Experiment No. 4
Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:16-08-2023
Date of Submission:27-09-2023

Aim: Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: Able to perform various feature engineering tasks, apply Random Forest Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

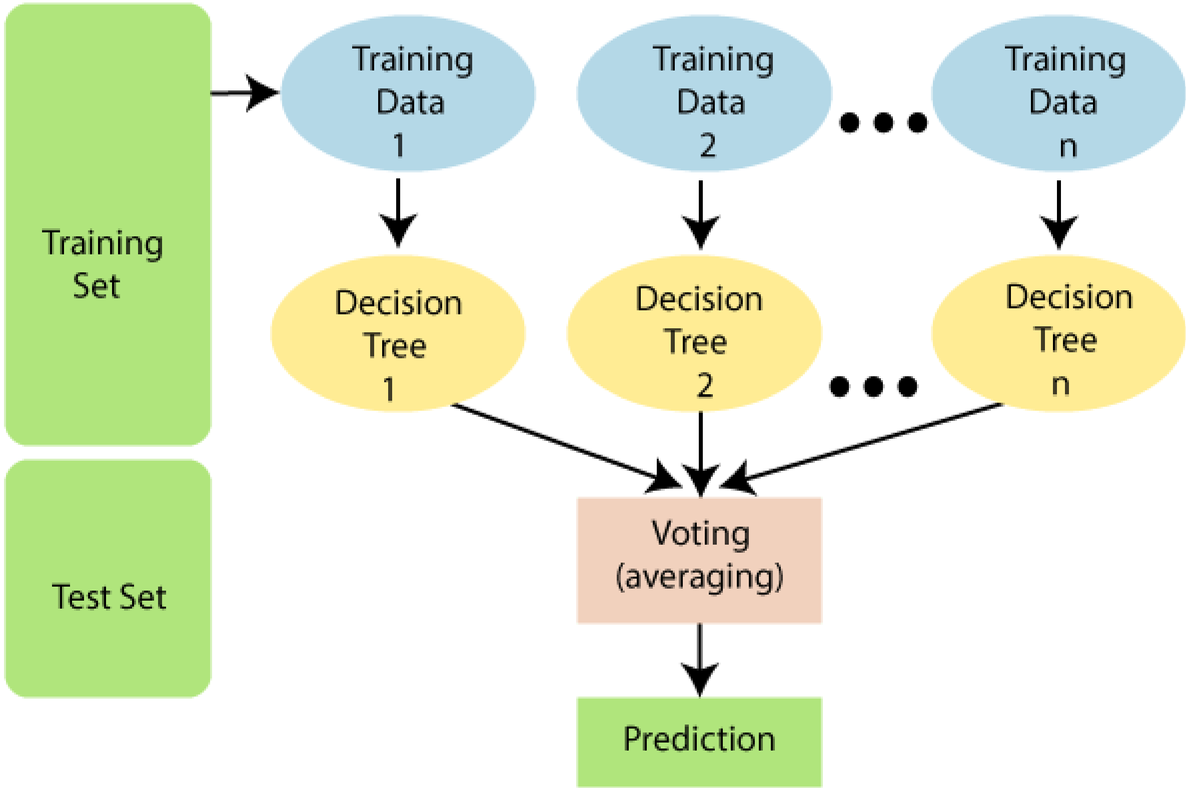
Theory:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



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.

Conclusion:

A correlation heatmap is a graphical representation of a correlation matrix, where each cell in the heatmap represents the correlation between two variables. The correlation values are typically color-coded to help you quickly identify patterns..The correlation heat map obtained from the dataset specifies significant positive correlations between education level and income, suggesting that higher education is associated with higher earnings.

Accuracy obtained in the decision tree model is 85.43%. A confusion matrix is a tabular representation used in machine learning to evaluate the performance of a classification model, especially for binary classification problems.Precision Obtain is 0.74 ,recall obtained is 0.60 and f1 score is 0.66.

