Experiment No. 6
Apply Boosting Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:04/09/23
Date of Submission: 25/09/23

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 model demonstrates strong accuracy but exhibits a preference for classifying instances as class 0 over class 1. The F1 score provides a balanced measure of its overall performance.

2. Compare the results obtained by applying boosting and random forest algorithm on the Adult Census Income Dataset.
   Random Forest Accuracy: 85.176
   Boosting algorithm Accuracy: 87.05
   Random Forest has lower precision, recall and F1 score compared to Boosting one.

```
# importing libraries
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('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	40	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

Data Treatment

```
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	:
8453	25	Private	308144	Bachelors	13	Never-married	Craft-repair	Not-in-family	White	Male	0	0	40	Mexico	
8645	90	Private	52386	Some-college	10	Never-married	Other-service	Not-in-family	Asian-Pac-Islander	Male	0	0	35	United-States	
12202	21	Private	250051	Some-college	10	Never-married	Prof-specialty	Own-child	White	Female	0	0	10	United-States	
14346	20	Private	107658	Some-college	10	Never-married	Tech-support	Not-in-family	White	Female	0	0	10	United-States	
15603	25	Private	195994	1st-4th	2	Never-married	Priv-house-serv	Not-in-family	White	Female	0	0	40	Guatemala	
17344	21	Private	243368	Preschool	1	Never-married	Farming-fishing	Not-in-family	White	Male	0	0	50	Mexico	
19067	46	Private	173243	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	
20388	30	Private	144593	HS-grad	9	Never-married	Other-service	Not-in-family	Black	Male	0	0	40	?	
20507	19	Private	97261	HS-grad	9	Never-married	Farming-fishing	Not-in-family	White	Male	0	0	40	United-States	
22783	19	Private	138153	Some-college	10	Never-married	Adm-clerical	Own-child	White	Female	0	0	10	United-States	
22934	19	Private	146679	Some-college	10	Never-married	Exec-managerial	Own-child	Black	Male	0	0	30	United-States	

```
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
32065	19	Private	251579	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	14	United-States	

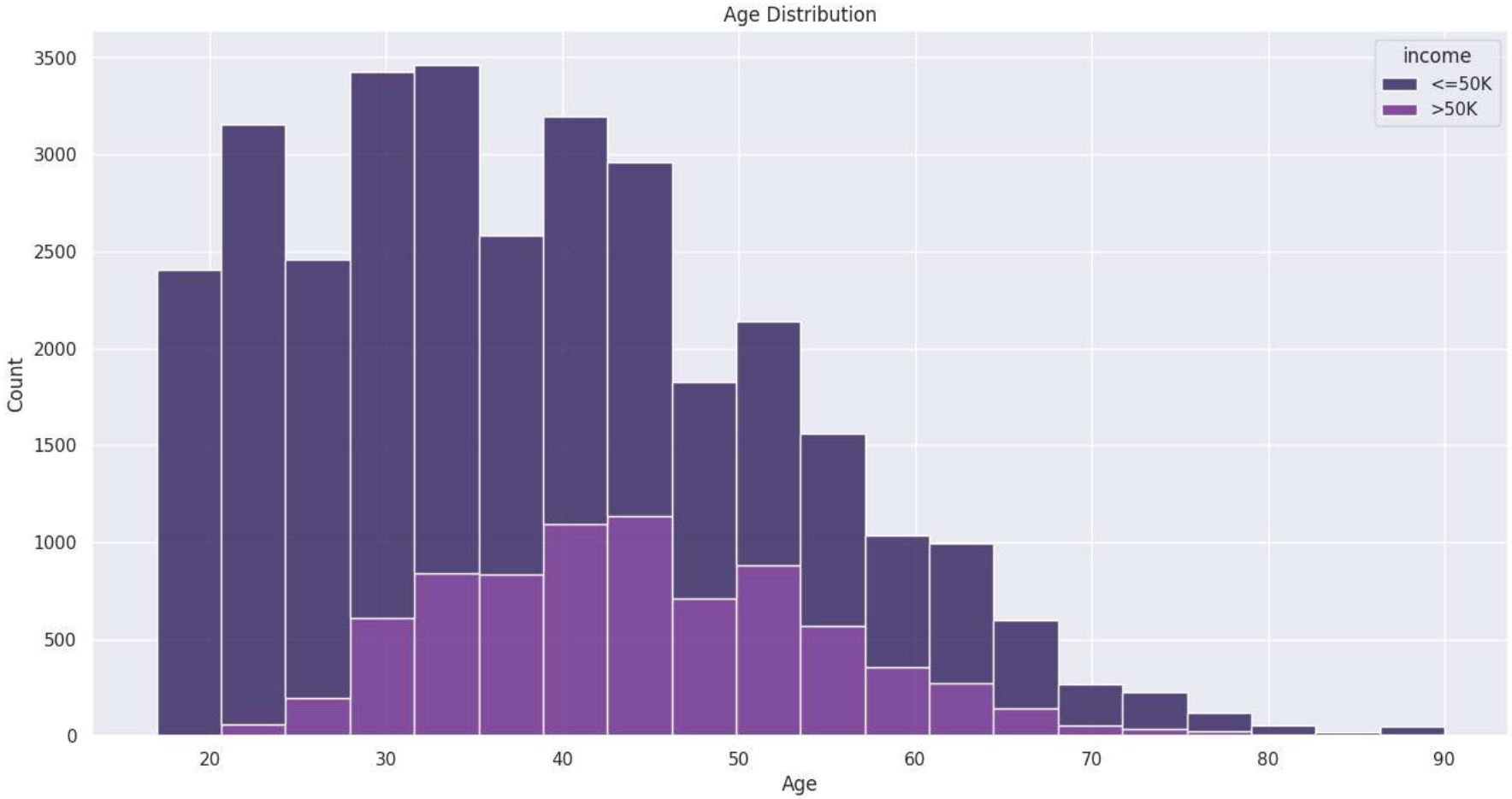
```
df['age'].describe()
```

```
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
```

Exploratory Data Analysis

Analysis per variable

