Classifying Personal Income

In [1]:
```
import pandas as pd#to work with dataframes
import seaborn as sns
```

In [2]:
```
import numpy as np#to perform numerical operation
```

In [3]:
```
from sklearn.model_selection import train_test_split#to partion he data
```

In [4]:
```
from sklearn.linear_model import LogisticRegression#importing library for logisti
```

In [5]:
```
from sklearn.metrics import accuracy_score,confusion_matrix#import performance me
```

In [6]:
```
data_income=pd.read_csv('income.csv')#importing data into spyder

data=data_income.copy()
```

In [7]:
```
#Exploratory data analysis
#Getting to know the data
#DATA PREPROCESSING(mISSING VALUES)
#CROSS TABLES AND DATA VISUALISATION
```

Getting to know the data

In [8]:
```
print(data.info())#to check variables data type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31978 entries, 0 to 31977
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            31978 non-null  int64 
 1   JobType        31978 non-null  object
 2   EdType         31978 non-null  object
 3   maritalstatus  31978 non-null  object
 4   occupation     31978 non-null  object
 5   relationship   31978 non-null  object
 6   race           31978 non-null  object
 7   gender         31978 non-null  object
 8   capitalgain    31978 non-null  int64 
 9   capitalloss    31978 non-null  int64 
 10  hoursperweek   31978 non-null  int64 
 11  nativecountry  31978 non-null  object
 12  SalStat        31978 non-null  object
dtypes: int64(4), object(9)
memory usage: 3.2+ MB
None
```

In [9]:
```
data.isnull()#to check the missing values
```

Out[9]:

	age	JobType	EdType	maritalstatus	occupation	relationship	race	gender	capitalgain
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
31973	False	False	False	False	False	False	False	False	False
31974	False	False	False	False	False	False	False	False	False
31975	False	False	False	False	False	False	False	False	False
31976	False	False	False	False	False	False	False	False	False
31977	False	False	False	False	False	False	False	False	False

31978 rows × 13 columns

In [10]:
```
data.isnull().sum()#to check how many variables values are missing
```

Out[10]:
```
age              0
JobType          0
EdType           0
maritalstatus    0
occupation       0
relationship     0
race             0
gender           0
capitalgain      0
capitalloss      0
hoursperweek     0
nativecountry    0
SalStat          0
dtype: int64
```

In [11]:
```
summary_num=data.describe()#to get summary of numerical varible
print(summary_num)
#total age
```

```
                age   capitalgain   capitalloss  hoursperweek
count  31978.000000  31978.000000  31978.000000  31978.000000
mean      38.579023   1064.360623     86.739352     40.417850
std       13.662085   7298.596271    401.594301     12.345285
min       17.000000      0.000000      0.000000      1.000000
25%       28.000000      0.000000      0.000000     40.000000
50%       37.000000      0.000000      0.000000     40.000000
75%       48.000000      0.000000      0.000000     45.000000
max       90.000000  99999.000000   4356.000000     99.000000
```

In [12]:
```
summary_cate=data.describe(include="O")#summary for categorical variable
print(summary_cate)
#include O GIVES YOU THE SUMMARY OF CATEGORICAL VARIABLE
#count gives you the value of total value enter under the variable
#unique tells the unique value
#top gives u the modal value
# freq gives you the idea of frequency of top value
```

```
         JobType    EdType        maritalstatus       occupation relationship
\
count      31978     31978                31978            31978        31978
unique         9        16                    7               15            6
top      Private   HS-grad   Married-civ-spouse   Prof-specialty      Husband
freq       22286     10368                14692             4038        12947

          race gender   nativecountry                        SalStat
count    31978  31978           31978                          31978
unique       5      2              41                              2
top      White   Male   United-States   less than or equal to 50,000
freq     27430  21370           29170                          24283
```

In [13]:
```
data['JobType'].value_counts()
#it gives the value of each value repeated under attribute name Jobtype
```

Out[13]:
```
 Private             22286
 Self-emp-not-inc     2499
 Local-gov            2067
 ?                    1809
 State-gov            1279
 Self-emp-inc         1074
 Federal-gov           943
 Without-pay            14
 Never-worked            7
Name: JobType, dtype: int64
```

In [14]:
```
data['occupation'].value_counts()
```

Out[14]:
```
 Prof-specialty       4038
 Craft-repair         4030
 Exec-managerial      3992
 Adm-clerical         3721
 Sales                3584
 Other-service        3212
 Machine-op-inspct    1966
 ?                    1816
 Transport-moving     1572
 Handlers-cleaners    1350
 Farming-fishing       989
 Tech-support          912
 Protective-serv       644
 Priv-house-serv       143
 Armed-Forces            9
Name: occupation, dtype: int64
```

In [15]:
```
print(np.unique(data['JobType']))#it gives the name of the unique values enter ur
#note there is a space before the each name of starting column
```

```
[' ?' ' Federal-gov' ' Local-gov' ' Never-worked' ' Private'
 ' Self-emp-inc' ' Self-emp-not-inc' ' State-gov' ' Without-pay']
```

In [16]:
```
print(np.unique(data['occupation']))
```

```
[' ?' ' Adm-clerical' ' Armed-Forces' ' Craft-repair' ' Exec-managerial'
 ' Farming-fishing' ' Handlers-cleaners' ' Machine-op-inspct'
 ' Other-service' ' Priv-house-serv' ' Prof-specialty' ' Protective-serv'
 ' Sales' ' Tech-support' ' Transport-moving']
```

