

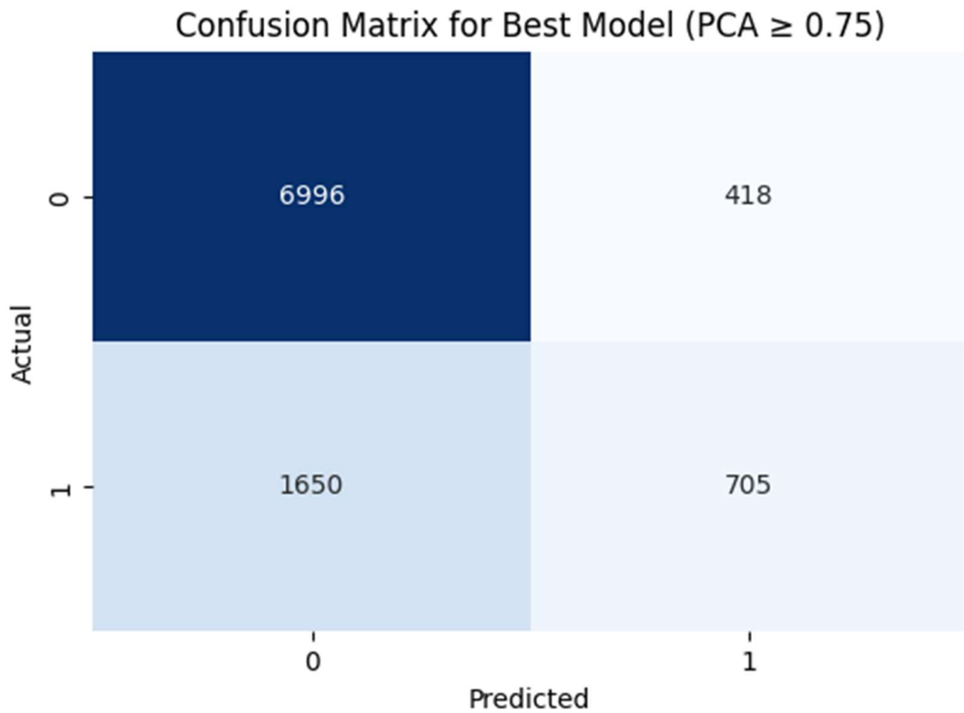


```
# Accuracy Comparison Graph
plt.figure(figsize=(8, 5))
scenarios = ['Without PCA', 'PCA All Components', 'PCA ≥ 0.5', 'PCA ≥ 0.75']
plt.bar(scenarios, accuracy_scores, color=['blue', 'green', 'orange', 'red'])
plt.title('Model Accuracy Across Different PCA Scenarios')
plt.xlabel('Scenario')
plt.ylabel('Accuracy')
plt.ylim(0.7, 1.0)
for i, v in enumerate(accuracy_scores):
plt.text(i, v + 0.005, f"{v:.4f}", ha='center', fontsize=12)
plt.show()
```




```
# Confusion Matrix for the Best Model (You can choose the best model here)
best_model_conf_matrix = confusion_matrix(y_test, y_pred_pca_75)

plt.figure(figsize=(6, 4))
sns.heatmap(best_model_conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
```

```
plt.title('Confusion Matrix for Best Model (PCA ≥ 0.75)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



Conclusion:

Dimensionality reduction can significantly influence the performance metrics of a model applied to the Adult Census Income dataset. By reducing the number of features, the model can focus on the most relevant information, often leading to improved accuracy and precision due to the elimination of noise and redundancy. However, while precision may increase, recall might remain stable or even decline if important features related to positive cases are lost. Consequently, the F1 score, which balances precision and recall, may improve if precision rises without a significant drop in recall, or decline if recall suffers. Overall, the impact of dimensionality reduction on these metrics highlights the importance of carefully selecting which features to retain.