Experiment No. 7
Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model
Date of Performance: 11/09/2023
Date of Submission: 06/10/2023

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 # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    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  \
0   90         ?   77053       HS-grad              9         Widowed

1   82   Private  132870       HS-grad              9         Widowed

2   66         ?  186061  Some-college             10         Widowed

3   54   Private  140359       7th-8th              4        Divorced

4   41   Private  264663  Some-college             10       Separated


          occupation   relationship   race     sex  capital.gain  \
0                  ?  Not-in-family  White  Female             0
1    Exec-managerial  Not-in-family  White  Female             0
2                  ?      Unmarried  Black  Female             0
3  Machine-op-inspct      Unmarried  White  Female             0
4     Prof-specialty      Own-child  White  Female             0

   capital.loss  hours.per.week native.country income
0          4356              40  United-States  <=50K
1          4356              18  United-States  <=50K
2          4356              40  United-States  <=50K
3          3900              40  United-States  <=50K
4          3900              40  United-States  <=50K

df.describe().T

                  count           mean            std      min
25%  \
age             32561.0      38.581647      13.640433     17.0
28.0
fnlwgt          32561.0  189778.366512  105549.977697  12285.0
```

```
117827.0
education.num   32561.0      10.080679       2.572720      1.0
9.0
capital.gain    32561.0    1077.648844    7385.292085      0.0
0.0
capital.loss    32561.0      87.303830     402.960219      0.0
0.0
hours.per.week  32561.0      40.437456      12.347429      1.0
40.0

                     50%       75%        max
age                 37.0      48.0       90.0
fnlwgt          178356.0  237051.0  1484705.0
education.num       10.0      12.0       16.0
capital.gain         0.0       0.0    99999.0
capital.loss         0.0       0.0     4356.0
hours.per.week      40.0      45.0       99.0
```

```
df.shape
```

```
(32561, 15)
```

```
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.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
```

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

```
y = df['income']
```

```python
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  \
0  0.101484   2.600478 -1.494279  -0.332263       1.133894       -
0.402341
1  0.028248  -1.884720  0.438778   0.184396      -0.423425       -
0.402341
2  0.247956  -0.090641  0.045292   1.217715      -0.034095
0.926666
3 -0.850587  -1.884720  0.793152   0.184396      -0.423425
0.926666
4 -0.044989  -2.781760 -0.853275   0.442726       1.523223       -
0.402341

   occupation  relationship     race       sex  capital.gain
capital.loss  \
0   -0.782234      2.214196  0.39298 -1.430470     -0.145189     -
0.217407
1   -0.026696     -0.899410  0.39298  0.699071     -0.145189     -
0.217407
2   -0.782234     -0.276689  0.39298 -1.430470     -0.145189     -
0.217407
3   -0.530388      0.968753  0.39298  0.699071     -0.145189     -
0.217407
4   -0.782234     -0.899410  0.39298  0.699071     -0.145189     -
0.217407

   hours.per.week  native.country
```

```
0       -1.662414       0.262317
1       -0.200753       0.262317
2       -0.038346       0.262317
3       -0.038346       0.262317
4       -0.038346       0.262317

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

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

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()
```

```
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()

y_pred = LR2.predict(X_test)

accuracy_score(y_test, y_pred)

0.8227044733340158

X = df.drop(['income','native.country', 'hours.per.week',
'capital.loss'], 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)
```

```
LR3 = LogisticRegression()
LR3.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
y_pred = LR3.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

```
0.8186098884225612
```

```
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
```

Conclusion:

Dimensionality reduction, specifically PCA, was applied to the Adult Census Income Dataset, aiming to simplify the dataset while preserving crucial information.

1. Original Dataset: Achieved 82.17% accuracy, and other metrics weren't provided.
2. Removing 'native.country': Slightly lower accuracy (82.13%) with comparable performance.
3. Removing 'hours.per.week': Slightly higher accuracy (82.27%) with similar performance.
4. Removing 'capital.loss': Slightly lower accuracy (81.86%) with similar performance.
5. PCA to retain 90% variance (12 dimensions): The dataset was reduced effectively while preserving essential information.

In this analysis, dimensionality reduction using PCA had a minor impact on model performance:

1. Accuracy: Slightly improved from 82.17% to 82.27%.
2. Precision: Improved slightly for the ">50K" class.
3. Recall: Decreased slightly for the ">50K" class.
4. F1 Score: Decreased slightly for the ">50K" class.