In [17]:
```
data=pd.read_csv('income.csv',na_values=[" ?"])#we do this because " ?" consider
#by adding any undercore value with the list of string value we are considering t
```

Data preprocessing

In [18]:
```
data.isnull().sum()#1809 missing values under variable jobtype
#1816 missing values under the variable occupation
```

Out[18]:
```
age                0
JobType         1809
EdType             0
maritalstatus      0
occupation      1816
relationship       0
race               0
gender             0
capitalgain        0
capitalloss        0
hoursperweek       0
nativecountry      0
SalStat            0
dtype: int64
```

In [19]:
```
missing=data[data.isnull().any(axis=1)]
#it will print the answer where atleast one row has nan values
print(missing)
#we have 7 rows where ocupaation has nan values and Job type has a value(Never wo
```

```
       age JobType         EdType        maritalstatus occupation  \
8       17     NaN           11th        Never-married        NaN
17      32     NaN   Some-college   Married-civ-spouse        NaN
29      22     NaN   Some-college        Never-married        NaN
42      52     NaN           12th        Never-married        NaN
44      63     NaN        1st-4th   Married-civ-spouse        NaN
...    ...     ...            ...                  ...        ...
31892   59     NaN      Bachelors   Married-civ-spouse        NaN
31934   20     NaN        HS-grad        Never-married        NaN
31945   28     NaN   Some-college   Married-civ-spouse        NaN
31967   80     NaN        HS-grad              Widowed        NaN
31968   17     NaN           11th        Never-married        NaN

          relationship    race   gender  capitalgain  capitalloss  \
8            Own-child   White   Female            0            0
17             Husband   White     Male            0            0
29           Own-child   White     Male            0            0
42      Other-relative   Black     Male          594            0
44             Husband   White     Male            0            0
...                ...     ...      ...          ...          ...
31892          Husband   White     Male            0            0
31934   Other-relative   White   Female            0            0
31945             Wife   White   Female            0         1887
31967    Not-in-family   White     Male            0            0
31968        Own-child   White     Male            0            0

       hoursperweek   nativecountry                       SalStat
8                 5   United-States   less than or equal to 50,000
17               40   United-States   less than or equal to 50,000
29               40   United-States   less than or equal to 50,000
42               40   United-States   less than or equal to 50,000
44               35   United-States   less than or equal to 50,000
...             ...             ...                            ...
31892            40   United-States            greater than 50,000
31934            35   United-States   less than or equal to 50,000
31945            40   United-States            greater than 50,000
31967            24   United-States   less than or equal to 50,000
31968            40   United-States   less than or equal to 50,000

[1816 rows x 13 columns]
```

In [20]:
```
data2=data.dropna(axis=0)#here we are deleting row that contain missing values
```

In [21]:
```
#Relationship between independent variable
correlation=data2.corr()
correlation
#here the nan of the values are nearer to one they are nearer to zero that none c
```

Out[21]:

	age	capitalgain	capitalloss	hoursperweek
age	1.000000	0.080154	0.060165	0.101599
capitalgain	0.080154	1.000000	-0.032229	0.080432
capitalloss	0.060165	-0.032229	1.000000	0.052417
hoursperweek	0.101599	0.080432	0.052417	1.000000

Cross table and data visualisation

In [22]:
```
data2.columns
#extracting the column names of each variable
```

Out[22]:
```
Index(['age', 'JobType', 'EdType', 'maritalstatus', 'occupation',
       'relationship', 'race', 'gender', 'capitalgain', 'capitalloss',
       'hoursperweek', 'nativecountry', 'SalStat'],
      dtype='object')
```

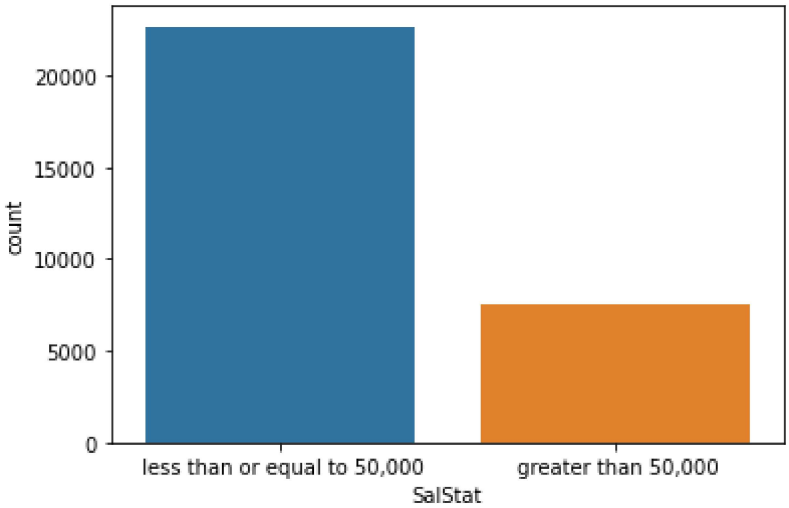
In [23]:
```
gender=pd.crosstab(index=data2["gender"],columns="count",normalize=True)
print(gender)
#from here we get index value of the gender and the different rows proportion nor
```

```
col_0       count
gender
 Female  0.324315
 Male    0.675685
```

In [24]:
```
gender_salstat=pd.crosstab(index=data2["gender"],columns=data2['SalStat'],margins
print(gender_salstat)
#from here only 11% of females earns greater than 50000 and 88% of females earns
#here man are more likely to earn 50000 as compare to females
```

