Charity Donors Prediction

In this project we will predict the chances of a certain person donating to CharityML.

Importing required libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing our data

In [2]:
```
data=pd.read_csv('census.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week
0	39	State-gov	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0
1	50	Self-emp-not-inc	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	13.0
2	38	Private	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0
3	53	Private	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	0.0	40.0
4	28	Private	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	0.0	40.0

Checking for null values

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
age               0
workclass         0
education_level   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
```

So, there are no null values.

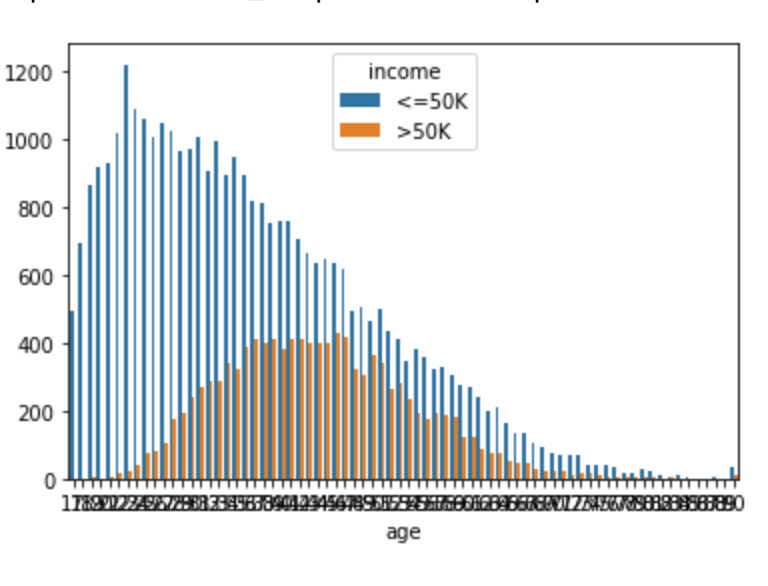
Finding relation between different features and our target_variable

Target variable is income.

Relation between age and income.

In [11]:
```
sns.countplot(x='age',hue='income',data=data)
```

Out[11]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc76d850c8>`

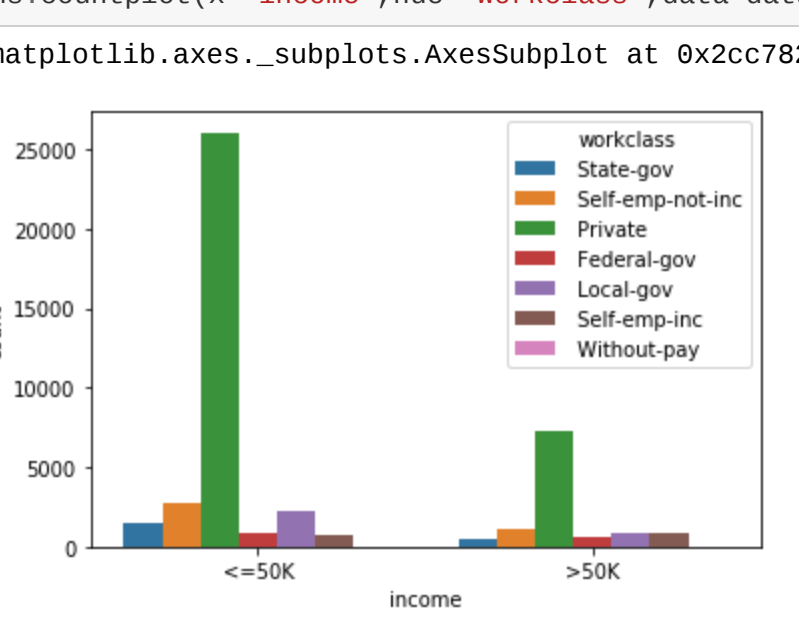


We can see that the number of people earning 50K or less decreases with the increase in age.

Relation between workclass and income.

In [14]:
```
sns.countplot(x='income',hue='workclass',data=data)
```

Out[14]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc782c0448>`

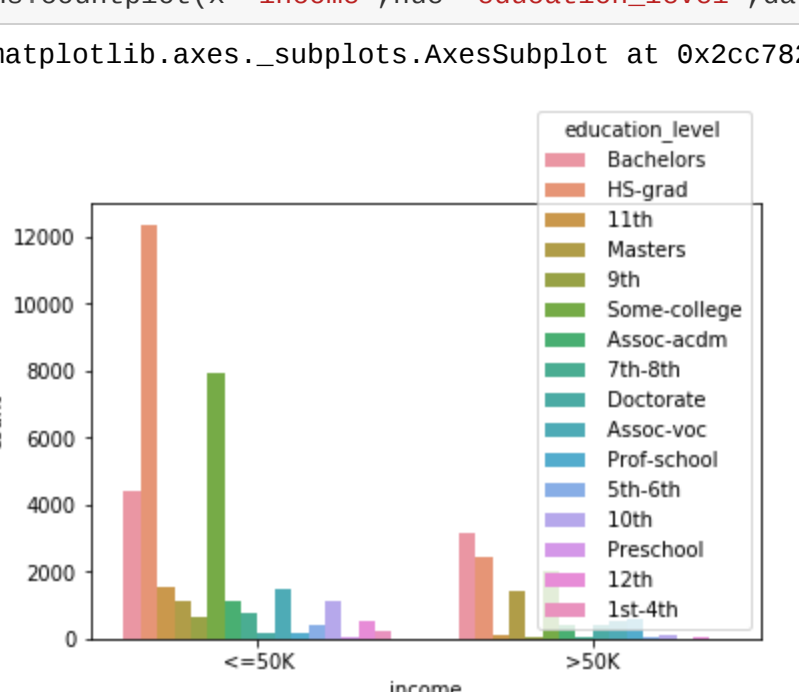


We can see that the majority of people who earn less than 50K work in private sector.

Relation between education level and income.

In [16]:
```
sns.countplot(x='income',hue='education_level',data=data)
```

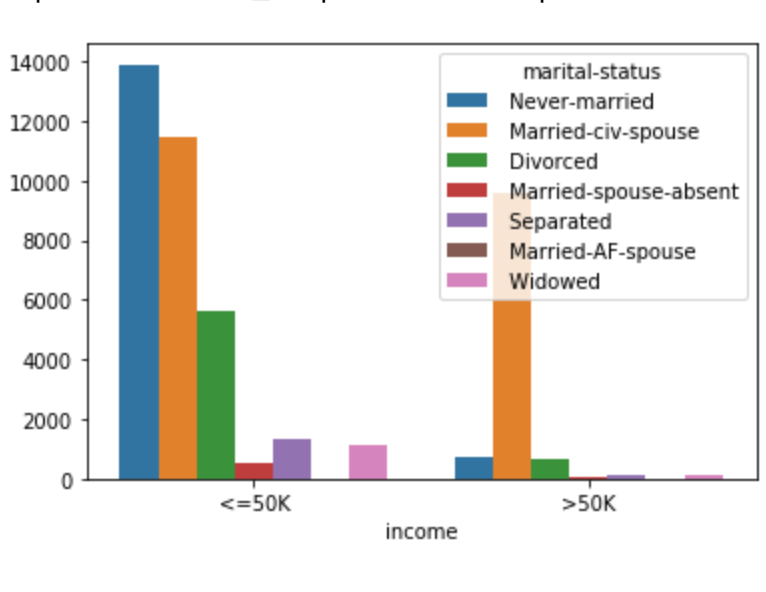
Out[16]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc782bc708>`



Relation between marital status and income.

