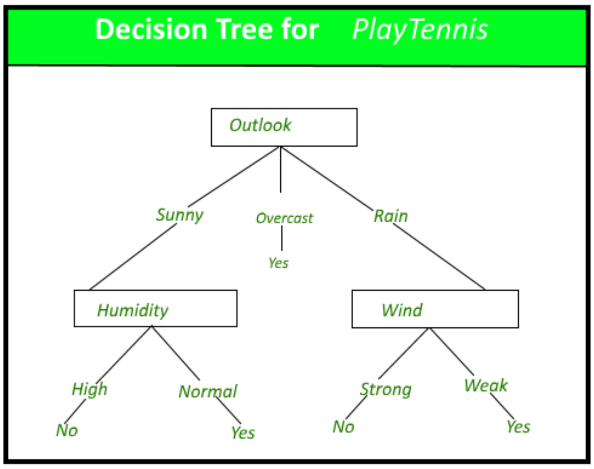
Experiment No. 3
Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance: 16/08/2023
Date of Submission: 13/09/2023

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

Objective: To perform various feature engineering tasks, apply Decision Tree Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score. Improve the performance by performing different data engineering and feature engineering tasks.

Theory:

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



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:

1. Discuss about the how categorical attributes have been dealt with during data pre-processing.

   Categorical attributes, also known as categorical variables or features, are those that represent discrete and distinct categories or groups. These attributes are common in data across various domains, such as gender, color, city names, and more.Label encoding is a simple technique where each category is assigned a unique integer label. For example, if you have a "color" attribute with categories like "red," "blue," and "green," you can encode them as 0, 1, and 2, respectively.Here Label encoder is used to convert categorical to numerical.
2. Discuss the hyper-parameter tuning done based on the decision tree obtained.

   Hyperparameter tuning is a critical step in optimizing the performance of a decision tree model. Decision trees have several hyperparameters that control their structure and behavior, and tuning these hyperparameters can help improve the model's accuracy, generalization, and robustness.Max depth controls the maximum depth of the tree. A deeper tree can capture more complex relationships in the data but is prone to overfitting.We have used max depth as 5.
3. Comment on the accuracy, confusion matrix, precision, recall and F1 score obtained.

   Accuracy obtained in the decision tree model is 85%. 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 0.98 ,recall obtained is 0.27 and f1 score is 0.43.

·adultcensus

Adult Census Income Dataset

```
[ ]: import pandas as pd
     import numpy as np
     import seaborn as sb
     from sklearn.model_selection import train_test_split
     from  sklearn.metrics import accuracy_score
     from sklearn.tree import DecisionTreeClassifier
     import matplotlib.pyplot as plt
```

```
[ ]: df=pd.read_csv("/content/adult.csv",error_bad_lines=False, engine="python")
```

```
<ipython-input-265-8dbf926b9719>:1: FutureWarning: The error_bad_lines argument
has been deprecated and will be removed in a future version. Use on_bad_lines in
the future.


  df=pd.read_csv("/content/adult.csv",error_bad_lines=False, engine="python")
```

```
[ ]: print(df)
```

```
       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]
```