Exp 4 : Random Forest

[101]:

```
```

Imports

[102]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[103]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score,␣
 ↪StratifiedKFold, learning_curve, train_test_split, KFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Reading Csv File

[104]:

```
df=pd.read_csv("/content/adult.csv")
```

Data Preprocessing

[105]:

```
df.head()
```

[105]:

```
   age workclass  fnlwgt     education  education.num marital.status  \
0   90         ?   77053       HS-grad              9        Widowed
1   82   Private  132870       HS-grad              9        Widowed
2   66         ?  186061  Some-college             10        Widowed
3   54   Private  140359       7th-8th              4       Divorced
4   41   Private  264663  Some-college             10      Separated

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

```
4     Prof-specialty      Own-child  White  Female             0

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

[106]:
```
print ("Rows : " ,df.shape[0])
print ("Columns : " ,df.shape[1])
print ("\nFeatures : \n" ,df.columns.tolist())
print ("\nMissing values : ", df.isnull().sum().values.sum())
print ("\nUnique values : \n",df.nunique())
```

```
Rows :  32561
Columns :  15

Features :
 ['age', 'workclass', 'fnlwgt', 'education', 'education.num', 'marital.status',
'occupation', 'relationship', 'race', 'sex', 'capital.gain', 'capital.loss',
'hours.per.week', 'native.country', 'income']

Missing values :  0

Unique values :
 age                  73
workclass             9
fnlwgt            21648
education            16
education.num        16
marital.status        7
occupation           15
relationship          6
race                  5
sex                   2
capital.gain        119
capital.loss         92
hours.per.week       94
native.country       42
income                2
dtype: int64
```

[107]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
```

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[108]:
```
df.describe
```

[108]:
```
<bound method NDFrame.describe of        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]>
```

[109]:
```
df.isnull().sum()
```

[109]:
```
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
```

[110]:
```
df[df == '?'] = np.nan
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       30725 non-null  object
 2   fnlwgt          32561 non-null  int64
```

```
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      30718 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  31978 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[111]:
```
df.isnull().sum()
```

[111]:
```
age                  0
workclass         1836
fnlwgt               0
education            0
education.num        0
marital.status       0
occupation        1843
relationship         0
race                 0
sex                  0
capital.gain         0
capital.loss         0
hours.per.week       0
native.country     583
income               0
dtype: int64
```

[112]:
```
max_category = df['workclass'].value_counts().idxmax()
df['workclass'].fillna(max_category, inplace=True)
max_category = df['occupation'].value_counts().idxmax()
df['occupation'].fillna(max_category, inplace=True)
max_category = df['native.country'].value_counts().idxmax()
df['native.country'].fillna(max_category, inplace=True)
max_category = df['relationship'].value_counts().idxmax()
df['relationship'].fillna(max_category, inplace=True)
max_category = df['race'].value_counts().idxmax()
df['race'].fillna(max_category, inplace=True)
```

