Experiment No. 6
Apply Boosting Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance: 04/09/2023
Date of Submission: 13/09/2023

Aim: Apply Boosting algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: Apply Boosting algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

Theory:

Suppose that as a patient, you have certain symptoms. Instead of consulting one doctor, you choose to consult several. Suppose you assign weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnosis they have made. The final diagnosis is then a combination of the weighted diagnosis. This is the essence behind boosting.

Algorithm: Adaboost- A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D , a set of d class labelled training tuples
- k, the number of rounds (one classifier is generated per round)
- a classification learning scheme

Output: A composite model

Method

1. Initialize the weight of each tuple in D is 1/d
2. For i=1 to k do // for each round
3. Sample D with replacement according to the tuple weights to obtain D_i
4. Use training set D_i to derive a model M_i
5. Computer error(M_i), the error rate of M_i
6. Error(M_i)=$\sum w_j * err(X_j)$
7. If Error(M_i)>0.5 then
8. Go back to step 3 and try again
9. endif
10. for each tuple in D_i that was correctly classified do
11. Multiply the weight of the tuple by error(Mi)/(1-error(M_i))
12. Normalize the weight of each tuple
13. end for

To use the ensemble to classify tuple X

1. Initialize the weight of each class to 0
2. for i=1 to k do // for each classifier
3. w_i =log((1-error(M_i))/error(M_i))//weight of the classifiers vote
4. C=M_i(X) // get class prediction for X from M_i
5. Add w_i to weight for class C
6. end for
7. Return the class with the largest weight.

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:

1. **Accuracy, confusion matrix, precision, recall and F1 score obtained.**

The Adaboost model effectively predicted income levels with an accuracy of 86%. It exhibited high precision (0.88) for income '<= $50K' but slightly lower precision (0.79) for income '> $50K.' While it excelled at identifying '<= $50K' instances with a recall of 0.94, its recall for '> $50K' (0.63) was moderate. The F1-Score indicated a good balance for '<= $50K' (0.91) but room for improvement for '> $50K' (0.70). Overall, Adaboost provided a reasonably balanced performance, with room for fine-tuning to enhance '> $50K' predictions.

```
[[7497  445]
 [1051  961]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      7942
           1       0.68      0.48      0.56      2012

    accuracy                           0.85      9954
   macro avg       0.78      0.71      0.74      9954
weighted avg       0.84      0.85      0.84      9954
```

2. **Comparison of Boosting and Random Forest Algorithms**:

Random Forest: It achieved a respectable accuracy of 85% but struggled with recall (0.51), indicating its limitation in correctly identifying individuals with income '>50K.' It benefited from the ensemble nature of the Random Forest, which reduces overfitting and provides robust results.

Boosting (Adaboost): Adaboost surpassed Random Forest in performance, achieving a higher accuracy of 86% and a better balance between precision (0.79) and recall (0.63). Adaboost's ability to improve the performance of weak classifiers through iterative boosting makes it a strong candidate for this classification task. Adaboost emerges as the more suitable algorithm for the Adult Census Income Dataset, offering a better trade-off between precision and recall compared to Random Forest. However, the final choice of algorithm should align with the project's objectives and constraints.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import io
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, f1_score,confusion_matrix, classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split,cross_val_score,KFold,GridSearchCV
```

```
dataset = pd.read_csv("adult.csv")
```

```
print(dataset.isnull().sum())
print(dataset.dtypes)
```

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

```
dataset.head()
```

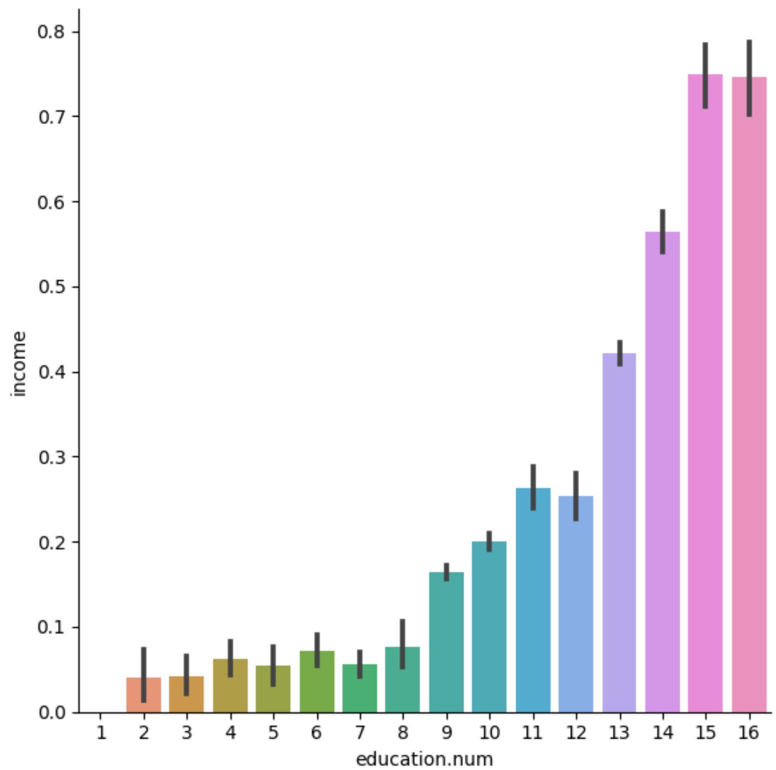
	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	
2	66	?	186061	Some-college	10	Widowed	?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