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

[ ]:
```
<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]>
```

[ ]:
```
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
```

[ ]:
```
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
```

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

[ ]:
```
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
```

[ ]:
```
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()
```

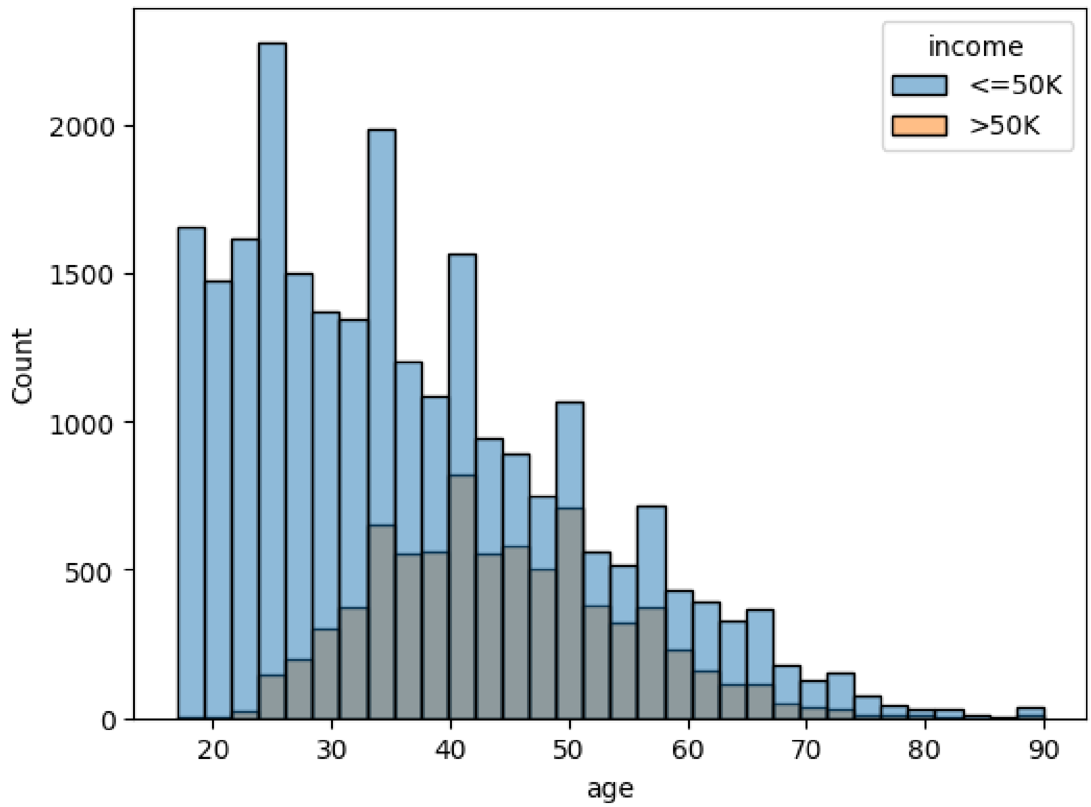
```
df['native.country'].fillna(max_category, inplace=True)
```

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

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

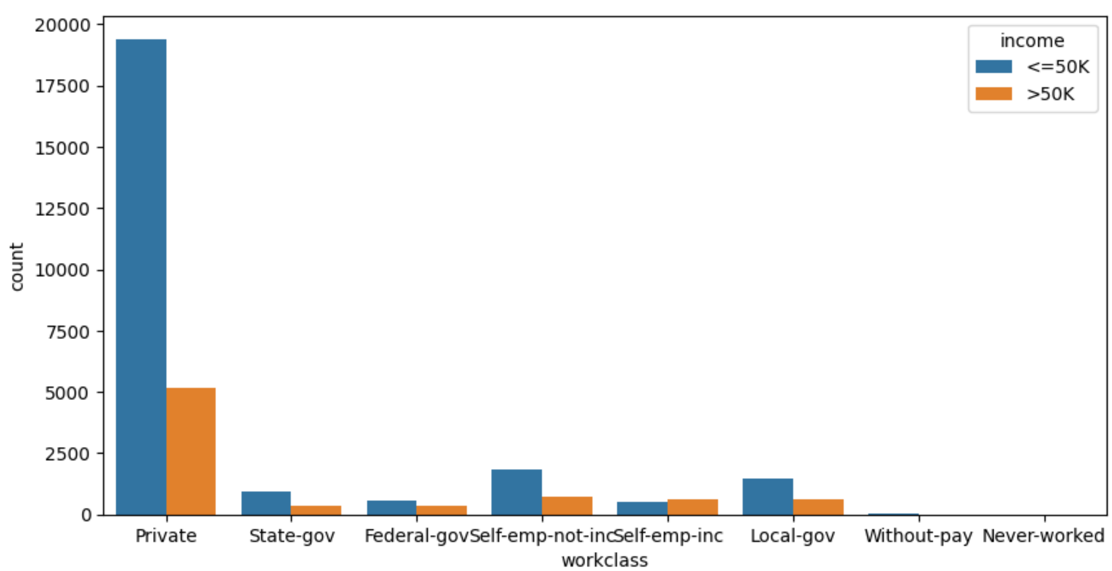
[ ]:
```
sb.histplot(df, x='age', hue='income', bins= 32)
```

