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

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. Comment on the accuracy, confusion matrix, precision, recall and F1 score obtained.

   The GradientBoostingClassifier outperforms the other two classifiers on this dataset, having the greatest accuracy (0.8732) and F1 score (0.71) of the three.

   2. Although they both perform well, the AdaBoostClassifier and XGBClassifier are significantly less accurate and have a lower F1 score than the GradientBoostingClassifier.

   3. The relatively high precision of all three classifiers suggests that they frequently anticipate a positive class (1) correctly.

   The GradientBoostingClassifier has the highest recall for the positive class (1), indicating that it correctly detects more positive cases. The recall values for the classifiers are different.

   5. The F1 score, which strikes a compromise between recall and precision, demonstrates how well the classifiers do in identifying both classes.

2. Compare the results obtained by applying boosting and random forest algorithm on the Adult Census Income Dataset.

   1. In terms of accuracy, precision, and F1 score, the boosting algorithms—AdaBoost, Gradient Boosting, and XGBoost—generally surpass the Random Forest Classifier.

2. The Random Forest Classifier performs admirably, with an F1 score that is balanced and an accuracy of about 87%. It lags a little behind the boosting algorithms, though.

3. The boosting algorithms show a tendency to have higher precision and recall for the positive class (income > $50,000), indicating a superior capacity to accurately categorize people with high incomes. On the Adult Census Income Dataset, all of these models perform about as well as they should. Gradient Boosting, in particular, seems to offer somewhat greater accuracy and an F1 score than Random Forest Classifier.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
```

```
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	hours.per.week	native.country	income
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-States	<=50K
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States	<=50K
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-States	<=50K
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	[illegible]	United-States	<=50K
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K

```
df.describe()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

```
df.isna().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.loc[df.duplicated() == True]
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
8453	25	Private	308144	Bachelors	13	Never-married	Craft-repair	Not-in-family	White	Male	0	0	40	Mexico	<=50K
8645	90	Private	52386	Some-college	10	Never-married	Other-service	Not-in-family	Asian-Pac-Islander	Male	0	0	35	United-States	<=50K
12202	21	Private	250051	Some-college	10	Never-married	Prof-specialty	Own-child	White	Female	0	0	10	United-States	<=50K
14346	20	Private	107658	Some-college	10	Never-married	Tech-support	Not-in-family	White	Female	0	0	10	United-States	<=50K
15603	25	Private	195994	1st-4th	2	Never-married	Priv-house-serv	Not-in-family	White	Female	0	0	40	Guatemala	<=50K
17344	21	Private	243368	Preschool	1	Never-married	Farming-fishing	Not-in-family	White	Male	0	0	50	Mexico	<=50K
19067	46	Private	173243	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K
20388	30	Private	144593	HS-grad	9	Never-married	Other-service	Not-in-family	Black	Male	0	0	40	?	<=50K
20507	19	Private	97261	HS-grad	9	Never-married	Farming-fishing	Not-in-family	White	Male	0	0	40	United-States	<=50K
22783	19	Private	138153	Some-college	10	Never-married	Adm-clerical	Own-child	White	Female	0	0	10	United-States	<=50K
22934	19	Private	146679	Some-college	10	Never-married	Exec-managerial	Own-child	Black	Male	0	0	30	United-States	<=50K
23276	49	Private	31267	7th-8th	4	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K
23660	25	Private	195994	1st-4th	2	Never-married	Priv-house-serv	Not-in-family	White	Female	0	0	40	Guatemala	<=50K
23720	44	Private	367749	Bachelors	13	Never-married	Prof-specialty	Not-in-family	White	Female	0	0	45	Mexico	<=50K
23827	49	Self-emp-not-inc	43479	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K
26738	23	Private	240137	5th-6th	3	Never-married	Handlers-cleaners	Not-in-family	White	Male	0	0	55	Mexico	<=50K
27133	28	Private	274679	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Male	0	0	50	United-States	<=50K
28796	27	Private	255582	HS-grad	9	Never-married	Machine-op-inspct	Not-in-family	White	Female	0	0	40	United-States	<=50K
29051	42	Private	204235	Some-college	10	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	40	United-States	>50K
29334	39	Private	30916	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K
29604	38	Private	207202	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	48	United-States	>50K
31060	46	Private	133616	Some-college	10	Divorced	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32065	19	Private	251579	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	14	United-States	<=50K
32419	35	Private	379959	HS-grad	9	Divorced	Other-service	Not-in-family	White	Female	0	0	40	United-States	<=50K

```
df = df.drop_duplicates()
```