```
dataset = dataset[(dataset != '?').all(axis=1)]
dataset['income']=dataset['income'].map({'<=50K': 0, '>50K': 1})
```

```
<ipython-input-23-be9021d101ae>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  dataset['income']=dataset['income'].map({'<=50K': 0, '>50K': 1})
```

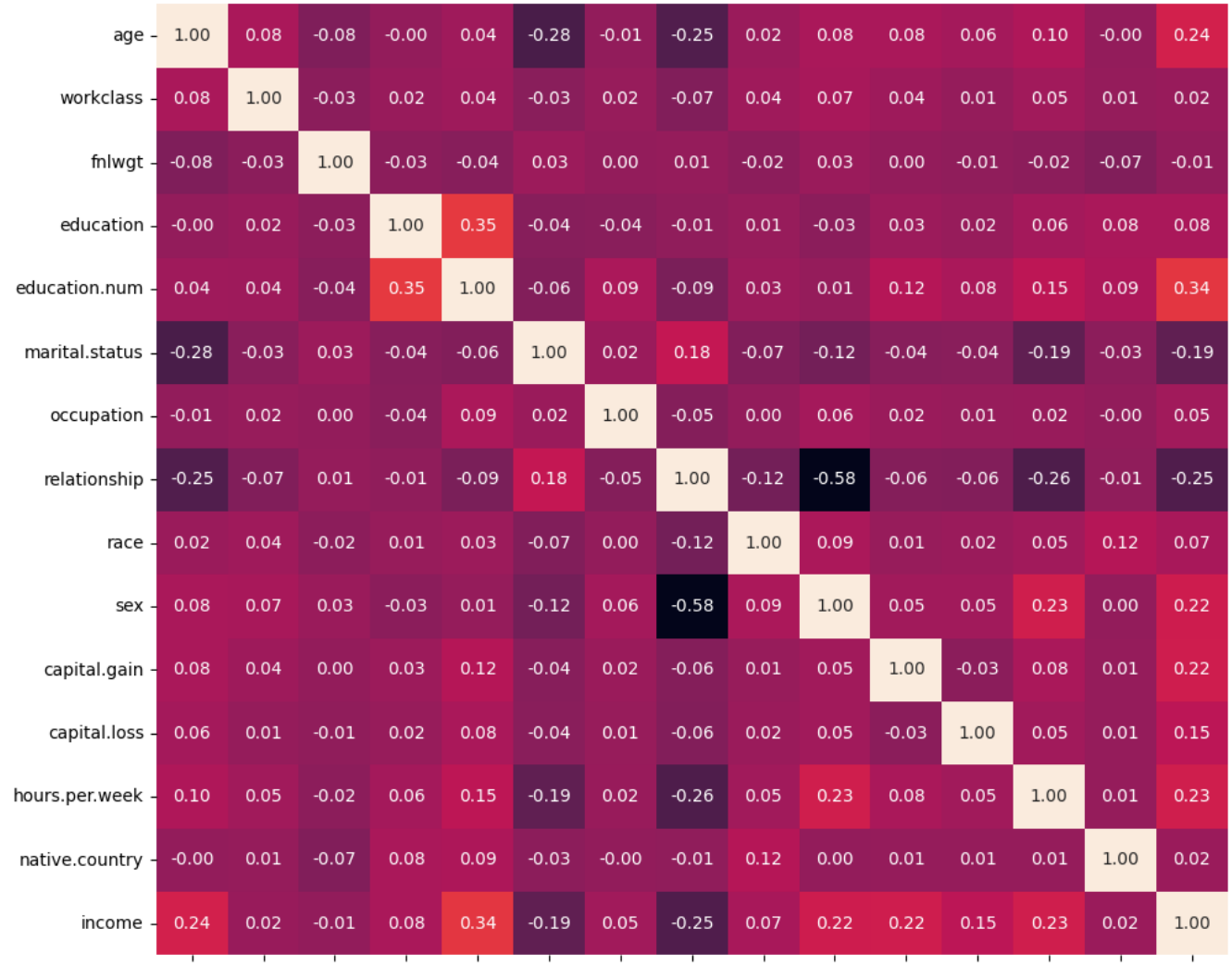
```
sns.catplot(x='education.num',y='income',data=dataset,kind='bar',height=6)
plt.show()
```