[113]:
```
df.isnull().sum()
```

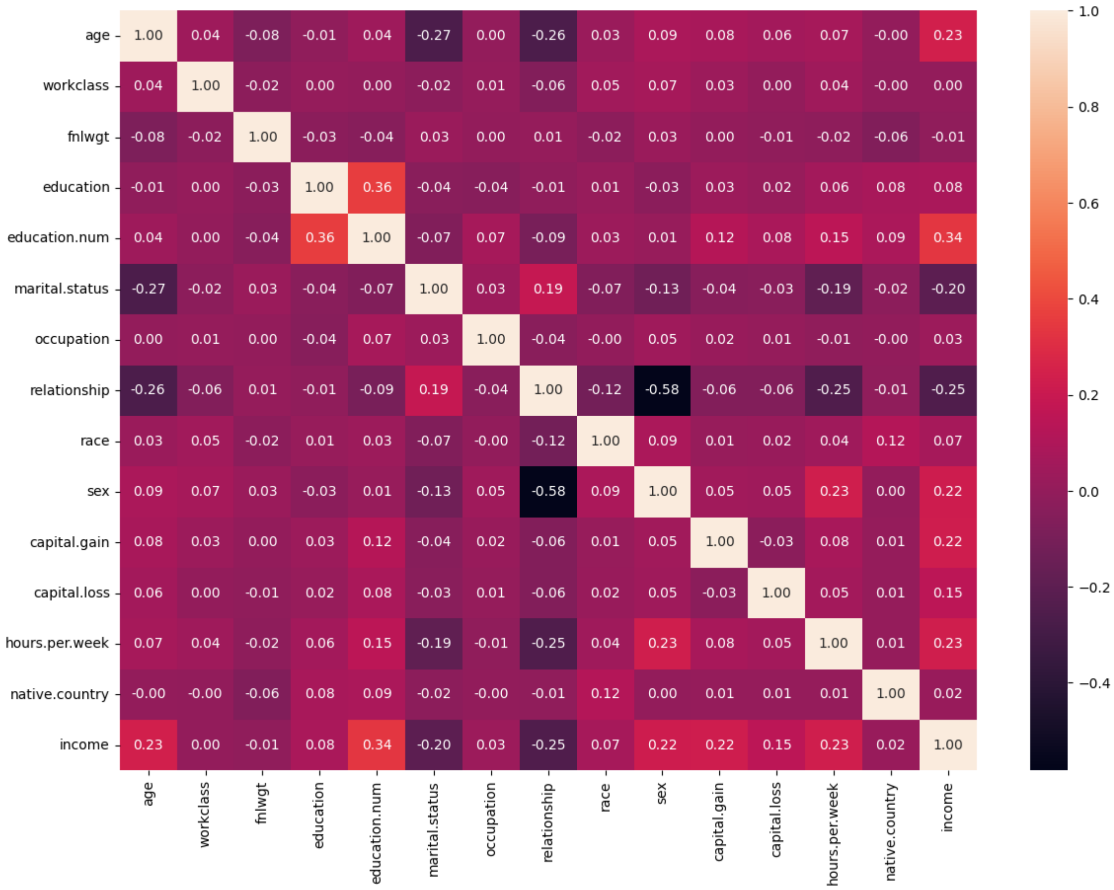
[113]:
```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

Label Encoding

[114]:
```
from sklearn.preprocessing import LabelEncoder
```

[115]:
```
labelencoder_x=LabelEncoder()
df["workclass"]=labelencoder_x.fit_transform(df["workclass"])
df["education"]=labelencoder_x.fit_transform(df["education"])
df["relationship"]=labelencoder_x.fit_transform(df["relationship"])
df["occupation"]=labelencoder_x.fit_transform(df["occupation"])
df["sex"]=labelencoder_x.fit_transform(df["sex"])
df["income"]=labelencoder_x.fit_transform(df["income"])
df["marital.status"]=labelencoder_x.fit_transform(df["marital.status"])
df["race"]=labelencoder_x.fit_transform(df["race"])
df["native.country"]=labelencoder_x.fit_transform(df["native.country"])
```

[116]:
```
plt.figure(figsize=(14,10))
sns.heatmap(df.corr(),annot=True,fmt='.2f')
plt.show()
```



[117]:
```
x=df.drop("income",axis=1)
y=df["income"]
```

[118]:
```
df.head(10)
```

[118]:
```
   age  workclass  fnlwgt  education  education.num  marital.status  \
0   90          3   77053         11              9               6
1   82          3  132870         11              9               6
2   66          3  186061         15             10               6
3   54          3  140359          5              4               0
4   41          3  264663         15             10               5
5   34          3  216864         11              9               0
6   38          3  150601          0              6               5
7   74          6   88638         10             16               4
8   68          0  422013         11              9               0
9   41          3   70037         15             10               4

   occupation  relationship  race  sex  capital.gain  capital.loss  \
0           9             1     4    0             0          4356
```