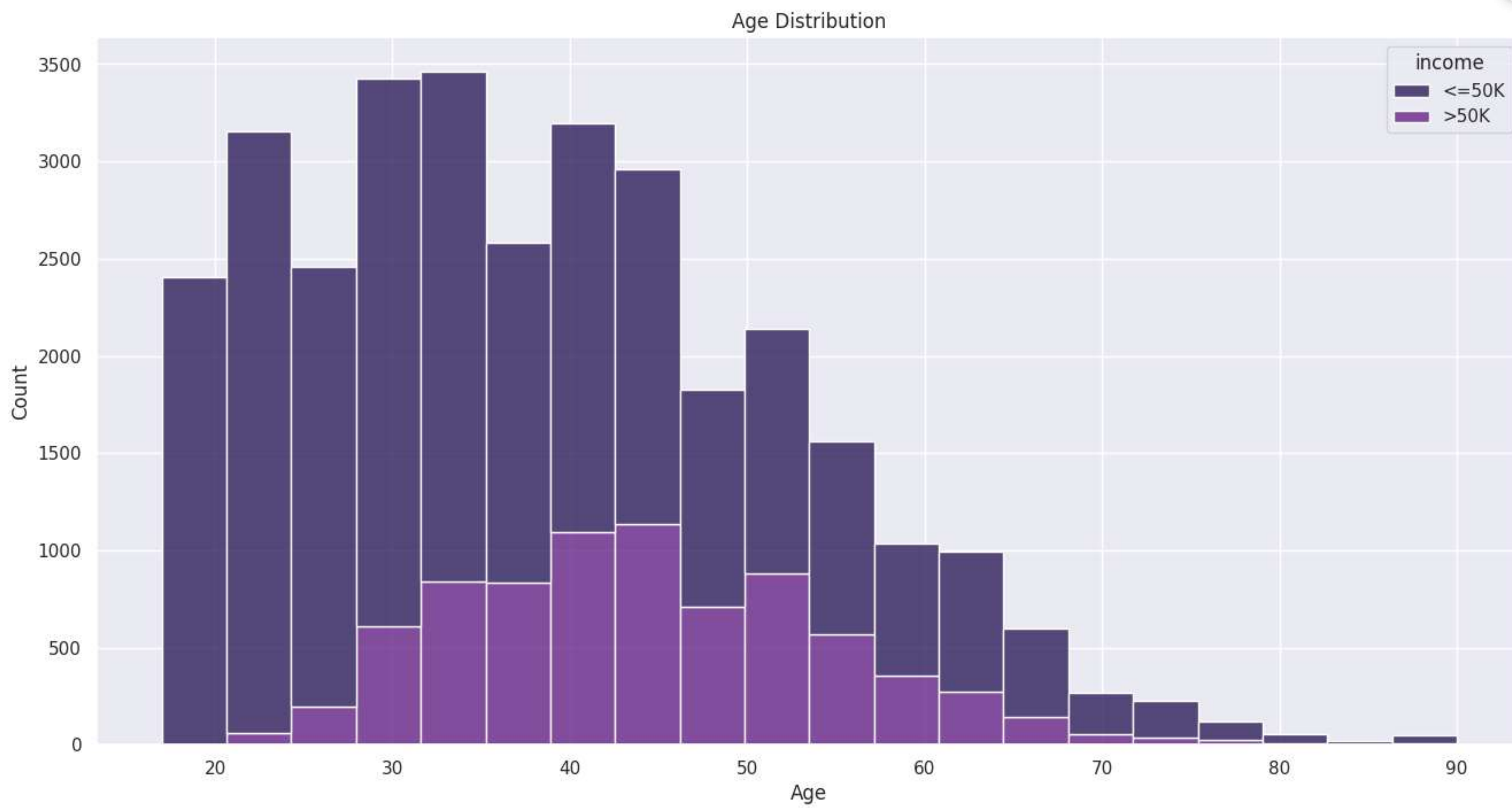
```
df.loc[df.duplicated() == True]
```

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

```
df['age'].describe()
```

```
count    32537.000000
mean        38.585549
std         13.637984
min         17.000000
25%         28.000000
50%         37.000000
75%         48.000000
max         90.000000
Name: age, dtype: float64
```

```
# Age
plt.figure(figsize=(16, 8))
sns.set_theme(style="darkgrid")
sns.set_palette("magma")
sns.histplot(data=df, x='age', hue='income', bins=20, multiple='stack')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Distribution')
plt.show()
```



```
df.groupby('workclass').size()
```

```
workclass
?                    1836
Federal-gov           960
Local-gov            2093
Never-worked            7
Private             22673
Self-emp-inc         1116
Self-emp-not-inc     2540
State-gov            1298
Without-pay            14
dtype: int64
```

```
workclass_unknown = df.loc[df['workclass'] == '?']
print('**age distribution for workclass "?"** \n', workclass_unknown['age'].describe())
plt.figure(figsize=(16, 8))
plt.title('age distribution for workclass "?"')
plt.hist(workclass_unknown['age'], bins=20)
sns.histplot(data=df.loc[df['workclass'] == '?'], x='age', hue='income', bins=20, multiple='stack')
```

```
**age distribution for workclass "?"**
 count    1836.000000
mean       40.960240
std        20.334587
min        17.000000
25%        21.000000
50%        35.000000
75%        61.000000
max        90.000000
Name: age, dtype: float64
<Axes: title={'center': 'age distribution for workclass "?"'}, xlabel='age', ylabel='Count'>
```

age distribution for workclass "?"

```
print(df.query('age < 20').groupby('workclass age less than 20').size())
print(df.query('age > 20 and age < 60').groupby('workclass').size())
print(df.query('age > 60').groupby('workclass age greater than 60').size())
```

```
workclass
?                    269
Federal-gov            9
Local-gov             35
Never-worked           4
Private             1249
Self-emp-inc          16
Self-emp-not-inc      37
State-gov             32
Without-pay            2
dtype: int64
workclass
?                    928
Federal-gov          874
Local-gov           1875
Never-worked           2
Private            19599
Self-emp-inc         940
Self-emp-not-inc    2108
State-gov           1158
Without-pay            5
dtype: int64
workclass
?                   493
Federal-gov          58
Local-gov           151
Private            1070
Self-emp-inc        143
Self-emp-not-inc    338
State-gov            71
Without-pay           7
dtype: int64
```

```
df.loc[df['workclass'] == '?', 'workclass'] = 'Private'
df.loc[df['workclass'] == '?' ]
```

age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.co

```
df.groupby(df['workclass']).size()
```

```
workclass
Federal-gov          960
Local-gov           2093
```

```
Never-worked           7
Private            24509
Self-emp-inc        1116
Self-emp-not-inc    2540
State-gov           1298
Without-pay           14
dtype: int64
```

```
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	nati
0	90	Private	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	U
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	U
2	66	Private	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	U

```
df.groupby('education').size()
```

```
education
10th             933
11th            1175
12th             433
1st-4th          166
5th-6th          332
7th-8th          645
9th              514
Assoc-acdm      1067
Assoc-voc       1382
Bachelors       5353
Doctorate        413
HS-grad        10494
Masters         1722
Preschool         50
Prof-school      576
Some-college    7282
dtype: int64
```

```
plt.figure(figsize=(16, 8))
plt.pie(df.groupby('education').size(), labels=df.groupby('education').size().index, autopct='%1.1f%%')
```

```
df.groupby('education.num').size()
```

```
education.num
1        50
2       166
3       332
4       645
5       514
6       933
7      1175
8       433
9     10494
10     7282
11     1382
```