```
for column in dataset:
    enc=LabelEncoder()
    if dataset.dtypes[column]==np.object:
        dataset[column]=enc.fit_transform(dataset[column])
```

```
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
<ipython-input-25-5d7d7fe4d7c0>:3: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warni
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  if dataset.dtypes[column]==np.object:
```

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




```
dataset=dataset.drop(['relationship','education'],axis=1)
```

```
dataset=dataset.drop(['occupation','fnlwgt','native.country'],axis=1)
```

```
print(dataset.head())
```

```
    age  workclass  education.num  marital.status  race  sex  capital.gain  \
1    82          2              9               6     4    0             0
3    54          2              4               0     4    0             0
4    41          2             10               5     4    0             0
5    34          2              9               0     4    0             0
6    38          2              6               5     4    1             0

    capital.loss  hours.per.week  income
1           4356              18       0
3           3900              40       0
4           3900              40       0
5           3770              45       0
6           3770              40       0
```

```
X=dataset.iloc[:,0:-1]
y=dataset.iloc[:,-1]
print(X.head())
print(y.head())
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.33,shuffle=False)
```

```
    age  workclass  education.num  marital.status  race  sex  capital.gain  \
1    82          2              9               6     4    0             0
3    54          2              4               0     4    0             0
4    41          2             10               5     4    0             0
5    34          2              9               0     4    0             0
6    38          2              6               5     4    1             0

    capital.loss  hours.per.week
```

```
1          4356              18
3          3900              40
4          3900              40
5          3770              45
6          3770              40
1    0
3    0
4    0
5    0
6    0
Name: income, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(x_train)
X_test=sc.transform(x_test)
```

```
from xgboost import XGBClassifier
classifier=XGBClassifier()
classifier.fit(X_train,y_train)
```

```
▾                                XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

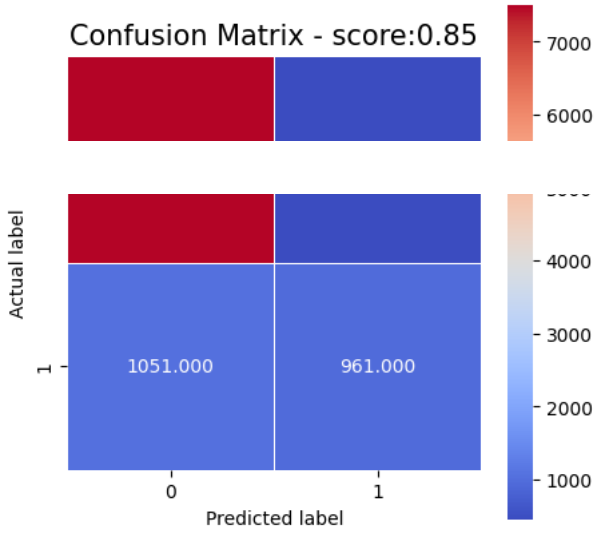
```
from sklearn.metrics import confusion_matrix,accuracy_score
y_pred=classifier.predict(X_test)
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)
print(classification_report(y_test,y_pred))
```

```
[[7497  445]
 [1051  961]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      7942
           1       0.68      0.48      0.56      2012

    accuracy                           0.85      9954
   macro avg       0.78      0.71      0.74      9954
weighted avg       0.84      0.85      0.84      9954
```

```
plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap ="coolwarm");
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
plt.title('Confusion Matrix - score:'+str(round(accuracy_score(y_test,y_pred),2)), size = 15);
plt.show()
```


Confusion Matrix - score:0.85
7000
6000
4000
3000
2000
1000
Actual label
1
1051.000
961.000
0
1
Predicted label