```
# 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()
```



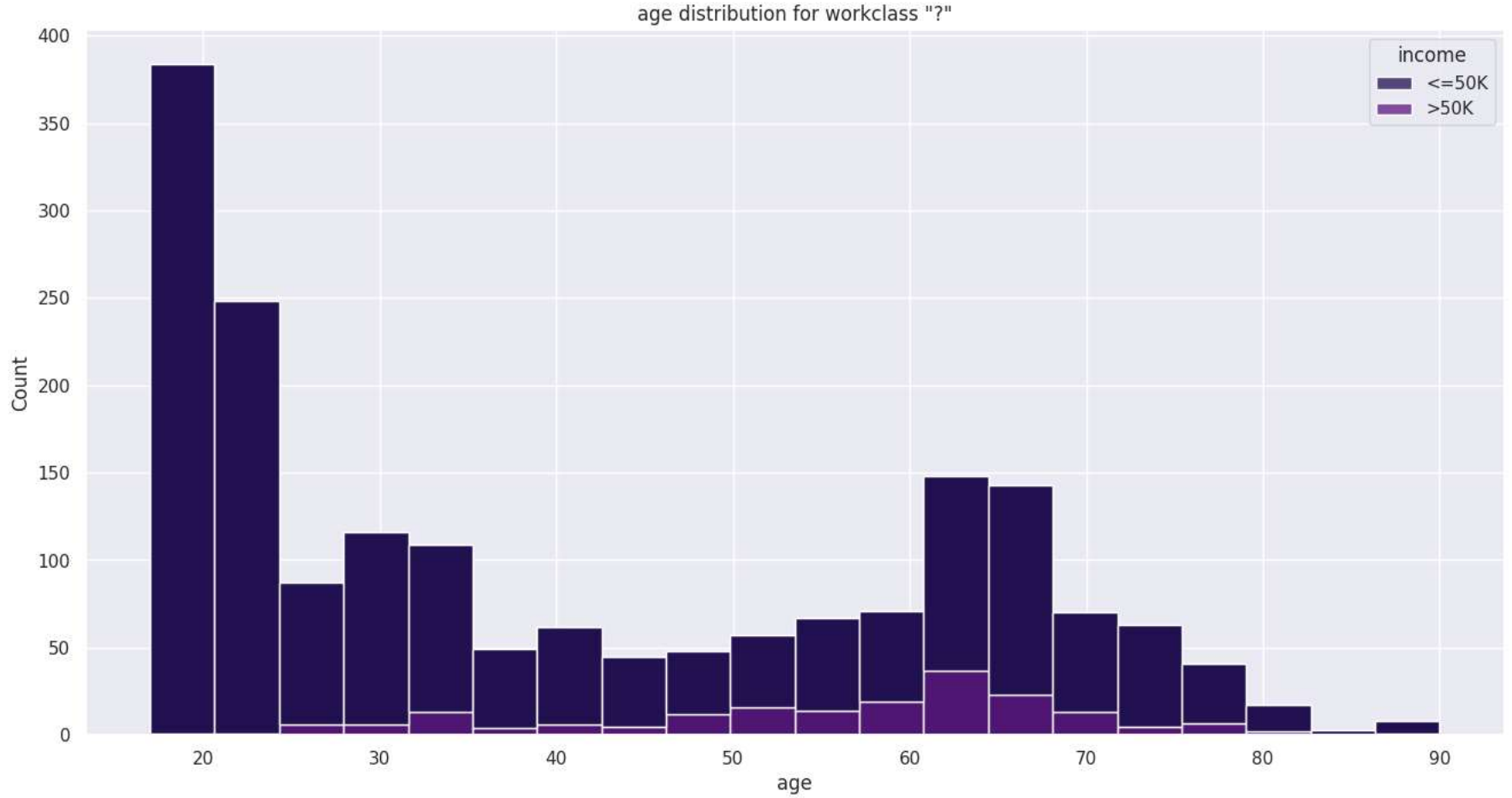

```
# workclass
df.groupby('workclass').size()
```

```
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'>
```




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

```
print(df.query('age > 60').groupby('workclass').size())
```

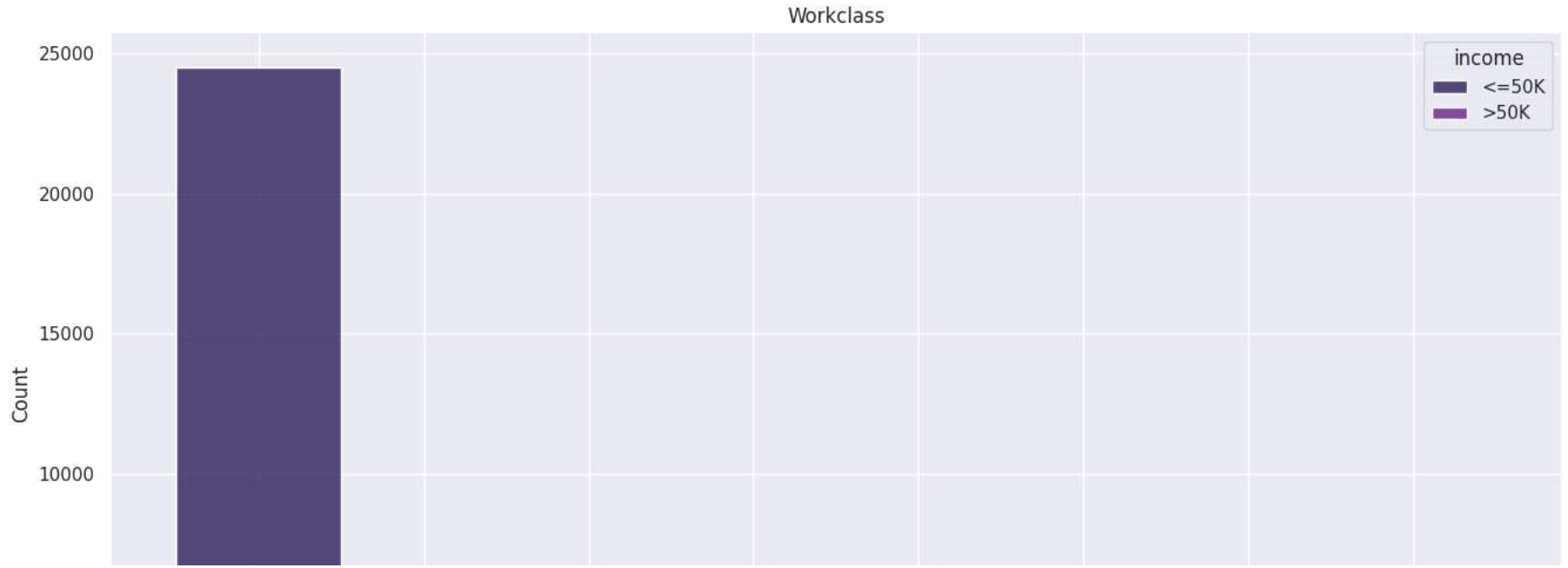
```
workclass
?                   269
Federal-gov           9
Local-gov            35
Never-worked          4
Private            1249
Self-emp-inc         16
Self-emp-not-inc     37
State-gov            32
Without-pay           2
dtype: int64
workclass
?                   928
Federal-gov         874
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.country	income

```
plt.figure(figsize=(16, 8))
sns.histplot(data=df, x='workclass', hue='income', multiple='stack')
plt.xlabel('Workclass')
plt.title('Workclass')
```

```
Text(0.5, 1.0, 'Workclass')
```




```
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
```

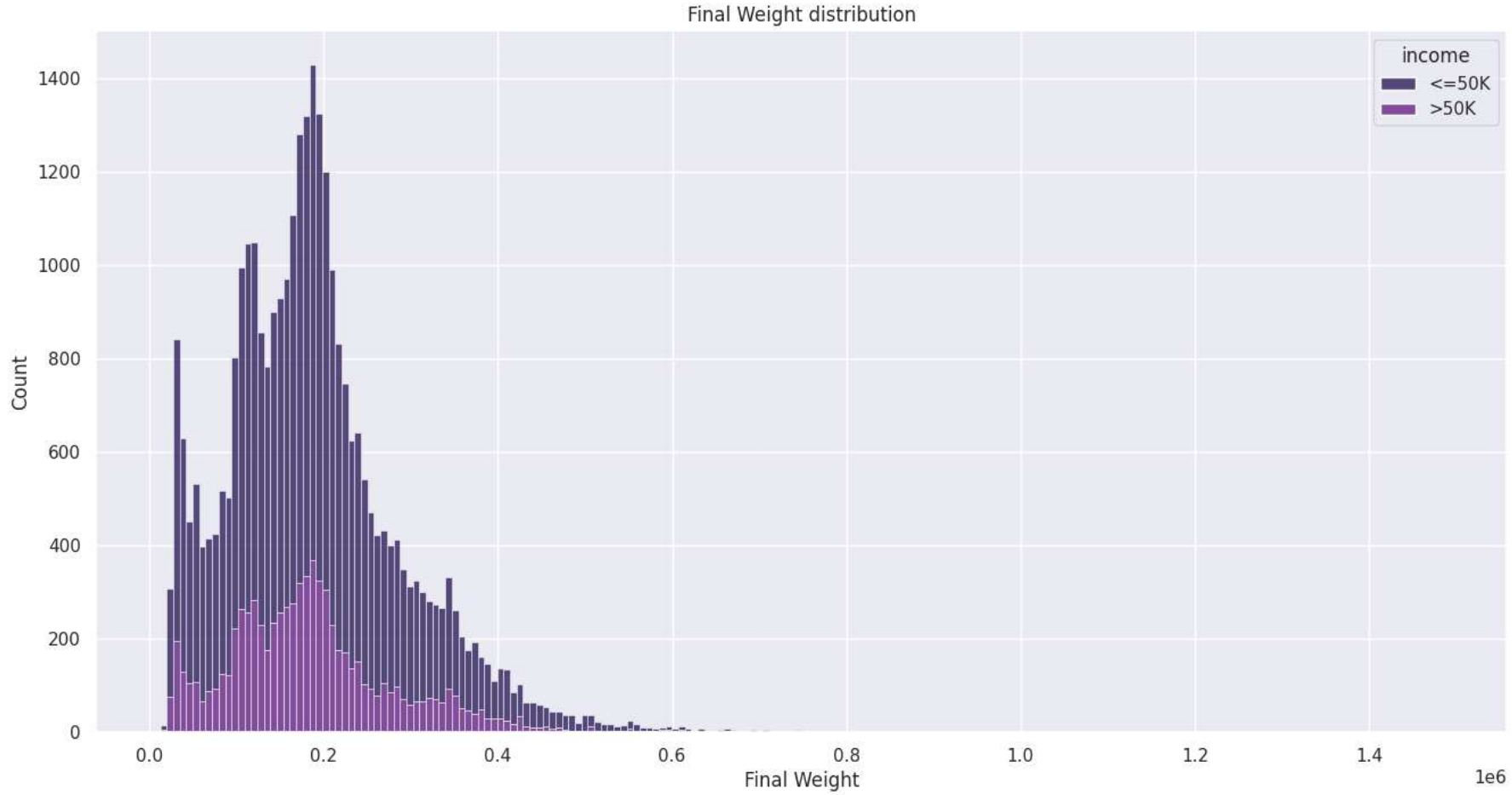
```
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
0	90	Private	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	Private	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	40	United-States	<=50K
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K

```
# Final Weight
plt.figure(figsize=(16, 8))
```

```
sns.histplot(x='fnlwgt', data=df, hue='income', multiple='stack')
plt.title('Final Weight distribution')
plt.xlabel('Final Weight')
```

```
Text(0.5, 0, 'Final Weight')
```




```
# Education
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
```

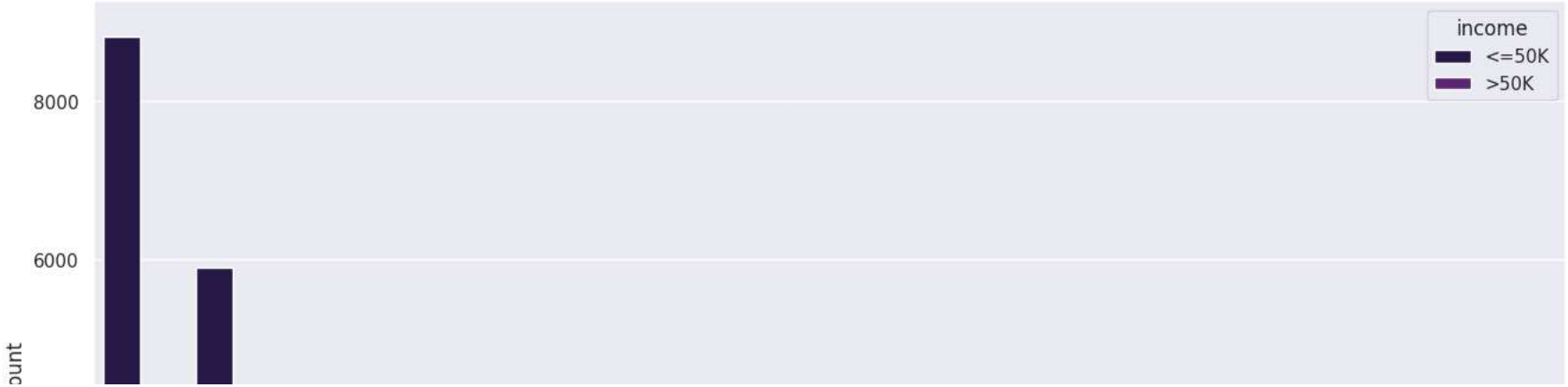
```
Bachelors        5353
Doctorate         413
HS-grad         10494
Masters          1722
Preschool          50
Prof-school       576
Some-college     7282
dtype: int64
```

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

```
([<matplotlib.patches.Wedge at 0x7f011fd8a020>,
  <matplotlib.patches.Wedge at 0x7f011fd89f00>,
  <matplotlib.patches.Wedge at 0x7f011fd8ad10>,
  <matplotlib.patches.Wedge at 0x7f011fd8b3a0>,
  <matplotlib.patches.Wedge at 0x7f011fd8ba30>,
  <matplotlib.patches.Wedge at 0x7f011fdac100>,
  <matplotlib.patches.Wedge at 0x7f011fd89ff0>,
  <matplotlib.patches.Wedge at 0x7f011fdacdf0>,
  <matplotlib.patches.Wedge at 0x7f011fdad480>,
  <matplotlib.patches.Wedge at 0x7f011fdadb10>,
  <matplotlib.patches.Wedge at 0x7f011fdae1a0>,
  <matplotlib.patches.Wedge at 0x7f011fdae830>,
  <matplotlib.patches.Wedge at 0x7f011fdaeec0>,
  <matplotlib.patches.Wedge at 0x7f011fdaf550>,
  <matplotlib.patches.Wedge at 0x7f011fdafbe0>,
  <matplotlib.patches.Wedge at 0x7f011fde42b0>],
 [Text(1.0955395671296693, 0.09895987496625551, '10th'),
  Text(1.05292201745472, 0.31836335398265037, '11th'),
  Text(0.9910262495685971, 0.4773541375813149, '12th'),
  Text(0.9617762876810843, 0.5338411490831262, '1st-4th'),
  Text(0.9350052562783607, 0.5794524749553126, '5th-6th'),
  Text(0.8762672392481279, 0.6649479118099886, '7th-8th'),
  Text(0.7965293158854829, 0.758644217624444, '9th'),
  Text(0.6719068697653994, 0.87094268374105, 'Assoc-acdm'),
  Text(0.4491787546624215, 1.0041107739487294, 'Assoc-voc'),
  Text(-0.2504055487350647, 1.071119536355626, 'Bachelors'),
  Text(-0.7785868926457222, 0.7770472640710339, 'Doctorate'),
  Text(-1.0605252834929717, -0.2920378795159155, 'HS-grad'),
  Text(-0.13449613832544505, -1.091746668772359, 'Masters'),
  Text(0.05334950660702595, -1.0987055247630217, 'Preschool'),
  Text(0.11962081577950918, -1.0934765020027841, 'Prof-school'),
  Text(0.8391187061846062, -0.7112522737616188, 'Some-college')],
 [Text(0.5975670366161832, 0.05397811361795755, '2.9%'),
  Text(0.5743211004298473, 0.17365273853599109, '3.6%'),
  Text(0.540559772491962, 0.2603749841352626, '1.3%'),
  Text(0.5246052478260459, 0.29118608131806883, '0.5%'),
  Text(0.5100286796092...
```

```
plt.figure(figsize=(16, 8))
sns.countplot(x='education', data=df, hue='income')
```

```
<Axes: xlabel='education', ylabel='count'>
```




```
# Education Number
df.groupby('education.num').size()
```

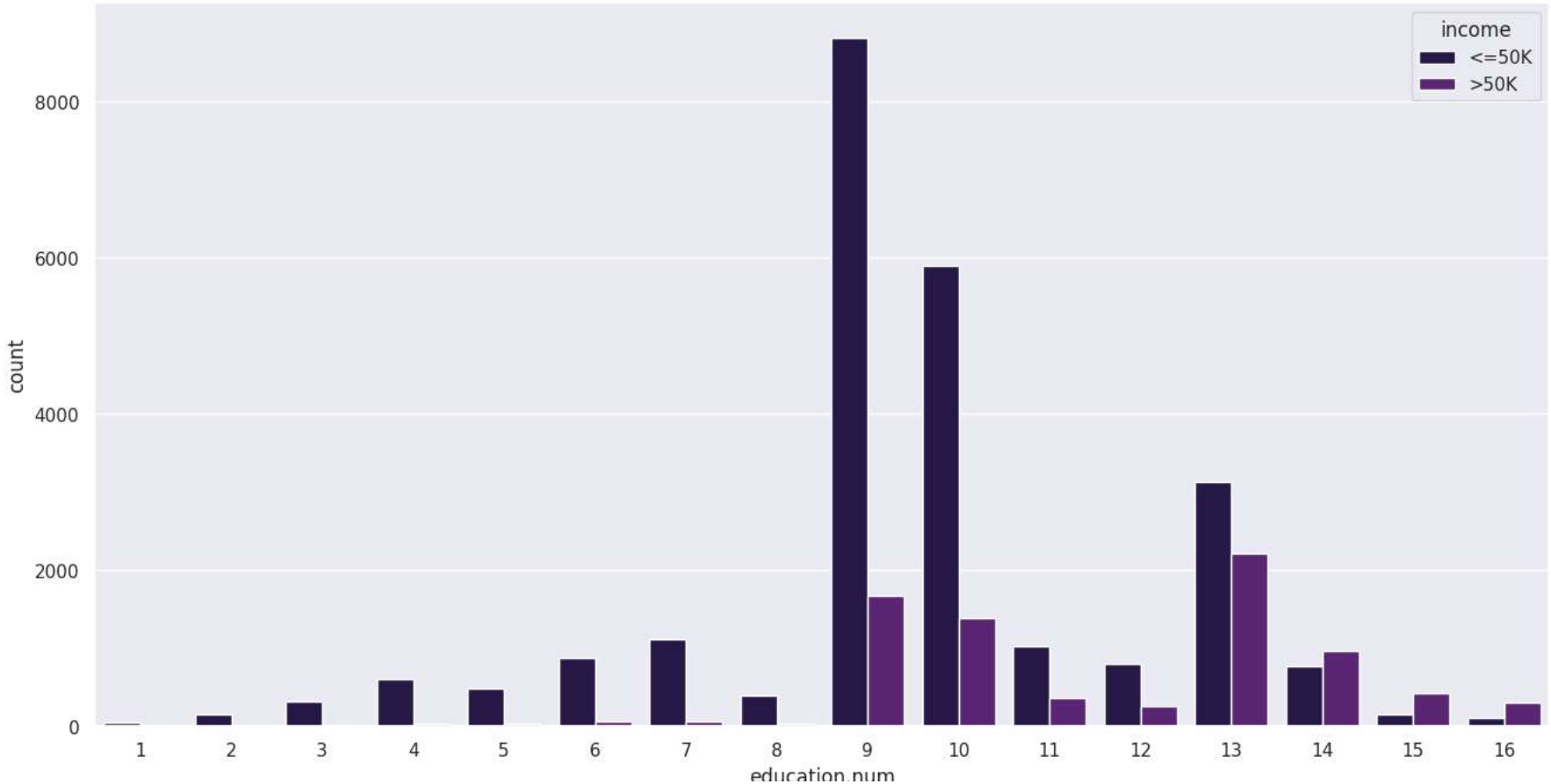
```
education.num
1        50
2       166
3       332
4       645
5       514
6       933
7      1175
8       433
9     10494
10     7282
11     1382
12     1067
13     5353
14     1722
15      576
16      413
dtype: int64
```

```
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=(16, 8))
sns.countplot(x='education.num', data=df, hue='income')
```

```
<Axes: xlabel='education.num', ylabel='count'>
```

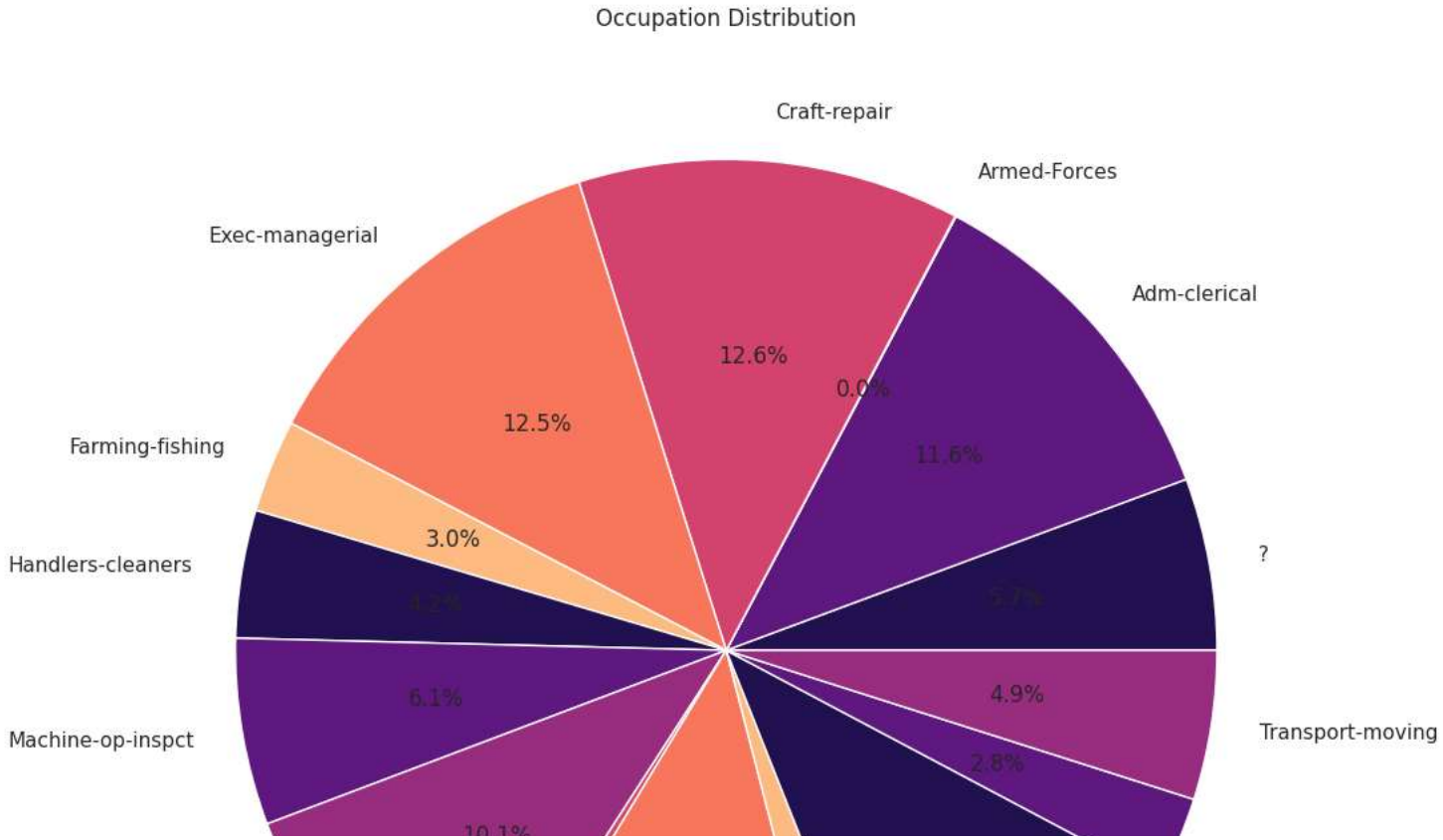



```
# marital status
print(df.groupby('marital.status').size())
plt.figure(figsize=(16, 8))
sns.countplot(data=df, x='marital.status', hue='income')
```

```
marital.status
Divorced                  4441
Married-AF-spouse           23
Married-civ-spouse       14970
Married-spouse-absent      418
Never-married            10667
Separated                 1025
Widowed                    993
dtype: int64
<Axes: xlabel='marital.status', ylabel='count'>
```

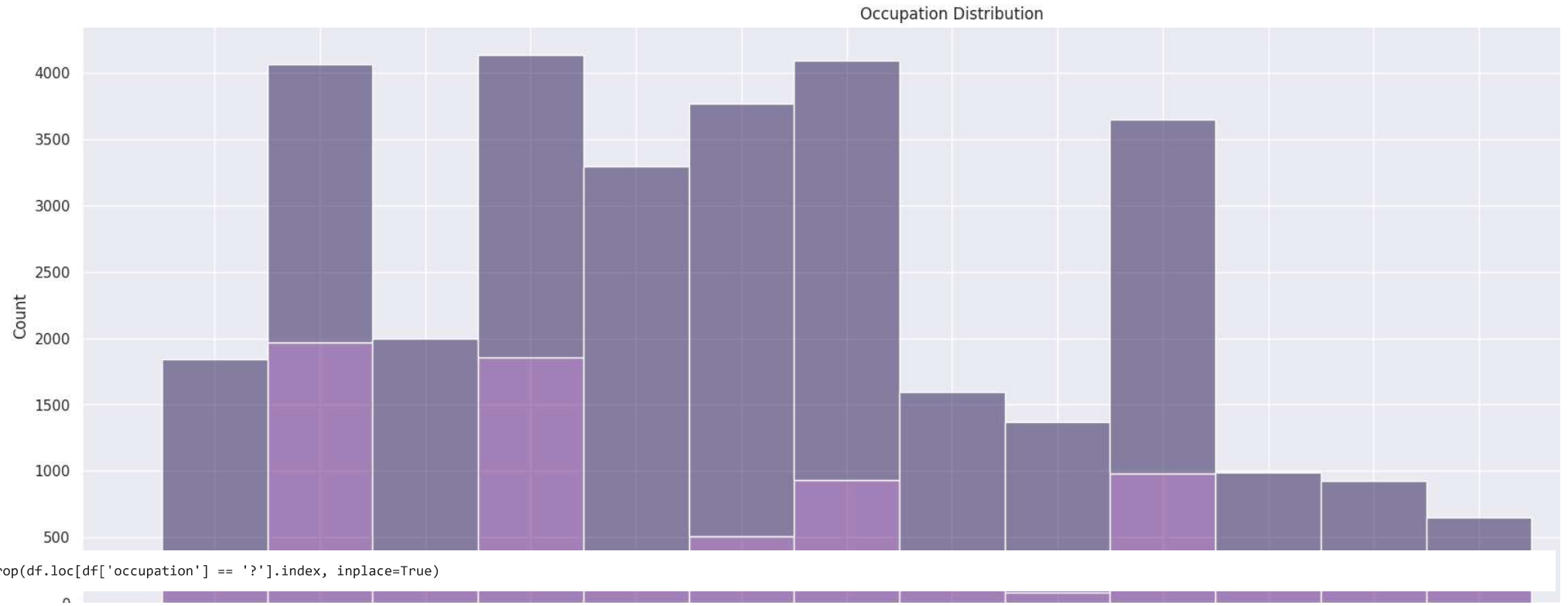

```
# Occupation
print(df.groupby('occupation').size())
plt.figure(figsize=(16, 12))
plt.pie(df.groupby('occupation').size(), labels=df.groupby('occupation').size().index, autopct='%1.1f%%')
plt.title('Occupation Distribution')
```

```
occupation
?                   1843
Adm-clerical        3768
Armed-Forces           9
Craft-repair        4094
Exec-managerial     4065
Farming-fishing      992
Handlers-cleaners   1369
Machine-op-inspct   2000
Other-service       3291
Priv-house-serv      147
Prof-specialty      4136
Protective-serv      649
Sales               3650
Tech-support         927
Transport-moving    1597
dtype: int64
Text(0.5, 1.0, 'Occupation Distribution')
```




```
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')
```

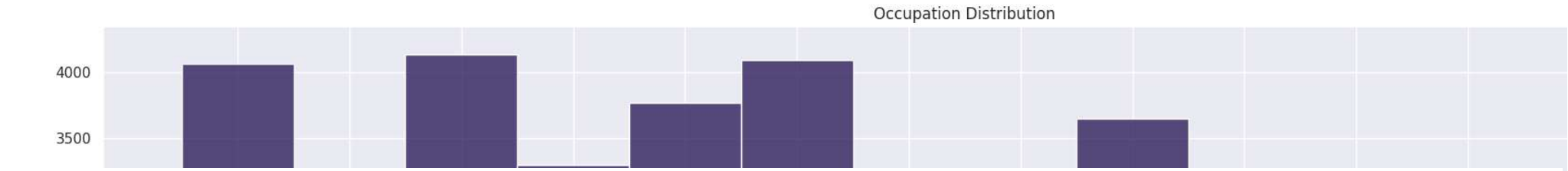
```
Text(0.5, 1.0, 'Occupation Distribution')
```




```
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')
```

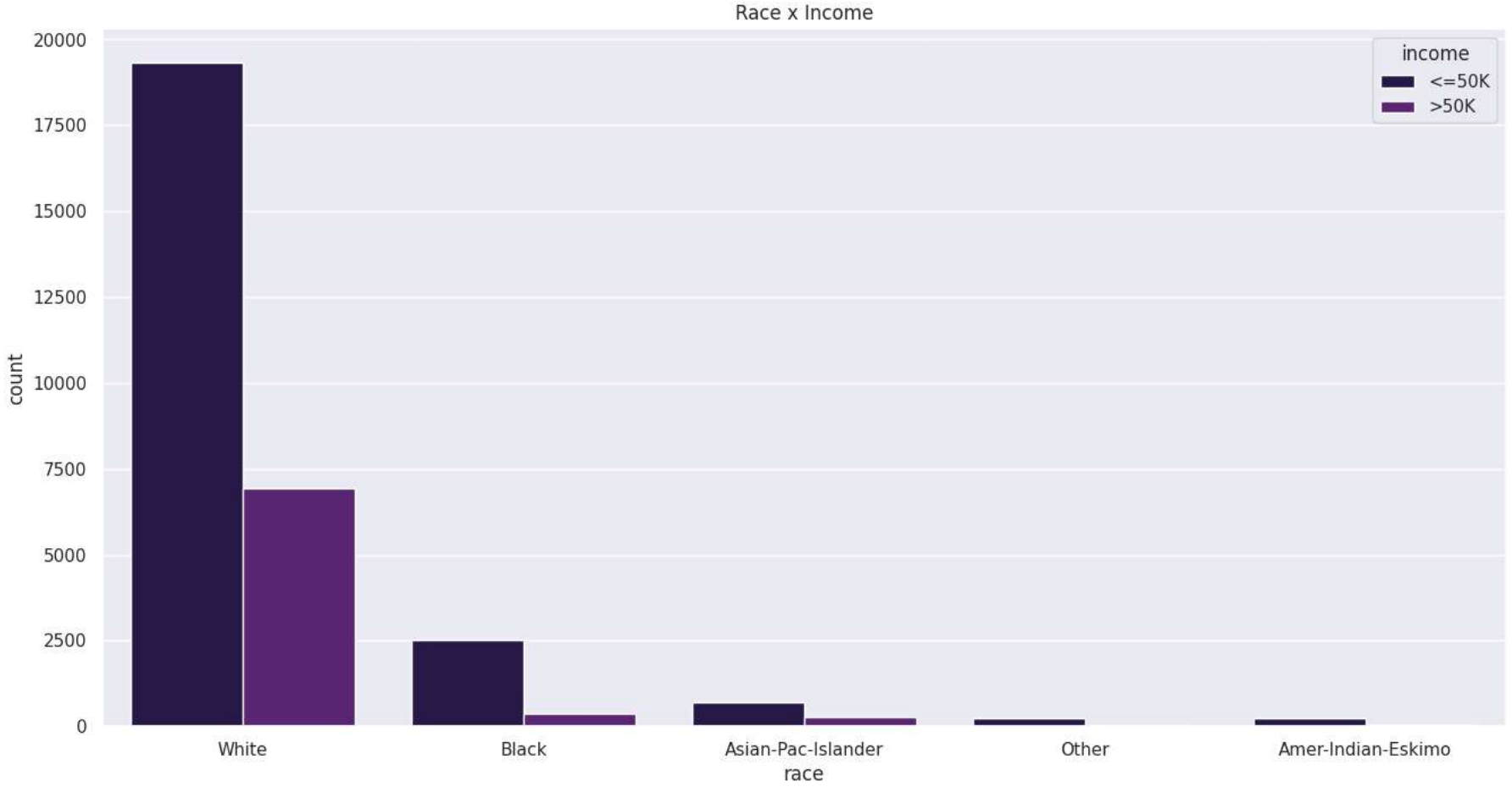



```
# Relationship
print(df.groupby('relationship').size())
plt.figure(figsize=(16, 8))
sns.histplot(data=df, x='relationship', hue='income', multiple='stack')
```

```
relationship
Husband           12698
Not-in-family      7852
Other-relative      918
```

```
# Race and Sex
plt.figure(figsize=(16, 8))
print(df.groupby(df.race).size())
sns.countplot(data=df, x='race', hue='income')
plt.title('Race x Income')
plt.show()
plt.figure(figsize=(16, 8))
print(df.groupby(df.sex).size())
sns.countplot(data=df, x='sex', hue='income')
plt.title('Sex x Income')
```

```
race
Amer-Indian-Eskimo      286
Asian-Pac-Islander      973
Black                  2907
Other                   248
White                 26280
dtype: int64
```



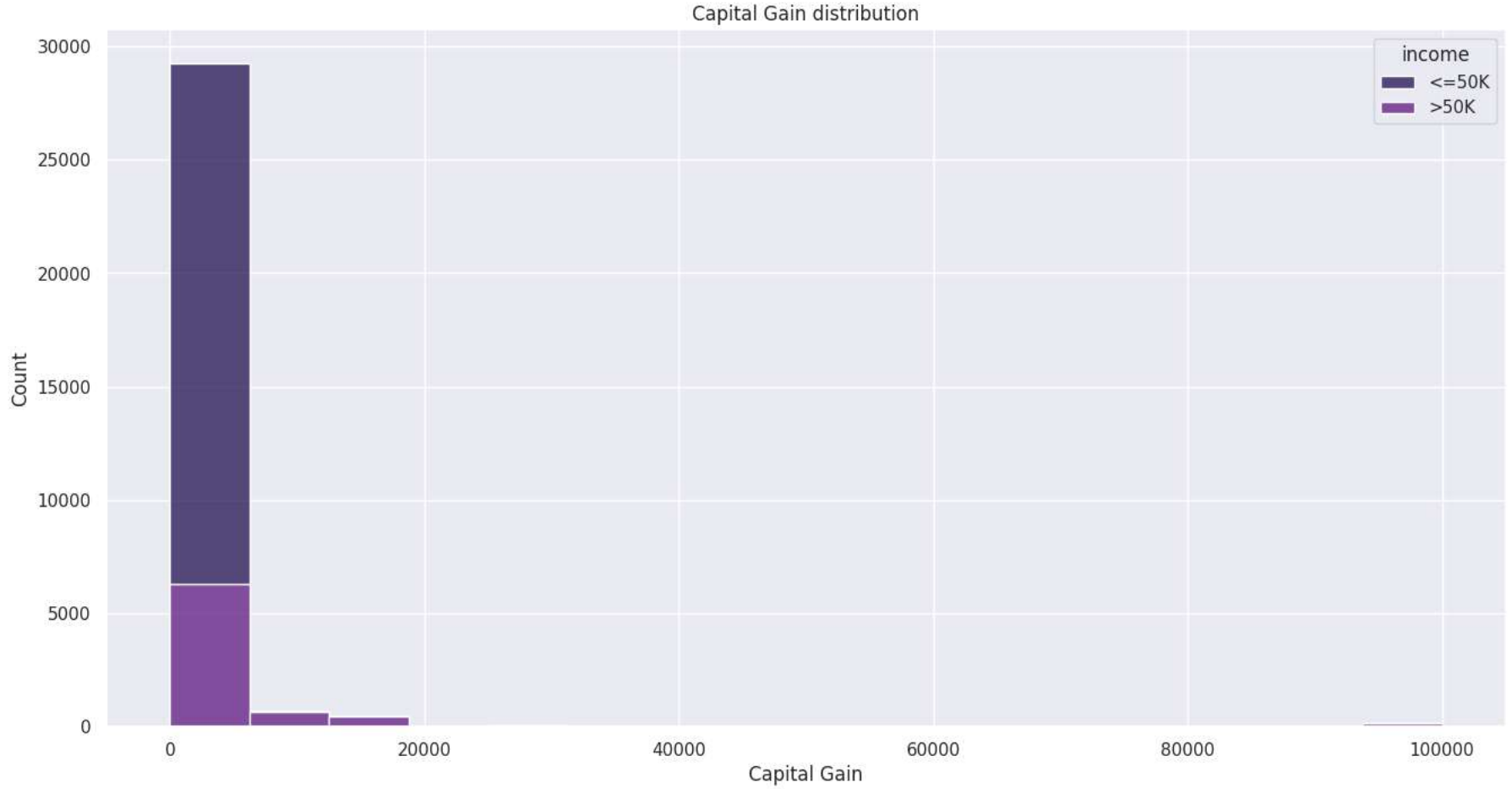

```
sex
Female     9921
Male      20773
dtype: int64
```

```
# Capital Gain & Capital Loss
print('**** capital gain **** \n ', df.groupby('capital.gain').size(), '\n')
print('**** capital loss **** \n ', df.groupby('capital.loss').size(), '\n')
```

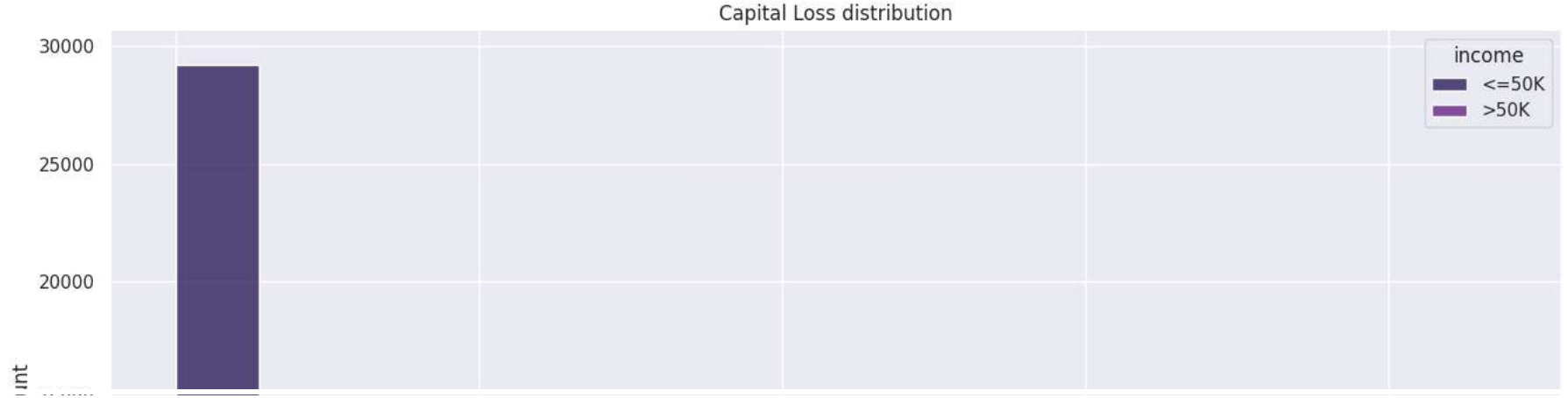
```
**** capital gain ****
  capital.gain
0        28105
114          6
```

```
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
```

```
plt.figure(figsize=(16, 8))
sns.histplot(x='capital.gain', data=df, hue='income', multiple='stack')
plt.title('Capital Gain distribution')
plt.xlabel('Capital Gain')
plt.show()
plt.figure(figsize=(16, 8))
sns.histplot(x='capital.loss', data=df, hue='income', multiple='stack')
plt.title('Capital Loss distribution')
plt.xlabel('Capital Loss')
```




```
Text(0.5, 0, 'Capital Loss')
```



```
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States	<=50K
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-States	<=50K
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	3770	45	United-States	<=50K
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0	3770	40	United-States	<=50K

```
# Hours per week
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'])
```

```
# Native Country
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%%')
```

```
([<matplotlib.patches.Wedge at 0x7f011ddd00d0>,
  <matplotlib.patches.Wedge at 0x7f011dd8bf70>,
  <matplotlib.patches.Wedge at 0x7f011ddd0dc0>,
  <matplotlib.patches.Wedge at 0x7f011ddd1450>,
  <matplotlib.patches.Wedge at 0x7f011ddd1ae0>,
  <matplotlib.patches.Wedge at 0x7f011ddd2170>,
  <matplotlib.patches.Wedge at 0x7f011ddd00a0>,
  <matplotlib.patches.Wedge at 0x7f011ddd2e60>,
  <matplotlib.patches.Wedge at 0x7f011ddd34f0>,
  <matplotlib.patches.Wedge at 0x7f011ddd3b80>,
  <matplotlib.patches.Wedge at 0x7f011de0c250>,
  <matplotlib.patches.Wedge at 0x7f011dd65930>,
  <matplotlib.patches.Wedge at 0x7f011de0ccd0>,
  <matplotlib.patches.Wedge at 0x7f011de0d360>,
  <matplotlib.patches.Wedge at 0x7f011de0d9f0>,
  <matplotlib.patches.Wedge at 0x7f011de0e080>,
  <matplotlib.patches.Wedge at 0x7f011de0e710>,
  <matplotlib.patches.Wedge at 0x7f011de0eda0>,
  <matplotlib.patches.Wedge at 0x7f011de0f430>,
  <matplotlib.patches.Wedge at 0x7f011de0fac0>,
  <matplotlib.patches.Wedge at 0x7f011dc48190>,
  <matplotlib.patches.Wedge at 0x7f011dc48820>,
  <matplotlib.patches.Wedge at 0x7f011dc48eb0>,
  <matplotlib.patches.Wedge at 0x7f011dc49540>,
  <matplotlib.patches.Wedge at 0x7f011dc49bd0>,
  <matplotlib.patches.Wedge at 0x7f011dc4a260>,
  <matplotlib.patches.Wedge at 0x7f011dc4a8f0>,
  <matplotlib.patches.Wedge at 0x7f011dc4af80>,
  <matplotlib.patches.Wedge at 0x7f011dc4b610>,
  <matplotlib.patches.Wedge at 0x7f011dc4bca0>,
  <matplotlib.patches.Wedge at 0x7f011dc80370>,
  <matplotlib.patches.Wedge at 0x7f011dc80a30>,
  <matplotlib.patches.Wedge at 0x7f011dc810c0>,
  <matplotlib.patches.Wedge at 0x7f011dc81750>,
  <matplotlib.patches.Wedge at 0x7f011dc81de0>,
  <matplotlib.patches.Wedge at 0x7f011dc82470>,
  <matplotlib.patches.Wedge at 0x7f011dc82b00>,
  <matplotlib.patches.Wedge at 0x7f011dc83190>,
  <matplotlib.patches.Wedge at 0x7f011dc83820>,
  <matplotlib.patches.Wedge at 0x7f011dc83eb0>,
  <matplotlib.patches.Wedge at 0x7f011dcbc580>,
  <matplotlib.patches.Wedge at 0x7f011dcbcc10>],
 [Text(1.0982257098767156, 0.062452303126338926, '?'),
  Text(1.0926769635963764, 0.12671642839743855, 'Cambodia'),
  Text(1.0909663706824744, 0.1406853867319212, 'Canada'),
  Text(1.0882716045748075, 0.1602027299273451, 'China'),
  Text(1.0861507751849457, 0.1740014182848571, 'Columbia'),
  Text(1.0833904711185112, 0.19043394416335158, 'Cuba'),
  Text(1.0801480310166247, 0.20803901338669392, 'Dominican-Republic'),
  Text(1.078096508189159, 0.2184214253005937, 'Ecuador'),
  Text(1.0751663126581934, 0.23241643686491748, 'El-Salvador'),
  Text(1.070547130189944, 0.2528415354368725, 'England'),
  Text(1.06755128529171, 0.2652060581359678, 'France'),
  Text(1.0632097437090118, 0.2821082077540777, 'Germany'),
  Text(1.0585393982717897, 0.2991560500915825, 'Greece'),
  Text(1.055738796305369, 0.3088941468783292, 'Guatemala'),
  Text(1.0524237487411885, 0.3200066453771607, 'Haiti'),
  Text(1.0510051676771455, 0.32463539165644795, 'Holand-Netherlands'),
  Text(1.0505722853263801, 0.3260335462864321, 'Honduras'),
  Text(1.0495325229278898, 0.32936527339204796, 'Hong'),
  Text(1.0484481371580436, 0.3328009971286569, 'Hungary'),
  Text(1.0445289976086, 0.3449045858095927, 'India'),
```

```
 Text(1.0394060163665135, 0.36004879272272994, 'Iran'),
 Text(1.0369501058827721, 0.36706195377580075, 'Ireland'),
 Text(1.03344778835222, 0.3768098575540526, 'Italy'),
 Text(1.0276214864433568, 0.392420795320465, 'Jamaica'),
 Text(1.021934735401079, 0.4070004871996181, 'Japan'),
 Text(1.0187378932148405, 0.41493747110641643, 'Laos'),
 Text(0.9902268053614884, 0.4790103067195746, 'Mexico'),
 Text(0.9568033395525671, 0.5427037584364559, 'Nicaragua'),
 Text(0.9541815781993336, 0.5473002063082282, 'Outlying-US(Guam-USVI-etc)'),
 Text(0.9517071502242159, 0.5515917876583207, 'Peru'),
 Text(0.9391637500763794, 0.5726879172310798, 'Philippines'),
 Text(0.9245701303027255, 0.5959614703586142, 'Poland'),
 Text(0.9190411740460781, 0.6044529100004454, 'Portugal'),
 Text(0.9100960753539995, 0.6178390839249711, 'Puerto-Rico'),
 Text(0.9024391728144983, 0.6289702213697277, 'Scotland'),
 Text(0.8971285888408105, 0.6365220303214149, 'South'),
 Text(0.889706878394594, 0.6468552160548351, 'Taiwan'),
 Text(0.8857844708901141, 0.6522161230220552, 'Thailand'),
 Text(0.883442345204158, 0.6553850949633944, 'Trinadad&Tobago'),
 Text(-1.0469387310657223, -0.337519322994248, 'United-States'),
 Text(1.0999469008422806, -0.01080811396416703, 'Vietnam'),
 Text(1.0999985251272877, -0.0018013100210502251, 'Yugoslavia')],
[Text(0.5990322053872994, 0.03406489261436668, '1.8%'),
 Text(0.5960056165071144, 0.06911805185314829, '0.1%'),
 Text(0.5950725658268041, 0.07673748367195703, '0.3%'),
 Text(0.5936026934044404, 0.08738330723309731, '0.2%'),
 Text(0.5924458773736067, 0.09490986451901295, '0.2%'),
 Text(0.5909402569737333, 0.10387306045273721, '0.3%'),
 Text(0.5891716532817952, 0.11347582548365122, '0.2%'),
 Text(0.5880526408304503, 0.11913895925486927, '0.1%'),
 Text(0.5864543523590146, 0.12677260192631862, '0.3%'),
 Text(0.5839347982854239, 0.1379135647837486, '0.3%'),
 Text(0.5823007010682053, 0.14465784989234606, '0.1%'),
 Text(0.5799325874776426, 0.15387720422949688, '0.4%'),
 Text(0.577385126330067, 0.16317602732268136, '0.1%'),
 Text(0.5758575252574739, 0.16848771647908867, '0.2%'),
 Text(0.5740493174951937, 0.1745490792966331, '0.1%'),
 Text(0.5732755460057156, 0.17707384999442613, '0.0%'),
 Text(0.5730394283598437, 0.1778364797925993, '0.0%'),
 Text(0.5724722852333944, 0.1796537854865716, '0.1%'),
 Text(0.5718808020862055, 0.18152781661563103, '0.0%'),
 Text(0.5697430896046909, 0.18812977395325048, '0.3%'),
 Text(0.5669487361999164, 0.19639025057603451, '0.1%'),
 Text(0.5656091486633301, 0.20021561115043676, '0.1%'),
 Text(0.5636987936466653, 0.20553264957493775, '0.2%'),
 Text(0.5605208107872854, 0.2140477065384354, '0.3%'),
 Text(0.5574189465824068, 0.22200026574524623, '0.2%'),
 Text(0.555675214480822, 0.22632952969440895, '0.1%'),
 Text(0.5401237120153572, 0.26127834911976794, '2.0%'),
 Text(0.5218927306650365, 0.2960202318744305, '0.1%'),
 Text(0.5204626790178183, 0.29852738525903355, '0.0%'),
 Text(0.5191129910313904, 0.3008682478136294, '0.1%'),
 Text(0.5122711364052978, 0.31237522758058894, '0.6%'),
 Text(0.5043109801651229, 0.32506989292288047, '0.2%'),
 Text(0.5012951858433153, 0.32970158727297016, '0.1%'),
 Text(0.4964160411021815, 0.33700313668634785, '0.4%'),
 Text(0.4922395488079081, 0.34307466620166965, '0.0%'),
 Text(0.489342866640442, 0.3471938347207717, '0.2%'),
 Text(0.48529466094250573, 0.3528301178480919, '0.1%'),
 Text(0.4831551659400622, 0.355754248921121, '0.1%'),
 Text(0.48187764283863155, 0.35748277907094234, '0.1%'),
 Text(-0.571057489672212, -0.18410144890595345, '89.6%'),
```

```
Text(0.5999710368230621, -0.005895334889545652, '0.2%'),
Text(0.5999991955239751, -0.0009825327387546683, '0.1%')])
```

Model preparation & building

```
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	income
1	82	2	132870	11	9	6	3	1	4	0	0	4356	39	0
3	54	2	140359	5	4	0	6	4	4	0	0	3900	39	0
4	41	2	264663	15	10	5	9	3	4	0	0	3900	39	0
5	34	2	216864	11	9	0	7	4	4	0	0	3770	39	0
6	38	2	150601	0	6	5	0	4	4	1	0	3770	39	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	22	2	310152	15	10	4	10	1	4	1	0	0	39	0
32557	27	2	257302	7	12	2	12	5	4	0	0	0	39	0
32558	40	2	154374	11	9	2	6	0	4	1	0	0	39	1
32559	58	2	151910	11	9	6	0	4	4	0	0	0	39	0
32560	22	2	201490	11	9	4	0	3	4	1	0	0	39	0

30694 rows × 14 columns

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

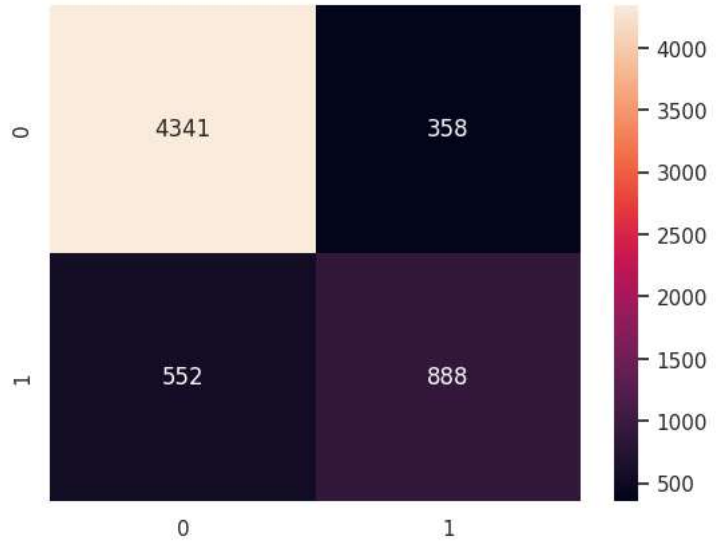
```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

Random Forest

```
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 ****

 0.8517673888255416

**** CLASSIFICATION_REPORT ****

              precision    recall  f1-score   support

           0       0.89      0.92      0.91      4699
           1       0.71      0.62      0.66      1440

    accuracy                           0.85      6139
   macro avg       0.80      0.77      0.78      6139
weighted avg       0.85      0.85      0.85      6139


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

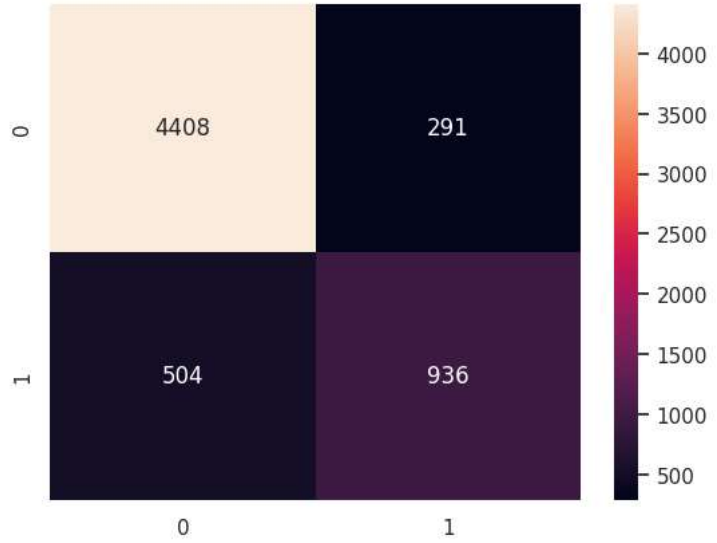


XGBoost Classifier

```
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 ****

 0.8705000814464896

**** CLASSIFICATION_REPORT ****

              precision    recall  f1-score   support

           0       0.90      0.94      0.92      4699
           1       0.76      0.65      0.70      1440

    accuracy                           0.87      6139
   macro avg       0.83      0.79      0.81      6139
weighted avg       0.87      0.87      0.87      6139


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




```
from sklearn. ensemble import GradientBoostingClassifier

#Training the model with gradient boosting
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)

# Predictions
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: >
```