```
12     1067
13     5353
14     1722
15      576
16      413
dtype: int64
```

```
df['education.num'].describe()
```

```
count    32537.000000
mean        10.081815
std          2.571633
min          1.000000
25%          9.000000
50%         10.000000
75%         12.000000
max         16.000000
Name: education.num, dtype: float64
```

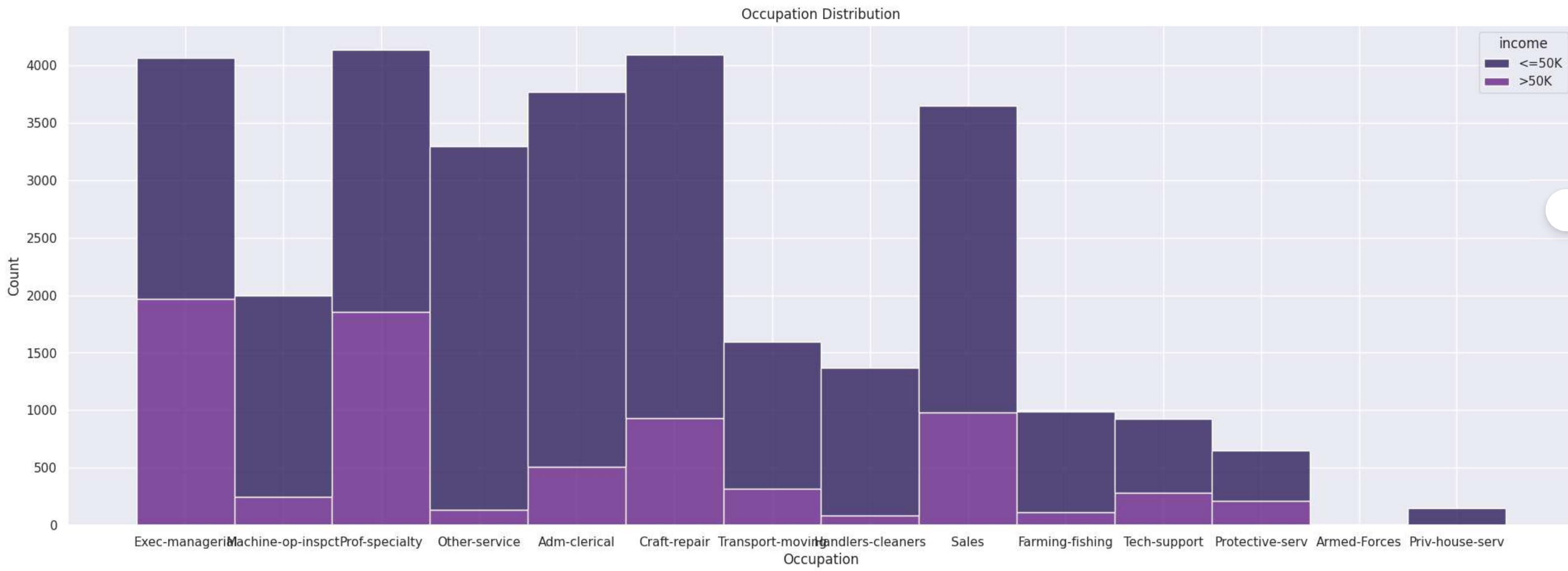
```
plt.figure(figsize=(24, 8))
sns.histplot(data=df, x='occupation', hue='income', multiple='stack', alpha=0.5)
plt.xlabel('Occupation')
plt.title('Occupation Distribution')
```

```
Text(0.5, 1.0, 'Occupation Distribution')
```

```
df.drop(df.loc[df['occupation'] == '?'].index, inplace=True)
```

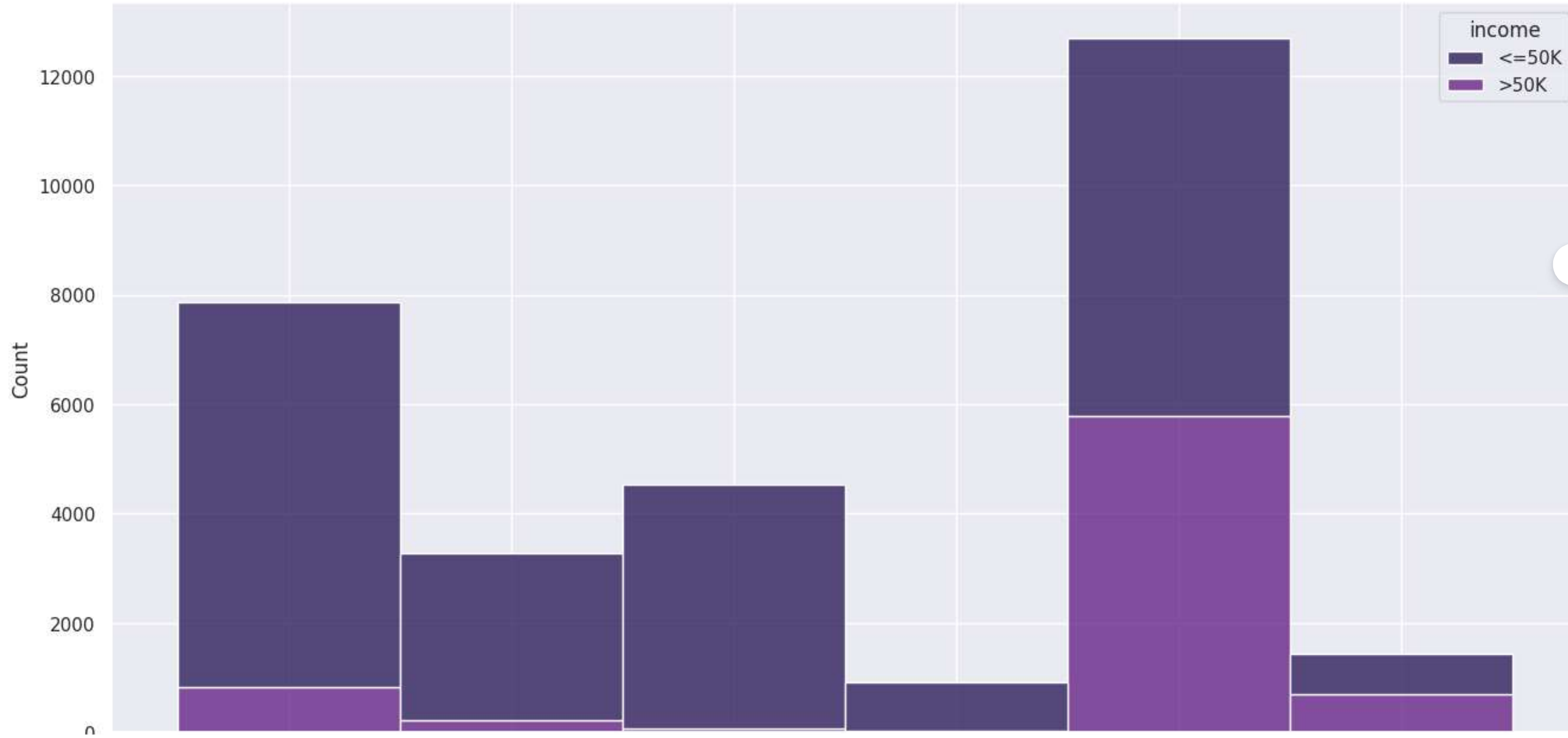
```
plt.figure(figsize=(24, 8))
sns.histplot(data=df, x='occupation', hue='income', multiple='stack')
plt.xlabel('Occupation')
plt.title('Occupation Distribution')
```

```
Text(0.5, 1.0, 'Occupation Distribution')
```




```
print(df.groupby('relationship').size())
plt.figure(figsize=(16, 8))
sns.histplot(data=df, x='relationship', hue='income', multiple='stack')
```

```
relationship
Husband           12698
Not-in-family      7852
Other-relative      918
Own-child          4521
Unmarried          3270
Wife               1435
dtype: int64
<Axes: xlabel='relationship', ylabel='Count'>
```




```
print('**** capital gain **** \n ', df.groupby('capital.gain').size(), '\n')
print('**** capital loss **** \n ', df.groupby('capital.loss').size(), '\n')
```

```
**** capital gain ****
  capital.gain
0        28105
114          6
401          1
594         29
914          8
         ...
25236       11
27828       33
34095        3
41310        2
99999      155
Length: 118, dtype: int64

**** capital loss ****
  capital.loss
0       29233
```

```
155        1
213        4
323        3
419        1
          ...
3004       2
3683       2
3770       2
3900       2
4356       1
Length: 90, dtype: int64
```

```
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	nati
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	U
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	U
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	U

```
df.groupby('hours.per.week').size()
```

```
hours.per.week
1      8
2     15
3     24
4     28
5     39
      ..
95     2
96     5
97     2
98    11
99    80
Length: 94, dtype: int64
```

```
df = df.drop(columns=['hours.per.week'])
```

```
df.groupby('native.country').size()
```

```
native.country
?                               555
Cambodia                         18
Canada                          107
China                            68
Columbia                         56
Cuba                             92
Dominican-Republic               67
Ecuador                          27
El-Salvador                     100
England                          86
France                           27
```

```
Germany                           128
Greece                             29
Guatemala                          61
Haiti                              42
Holand-Netherlands                  1
Honduras                           12
Hong                               19
Hungary                            13
India                             100
Iran                               42
Ireland                            24
Italy                              68
Jamaica                            80
Japan                              59
Laos                               17
Mexico                            606
Nicaragua                          33
Outlying-US(Guam-USVI-etc)         14
Peru                               30
Philippines                       188
Poland                             56
Portugal                           34
Puerto-Rico                       109
Scotland                           11
South                              71
Taiwan                             42
Thailand                           17
Trinadad&Tobago                    18
United-States                   27487
Vietnam                            64
Yugoslavia                         16
dtype: int64
```

```
plt.figure(figsize=(16, 8))
plt.pie(df.groupby('native.country').size(), labels=df.groupby('native.country').size().index, autopct='%1.1f%%')
```

```
label_encoder = LabelEncoder()
categorical_columns = ['income', 'workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.country']
df[categorical_columns] = df[categorical_columns].apply(label_encoder.fit_transform)
```

```
df
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	native.country	inco
1	82	2	132870	11	9	6	3	1	4	0	0	4356	39	
3	54	2	140359	5	4	0	6	4	4	0	0	3900	39	
5	34	2	216864	11	9	0	7	4	4	0	0	3770	39	
32556	22	2	310152	15	10	4	10	1	4	1	0	0	39	