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

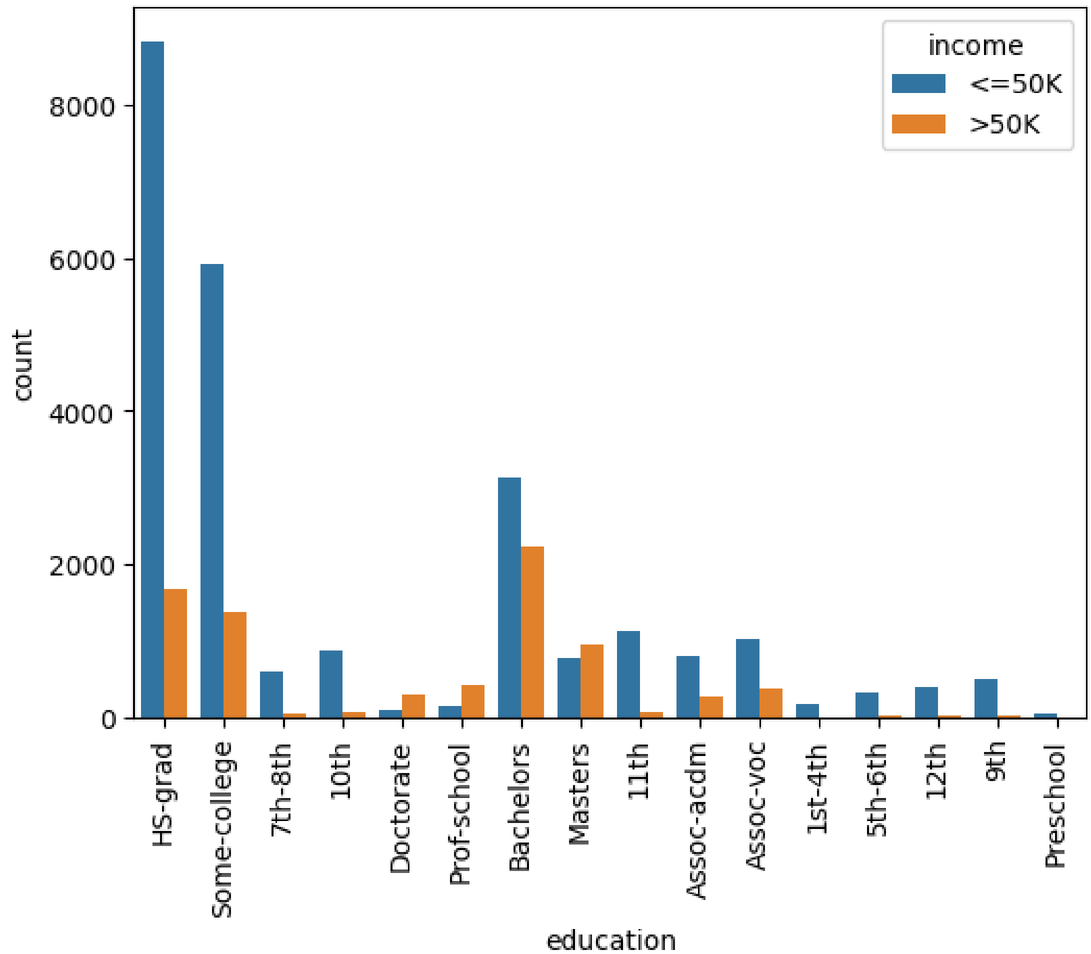



```
[ ]: fig=plt.figure(figsize=(10,5))
     sb.countplot(data = df, x = 'workclass', hue = 'income')
```

