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.

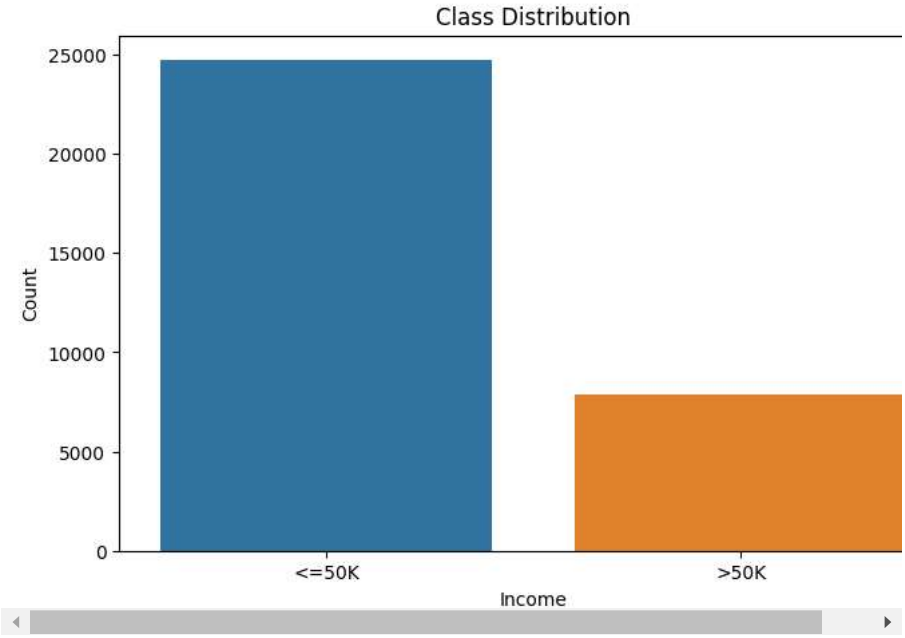
Code:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df=pd.read_csv("/content/adult.csv")
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900

```
plt.figure(figsize=(8,5))
sns.countplot(data=df, x='income')
plt.title("Class Distribution")
plt.xlabel("Income")
plt.ylabel("Count")
plt.show()
```




```
plt.figure(figsize=(14,10))
sns.heatmap(df.corr(),annot=True,fmt='.2f')
plt.show()
```

```
<ipython-input-120-ce7fadf8682c>:2: FutureWarning: The default value of numeric_only in
  sns.heatmap(df.corr(),annot=True,fmt='.2f')
```


```
df[df == '?'] = np.nan
df.isnull().sum()
```

```
age                0
workclass       1836
fnlwgt             0
education          0
education.num      0
marital.status     0
occupation      1843
relationship       0
race               0
sex                0
capital.gain       0
capital.loss       0
hours.per.week     0
native.country   583
income             0
dtype: int64
```

```
for col in ['workclass', 'occupation', 'native.country']:
    df[col].fillna(df[col].mode()[0], inplace=True)
df.isnull().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

```
X = df.drop(['income'], axis=1)

y = df['income']
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)


from sklearn import preprocessing

categorical = ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.country']
for feature in categorical:
        label = preprocessing.LabelEncoder()
        X_train[feature] = label.fit_transform(X_train[feature])
        X_test[feature] = label.transform(X_test[feature])


from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = pd.DataFrame(scaler.fit_transform(X_train), columns = X.columns)

X_test = pd.DataFrame(scaler.transform(X_test), columns = X.columns)
X_train.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	ca
0	0.101484	2.600478	-1.494279	-0.332263	1.133894	-0.402341	-0.782234	2.214196	0.39298	-1.430470	-0.145189	
1	0.028248	-1.884720	0.438778	0.184396	-0.423425	-0.402341	-0.026696	-0.899410	0.39298	0.699071	-0.145189	
2	0.247956	-0.090641	0.045292	1.217715	-0.034095	0.926666	-0.782234	-0.276689	0.39298	-1.430470	-0.145189	
3	-0.850587	-1.884720	0.793152	0.184396	-0.423425	0.926666	-0.530388	0.968753	0.39298	0.699071	-0.145189	
4	-0.044989	-2.781760	-0.853275	0.442726	1.523223	-0.402341	-0.782234	-0.899410	0.39298	0.699071	-0.145189	

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
LR = LogisticRegression()
LR.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = LR.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.8216808271061521
```

```
from sklearn.decomposition import PCA
pca = PCA()
X_train = pca.fit_transform(X_train)
pca.explained_variance_ratio_
```

```
array([0.14757168, 0.10182915, 0.08147199, 0.07880174, 0.07463545,
       0.07274281, 0.07009602, 0.06750902, 0.0647268 , 0.06131155,
       0.06084207, 0.04839584, 0.04265038, 0.02741548])
```

```
X = df.drop(['income','native.country'], axis=1)
y = df['income']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
categorical = ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex']
for feature in categorical:
        label = preprocessing.LabelEncoder()
        X_train[feature] = label.fit_transform(X_train[feature])
        X_test[feature] = label.transform(X_test[feature])


X_train = pd.DataFrame(scaler.fit_transform(X_train), columns = X.columns)

X_test = pd.DataFrame(scaler.transform(X_test), columns = X.columns)
LR1 = LogisticRegression()
LR1.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = LR1.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.8212713686150066
```

```
X = df.drop(['income','native.country', 'hours.per.week'], axis=1)
y = df['income']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

```
categorical = ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex']
for feature in categorical:
        label = preprocessing.LabelEncoder()
        X_train[feature] = label.fit_transform(X_train[feature])
        X_test[feature] = label.transform(X_test[feature])
```

```
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns = X.columns)

X_test = pd.DataFrame(scaler.transform(X_test), columns = X.columns)
LR2 = LogisticRegression()
LR2.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = LR2.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.8227044733340158
```

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:")
print(classification_rep)
```

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

       <=50K       0.84      0.95      0.89      7410
        >50K       0.72      0.43      0.54      2359

    accuracy                           0.82      9769
   macro avg       0.78      0.69      0.72      9769
weighted avg       0.81      0.82      0.81      9769
```

```
confusion_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion_mat)
```

```
Confusion Matrix:
[[7012  398]
 [1334 1025]]
```

Conclusion:

1. The accuracy of the model using logistic regression was 82.16% After applying dimensionality reduction the accuracy obtained using logistic regression was 82.27%.
2. The precision after applying PCA was 0.84 for income <=50 and 0.72 for income >50.
3. The recall for income <=50 came out to be 0.95 and 0.43 for income >50.
4. The F1 Score for income <=50 was 0.89 and 0.54 for income >50..
5. We can conclude that by taking 12 features we obtain the optimal results. By implementing dimensionality reduction the performance metrics like accuracy recall and F1 score somewhat improved.