Experiment No. 7
Apply Dimensionality Reduction on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:27/9/23
Date of Submission:9/10/23

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:

Performance on Original Data:

- Accuracy: The accuracy on the original data is approximately 0.849, which means that the classifier correctly predicts the income category for around 84.9% of the instances in the test set.
- Precision for '>50K' is 0.73, meaning it's correct about 73% of the time when predicting '>50K'. For '<=50K', it's 0.88, indicating an 88% correctness in predicting '<=50K'.
- Recall for '>50K' is 0.61, correctly identifying about 61% of '>50K' instances. For '<=50K', it's 0.93, correctly identifying about 93% of '<=50K' instances.
- F1-score For the '>50K' category, the F1-score is 0.66, and for the '<=50K' category, it is 0.91.

Performance on PCA Reduced Data:

- Accuracy: The accuracy on the PCA-reduced data is approximately 0.825, which is slightly lower than the accuracy on the original data.
- Precision: The precision for the '>50K' category is 0.72, and for the '<=50K' category, it is 0.87.
- Recall: The recall for the '>50K' category is 0.54, and for the '<=50K' category, it is 0.94.
- F1-score: For the '>50K' category, the F1-score is 0.61, and for the '<=50K' category, it is 0.90.

The original data outperforms the PCA-reduced data in terms of accuracy, precision, and recall. While PCA simplifies the model and reduces dimensionality, it results in providing , with a slight decrease in predictive performance. Therefore, the decision to use PCA or the original data should depend on the specific requirements and priorities of the problem, considering factors such as model complexity, training time, and the importance of accuracy in the application.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

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

```
print(data)
```

```
         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
...    ...       ...     ...           ...            ...                 ...
32556   22   Private  310152  Some-college             10       Never-married
32557   27   Private  257302    Assoc-acdm             12  Married-civ-spouse
32558   40   Private  154374       HS-grad              9  Married-civ-spouse
32559   58   Private  151910       HS-grad              9             Widowed
32560   22   Private  201490       HS-grad              9       Never-married

              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
...                  ...            ...    ...     ...           ...
32556    Protective-serv  Not-in-family  White    Male             0
32557       Tech-support           Wife  White  Female             0
32558  Machine-op-inspct        Husband  White    Male             0
32559       Adm-clerical      Unmarried  White  Female             0
32560       Adm-clerical      Own-child  White    Male             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
...             ...             ...            ...    ...
32556             0              40  United-States  <=50K
32557             0              38  United-States  <=50K
32558             0              40  United-States   >50K
32559             0              40  United-States  <=50K
32560             0              20  United-States  <=50K

[32561 rows x 15 columns]
```

```
data.describe()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.\
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.00(
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.43
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.34
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.00(
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.00(
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.00(
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.00(
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.00(

```
data.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
```

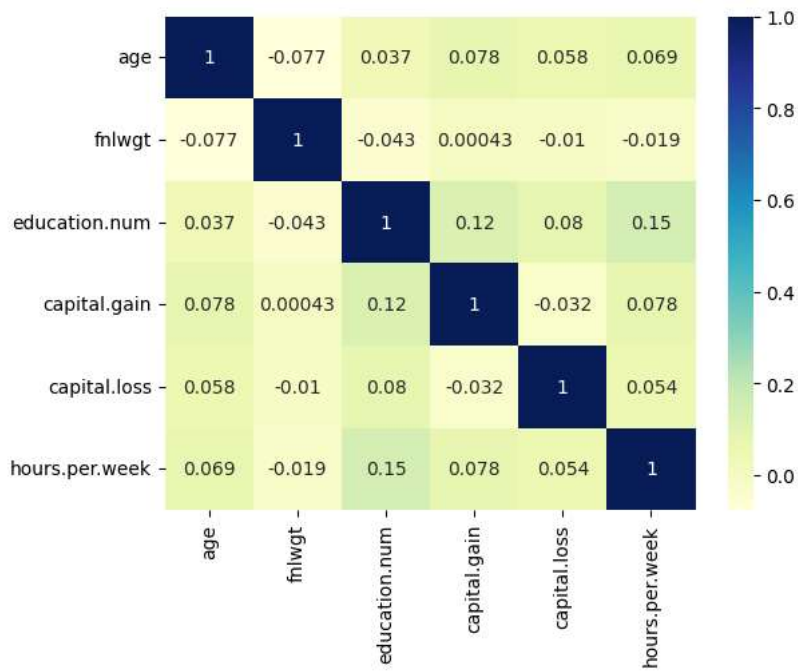
```
import matplotlib.pyplot as mp
import pandas as pd
import seaborn as sb
print(data.corr())

# plotting correlation heatmap
dataplot = sb.heatmap(data.corr(), cmap="YlGnBu", annot=True)

# displaying heatmap
mp.show()
```

```
<ipython-input-28-b698e0a536da>:4: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versior
  print(data.corr())
<ipython-input-28-b698e0a536da>:7: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versior
  dataplot = sb.heatmap(data.corr(), cmap="YlGnBu", annot=True)
                     age    fnlwgt  education.num  capital.gain  capital.loss  \
age             1.000000 -0.076646       0.036527      0.077674      0.057775
fnlwgt         -0.076646  1.000000      -0.043195      0.000432     -0.010252
education.num   0.036527 -0.043195       1.000000      0.122630      0.079923
capital.gain    0.077674  0.000432       0.122630      1.000000     -0.031615
capital.loss    0.057775 -0.010252       0.079923     -0.031615      1.000000
hours.per.week  0.068756 -0.018768       0.148123      0.078409      0.054256

                hours.per.week
age                   0.068756
fnlwgt               -0.018768
education.num         0.148123
capital.gain          0.078409
capital.loss          0.054256
hours.per.week        1.000000
```