In [20]:
```
sns.countplot(x='income',hue='marital-status',data=data)
```

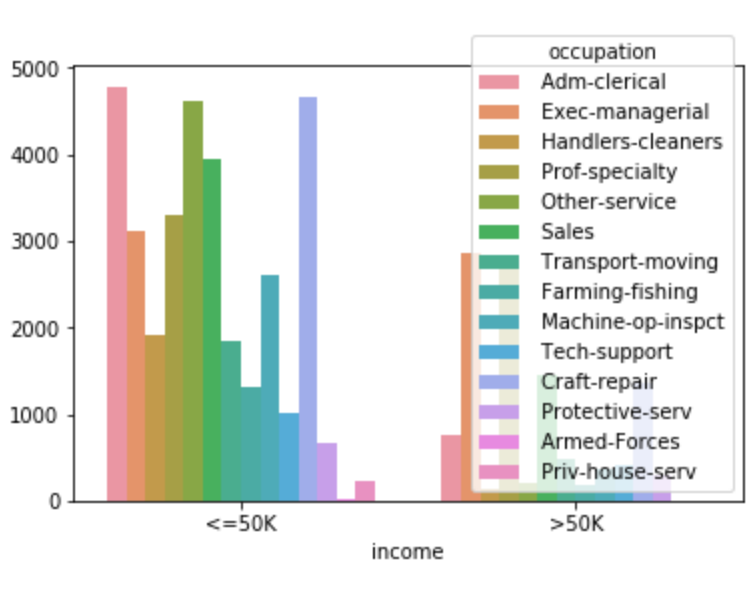
Out[20]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc7869e308>`



We see that the majority of people earning more than 50K are married.

In [21]:
```
sns.countplot(x='income',hue='occupation',data=data)
```

Out[21]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc78720b08>`

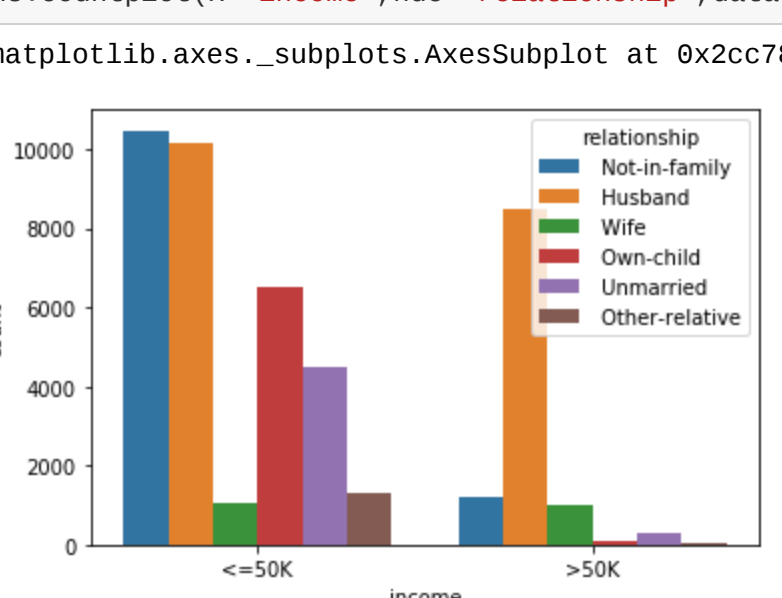

occupation
Adm-clerical
Exec-managerial
Handlers-cleaners
Prof-specialty
Other-service
Sales
Transport-moving
Farming-fishing
Machine-op-inspct
Tech-support
Craft-repair
Protective-serv
Armed-Forces
Priv-house-serv
count
<=50K
>50K
income

This graphs does not help us in drawing useful conclusions.

Relation between relationship and income.

In [22]:
```
sns.countplot(x='income',hue='relationship',data=data)
```

Out[22]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc78720a88>`

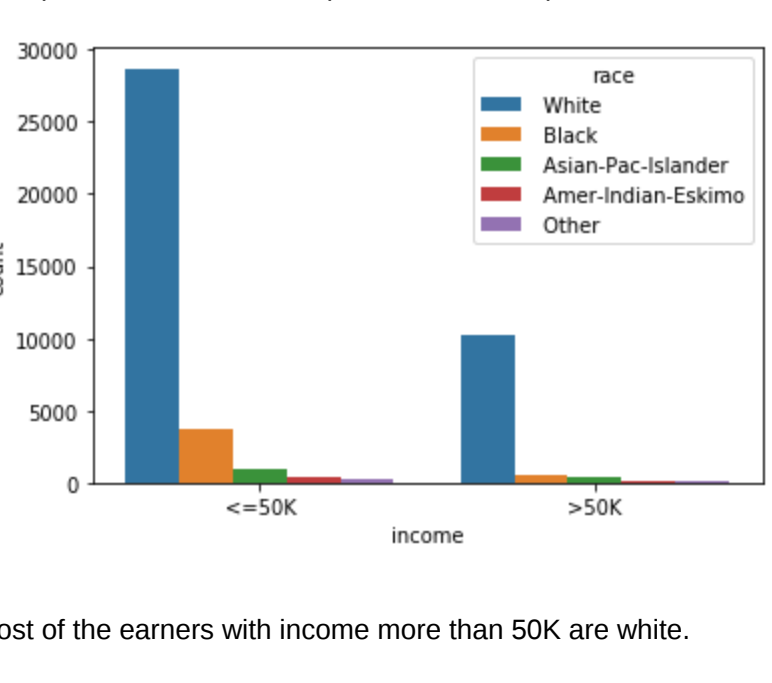


We can conclude that most of the people with income more than 50K are married and are male.

Relation between race and income.

In [23]:
```
sns.countplot(x='income',hue='race',data=data)
```

Out[23]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc788ef748>`

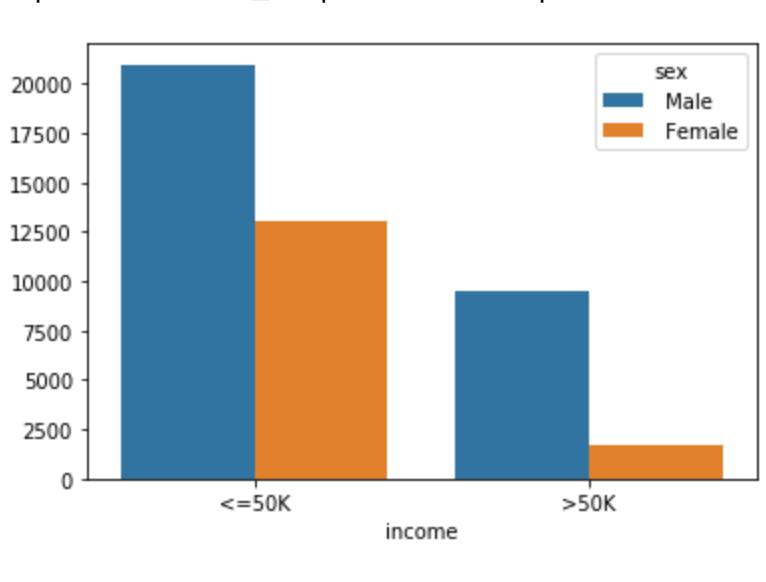


Most of the earners with income more than 50K are white.

Relation between sex and income.

In [24]:
```
sns.countplot(x='income',hue='sex',data=data)
```

Out[24]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc78969cc8>`



Most of the earners with income more than 50K are male.

Relation between one's native country and income.

Converting categorical data into indicator data

In [29]:
```
data.head()
```

Out[29]:

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week
0	39	State-gov	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0
1	50	Self-emp-not-inc	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	13.0
2	38	Private	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0
3	53	Private	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	0.0	40.0
4	28	Private	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	0.0	40.0

In [30]:
```
marital_status=pd.get_dummies(data['marital-status'],drop_first=True)
relationship=pd.get_dummies(data['relationship'],drop_first=True)
race=pd.get_dummies(data['race'],drop_first=True)
sex=pd.get_dummies(data['sex'],drop_first=True)
```

Concatinating this data into our main dataframe.