```
x_train, x_test, y_train, y_test = train_test_split(df.drop(columns=['income']), df['income'], test_size=0.2, random_state=42)
```

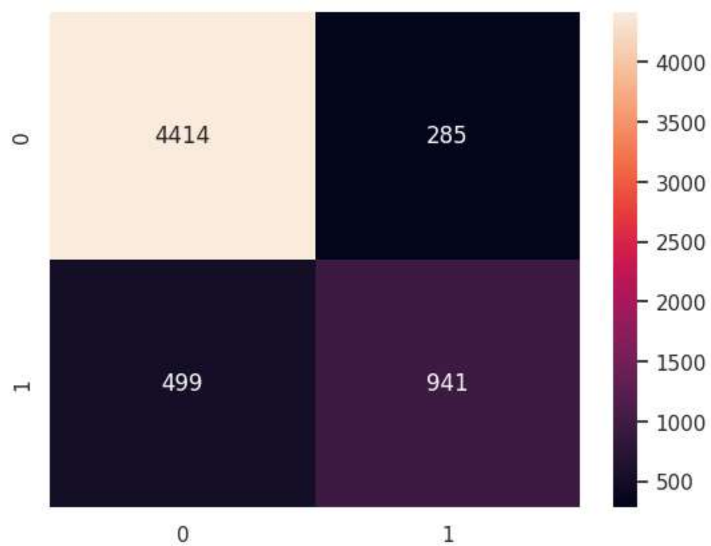
```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_pred = rfc.predict(x_test)
print('**** ACCURACY_SCORE **** \n\n', accuracy_score(y_test, y_pred), '\n')
print('**** CLASSIFICATION_REPORT **** \n\n', classification_report(y_test, y_pred), '\n')
print('**** CONFUSION MATRIX ****')
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d')
```

```
**** ACCURACY_SCORE ****

 0.8579573220394201

**** CLASSIFICATION_REPORT ****

               precision    recall  f1-score   support
```

```
from xgboost import XGBClassifier

xgb = XGBClassifier()
xgb.fit(x_train, y_train)
y_pred = xgb.predict(x_test)
print('**** ACCURACY_SCORE **** \n\n', accuracy_score(y_test, y_pred), '\n')
print('**** CLASSIFICATION_REPORT **** \n\n', classification_report(y_test, y_pred), '\n')
print('**** CONFUSION MATRIX ****')
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d')
```

```
**** ACCURACY_SCORE ****

 0.8722919042189282

**** CLASSIFICATION_REPORT ****

               precision    recall  f1-score   support

           0       0.90      0.94      0.92      4699
           1       0.77      0.65      0.71      1440

    accuracy                           0.87      6139
   macro avg       0.83      0.80      0.81      6139
weighted avg       0.87      0.87      0.87      6139


**** CONFUSION MATRIX ****
<Axes: >
```



```python
from sklearn. ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(
    learning_rate=0.1,
    n_estimators = 500,
    max_depth=5,
    subsample=0.9,
    min_samples_split = 100,
    max_features='sqrt',
    random_state=10)
gbc.fit(x_train,y_train)

y_pred_gbc =gbc.predict (x_test)
print("Accuracy: ",accuracy_score (y_test, y_pred_gbc))
print('**** CLASSIFICATION_REPORT **** \n\n', classification_report(y_test, y_pred_gbc), '\n')
print('**** CONFUSION MATRIX ****')
sns.heatmap(confusion_matrix(y_test, y_pred_gbc), annot=True, fmt='d')
```

```
Accuracy:  0.8732692620948037
**** CLASSIFICATION_REPORT ****

              precision    recall  f1-score   support

           0       0.90      0.94      0.92      4699
           1       0.77      0.65      0.71      1440

    accuracy                           0.87      6139
   macro avg       0.84      0.80      0.81      6139
weighted avg       0.87      0.87      0.87      6139


**** CONFUSION MATRIX ****
<Axes: >
```