```
# Encode categorical features using Label Encoding
categorical_features = ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.country', 'income']
for feature in categorical_features:
    label_encoder = LabelEncoder()
```

```
        data[feature] = label_encoder.fit_transform(data[feature])

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


    print(x)
    print(y)
```

```
           age workclass  fnlwgt     education  education.num      marital.status  \
    1       82   Private  132870       HS-grad              9             Widowed
    3       54   Private  140359       7th-8th              4            Divorced
    4       41   Private  264663  Some-college             10           Separated
    5       34   Private  216864       HS-grad              9            Divorced
    6       38   Private  150601          10th              6           Separated
    ...    ...       ...     ...           ...            ...                 ...
    32556   22   Private  310152  Some-college             10       Never-married
    32557   27   Private  257302    Assoc-acdm             12  Married-civ-spouse
    32558   40   Private  154374       HS-grad              9  Married-civ-spouse
    32559   58   Private  151910       HS-grad              9             Widowed
    32560   22   Private  201490       HS-grad              9       Never-married

                  occupation   relationship   race     sex  capital.gain  \
    1        Exec-managerial  Not-in-family  White  Female             0
    3      Machine-op-inspct      Unmarried  White  Female             0
    4         Prof-specialty      Own-child  White  Female             0
    5          Other-service      Unmarried  White  Female             0
    6           Adm-clerical      Unmarried  White    Male             0
    ...                  ...            ...    ...     ...           ...
    32556    Protective-serv  Not-in-family  White    Male             0
    32557       Tech-support           Wife  White  Female             0
    32558  Machine-op-inspct        Husband  White    Male             0
    32559       Adm-clerical      Unmarried  White  Female             0
    32560       Adm-clerical      Own-child  White    Male             0

           capital.loss  hours.per.week native.country
    1              4356              18  United-States
    3              3900              40  United-States
    4              3900              40  United-States
    5              3770              45  United-States
    6              3770              40  United-States
    ...             ...             ...            ...
    32556             0              40  United-States
    32557             0              38  United-States
    32558             0              40  United-States
    32559             0              40  United-States
    32560             0              20  United-States

    [30162 rows x 14 columns]
    1        0
    3        0
    4        0
    5        0
    6        0
            ..
    32556    0
    32557    0
    32558    1
    32559    0
    32560    0
    Name: income, Length: 30162, dtype: int64
```

```
    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Standardize numerical features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Apply PCA for dimensionality reduction
    pca = PCA(n_components=10)  # Adjust the number of components as needed
    X_train_pca = pca.fit_transform(X_train)
    X_test_pca = pca.transform(X_test)

    # Train a classifier on the original and reduced-dimension data
    classifier_original = RandomForestClassifier(random_state=42)
    classifier_original.fit(X_train, y_train)

    classifier_pca = RandomForestClassifier(random_state=42)
```

```
classifier_pca.fit(X_train_pca, y_train)

# Evaluate model performance
y_pred_original = classifier_original.predict(X_test)
y_pred_pca = classifier_pca.predict(X_test_pca)

accuracy_original = accuracy_score(y_test, y_pred_original)
accuracy_pca = accuracy_score(y_test, y_pred_pca)

print("Accuracy (Original Data):", accuracy_original)
print("Accuracy (PCA Reduced Data):", accuracy_pca)
```

```
Accuracy (Original Data): 0.8491629371788496
Accuracy (PCA Reduced Data): 0.8252942151500083
```

```
from sklearn.metrics import classification_report, confusion_matrix

# Evaluate model performance on the original data
print("Performance on Original Data:")
print("Confusion Matrix:")
confusion_original = confusion_matrix(y_test, y_pred_original)
print(confusion_original)

report_original = classification_report(y_test, y_pred_original)
print("Classification Report:")
print(report_original)

# Evaluate model performance on PCA-reduced data
print("\nPerformance on PCA Reduced Data:")
print("Confusion Matrix:")
confusion_pca = confusion_matrix(y_test, y_pred_pca)
print(confusion_pca)

report_pca = classification_report(y_test, y_pred_pca)
print("Classification Report:")
print(report_pca)
```

```
Performance on Original Data:
Confusion Matrix:
[[4630  346]
 [ 602  935]]
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.93      0.91      4976
           1       0.73      0.61      0.66      1537

    accuracy                           0.85      6513
   macro avg       0.81      0.77      0.79      6513
weighted avg       0.85      0.85      0.85      6513


Performance on PCA Reduced Data:
Confusion Matrix:
[[4655  321]
 [ 712  825]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.94      0.90      4976
           1       0.72      0.54      0.61      1537

    accuracy                           0.84      6513
   macro avg       0.79      0.74      0.76      6513
weighted avg       0.83      0.84      0.83      6513
```

```
import numpy as np
import matplotlib.pyplot as plt
import itertools

def plot_confusion_matrix(cm, classes, title, cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
```

```
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

# Plot confusion matrix for the classifier on original data
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plot_confusion_matrix(confusion_original, classes=['<=50K', '>50K'], title="Confusion Matrix (Original Data)")

# Plot confusion matrix for the classifier on PCA-reduced data
plt.subplot(1, 2, 2)
plot_confusion_matrix(confusion_pca, classes=['<=50K', '>50K'], title="Confusion Matrix (PCA Reduced Data)")

plt.tight_layout()
plt.show()
```