In [31]:
```
data=pd.concat([data,marital_status,relationship,race,sex],axis=1)
data.head()
```

Out[31]:

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	...	Not-in-family	Other-relative
0	39	State-gov	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	...	1	0
1	50	Self-emp-not-inc	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	...	0	0
2	38	Private	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	...	1	0
3	53	Private	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	...	0	0
4	28	Private	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	...	0	0

5 rows × 30 columns

In [33]:
```
workclass=pd.get_dummies(data['workclass'],drop_first=True)
```

In [34]:
```
data=pd.concat([data,workclass],axis=1)
data.head()
```

Out[34]:

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	...	Black	Other
0	39	State-gov	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	...	0	0
1	50	Self-emp-not-inc	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	...	0	0
2	38	Private	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	...	0	0
3	53	Private	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	...	1	0
4	28	Private	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	...	1	0

5 rows × 36 columns

Separating our required features

In [35]:
```
data.drop(['workclass','education_level','education-num','marital-status','occupation','rela
tionship','race','sex'],axis=1,inplace=True)
```

In [36]:
```
data.head()
```

Out[36]:

	age	capital-gain	capital-loss	hours-per-week	native-country	income	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	...	Black	Other	White	Male
0	39	2174.0	0.0	40.0	United-States	<=50K	0	0	0	1	...	0	0	1	1
1	50	0.0	0.0	13.0	United-States	<=50K	0	1	0	0	...	0	0	1	1
2	38	0.0	0.0	40.0	United-States	<=50K	0	0	0	0	...	0	0	1	1
3	53	0.0	0.0	40.0	United-States	<=50K	0	1	0	0	...	1	0	0	1
4	28	0.0	0.0	40.0	Cuba	<=50K	0	1	0	0	...	1	0	0	0

5 rows × 28 columns

In [37]:
```
data.drop('native-country',axis=1,inplace=True)
```

In [38]:
```
data.head()
```

Out[38]:

	age	capital-gain	capital-loss	hours-per-week	income	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	Separated	...	Black	Other	White	Male
0	39	2174.0	0.0	40.0	<=50K	0	0	0	1	0	...	0	0	1	1
1	50	0.0	0.0	13.0	<=50K	0	1	0	0	0	...	0	0	1	1
2	38	0.0	0.0	40.0	<=50K	0	0	0	0	0	...	0	0	1	1
3	53	0.0	0.0	40.0	<=50K	0	1	0	0	0	...	1	0	0	1
4	28	0.0	0.0	40.0	<=50K	0	1	0	0	0	...	1	0	0	0

5 rows × 27 columns

In [39]:
```
data.drop('hours-per-week',axis=1,inplace=True)
```

In [40]:
```
data.head()
```

Out[40]:

	age	capital-gain	capital-loss	income	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	Separated	Widowed	...	Black	Other	White	Male
0	39	2174.0	0.0	<=50K	0	0	0	1	0	0	...	0	0	1	1
1	50	0.0	0.0	<=50K	0	1	0	0	0	0	...	0	0	1	1
2	38	0.0	0.0	<=50K	0	0	0	0	0	0	...	0	0	1	1
3	53	0.0	0.0	<=50K	0	1	0	0	0	0	...	1	0	0	1
4	28	0.0	0.0	<=50K	0	1	0	0	0	0	...	1	0	0	0

5 rows × 26 columns

Checking the relation between profit and income

In [41]:
```
def profit(cols):
    gain=cols[0]
    loss=cols[1]

    return gain-loss
```

In [43]:
```
data['profit']=data[['capital-gain','capital-loss']].apply(profit,axis=1)
data.head()
```