```
1          3          1     4    0          0         4356
2          9          4     2    0          0         4356
3          6          4     4    0          0         3900
4          9          3     4    0          0         3900
5          7          4     4    0          0         3770
6          0          4     4    1          0         3770
7          9          2     4    0          0         3683
8          9          1     4    0          0         3683
9          2          4     4    1          0         3004

   hours.per.week  native.country  income
0              40              38       0
1              18              38       0
2              40              38       0
3              40              38       0
4              40              38       0
5              45              38       0
6              40              38       0
7              20              38       1
8              40              38       0
9              60              38       1
```

Data Accuracy

[119]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.3)
```

[120]:
```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score,classification_report
from sklearn.ensemble import RandomForestClassifier
```

[121]:
```
features = list(df.columns[1:])
features
```

[121]:
```
['workclass',
 'fnlwgt',
 'education',
 'education.num',
 'marital.status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'capital.gain',
 'capital.loss',
 'hours.per.week',
```

```
 'native.country',
 'income']
```

[122]:
```
rf=RandomForestClassifier(random_state=2)
rf.fit(x_train,y_train)
y_rfp=rf.predict(x_test)
```

[123]:
```
print('Random Forest : ' ,accuracy_score(y_test,y_rfp)*100)
```

```
Random Forest :  85.43351417750026
```

[124]:
```
print('Recall : ' ,recall_score(y_test,y_rfp)*100)
```

```
Recall :  60.44934294192454
```

[125]:
```
print('Precision : ' ,precision_score(y_test,y_rfp)*100)
```

```
Precision :  74.4258872651357
```

[126]:
```
print('F1-Score : ' ,f1_score(y_test,y_rfp)*100)
```

```
F1-Score :  66.71345029239765
```

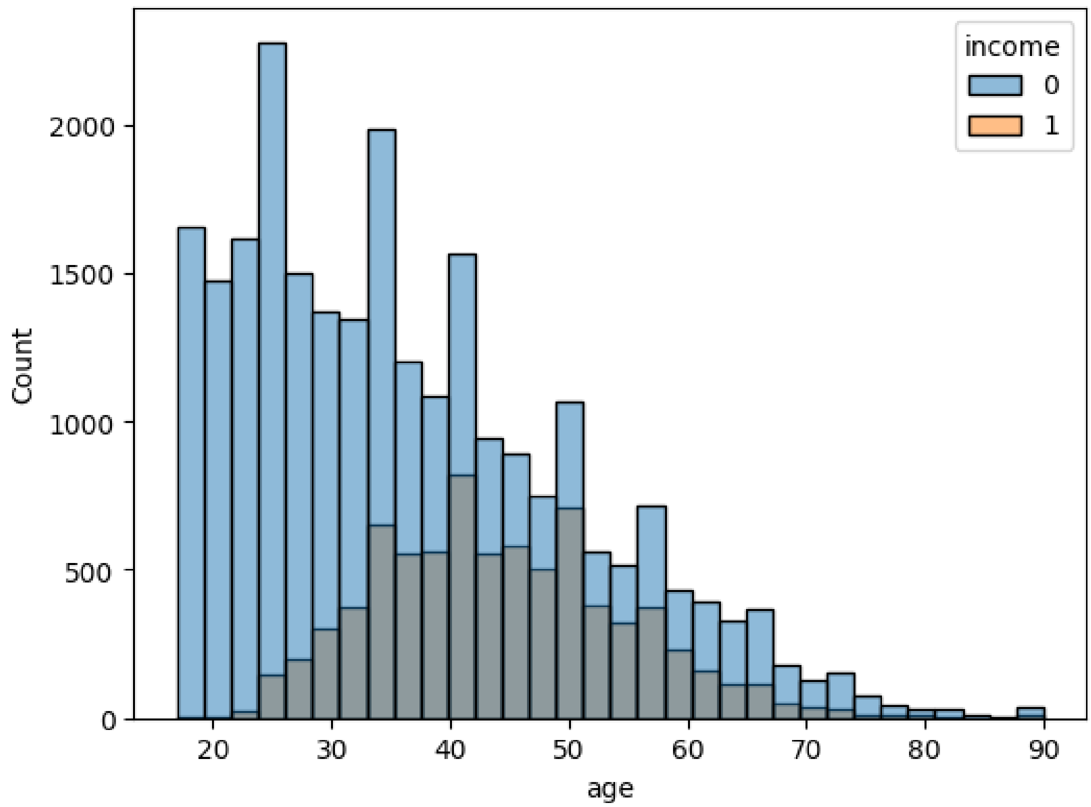
[127]:
```
print(classification_report(y_test,y_rfp))
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.91      7410
           1       0.74      0.60      0.67      2359

    accuracy                           0.85      9769
   macro avg       0.81      0.77      0.79      9769
weighted avg       0.85      0.85      0.85      9769
```

Data Visualization

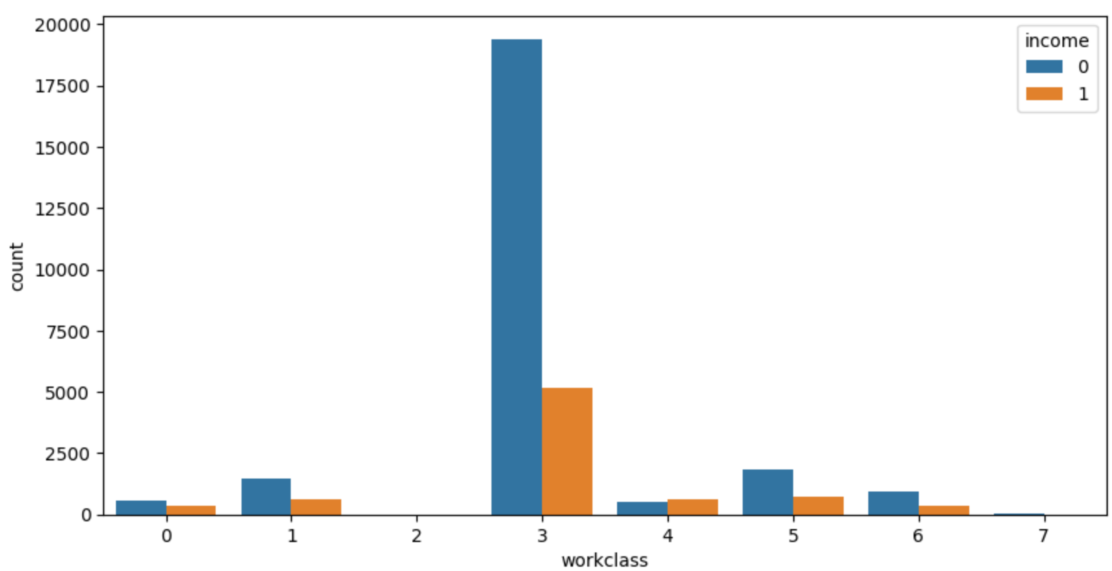
[129]:
```
sns.histplot(df, x='age', hue='income', bins= 32)
```

[129]:
```
<Axes: xlabel='age', ylabel='Count'>
```

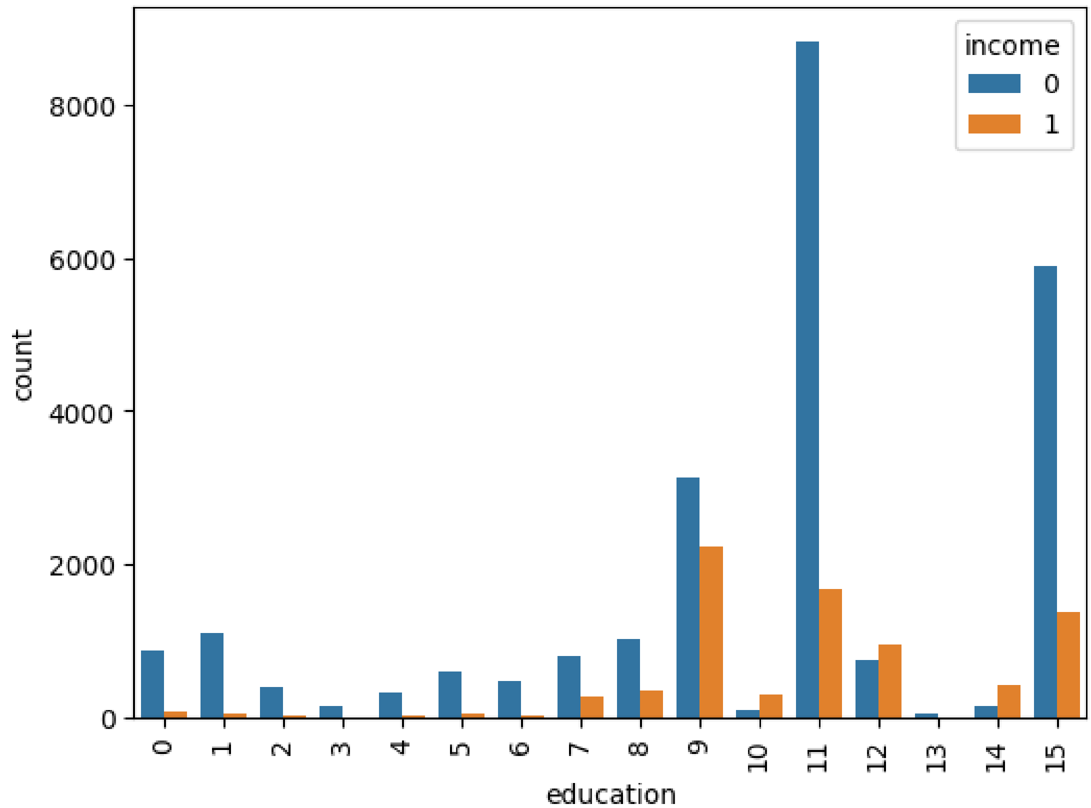


[130]:
```
fig=plt.figure(figsize=(10,5))
sns.countplot(data = df, x = 'workclass', hue = 'income')
```

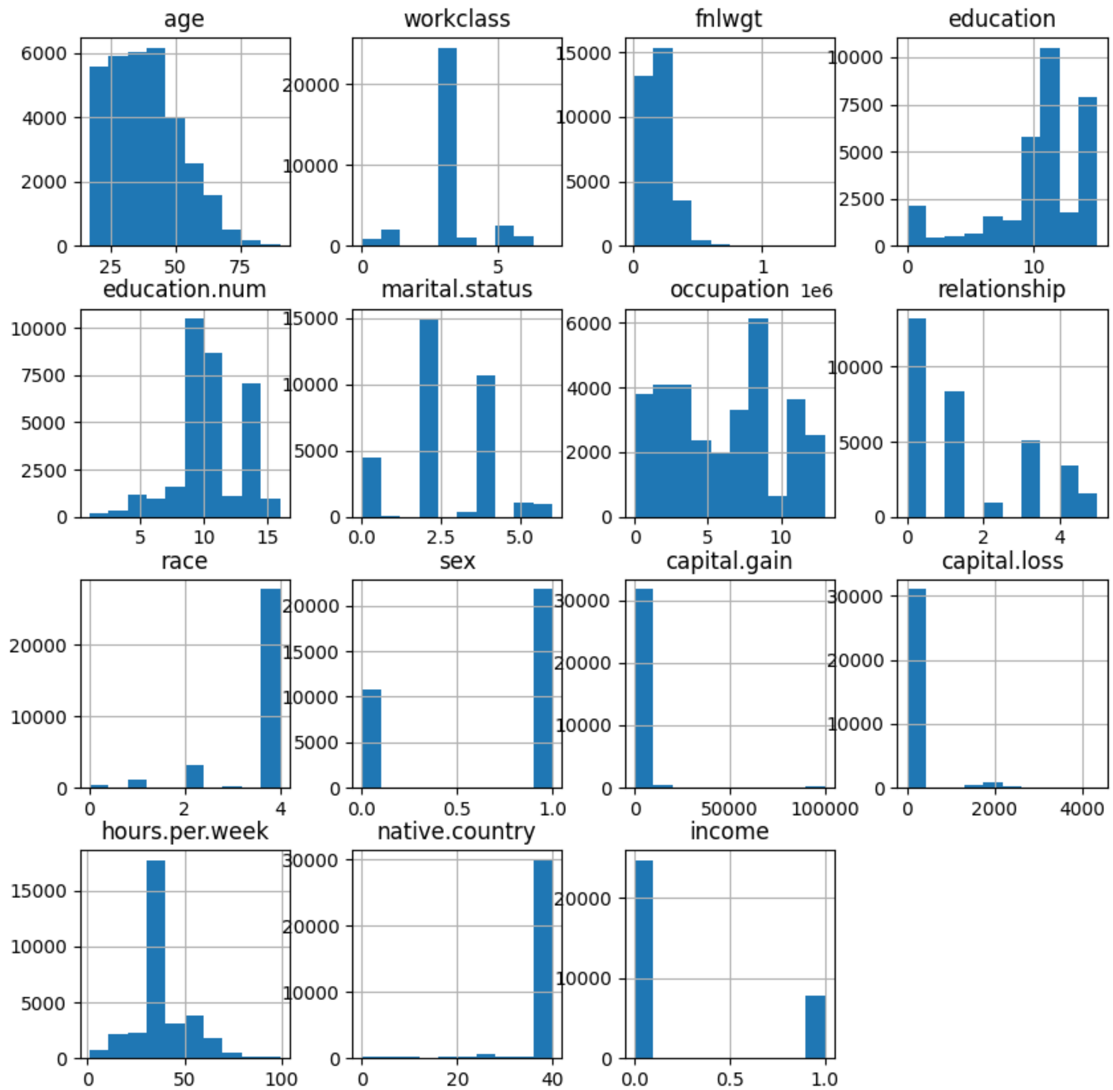
[130]: <Axes: xlabel='workclass', ylabel='count'>



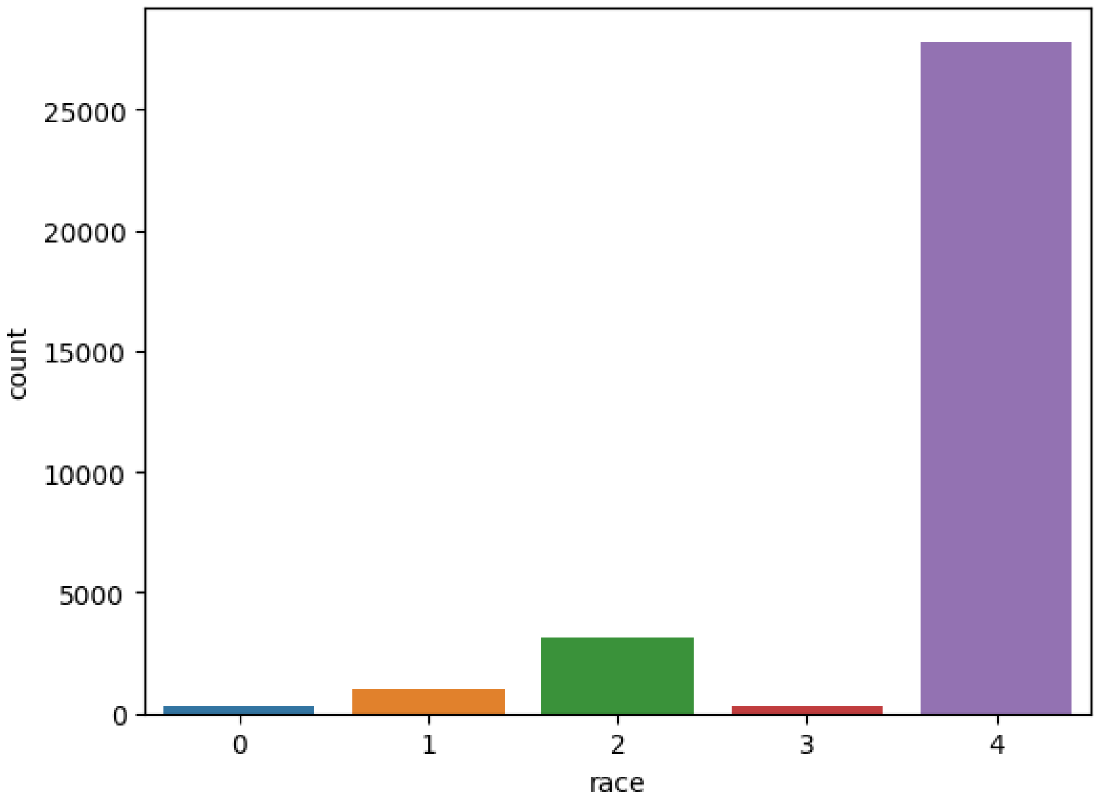
[131]:
```
sns.countplot(data = df, x = 'education', hue = 'income')
plt.tick_params(axis='x', rotation=90)
```



[132]:
```
df.hist(bins=10, figsize=(10, 10))
plt.show()
```

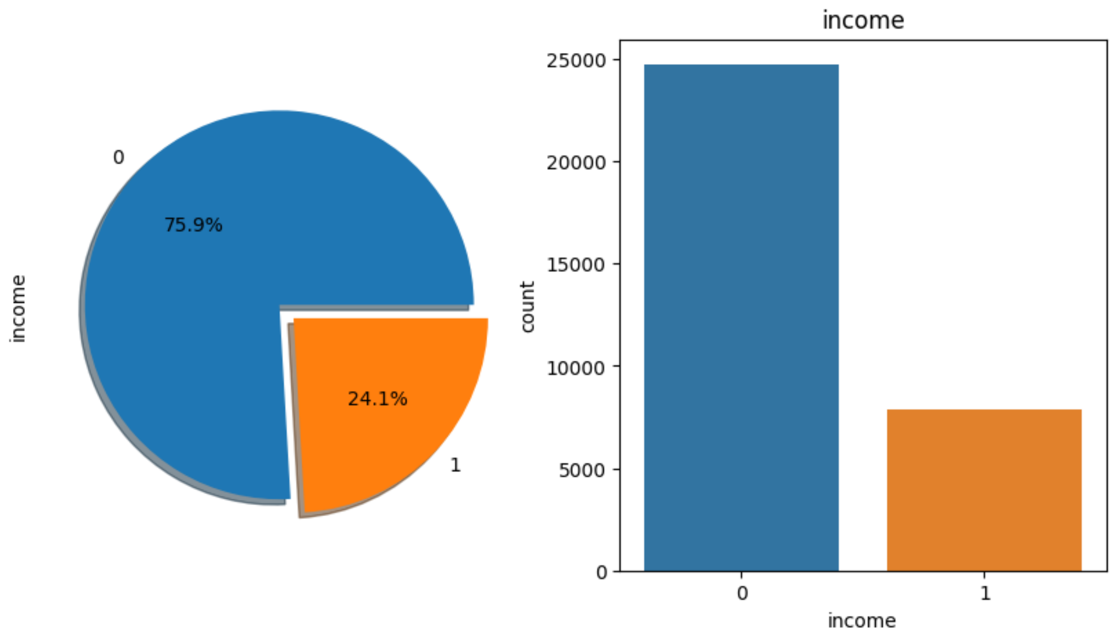


[133]:
```
sns.countplot(x = "race", data=df);
```



[134]:
```
f, ax = plt.subplots(1, 2, figsize=(10, 5))
df['income'].value_counts().plot.pie(explode=[0, 0.1], autopct='%1.1f%%',␣
↪ax=ax[0], shadow=True)
sns.countplot(x='income', data=df, ax=ax[1])
ax[1].set_title('income')
```

[134]: Text(0.5, 1.0, 'income')


income
0
75.9%
24.1%
1
income
25000
20000
15000
10000
5000
0
count
0
1
income