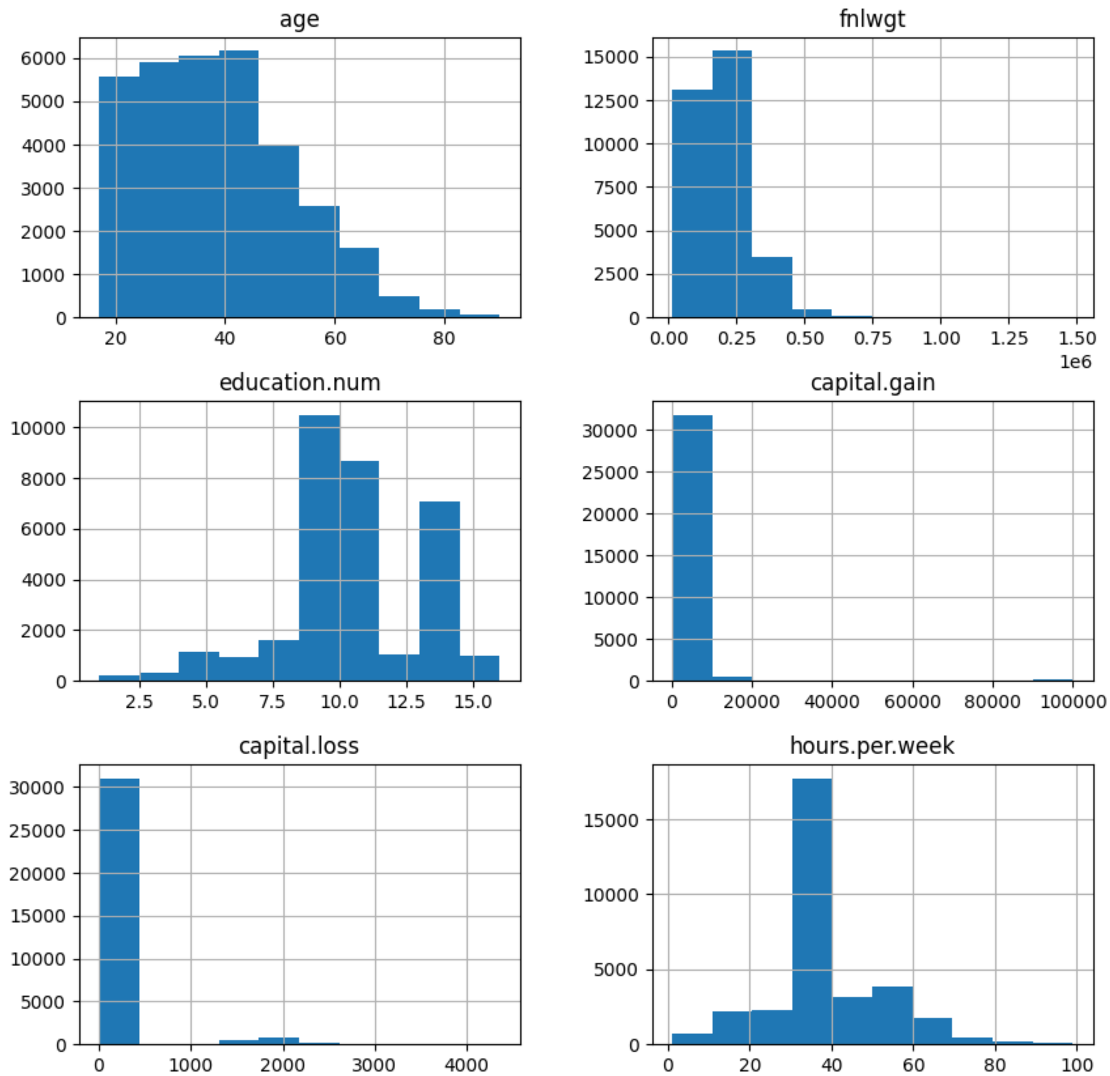
```
[ ]: <Axes: xlabel='workclass', ylabel='count'>
```