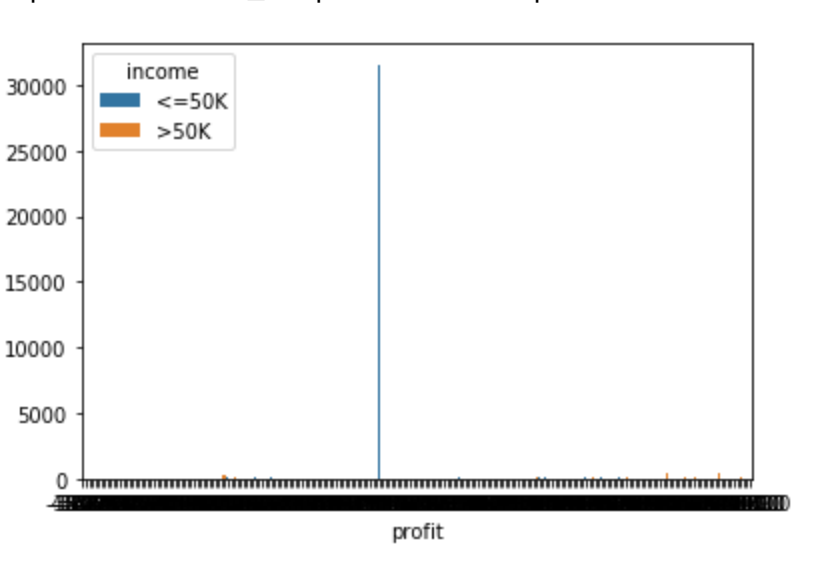
Out[43]:

	age	capital-gain	capital-loss	income	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	Separated	Widowed	...	Other	White	Male	Local-gov
0	39	2174.0	0.0	<=50K	0	0	0	1	0	0	...	0	1	1	0
1	50	0.0	0.0	<=50K	0	1	0	0	0	0	...	0	1	1	0
2	38	0.0	0.0	<=50K	0	0	0	0	0	0	...	0	1	1	0
3	53	0.0	0.0	<=50K	0	1	0	0	0	0	...	0	0	1	0
4	28	0.0	0.0	<=50K	0	1	0	0	0	0	...	0	0	0	0

5 rows × 27 columns

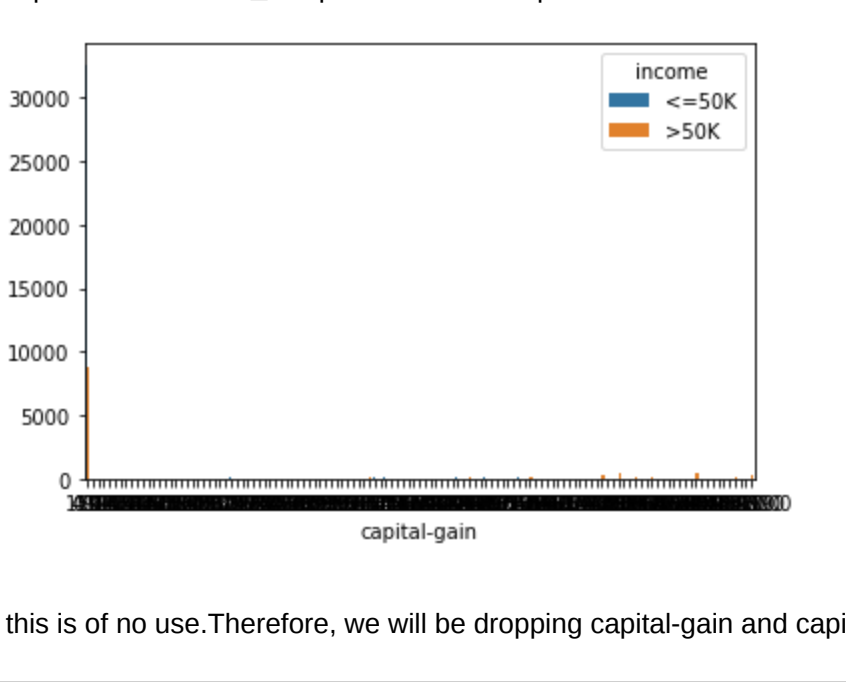
In [47]:
```
sns.countplot(x='profit',hue='income',data=data)
```

Out[47]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc7c0c2ec8>`



In [48]:
```
sns.countplot(x='capital-gain',hue='income',data=data)
```

Out[48]: `<matplotlib.axes._subplots.AxesSubplot at 0x2cc7c8b3248>`



As this is of no use.Therefore, we will be dropping capital-gain and capital loss.

In [51]:
```
data.drop(['capital-gain','capital-loss'],axis=1,inplace=True)
```

In [52]:
```
data.head()
```

Out[52]:

	age	income	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	Separated	Widowed	Not-in-family	Other-relative	...	Other	White	Male	Local-gov
0	39	<=50K	0	0	0	1	0	0	1	0	...	0	1	1	0
1	50	<=50K	0	1	0	0	0	0	0	0	...	0	1	1	0
2	38	<=50K	0	0	0	0	0	0	1	0	...	0	1	1	0
3	53	<=50K	0	1	0	0	0	0	0	0	...	0	0	1	0
4	28	<=50K	0	1	0	0	0	0	0	0	...	0	0	0	0

5 rows × 25 columns

Converting income column

In [55]:
```
income=pd.get_dummies(data['income'],drop_first=True)
```

In [56]:
```
data=pd.concat([data,income],axis=1)
data.head()
```

Out[56]:

	age	income	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	Separated	Widowed	Not-in-family	Other-relative	...	White	Male	Local-gov	Private
0	39	<=50K	0	0	0	1	0	0	1	0	...	1	1	0	0
1	50	<=50K	0	1	0	0	0	0	0	0	...	1	1	0	0
2	38	<=50K	0	0	0	0	0	0	1	0	...	1	1	0	1
3	53	<=50K	0	1	0	0	0	0	0	0	...	0	1	0	1
4	28	<=50K	0	1	0	0	0	0	0	0	...	0	0	0	1

5 rows × 26 columns

Dropping income column.

In [57]:
```
data.drop('income',axis=1,inplace=True)
```

In [59]:
```
data.head()
```

Out[59]:

	age	Married-AF-spouse	Married-civ-spouse	Married-spouse-absent	Never-married	Separated	Widowed	Not-in-family	Other-relative	Own-child	...	White	Male	Local-gov	Private
0	39	0	0	0	1	0	0	1	0	0	...	1	1	0	0
1	50	0	1	0	0	0	0	0	0	0	...	1	1	0	0
2	38	0	0	0	0	0	0	1	0	0	...	1	1	0	1
3	53	0	1	0	0	0	0	0	0	0	...	0	1	0	1
4	28	0	1	0	0	0	0	0	0	0	...	0	0	0	1

5 rows × 25 columns

Building our model

Importing required libraries.

In [60]:
```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier()
```

Training and testing our model

Importing required libraries.

In [61]:
```
from sklearn.model_selection import train_test_split
```

In [62]:
```
x=data.drop('>50K',axis=1)
y=data['>50K']
```

Splitting our data.

In [63]:
```
x_test,x_train,y_test,y_train=train_test_split(x,y,test_size=0.2)
```

In [64]:
```
classifier.fit(x_train,y_train)
```

Out[64]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Predicting from testing set.

In [65]:
```
test_predictions=classifier.predict(x_test)
```

In [66]:
```
test_predictions
```

Out[66]:
```
array([0, 0, 0, ..., 1, 0, 0], dtype=uint8)
```

Checking accuracy

In [67]:
```
from sklearn.metrics import accuracy_score
```

In [68]:
```
accuracy=accuracy_score(y_test,test_predictions)
```