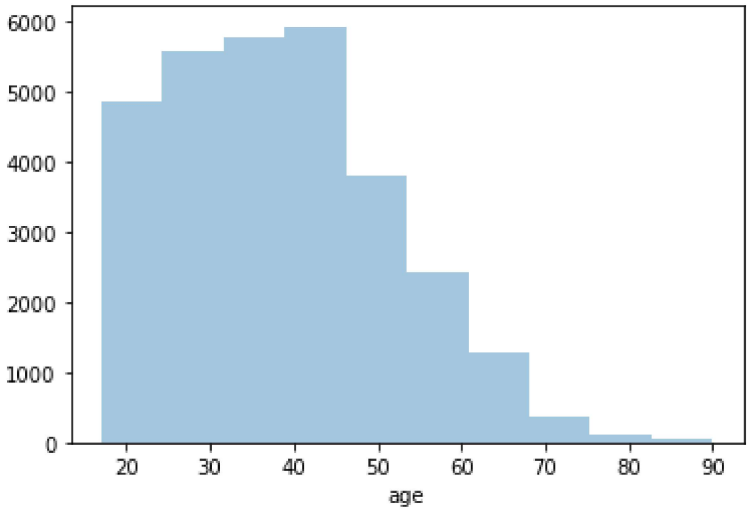
```
SalStat   greater than 50,000   less than or equal to 50,000
gender
 Female              0.113678                       0.886322
 Male                0.313837                       0.686163
All                  0.248922                       0.751078
```

In [25]:
```
SalStat=sns.countplot(data2['SalStat'])
#from here we know 75% person has less than 50000 income
#from here 25% person has more than 50000 salary
```



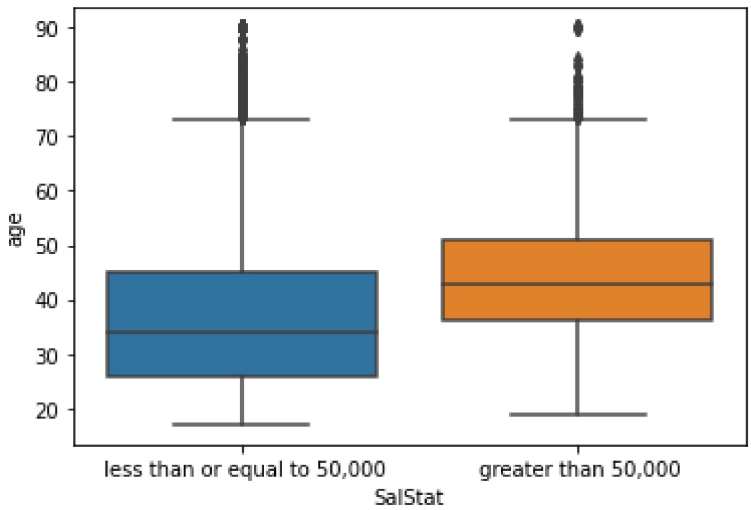
In [26]:
```
sns.distplot(data2['age'],bins=10,kde=False)
#kde=false so that the frequency is on  y axis
#people with age 20-45 age are high in frequency
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1c9a6107c40>



In [27]:
```
#to know how the age affecting the salary we can use box plot
sns.boxplot('SalStat','age',data=data2)
data2.groupby('SalStat')['age'].median()
#people with age 35-50 age are more likely to earn>50000
#people with age 25-35 are likely to earn less than 50000
```

Out[27]:
```
SalStat
 greater than 50,000             43
 less than or equal to 50,000    34
Name: age, dtype: int64
```



similarly we can use graphs and crosstab to get the data relation about the other variable

Exploratory data analysis---Analyse the data yourself with different categorical data and Salstat

Logistic regression --It is machine learning classification alogorithim that is use to predict the probablity of a data categorical dependent variable

In [28]:
```
#it is difficult to deal with categorical data so we are converting categorical r
#revaluing the salary status name from 0,1

data2['SalStat']=data2['SalStat'].map({' less than or equal to 50,000':0,' greate
print(data2['SalStat'])
#the method here we use is integer incoding so that we can invert the inodingg lo
#as in case of prediction
```

```
0        0
1        0
2        1
3        0
4        0
        ..
31973    0
31974    0
31975    0
31976    0
31977    0
Name: SalStat, Length: 30162, dtype: int64


<ipython-input-28-c8b61d3704b9>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  data2['SalStat']=data2['SalStat'].map({' less than or equal to 50,000':0,' gr
eater than 50,000':1})
```

In [29]:
```
new_data=pd.get_dummies(data2,drop_first=True)
#WE CAN CONVERT THE CATEGORICAL VALUES INTO THE DUMY VARIABLES WHICH IS CALLED AS
#COLUMN WHICH HAS CATEGORICAL DATA TO MANY COLUMNS DEPENDING ON THE NUMBER CAEGOR
print(new_data)
#we can take the example of Job type variable intially it has seven unique elemer
#all seven unique values are now columns and each column has value 0 and 1
#here where ever we see 1 it means at index position it is written if it 0 then i
#Dependent variable are refer as y
#independent variable are refer as x
```