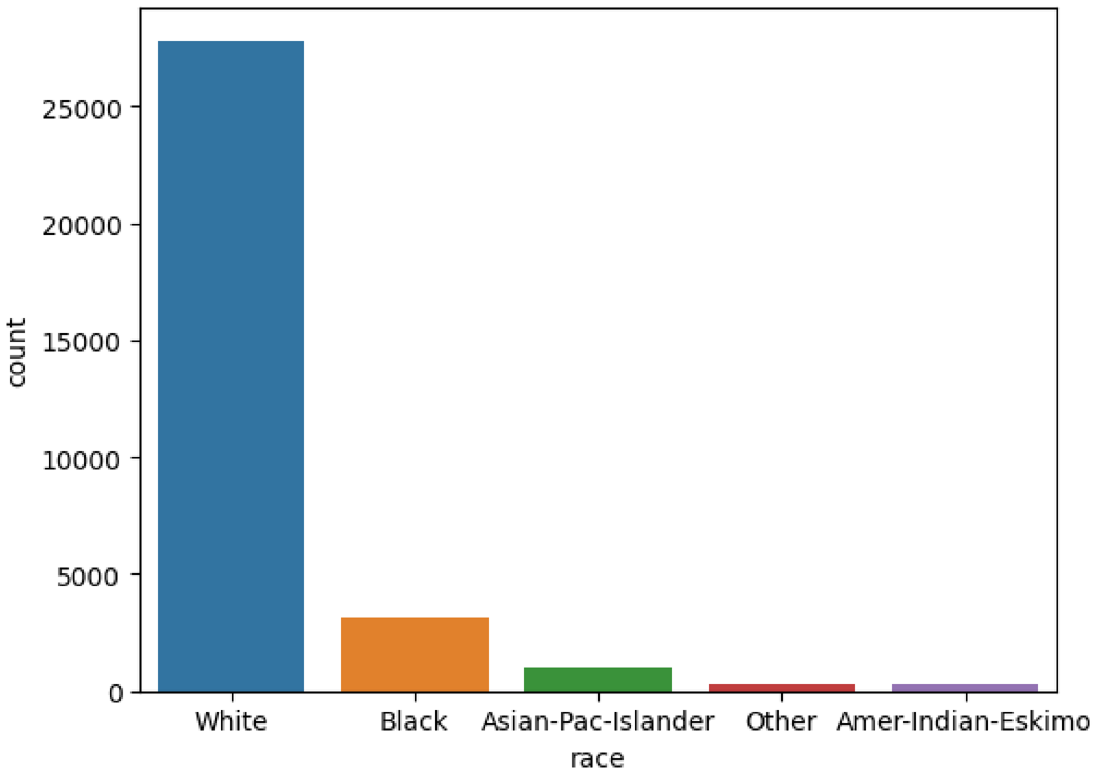
[ ]:
```
sb.countplot(data = df, x = 'education', hue = 'income')
plt.tick_params(axis='x', rotation=90)
```



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

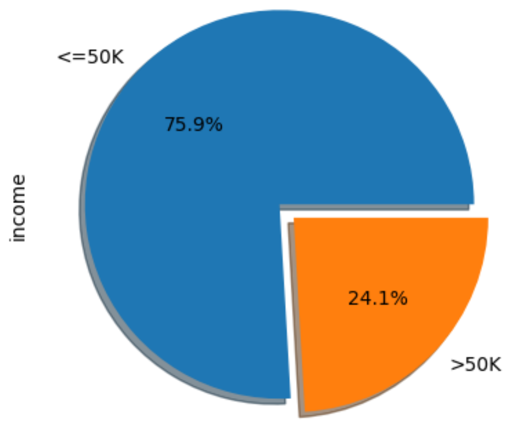


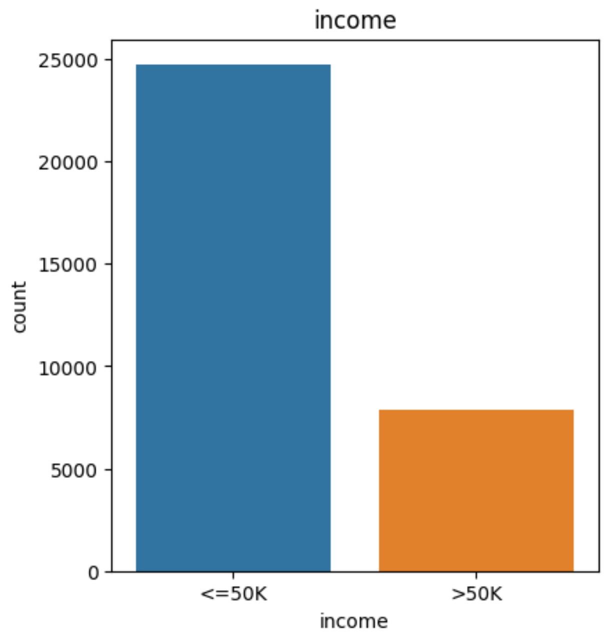
[ ]:
```
sb.countplot(x = "race", data=df);
```



[ ]:
```
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)
sb.countplot(x='income', data=df, ax=ax[1])
ax[1].set_title('income')
```

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

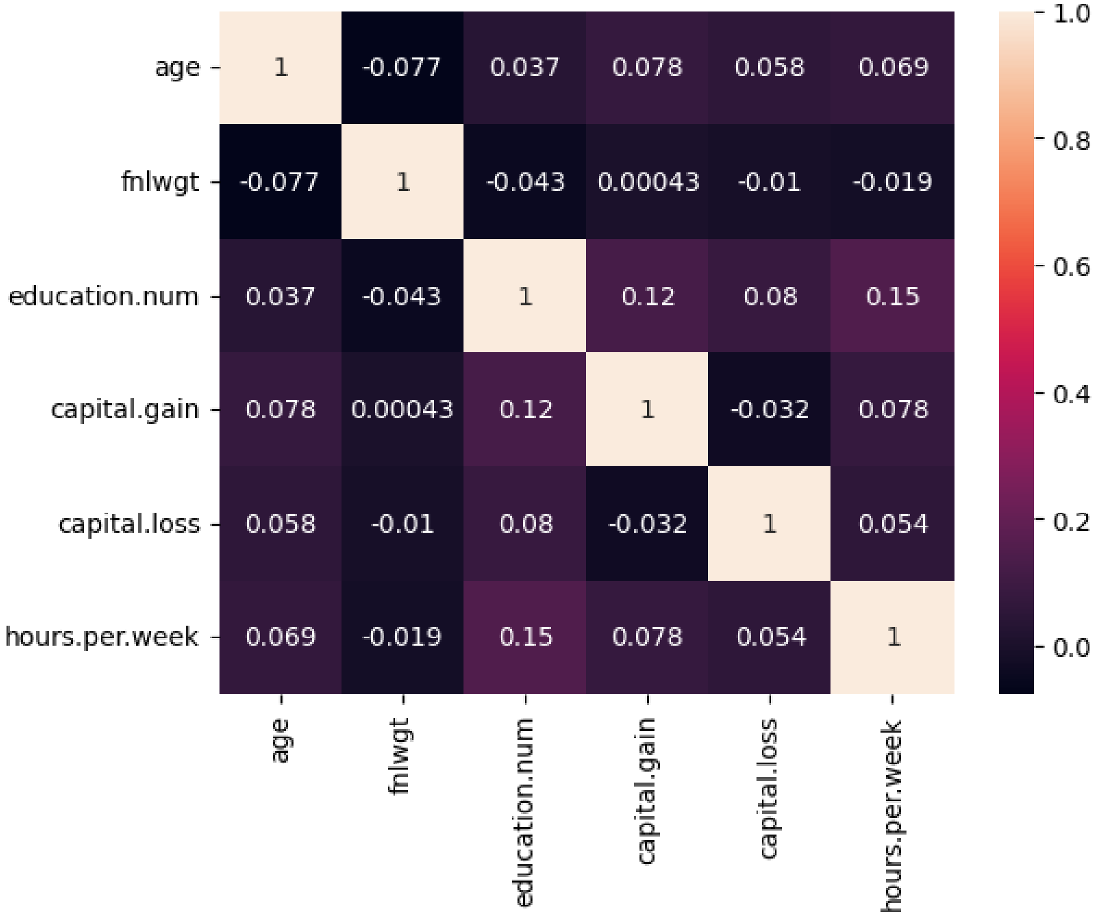




[ ]:
```
corr=df.corr()
sb.heatmap(corr,annot=True)
```

```
<ipython-input-279-5b574c6aa484>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr=df.corr()
```

[ ]: <Axes: >



[ ]:
```
df=df.drop(columns='race')
df=df.drop(columns='fnlwgt')
df=df.drop(columns='education.num')
df=df.drop(columns='relationship')
df=df.drop(columns='native.country')
# df=df.drop(columns='marital.status')
```

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

[ ]:
```
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["native.country"]=labelencoder_x.fit_transform(df["native.country"])
df["income"]=labelencoder_x.fit_transform(df["income"])
df["marital.status"]=labelencoder_x.fit_transform(df["marital.status"])
```

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

```
[ ]: df.head(20)
```

```
[ ]:     age  workclass  education  marital.status  occupation  sex  capital.gain  \
     0    90          3         11               6           9    0             0
     1    82          3         11               6           3    0             0
     2    66          3         15               6           9    0             0
     3    54          3          5               0           6    0             0
     4    41          3         15               5           9    0             0
     5    34          3         11               0           7    0             0
     6    38          3          0               5           0    1             0
     7    74          6         10               4           9    0             0
     8    68          0         11               0           9    0             0
     9    41          3         15               4           2    1             0
     10   45          3         10               0           9    0             0
     11   38          5         14               4           9    1             0
     12   52          3          9               6           7    0             0
     13   32          3         12               5           3    1             0
     14   51          3         10               4           9    1             0
     15   46          3         14               0           9    1             0
     16   45          3          1               0          13    1             0
     17   57          3         12               0           3    1             0
     18   22          3          7               4           5    1             0
     19   34          3          9               5          11    1             0

         capital.loss  hours.per.week  income
     0           4356              40       0
     1           4356              18       0
     2           4356              40       0
     3           3900              40       0
     4           3900              40       0
     5           3770              45       0
     6           3770              40       0
     7           3683              20       1
     8           3683              40       0
     9           3004              60       1
     10          3004              35       1
     11          2824              45       1
     12          2824              20       1
     13          2824              55       1
     14          2824              40       1
     15          2824              40       1
     16          2824              76       1
     17          2824              50       1
     18          2824              40       1
```

19 2824 50 1

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

[ ]:
```
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
```

[ ]:
```
dt=DecisionTreeClassifier(max_depth=5)
dt.fit(x_train,y_train)
```

[ ]:
```
DecisionTreeClassifier(max_depth=5)
```

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

[ ]:
```
['workclass',
 'education',
 'marital.status',
 'occupation',
 'sex',
 'capital.gain',
 'capital.loss',
 'hours.per.week',
 'income']
```

[ ]:
```
y_dtp=dt.predict(x_test)
```

[ ]:
```
print(classification_report(y_test,y_dtp))
```

```
              precision    recall  f1-score   support

           0       0.81      1.00      0.90      7410
           1       0.98      0.27      0.43      2359

    accuracy                           0.82      9769
   macro avg       0.90      0.64      0.66      9769
weighted avg       0.85      0.82      0.78      9769
```

[ ]:
```
rf=RandomForestClassifier(random_state=1)
rf.fit(x_train,y_train)
y_rfp=rf.predict(x_test)
```

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

```
Random Forest :  84.58388780837342
```

[ ]:
```
!pip install my-package
```

```
Requirement already satisfied: my-package in /usr/local/lib/python3.10/dist-
packages (0.0.0)
```

[ ]:
```
!pip install pydotplus
```

```
Requirement already satisfied: pydotplus in /usr/local/lib/python3.10/dist-
packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in
/usr/local/lib/python3.10/dist-packages (from pydotplus) (3.1.1)
```

[ ]:
```
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz
import pydotplus,graphviz
```

[ ]:
```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-
packages (0.20.1)
```

[ ]:
```
dot_data = StringIO()
export_graphviz(dt, out_file=dot_data,
feature_names=features, filled=True,rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

[ ]: