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:

Comment on the impact of dimensionality reduction on the accuracy, precision, recall and F1 score.

	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

Accuracy: The accuracy has decreased from what it might have been with the original, higher-dimensional data. This reduction in accuracy is likely due to the loss of information when reducing dimensionality.

Precision: Precision for the '>50K' class has decreased significantly, indicating a higher rate of false positives compared to the original dataset. This is a common trade-off with dimensionality reduction, as it might lead to a less precise separation of classes.

Recall: Recall for the '>50K' class has also decreased, indicating that the dimensionality reduction might be causing the model to miss some positive instances. It's a trade-off between precision and recall.

F1-score: The F1 score, which balances precision and recall, has also dropped. This suggests that the dimensionality reduction has affected the model's ability to provide a balance between precision and recall.

```
import numpy as np
import pandas as pd
import os
```

```
for dirname, _, filenames in os.walk('/content/adult (1).csv'):
      for filename in filenames:
            print(os.path.join(dirname, filename))
```

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

```
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	rela
0	90	Private	77053	HS-grad	9	Widowed	Prof-specialty	No
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	No
2	66	Private	186061	Some-college	10	Widowed	Prof-specialty	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	

```
df.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education.num',
       'marital.status', 'occupation', 'relationship', 'race', 'sex',
       'capital.gain', 'capital.loss', 'hours.per.week', 'native.country',
       'income'],
      dtype='object')
```

```
df.shape
```

```
(32561, 15)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
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
```

```
df.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}, inplace=True)
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	occupati
0	0.101484	2.600478	-1.494279	-0.332263	1.133894	-0.402341	-0.7822
1	0.028248	-1.884720	0.438778	0.184396	-0.423425	-0.402341	-0.0266
2	0.247956	-0.090641	0.045292	1.217715	-0.034095	0.926666	-0.7822
3	-0.850587	-1.884720	0.793152	0.184396	-0.423425	0.926666	-0.5303
4	-0.044989	-2.781760	-0.853275	0.442726	1.523223	-0.402341	-0.7822

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

LR = LogisticRegression()
LR.fit(X_train, y_train)
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_

X = df.drop(['income'], 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', 'native.country']
for feature in categorical:
          lable = 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)
pca= PCA()
pca.fit(X_train)
cumsum = np.cumsum(pca.explained_variance_ratio_)
dim = np.argmax(cumsum >= 0.90) + 1
print('The number of dimensions required to preserve 90% of variance is',dim)
```

```
The number of dimensions required to preserve 90% of variance is 12
```

```
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

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

```
0.8227044733340158
```

```
from sklearn.metrics import confusion_matrix
import pandas as pd
confusion = confusion_matrix(y_test, y_pred)
df_confusion = pd.DataFrame(confusion, columns=['Predicted No', 'Predicted Yes'], index=['Actual No', 'Actual Yes'])
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              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
```