```
       age  capitalgain  capitalloss  hoursperweek  SalStat  \
0       45            0            0            28        0
1       24            0            0            40        0
2       44            0            0            40        1
3       27            0            0            40        0
4       20            0            0            35        0
...    ...          ...          ...           ...      ...
31973   34          594            0            60        0
31974   34            0            0            40        0
31975   23            0            0            40        0
31976   42            0            0            40        0
31977   29            0            0            40        0

       JobType_ Local-gov  JobType_ Private  JobType_ Self-emp-inc  \
0                       0                 1                      0
1                       0                 0                      0
2                       0                 1                      0
3                       0                 1                      0
4                       0                 1                      0
...                   ...               ...                    ...
31973                   1                 0                      0
31974                   1                 0                      0
31975                   0                 1                      0
31976                   1                 0                      0
31977                   0                 1                      0

       JobType_ Self-emp-not-inc  JobType_ State-gov  ...  \
0                              0                   0  ...
1                              0                   0  ...
2                              0                   0  ...
3                              0                   0  ...
4                              0                   0  ...
...                          ...                 ...  ...
31973                          0                   0  ...
31974                          0                   0  ...
31975                          0                   0  ...
31976                          0                   0  ...
31977                          0                   0  ...

       nativecountry_ Portugal  nativecountry_ Puerto-Rico  \
0                            0                            0
1                            0                            0
2                            0                            0
3                            0                            0
4                            0                            0
...                        ...                          ...
```

```
31973                        0                           0
31974                        0                           0
31975                        0                           0
31976                        0                           0
31977                        0                           0

       nativecountry_ Scotland  nativecountry_ South  nativecountry_ Taiwan  \
0                            0                     0                      0
1                            0                     0                      0
2                            0                     0                      0
3                            0                     0                      0
4                            0                     0                      0
...                        ...                   ...                    ...
31973                        0                     0                      0
31974                        0                     0                      0
31975                        0                     0                      0
31976                        0                     0                      0
31977                        0                     0                      0

       nativecountry_ Thailand  nativecountry_ Trinadad&Tobago  \
0                            0                               0
1                            0                               0
2                            0                               0
3                            0                               0
4                            0                               0
...                        ...                             ...
31973                        0                               0
31974                        0                               0
31975                        0                               0
31976                        0                               0
31977                        0                               0

       nativecountry_ United-States  nativecountry_ Vietnam  \
0                                 1                       0
1                                 1                       0
2                                 1                       0
3                                 0                       0
4                                 1                       0
...                             ...                     ...
31973                             1                       0
31974                             1                       0
31975                             1                       0
31976                             1                       0
31977                             1                       0

       nativecountry_ Yugoslavia
0                              0
1                              0
2                              0
3                              0
4                              0
...                          ...
31973                          0
31974                          0
31975                          0
31976                          0
31977                          0
```

```
[30162 rows x 95 columns]
```

In [30]:
```
#Storing column names
columns_list=list(new_data.columns)
print(columns_list)
```

```
['age', 'capitalgain', 'capitalloss', 'hoursperweek', 'SalStat', 'JobType_ Loca
l-gov', 'JobType_ Private', 'JobType_ Self-emp-inc', 'JobType_ Self-emp-not-in
c', 'JobType_ State-gov', 'JobType_ Without-pay', 'EdType_ 11th', 'EdType_ 12t
h', 'EdType_ 1st-4th', 'EdType_ 5th-6th', 'EdType_ 7th-8th', 'EdType_ 9th', 'Ed
Type_ Assoc-acdm', 'EdType_ Assoc-voc', 'EdType_ Bachelors', 'EdType_ Doctorat
e', 'EdType_ HS-grad', 'EdType_ Masters', 'EdType_ Preschool', 'EdType_ Prof-sc
hool', 'EdType_ Some-college', 'maritalstatus_ Married-AF-spouse', 'maritalstat
us_ Married-civ-spouse', 'maritalstatus_ Married-spouse-absent', 'maritalstatus
_ Never-married', 'maritalstatus_ Separated', 'maritalstatus_ Widowed', 'occupa
tion_ Armed-Forces', 'occupation_ Craft-repair', 'occupation_ Exec-managerial',
'occupation_ Farming-fishing', 'occupation_ Handlers-cleaners', 'occupation_ Ma
chine-op-inspct', 'occupation_ Other-service', 'occupation_ Priv-house-serv',
'occupation_ Prof-specialty', 'occupation_ Protective-serv', 'occupation_ Sale
s', 'occupation_ Tech-support', 'occupation_ Transport-moving', 'relationship_
Not-in-family', 'relationship_ Other-relative', 'relationship_ Own-child', 'rel
ationship_ Unmarried', 'relationship_ Wife', 'race_ Asian-Pac-Islander', 'race_
Black', 'race_ Other', 'race_ White', 'gender_ Male', 'nativecountry_ Canada',
'nativecountry_ China', 'nativecountry_ Columbia', 'nativecountry_ Cuba', 'nati
vecountry_ Dominican-Republic', 'nativecountry_ Ecuador', 'nativecountry_ El-Sa
lvador', 'nativecountry_ England', 'nativecountry_ France', 'nativecountry_ Ger
many', 'nativecountry_ Greece', 'nativecountry_ Guatemala', 'nativecountry_ Hai
ti', 'nativecountry_ Holand-Netherlands', 'nativecountry_ Honduras', 'nativecou
ntry_ Hong', 'nativecountry_ Hungary', 'nativecountry_ India', 'nativecountry_
Iran', 'nativecountry_ Ireland', 'nativecountry_ Italy', 'nativecountry_ Jamaic
a', 'nativecountry_ Japan', 'nativecountry_ Laos', 'nativecountry_ Mexico', 'na
tivecountry_ Nicaragua', 'nativecountry_ Outlying-US(Guam-USVI-etc)', 'nativeco
untry_ Peru', 'nativecountry_ Philippines', 'nativecountry_ Poland', 'nativecou
ntry_ Portugal', 'nativecountry_ Puerto-Rico', 'nativecountry_ Scotland', 'nati
vecountry_ South', 'nativecountry_ Taiwan', 'nativecountry_ Thailand', 'nativec
ountry_ Trinadad&Tobago', 'nativecountry_ United-States', 'nativecountry_ Vietn
am', 'nativecountry_ Yugoslavia']
```

In [31]:
```
features=list(set(columns_list)-set(['SalStat']))
print(features)
#features contain input variables
```

```
['nativecountry_ United-States', 'occupation_ Handlers-cleaners', 'nativecountr
y_ Germany', 'nativecountry_ Italy', 'race_ Other', 'nativecountry_ Canada', 'n
ativecountry_ Haiti', 'EdType_ Masters', 'JobType_ Self-emp-inc', 'JobType_ Sel
f-emp-not-inc', 'EdType_ Assoc-voc', 'capitalloss', 'EdType_ Assoc-acdm', 'occu
pation_ Farming-fishing', 'race_ Black', 'race_ Asian-Pac-Islander', 'maritalst
atus_ Married-civ-spouse', 'EdType_ 12th', 'nativecountry_ Poland', 'occupation
_ Protective-serv', 'EdType_ 9th', 'nativecountry_ Scotland', 'nativecountry_ T
hailand', 'JobType_ Local-gov', 'maritalstatus_ Never-married', 'JobType_ Witho
ut-pay', 'nativecountry_ Jamaica', 'nativecountry_ Yugoslavia', 'occupation_ Tr
ansport-moving', 'nativecountry_ Hong', 'nativecountry_ Cuba', 'occupation_ Cra
ft-repair', 'occupation_ Exec-managerial', 'EdType_ 7th-8th', 'EdType_ Prof-sch
ool', 'JobType_ State-gov', 'occupation_ Tech-support', 'nativecountry_ Peru',
'occupation_ Other-service', 'occupation_ Machine-op-inspct', 'occupation_ Prof
-specialty', 'EdType_ 1st-4th', 'occupation_ Sales', 'nativecountry_ Dominican-
Republic', 'occupation_ Priv-house-serv', 'nativecountry_ Honduras', 'relations
hip_ Unmarried', 'nativecountry_ Nicaragua', 'nativecountry_ Ireland', 'marital
status_ Married-spouse-absent', 'nativecountry_ El-Salvador', 'EdType_ 5th-6t
h', 'capitalgain', 'maritalstatus_ Separated', 'nativecountry_ China', 'marital
status_ Married-AF-spouse', 'nativecountry_ Taiwan', 'nativecountry_ Ecuador',
'relationship_ Own-child', 'EdType_ Bachelors', 'gender_ Male', 'age', 'nativec
ountry_ India', 'nativecountry_ Outlying-US(Guam-USVI-etc)', 'nativecountry_ Tr
inadad&Tobago', 'maritalstatus_ Widowed', 'EdType_ Doctorate', 'nativecountry_
South', 'nativecountry_ Philippines', 'nativecountry_ Guatemala', 'hoursperwee
k', 'nativecountry_ Vietnam', 'JobType_ Private', 'EdType_ Some-college', 'nati
vecountry_ France', 'nativecountry_ Greece', 'nativecountry_ Mexico', 'occupati
on_ Armed-Forces', 'relationship_ Wife', 'nativecountry_ England', 'relationshi
p_ Not-in-family', 'EdType_ Preschool', 'nativecountry_ Laos', 'EdType_ HS-gra
d', 'EdType_ 11th', 'nativecountry_ Puerto-Rico', 'nativecountry_ Japan', 'nati
vecountry_ Columbia', 'nativecountry_ Hungary', 'race_ White', 'nativecountry_
Iran', 'relationship_ Other-relative', 'nativecountry_ Portugal', 'nativecountr
y_ Holand-Netherlands']
```

In [32]:
```
#Storing the output variable in y
y=new_data['SalStat'].values#values helps to extract the data from the vactor
print(y)
```

```
[0 0 1 ... 0 0 0]
```

In [33]:
```
#Storing the input values in x
x=new_data[features].values
print(x)
```

```
[[1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]]
```

Split the data into train and test

In [34]:
```
#1.we are dividing the data into training set and test set so that we can role th
#of data into test set to test the model this can be done by train test split con
#2.here x represent the input value and y output represent the output value
#3.test_size represent as .3 that shows the proprtion of data include in test spl
#4.random_state=0 here random state is seed used by random number generatorso eac
#the same set of sample will be chosen if u do not set random_state different set
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [35]:
```
print(train_x) #iit has many values
```

```
[[1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 1 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]]
```

In [36]:
```
print(train_y)#it ccontain only one row with zero and one
```

```
[0 0 0 ... 1 0 0]
```

In [37]:
```
print(test_x)#less than train data
```

```
[[1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 1 0 ... 0 0 0]]
```

In [38]:
```
print(test_y)
```

```
[0 0 0 ... 0 0 0]
```

In [39]:
```
#make an instance of the model
logistic=LogisticRegression()
#an instance of logistic classifier
```

In [40]:
```
#we are fitting the model on train data
logistic.fit(train_x,train_y)
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:7
62: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[40]:
```
LogisticRegression()
```

In [41]:
```
logistic.coef_
#coffecient values of all the variable
```

Out[41]:
```
array([[-4.24873770e-01, -3.39117057e-01,  1.45417983e-02,
        -1.86615701e-03, -8.39829775e-02,  3.59381747e-03,
        -1.18286455e-02,  5.26508294e-01,  1.61497118e-01,
        -5.83108150e-01, -3.85614412e-02,  6.71555249e-04,
        -5.96526034e-02, -3.90196893e-01, -4.33748420e-01,
        -1.17957738e-01,  1.13242860e+00, -9.57928455e-02,
        -9.34947699e-03, -1.42220751e-02, -1.72381984e-01,
        -3.29595774e-03, -3.33876091e-03, -2.48114693e-01,
        -1.34689264e+00, -8.12900446e-03, -1.19764305e-02,
        -5.22273669e-04, -1.95325188e-01, -1.26427371e-03,
        -8.19582365e-03, -3.21744136e-01,  8.97254282e-01,
        -2.50093191e-01,  2.33893131e-01, -1.26816298e-01,
         1.17069011e-01, -1.20804367e-02, -7.40453909e-01,
        -3.50567801e-01,  6.83313658e-01, -7.16630519e-02,
         9.81491841e-02, -2.57041373e-02, -4.06257198e-02,
        -2.24526761e-03, -5.75439757e-01, -1.28851408e-02,
        -3.88142742e-03, -5.83618116e-02, -3.96646213e-02,
        -1.40308729e-01,  3.16594759e-04, -1.79334760e-01,
        -2.24438657e-02,  1.65057471e-02,  1.95102297e-03,
        -7.24170819e-03, -9.40988883e-01,  7.21947689e-01,
         1.64320135e-01,  4.35660765e-04, -2.02963429e-02,
        -7.41553856e-03, -4.48967778e-03, -9.74281847e-02,
         2.48410437e-01, -2.65522737e-02, -8.20969444e-03,
        -2.17549712e-02,  9.04726396e-03, -3.36938149e-02,
        -3.25305206e-01, -3.95479959e-01,  5.71030512e-03,
        -8.33336955e-03, -2.57511673e-01, -1.94000723e-03,
         2.46287799e-01,  1.26945392e-02, -4.99044492e-01,
        -2.63626261e-02, -6.14996245e-03, -9.34989415e-01,
        -3.04418448e-01, -4.13705297e-02, -4.78557027e-04,
        -2.01974777e-02, -3.27264624e-03, -3.25353404e-01,
        -2.70633834e-03, -2.44469534e-01, -1.37343796e-02,
        -8.91618327e-04]])
```

In [42]:
```
logistic.intercept_
```

Out[42]:
```
array([-1.04059467])
```

In [43]:
```
#prediction from test data
prediction=logistic.predict(test_x)
print(prediction)
#the value is of 0 and 1 which gives you salary status of test dataframe
```

```
[0 0 0 ... 0 0 0]
```

In [44]:
```
#1.confusion table used to evalvulate the performance of classification module th
#the coorect prediction and incorrect prediction
```

In [45]:
```
#confusion matrix
confusion_matrix=confusion_matrix(test_y,prediction)
#here test_y store the actual result and prediction store the predicted result
#we are storing the result in confusion matrix
print(confusion_matrix)
#the diagnol value gives you the total number of correctly classified sample off
#2.in column we have prediction and rows have actual classes if the actual classe
#6332 as less than or equal to 50000 431 as greater than 50000
#3.947 as less than 947 and 1279 as greater than
#4.there are missclassification
```

```
[[6291  532]
 [ 946 1280]]
```

In [46]:
```
#calculating accuracy will give you how accurate our model is
accuracy_score=accuracy_score(test_y,prediction)
print(accuracy_score)
#84% our data is correct
```

```
0.8366670350314952
```

In [47]:
```
#printing the misclassified value from predictio
print('Misclassified samples: %d' % (test_y!=prediction).sum())
```

```
Misclassified samples: 1478
```

Improving the singnificant result

In [48]:
```
#1.Logistic regression by removing insingnificant variable to improve the resultw
#insignificant variABLEwho do  not contribute in evaluting the salary status so w
```

In [ ]:
```

```

In [49]:
```
data_income=pd.read_csv('income.csv')#importing data into spyder

data=data_income.copy()
data2=data.dropna(axis=0)
```

In [50]:
```
data2['SalStat']=data2['SalStat'].map({' less than or equal to 50,000':0,' greate
print(data2['SalStat'])
```

```
0        0
1        0
2        1
3        0
4        0
        ..
31973    0
31974    0
31975    0
31976    0
31977    0
Name: SalStat, Length: 31978, dtype: int64
```

In [51]:
```
#These are the insignificant variable so we are storing in the another variable
cols=['gender','nativecountry','race','JobType']
new_data=data2.drop(cols,axis=1)
#here we drop all the insignificant variable hence we do not consider when we moc
```

In [52]:
```
new_data=pd.get_dummies(data2,drop_first=True)
```

In [53]:
```
#Storing column names
columns_list=list(new_data.columns)
print(columns_list)
```

```
['age', 'capitalgain', 'capitalloss', 'hoursperweek', 'SalStat', 'JobType_ Fede
ral-gov', 'JobType_ Local-gov', 'JobType_ Never-worked', 'JobType_ Private', 'J
obType_ Self-emp-inc', 'JobType_ Self-emp-not-inc', 'JobType_ State-gov', 'JobT
ype_ Without-pay', 'EdType_ 11th', 'EdType_ 12th', 'EdType_ 1st-4th', 'EdType_
5th-6th', 'EdType_ 7th-8th', 'EdType_ 9th', 'EdType_ Assoc-acdm', 'EdType_ Asso
c-voc', 'EdType_ Bachelors', 'EdType_ Doctorate', 'EdType_ HS-grad', 'EdType_ M
asters', 'EdType_ Preschool', 'EdType_ Prof-school', 'EdType_ Some-college', 'm
aritalstatus_ Married-AF-spouse', 'maritalstatus_ Married-civ-spouse', 'marital
status_ Married-spouse-absent', 'maritalstatus_ Never-married', 'maritalstatus_
Separated', 'maritalstatus_ Widowed', 'occupation_ Adm-clerical', 'occupation_
Armed-Forces', 'occupation_ Craft-repair', 'occupation_ Exec-managerial', 'occu
pation_ Farming-fishing', 'occupation_ Handlers-cleaners', 'occupation_ Machine
-op-inspct', 'occupation_ Other-service', 'occupation_ Priv-house-serv', 'occup
ation_ Prof-specialty', 'occupation_ Protective-serv', 'occupation_ Sales', 'oc
cupation_ Tech-support', 'occupation_ Transport-moving', 'relationship_ Not-in-
family', 'relationship_ Other-relative', 'relationship_ Own-child', 'relationsh
ip_ Unmarried', 'relationship_ Wife', 'race_ Asian-Pac-Islander', 'race_ Blac
k', 'race_ Other', 'race_ White', 'gender_ Male', 'nativecountry_ Canada', 'nat
ivecountry_ China', 'nativecountry_ Columbia', 'nativecountry_ Cuba', 'nativeco
untry_ Dominican-Republic', 'nativecountry_ Ecuador', 'nativecountry_ El-Salvad
or', 'nativecountry_ England', 'nativecountry_ France', 'nativecountry_ German
y', 'nativecountry_ Greece', 'nativecountry_ Guatemala', 'nativecountry_ Hait
i', 'nativecountry_ Holand-Netherlands', 'nativecountry_ Honduras', 'nativecoun
try_ Hong', 'nativecountry_ Hungary', 'nativecountry_ India', 'nativecountry_ I
ran', 'nativecountry_ Ireland', 'nativecountry_ Italy', 'nativecountry_ Jamaic
a', 'nativecountry_ Japan', 'nativecountry_ Laos', 'nativecountry_ Mexico', 'na
tivecountry_ Nicaragua', 'nativecountry_ Outlying-US(Guam-USVI-etc)', 'nativeco
untry_ Peru', 'nativecountry_ Philippines', 'nativecountry_ Poland', 'nativecou
ntry_ Portugal', 'nativecountry_ Puerto-Rico', 'nativecountry_ Scotland', 'nati
vecountry_ South', 'nativecountry_ Taiwan', 'nativecountry_ Thailand', 'nativec
ountry_ Trinadad&Tobago', 'nativecountry_ United-States', 'nativecountry_ Vietn
am', 'nativecountry_ Yugoslavia']
```

In [54]:
```
features=list(set(columns_list)-set(['SalStat']))
print(features)
#features contain input variables
```

```
['nativecountry_ United-States', 'occupation_ Handlers-cleaners', 'nativecountr
y_ Germany', 'nativecountry_ Italy', 'race_ Other', 'nativecountry_ Canada', 'n
ativecountry_ Haiti', 'EdType_ Masters', 'JobType_ Self-emp-inc', 'JobType_ Sel
f-emp-not-inc', 'EdType_ Assoc-voc', 'JobType_ Federal-gov', 'capitalloss', 'Ed
Type_ Assoc-acdm', 'occupation_ Farming-fishing', 'race_ Black', 'race_ Asian-P
ac-Islander', 'maritalstatus_ Married-civ-spouse', 'EdType_ 12th', 'nativecount
ry_ Poland', 'occupation_ Protective-serv', 'EdType_ 9th', 'nativecountry_ Scot
land', 'nativecountry_ Thailand', 'JobType_ Local-gov', 'maritalstatus_ Never-m
arried', 'JobType_ Without-pay', 'nativecountry_ Jamaica', 'nativecountry_ Yugo
slavia', 'occupation_ Transport-moving', 'nativecountry_ Hong', 'nativecountry_
Cuba', 'occupation_ Craft-repair', 'occupation_ Exec-managerial', 'occupation_
Adm-clerical', 'EdType_ 7th-8th', 'EdType_ Prof-school', 'JobType_ State-gov',
'occupation_ Tech-support', 'nativecountry_ Peru', 'occupation_ Other-service',
'occupation_ Machine-op-inspct', 'occupation_ Prof-specialty', 'EdType_ 1st-4t
h', 'occupation_ Sales', 'nativecountry_ Dominican-Republic', 'occupation_ Priv
-house-serv', 'nativecountry_ Honduras', 'relationship_ Unmarried', 'nativecoun
try_ Nicaragua', 'nativecountry_ Ireland', 'maritalstatus_ Married-spouse-absen
t', 'nativecountry_ El-Salvador', 'EdType_ 5th-6th', 'capitalgain', 'maritalsta
tus_ Separated', 'nativecountry_ China', 'maritalstatus_ Married-AF-spouse', 'n
ativecountry_ Taiwan', 'nativecountry_ Ecuador', 'relationship_ Own-child', 'Ed
Type_ Bachelors', 'gender_ Male', 'age', 'nativecountry_ India', 'nativecountry
_ Outlying-US(Guam-USVI-etc)', 'nativecountry_ Trinadad&Tobago', 'maritalstatus
_ Widowed', 'EdType_ Doctorate', 'nativecountry_ South', 'nativecountry_ Philip
pines', 'nativecountry_ Guatemala', 'hoursperweek', 'nativecountry_ Vietnam',
'JobType_ Private', 'EdType_ Some-college', 'nativecountry_ France', 'nativecou
ntry_ Greece', 'nativecountry_ Mexico', 'occupation_ Armed-Forces', 'relationsh
ip_ Wife', 'nativecountry_ England', 'relationship_ Not-in-family', 'JobType_ N
ever-worked', 'EdType_ Preschool', 'nativecountry_ Laos', 'EdType_ HS-grad', 'E
dType_ 11th', 'nativecountry_ Puerto-Rico', 'nativecountry_ Japan', 'nativecoun
try_ Columbia', 'nativecountry_ Hungary', 'race_ White', 'nativecountry_ Iran',
'relationship_ Other-relative', 'nativecountry_ Portugal', 'nativecountry_ Hola
nd-Netherlands']
```

In [55]:
```
#Storing the output variable in y
y=new_data['SalStat'].values#values helps to extract the data from the vactor
print(y)
```

```
[0 0 1 ... 0 0 0]
```

In [56]:
```
#Storing the input values in x
x=new_data[features].values
print(x)
```

```
[[1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]]
```

In [57]:
```
#1.we are dividing the data into training set and test set so that we can role th
#of data into test set to test the model this can be done by train test split con
#2.here x represent the input value and y output represent the output value
#3.test_size represent as .3 that shows the proprtion of data include in test spl
#4.random_state=0 here random state is seed used by random number generatorso eac
#the same set of sample will be chosen if u do not set random_state different set
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [58]:
```
print(train_x) #iit has many values
```

```
[[1 0 0 ... 1 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]]
```

In [59]:
```
print(train_y) #iit has many values
```

```
[0 0 0 ... 0 1 0]
```

In [60]:
```
print(test_x)
```

```
[[1 0 0 ... 0 0 0]
 [0 0 0 ... 1 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]]
```

In [61]:
```
print(test_y)
```

```
[0 0 1 ... 0 0 0]
```

In [62]:
```
logistic=LogisticRegression()
```

In [63]:
```
logistic.fit(train_x,train_y)
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:7
62: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[63]:
```
LogisticRegression()
```

In [64]:
```
prediction=logistic.predict(test_x)
print(prediction)
```

```
[0 0 0 ... 0 0 0]
```

In [65]:
```
import numpy as  np
accuracy_score=accuracy_score(test_y,prediction)
print(accuracy_score)
#here the accuracy decrease accuracy
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-65-19ad7bc8e24c> in <module>
      1 import numpy as  np
----> 2 accuracy_score=accuracy_score(test_y,prediction)
      3 print(accuracy_score)
      4 #here the accuracy decrease accuracy

TypeError: 'numpy.float64' object is not callable
```

knn model

to class records in any of the category of the salary status

In [66]:
```
#importing the library of knn
from sklearn.neighbors import KNeighborsClassifier
```

In [67]:
```
#importing matplotlib as plt
import matplotlib.pyplot as plt
```

In [81]:
```
knn=KNeighborsClassifier(n_neighbors=5)#storing the k nearest neighbours classifi
#1.k=5 so it consider 5 neighbours when classifying the data less than or equal t
#majority from 5 classes
```

In [73]:
```
KNN_classifier.fit(train_x,train_y)
#input value and output value should be there from trained value
```

Out[73]:
```
KNeighborsClassifier()
```

In [74]:
```
#prediction have the value salary status
prediction=KNN_classifier.predict(test_x)
print(prediction)
#prediction contain the value greater 0(less than 50000) and 1(more than 50000)
```

```
[0 0 0 ... 0 0 0]
```

In [75]:
```
confusion_matrix=confusion_matrix(test_y,prediction)
#6338 less than value 50000
#1285 greater than 50000
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-75-2ced6ed1321b> in <module>
----> 1 confusion_matrix=confusion_matrix(test_y,prediction)

TypeError: 'numpy.ndarray' object is not callable
```

In [78]:
```
#to know the accuracy percentage
import numpy as  np
accuracy_score=accuracy_score(test_y,prediction)
print(accuracy_score)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-78-70e4703c3e45> in <module>
      1 #to know the accuracy percentage
      2 import numpy as  np
----> 3 accuracy_score=accuracy_score(test_y,prediction)
      4 print(accuracy_score)

TypeError: 'numpy.float64' object is not callable
```

In [77]:
```
#printing the misclassified value from predictio
print('Misclassified samples: %d' % (test_y!=prediction).sum())
```

```
Misclassified samples: 1534
```

In [ ]:
```
Misclassified_sample=[]
for i in range(1, 20):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(train_x,train_y)
    pred_i=knn.predict(test_x)
    Misclassified_sample.append((test_y!=prediction).sum())
print(Misclassified_sample)
```

In [ ]: