project

July 4, 2024

1 ADULT INCOME PREDICTION

This dataset has been taken from census bureau database. The goal of this project is to accurately predict whether or not an adult makes more than 50000 US Dollars in an year on the basis of the feautures given

ABOUT DATASET

- Age: Describes the age of individuals. Continuous.
- Workclass: a general term to represent the employment status of an individual
- fnlwgt: estimates of the civilian noninstitutional population of the US
- education: the highest level of education achieved by an individual
- educationnum: the highest level of education achieved in numerical form.
- maritalstatus: marital status of an individual.
- occupation: the general type of occupation of an individual
- relationship: represents what this individual is relative to others.
- race: Descriptions of an individual's race
- sex: the sex of the individual
- capitalgain: capital gains for an individual
- capitalloss: capital loss for an individual
- hoursperweek: the hours an individual has reported to work per week
- nativecountry: country of origin for an individual

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/adult[1].csv')
df
```

[1]:
```
       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]
```

DATA PREPROCESSING

[2]:
```
df.head()
```

[2]:
```
   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
```

[3]:
```
df.tail()
```

[3]:
```
       age  workclass  fnlwgt     education  education.num      marital.status  \
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  \
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
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
```

[4]:
```
df.shape
```

[4]: (32561, 15)

[5]:
```
df.columns
```

```
[5]: Index(['age', 'workclass', 'fnlwgt', 'education', 'education.num',
       'marital.status', 'occupation', 'relationship', 'race', 'sex',
       'capital.gain', 'capital.loss', 'hours.per.week', 'native.country',
       'income'],
      dtype='object')
```

```
[6]: 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
```

```
[7]: [(col, df[col].nunique(), df[col].dtype) for col in df.columns]
```

```
[7]: [('age', 73, dtype('int64')),
 ('workclass', 9, dtype('O')),
 ('fnlwgt', 21648, dtype('int64')),
 ('education', 16, dtype('O')),
 ('education.num', 16, dtype('int64')),
 ('marital.status', 7, dtype('O')),
 ('occupation', 15, dtype('O')),
 ('relationship', 6, dtype('O')),
 ('race', 5, dtype('O')),
 ('sex', 2, dtype('O')),
 ('capital.gain', 119, dtype('int64')),
 ('capital.loss', 92, dtype('int64')),
 ('hours.per.week', 94, dtype('int64')),
 ('native.country', 42, dtype('O')),
```

```
('income', 2, dtype('O'))]
```

Insights

More than half the values of fnlwgt are unique. This probably doesn't give a sigificant information about the data.

[8]:
```python
df.describe()
```

[8]:
```
                age        fnlwgt  education.num  capital.gain  capital.loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000  32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844     87.303830
std       13.640433  1.055500e+05       2.572720   7385.292085    402.960219
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.178270e+05       9.000000      0.000000      0.000000
50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       hours.per.week
count    32561.000000
mean        40.437456
std         12.347429
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```

Insights

Count tell us if there's a missing data or not.All the numerical columns seems to have no missing data here.

[9]:
```python
df.isna().sum()
```

[9]:
```
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
```

[10]:
```
df[df == '?'] = np.nan
df.isna().sum()
```

[10]:
```
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
```

Insights

workclass,occupation,native country are found to have missing values('?'). The percetage of the entries with (?) is very low as compared to the length of the data of that particular columns. Therefore, it seems that dropping the missing values should be good choice

[11]:
```
for i in ['workclass','occupation','native.country']:
  df[i].fillna(df[i].mode()[0],inplace=True)
```

[12]:
```
df.isna().sum()
```

[12]:
```
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
```

[13]:
```
df[df.duplicated()]
```

[13]:
```
       age         workclass  fnlwgt     education  education.num  \
8453    25           Private  308144     Bachelors             13
8645    90           Private   52386  Some-college             10
12202   21           Private  250051  Some-college             10
14346   20           Private  107658  Some-college             10
15603   25           Private  195994       1st-4th              2
17344   21           Private  243368     Preschool              1
19067   46           Private  173243       HS-grad              9
20388   30           Private  144593       HS-grad              9
20507   19           Private   97261       HS-grad              9
22783   19           Private  138153  Some-college             10
22934   19           Private  146679  Some-college             10
23276   49           Private   31267       7th-8th              4
23660   25           Private  195994       1st-4th              2
23720   44           Private  367749     Bachelors             13
23827   49  Self-emp-not-inc   43479  Some-college             10
26738   23           Private  240137       5th-6th              3
27133   28           Private  274679       Masters             14
28796   27           Private  255582       HS-grad              9
29051   42           Private  204235  Some-college             10
29334   39           Private   30916       HS-grad              9
29604   38           Private  207202       HS-grad              9
31060   46           Private  133616  Some-college             10
32065   19           Private  251579  Some-college             10
32419   35           Private  379959       HS-grad              9

           marital.status         occupation   relationship  \
8453        Never-married       Craft-repair  Not-in-family
8645        Never-married      Other-service  Not-in-family
12202       Never-married     Prof-specialty      Own-child
14346       Never-married       Tech-support  Not-in-family
15603       Never-married    Priv-house-serv  Not-in-family
17344       Never-married    Farming-fishing  Not-in-family
19067  Married-civ-spouse       Craft-repair        Husband
20388       Never-married      Other-service  Not-in-family
20507       Never-married    Farming-fishing  Not-in-family
22783       Never-married       Adm-clerical      Own-child
22934       Never-married    Exec-managerial      Own-child
23276  Married-civ-spouse       Craft-repair        Husband
23660       Never-married    Priv-house-serv  Not-in-family
23720       Never-married     Prof-specialty  Not-in-family
```

```
23827  Married-civ-spouse       Craft-repair        Husband
26738       Never-married  Handlers-cleaners  Not-in-family
27133       Never-married     Prof-specialty  Not-in-family
28796       Never-married  Machine-op-inspct  Not-in-family
29051  Married-civ-spouse     Prof-specialty        Husband
29334  Married-civ-spouse       Craft-repair        Husband
29604  Married-civ-spouse  Machine-op-inspct        Husband
31060            Divorced       Adm-clerical      Unmarried
32065       Never-married      Other-service      Own-child
32419            Divorced      Other-service  Not-in-family

                     race     sex  capital.gain  capital.loss  hours.per.week  \
8453                White    Male             0             0              40
8645   Asian-Pac-Islander    Male             0             0              35
12202               White  Female             0             0              10
14346               White  Female             0             0              10
15603               White  Female             0             0              40
17344               White    Male             0             0              50
19067               White    Male             0             0              40
20388               Black    Male             0             0              40
20507               White    Male             0             0              40
22783               White  Female             0             0              10
22934               Black    Male             0             0              30
23276               White    Male             0             0              40
23660               White  Female             0             0              40
23720               White  Female             0             0              45
23827               White    Male             0             0              40
26738               White    Male             0             0              55
27133               White    Male             0             0              50
28796               White  Female             0             0              40
29051               White    Male             0             0              40
29334               White    Male             0             0              40
29604               White    Male             0             0              48
31060               White  Female             0             0              40
32065               White    Male             0             0              14
32419               White  Female             0             0              40

      native.country income
8453          Mexico  <=50K
8645   United-States  <=50K
12202  United-States  <=50K
14346  United-States  <=50K
15603      Guatemala  <=50K
17344         Mexico  <=50K
19067  United-States  <=50K
20388  United-States  <=50K
20507  United-States  <=50K
```

```
22783  United-States  <=50K
22934  United-States  <=50K
23276  United-States  <=50K
23660      Guatemala  <=50K
23720         Mexico  <=50K
23827  United-States  <=50K
26738         Mexico  <=50K
27133  United-States  <=50K
28796  United-States  <=50K
29051  United-States   >50K
29334  United-States  <=50K
29604  United-States   >50K
31060  United-States  <=50K
32065  United-States  <=50K
32419  United-States  <=50K
```

[14]:
```
df = df.drop_duplicates()
```

[15]:
```
df[df.duplicated()]
```

[15]:
```
Empty DataFrame
Columns: [age, workclass, fnlwgt, education, education.num, marital.status,
occupation, relationship, race, sex, capital.gain, capital.loss, hours.per.week,
native.country, income]
Index: []
```

[16]:
```
df.dtypes
```

[16]:
```
age                int64
workclass         object
fnlwgt             int64
education         object
education.num      int64
marital.status    object
occupation        object
relationship      object
race              object
sex               object
capital.gain       int64
capital.loss       int64
hours.per.week     int64
native.country    object
income            object
dtype: object
```

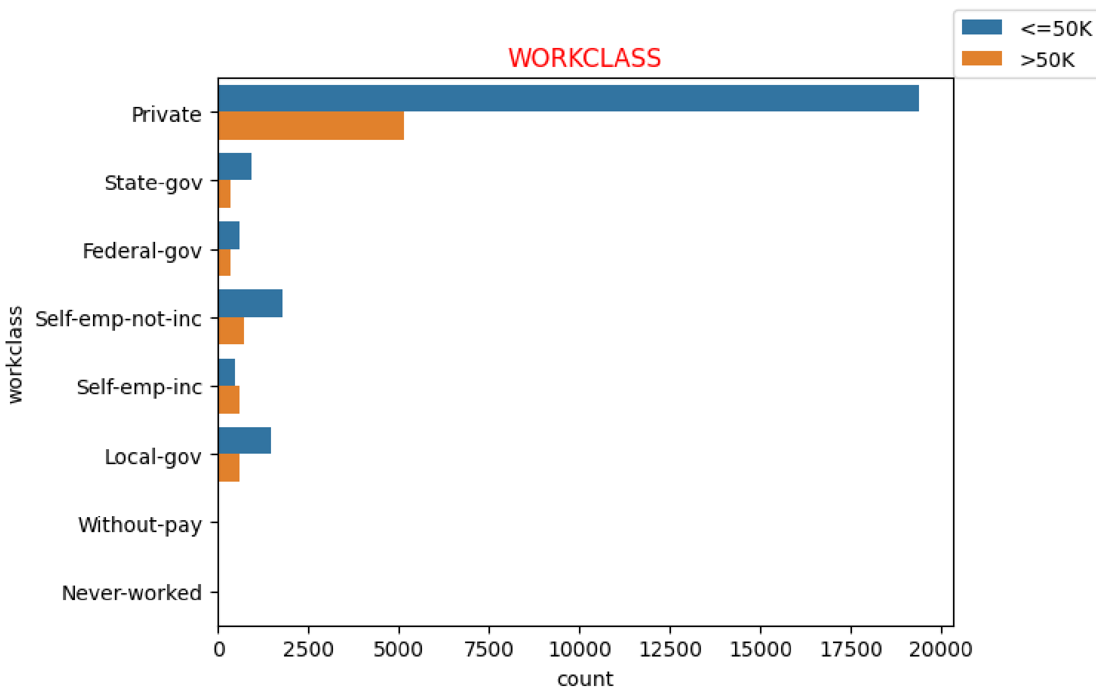
EXPLORATORY DATA ANALYSIS

[17]:
```
df['workclass'].value_counts()
```

[17]:
```
workclass
Private             24509
Self-emp-not-inc     2540
Local-gov            2093
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: count, dtype: int64
```

[18]:
```
sns.countplot(y='workclass',data=df,hue='income')
plt.title('WORKCLASS',color='r')
plt.legend(loc=(1,1))
```

[18]:
```
<matplotlib.legend.Legend at 0x7a5f6e808640>
```



[19]:
```
df['education'].value_counts()
```

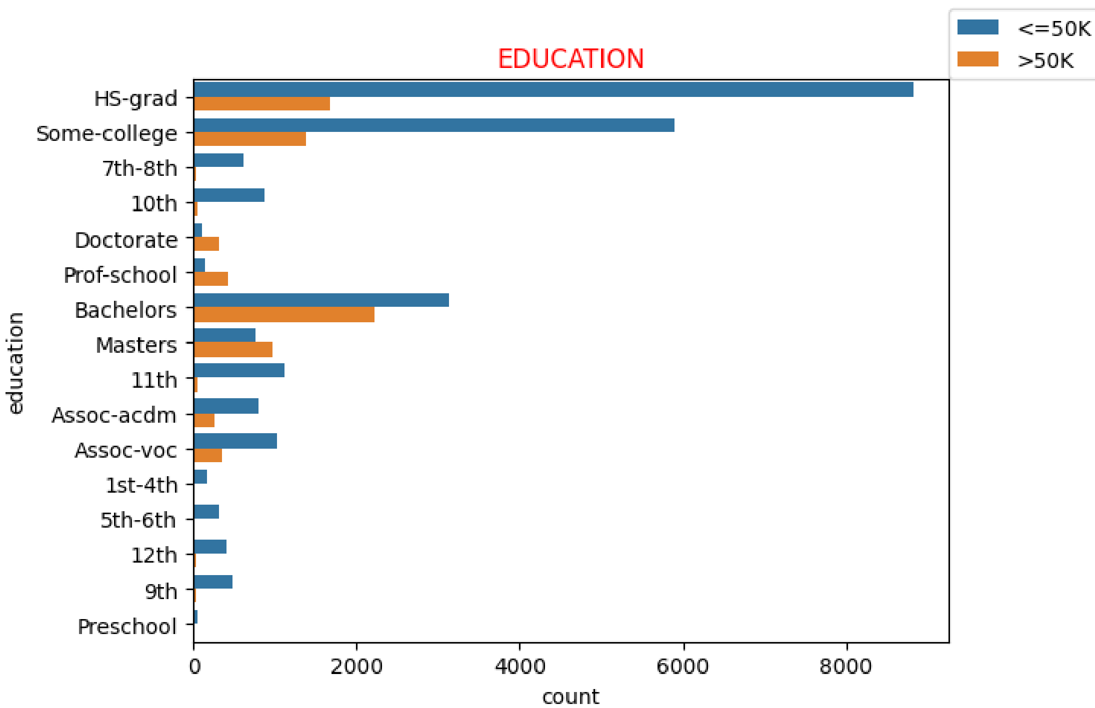
[19]:
```
education
HS-grad         10494
Some-college     7282
```

```
Bachelors        5353
Masters          1722
Assoc-voc        1382
11th             1175
Assoc-acdm       1067
10th              933
7th-8th           645
Prof-school       576
9th               514
12th              433
Doctorate         413
5th-6th           332
1st-4th           166
Preschool          50
Name: count, dtype: int64
```

[20]:

```python
sns.countplot(y='education',data=df,hue='income')
plt.title('EDUCATION',color='r')
plt.legend(loc=(1,1))
```

[20]: <matplotlib.legend.Legend at 0x7a5f6e3ac580>



[21]:

```python
df['marital.status'].value_counts()
```

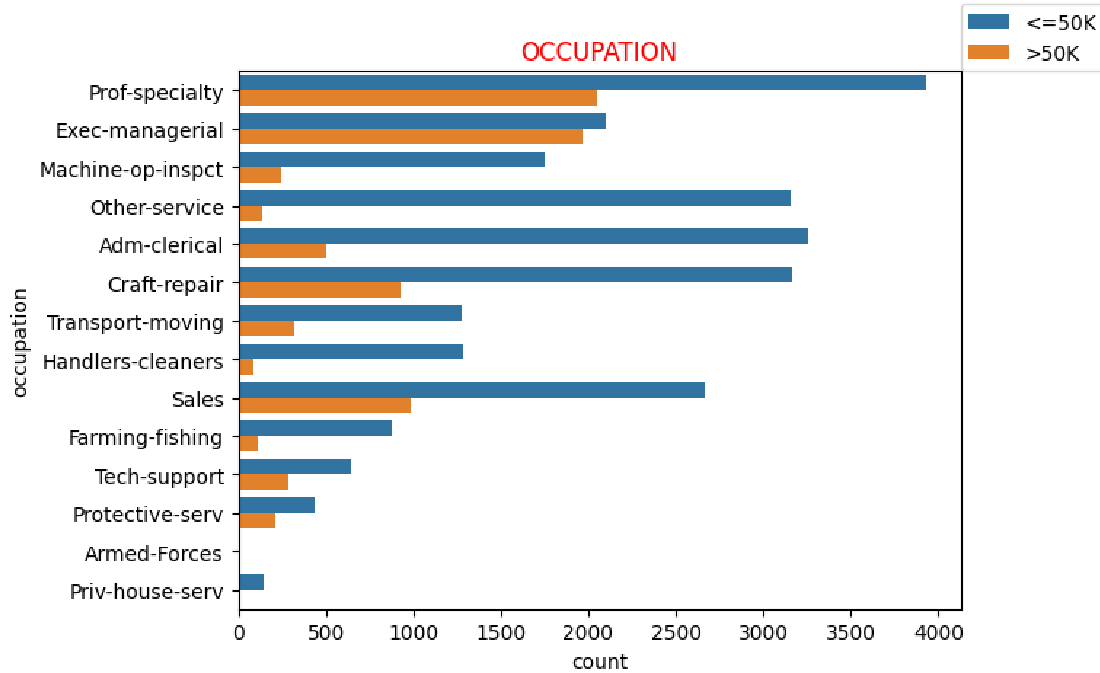
[21]:
```
marital.status
Married-civ-spouse       14970
Never-married            10667
Divorced                  4441
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: count, dtype: int64
```

[22]:
```
df['occupation'].value_counts()
```

[22]:
```
occupation
Prof-specialty       5979
Craft-repair         4094
Exec-managerial      4065
Adm-clerical         3768
Sales                3650
Other-service        3291
Machine-op-inspct    2000
Transport-moving     1597
Handlers-cleaners    1369
Farming-fishing       992
Tech-support          927
Protective-serv       649
Priv-house-serv       147
Armed-Forces            9
Name: count, dtype: int64
```

[23]:
```
sns.countplot(y='occupation',data=df,hue='income')
plt.title('OCCUPATION',color='r')
plt.legend(loc=(1,1))
```

[23]:
```
<matplotlib.legend.Legend at 0x7a5fac9a2920>
```



[24]:
```
df['relationship'].value_counts()
```

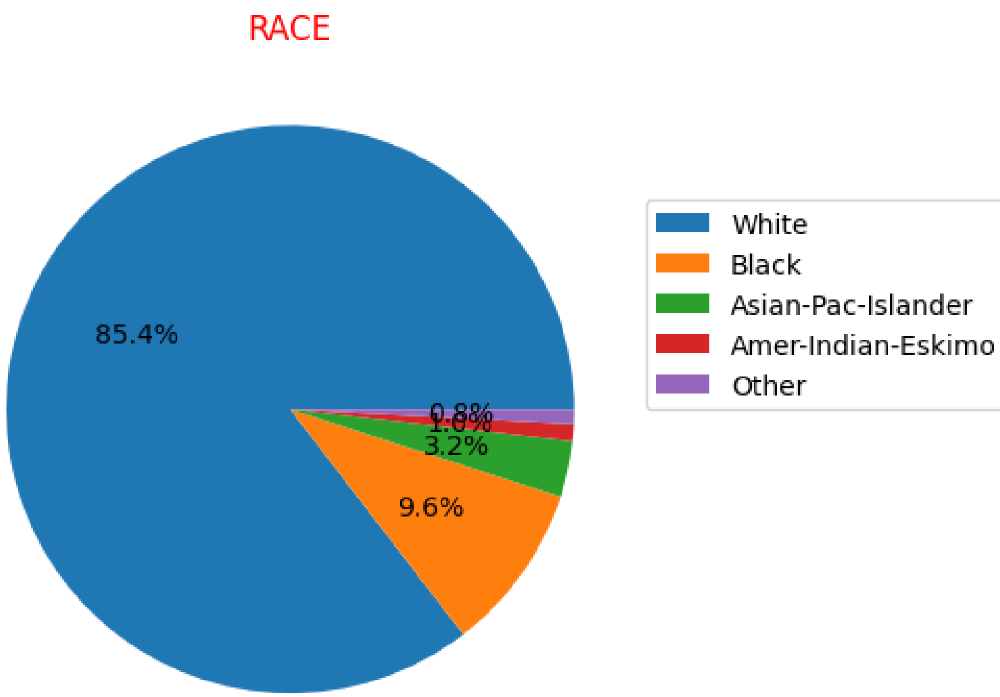
[24]:
```
relationship
Husband           13187
Not-in-family      8292
Own-child          5064
Unmarried          3445
Wife               1568
Other-relative      981
Name: count, dtype: int64
```

[25]:
```
df['race'].value_counts()
```

[25]:
```
race
White                 27795
Black                  3122
Asian-Pac-Islander     1038
Amer-Indian-Eskimo      311
Other                   271
Name: count, dtype: int64
```

[26]:
```
plt.pie(df['race'].value_counts(),autopct='%1.1f%%')
plt.legend(df['race'].value_counts().index,loc=[1,0.5])
plt.title('RACE',color='red')
```

[26]: Text(0.5, 1.0, 'RACE')



[27]:
```
df['sex'].value_counts()
```

[27]:
```
sex
Male      21775
Female    10762
Name: count, dtype: int64
```

[28]:
```
df['native.country'].value_counts()
```

[28]:
```
native.country
United-States                 29735
Mexico                          639
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
```

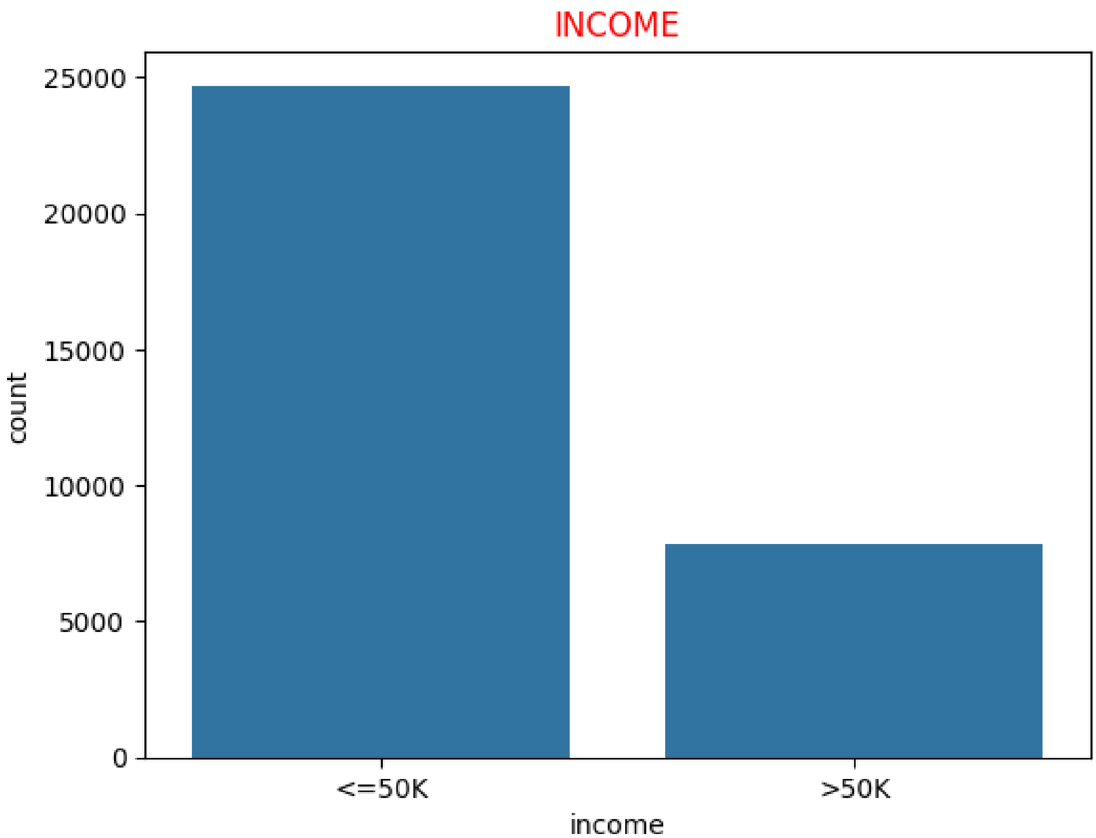
```
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        62
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
Greece                           29
France                           29
Ecuador                          28
Ireland                          24
Hong                             20
Trinadad&Tobago                  19
Cambodia                         19
Thailand                         18
Laos                             18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Hungary                          13
Honduras                         13
Scotland                         12
Holand-Netherlands                1
Name: count, dtype: int64
```

[29]:
```
df['income'].value_counts()
```

[29]:
```
income
<=50K    24698
>50K      7839
Name: count, dtype: int64
```

[30]:
```
sns.countplot(x='income',data=df)
plt.title('INCOME',color='r')
```

[30]:
```
Text(0.5, 1.0, 'INCOME')
```



FEATURE ENGINEERING

[31]:
```
#education
df['education'].
  ↪replace(['Preschool','1st-4th','5th-6th','9th','7th-8th','10th','HS-grad'],'school',inplace
df['education'].
  ↪replace(['11th','12th','Assoc-acdm','Assoc-voc','Prof-school'],'higher_school',inplace=True
df['education'].replace(['Bachelors','Some-college'],'ug',inplace=True)
```

```
<ipython-input-31-39bedb34f37e>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['education'].replace(['Preschool','1st-4th','5th-6th','9th','7th-8th','10th
','HS-grad'],'school',inplace=True)
<ipython-input-31-39bedb34f37e>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  df['education'].replace(['11th','12th','Assoc-acdm','Assoc-voc','Prof-
school'],'higher_school',inplace=True)
<ipython-input-31-39bedb34f37e>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['education'].replace(['Bachelors','Some-college'],'ug',inplace=True)
```

[32]:
```
df['education'].value_counts()
```

[32]:
```
education
school           13134
ug               12635
higher_school     4633
Masters           1722
Doctorate          413
Name: count, dtype: int64
```

[33]:
```
#marital status
df['marital.status'].
 ↪replace(['Married-civ-spouse','Married-spouse-absent','Married-AF-spouse'],'married',inplac
df['marital.status'].
 ↪replace(['Divorced','Separated','Widowed'],'other',inplace=True)
```

```
<ipython-input-33-db7c554eae8b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['marital.status'].replace(['Married-civ-spouse','Married-spouse-
absent','Married-AF-spouse'],'married',inplace=True)
<ipython-input-33-db7c554eae8b>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['marital.status'].replace(['Divorced','Separated','Widowed'],'other',inplac
e=True)
```

[34]:
```
df['marital.status'].value_counts()
```

[34]:
```
marital.status
married          15411
Never-married    10667
other             6459
Name: count, dtype: int64
```

[35]:
```
#native country
df['native.country'] = df['native.country'].apply(lambda x: 'Other' if x !=␣
↪'United-States' else x)
```

```
<ipython-input-35-29f29daae522>:2: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['native.country'] = df['native.country'].apply(lambda x: 'Other' if x !=
'United-States' else x)
```

[36]:
```
df['native.country'].value_counts()
```

[36]:
```
native.country
United-States    29735
Other             2802
Name: count, dtype: int64
```

[37]:
```
#income
df['income'] = df['income'].apply(lambda x:x.replace("<=50K", "0"))
df['income'] = df['income'].apply(lambda x:x.replace(">50K", "1"))
df['income'] = df['income'].astype(int)
```

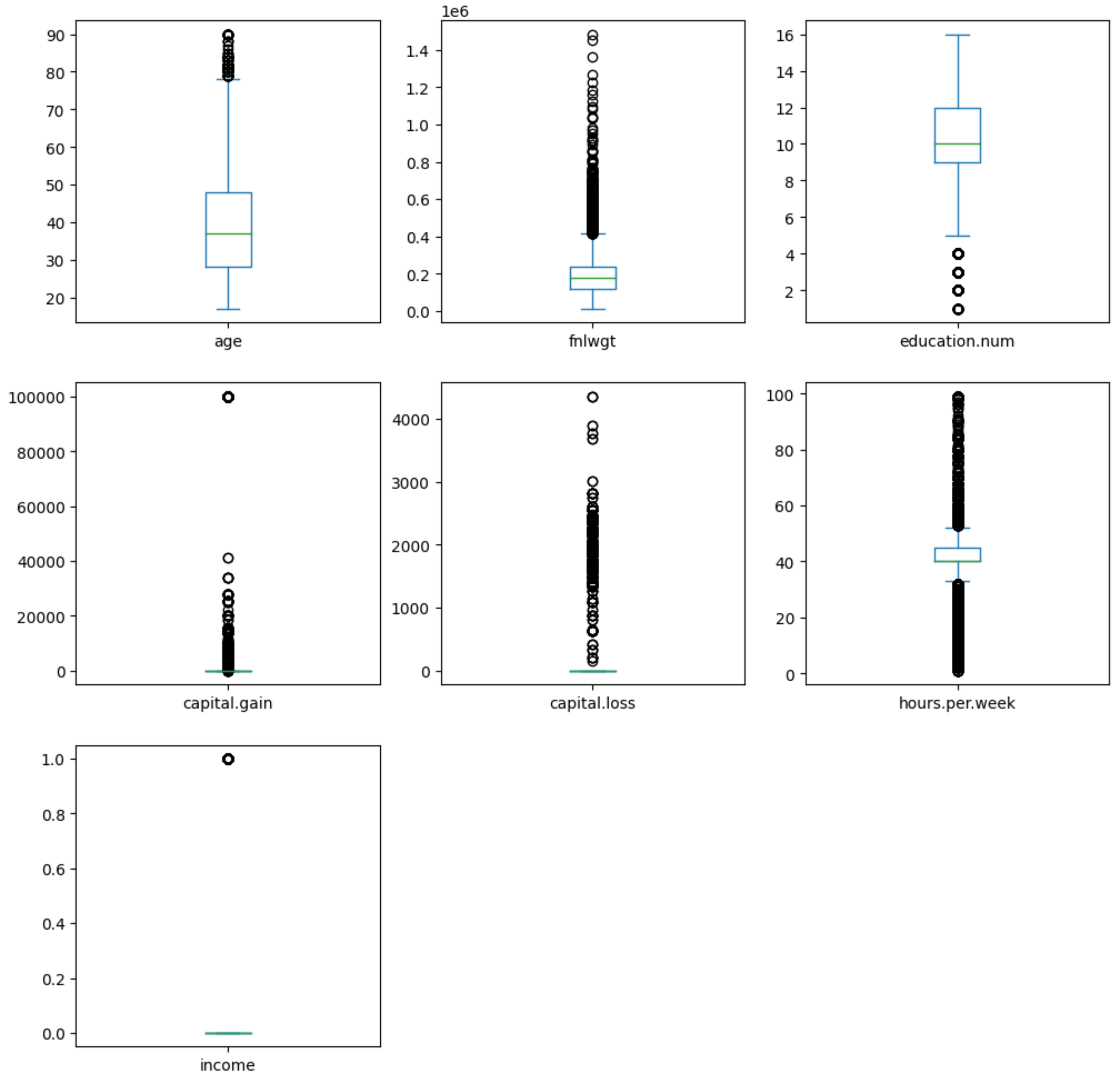
```
<ipython-input-37-ad81c6605bdb>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['income'] = df['income'].apply(lambda x:x.replace("<=50K", "0"))
<ipython-input-37-ad81c6605bdb>:3: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['income'] = df['income'].apply(lambda x:x.replace(">50K", "1"))
<ipython-input-37-ad81c6605bdb>: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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['income'] = df['income'].astype(int)
```

[38]:
```
df['income'].value_counts()
```

[38]:
```
income
0    24698
1     7839
Name: count, dtype: int64
```

REMOVING OUTLIERS

[39]:
```
df.plot(kind='box',figsize=(12,12),layout=(3,3),sharex=False,subplots
=True);
```



[40]:
```
def iqr_method(df,variables):
  q1=df[variables].quantile(0.25)
```

```
    q3=df[variables].quantile(0.75)
    iqr=q3-q1
    upper=q3+(1.5*iqr)
    lower=q1-(1.5*iqr)
    return lower,upper
```

[41]:
```
lower_lim,upper_lim=iqr_method(df,'age')
print('lower limit = ',lower_lim)
print('upper limit = ',upper_lim )
```

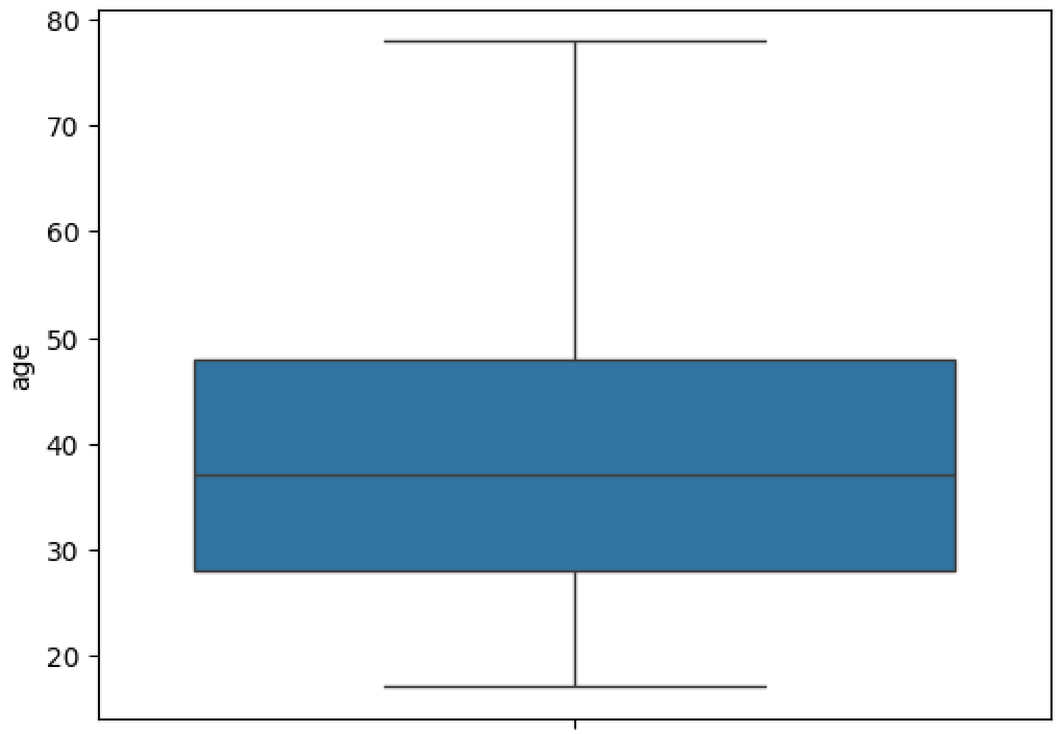
```
lower limit =  -2.0
upper limit =  78.0
```

[42]:
```
df['age']=np.where(df['age']>upper_lim,upper_lim,
                   np.where(df['age']<lower_lim,lower_lim,df['age']))
```

```
<ipython-input-42-d36c3dc4bade>:1: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['age']=np.where(df['age']>upper_lim,upper_lim,
```

[43]:
```
sns.boxplot(y=df['age'])
```

[43]: `<Axes: ylabel='age'>`



[44]:
```
df
```

[44]:
```
         age workclass  fnlwgt      education  education.num marital.status  \
0       78.0   Private   77053         school              9          other
1       78.0   Private  132870         school              9          other
2       66.0   Private  186061             ug             10          other
3       54.0   Private  140359         school              4          other
4       41.0   Private  264663             ug             10          other
…        …        …      …              …            …            …
32556   22.0   Private  310152             ug             10  Never-married
32557   27.0   Private  257302  higher_school             12        married
32558   40.0   Private  154374         school              9        married
32559   58.0   Private  151910         school              9          other
32560   22.0   Private  201490         school              9  Never-married

              occupation   relationship   race     sex  capital.gain  \
0         Prof-specialty  Not-in-family  White  Female             0
1        Exec-managerial  Not-in-family  White  Female             0
2         Prof-specialty      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       0
1              4356              18  United-States       0
2              4356              40  United-States       0
3              3900              40  United-States       0
4              3900              40  United-States       0
…               …               …             …      …
32556             0              40  United-States       0
32557             0              38  United-States       0
32558             0              40  United-States       1
32559             0              40  United-States       0
32560             0              20  United-States       0

[32537 rows x 15 columns]
```

[45]:
```
df.reset_index(drop=True,inplace=True)
df
```

[45]:
```
        age workclass  fnlwgt      education  education.num marital.status  \
0      78.0   Private   77053         school              9          other
1      78.0   Private  132870         school              9          other
2      66.0   Private  186061             ug             10          other
3      54.0   Private  140359         school              4          other
4      41.0   Private  264663             ug             10          other
…      …        …      …             …             …             …
32532  22.0   Private  310152             ug             10  Never-married
32533  27.0   Private  257302  higher_school             12        married
32534  40.0   Private  154374         school              9        married
32535  58.0   Private  151910         school              9          other
32536  22.0   Private  201490         school              9  Never-married

              occupation   relationship   race     sex  capital.gain  \
0         Prof-specialty  Not-in-family  White  Female             0
1        Exec-managerial  Not-in-family  White  Female             0
2         Prof-specialty      Unmarried  Black  Female             0
3      Machine-op-inspct      Unmarried  White  Female             0
4         Prof-specialty      Own-child  White  Female             0
…                    …            …    …     …           …
32532    Protective-serv  Not-in-family  White    Male             0
32533       Tech-support           Wife  White  Female             0
32534  Machine-op-inspct        Husband  White    Male             0
32535       Adm-clerical      Unmarried  White  Female             0
```

```
32536       Adm-clerical      Own-child  White     Male              0

       capital.loss  hours.per.week native.country  income
0              4356              40  United-States       0
1              4356              18  United-States       0
2              4356              40  United-States       0
3              3900              40  United-States       0
4              3900              40  United-States       0
…                 …               …             …     …
32532             0              40  United-States       0
32533             0              38  United-States       0
32534             0              40  United-States       1
32535             0              40  United-States       0
32536             0              20  United-States       0

[32537 rows x 15 columns]
```

[46]:
```
df.dtypes
```

[46]:
```
age               float64
workclass          object
fnlwgt              int64
education          object
education.num       int64
marital.status     object
occupation         object
relationship       object
race               object
sex                object
capital.gain        int64
capital.loss        int64
hours.per.week      int64
native.country     object
income              int64
dtype: object
```

ENCODING

[47]:
```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
new=['workclass','education','marital.
 ↪status','occupation','relationship','race','sex','native.country']
for i in new:
  df[i]=lb.fit_transform(df[i])

df
```

```
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
```

```
<ipython-input-47-b480d3e0c516>:5: 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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[i]=lb.fit_transform(df[i])
```

[47]:

```
        age  workclass  fnlwgt  education  education.num  marital.status  \
0      78.0          3   77053          3              9               2
1      78.0          3  132870          3              9               2
2      66.0          3  186061          4             10               2
3      54.0          3  140359          3              4               2
4      41.0          3  264663          4             10               2
…       …          …       …          …              …               …
32532  22.0          3  310152          4             10               0
32533  27.0          3  257302          2             12               1
32534  40.0          3  154374          3              9               1
32535  58.0          3  151910          3              9               2
32536  22.0          3  201490          3              9               0

       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
…             …             …     …    …             …             …
32532          10             1     4    1             0             0
32533          12             5     4    0             0             0
32534           6             0     4    1             0             0
32535           0             4     4    0             0             0
32536           0             3     4    1             0             0

       hours.per.week  native.country  income
0                  40               1       0
1                  18               1       0
2                  40               1       0
3                  40               1       0
4                  40               1       0
…                 …               …     …
32532              40               1       0
32533              38               1       0
32534              40               1       1
32535              40               1       0
32536              20               1       0
```

```
[32537 rows x 15 columns]
```

[48]:
```
df.dtypes
```

[48]:
```
age                float64
workclass            int64
fnlwgt               int64
education            int64
education.num        int64
marital.status       int64
occupation           int64
relationship         int64
race                 int64
sex                  int64
capital.gain         int64
capital.loss         int64
hours.per.week       int64
native.country       int64
income               int64
dtype: object
```

[49]:
```
df.drop(['fnlwgt','education.num'],axis=1,inplace=True)
```

```
<ipython-input-49-5e8085fc6d71>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(['fnlwgt','education.num'],axis=1,inplace=True)
```

SEPARATE X AND Y

[50]:
```
x=df.drop(['income'],axis=1)
x
```

[50]:

	age	workclass	education	marital.status	occupation	relationship	\
0	78.0	3	3	2	9	1	
1	78.0	3	3	2	3	1	
2	66.0	3	4	2	9	4	
3	54.0	3	3	2	6	4	
4	41.0	3	4	2	9	3	
…	…	…	…	…	…	…	
32532	22.0	3	4	0	10	1	
32533	27.0	3	2	1	12	5	
32534	40.0	3	3	1	6	0	
32535	58.0	3	3	2	0	4	
32536	22.0	3	3	0	0	3	

```
       race  sex  capital.gain  capital.loss  hours.per.week  native.country
0         4    0             0          4356              40               1
1         4    0             0          4356              18               1
2         2    0             0          4356              40               1
3         4    0             0          3900              40               1
4         4    0             0          3900              40               1
…       …  …             …             …               …              …
32532     4    1             0             0              40               1
32533     4    0             0             0              38               1
32534     4    1             0             0              40               1
32535     4    0             0             0              40               1
32536     4    1             0             0              20               1

[32537 rows x 12 columns]
```

[51]:

```python
y=df['income']
y
```

[51]:

```
0        0
1        0
2        0
3        0
4        0
        ..
32532    0
32533    0
32534    1
32535    0
32536    0
Name: income, Length: 32537, dtype: int64
```

[52]:

```python
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)
x_resampled, y_resampled = smote.fit_resample(x,y)
print(x_resampled.shape)
print(y_resampled.shape)
```

```
(49396, 12)
(49396,)
```

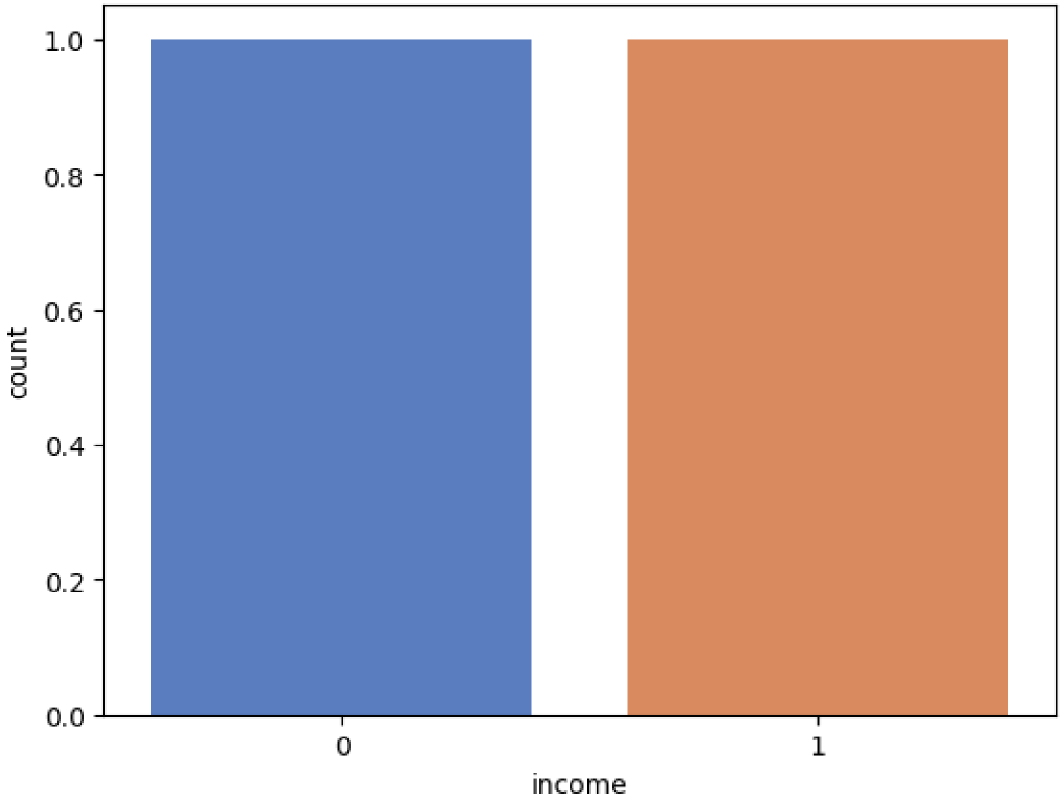
[53]:

```python
sns.countplot(y_resampled.value_counts(), palette='muted')
```

```
<ipython-input-53-1875fc4e2d21>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```

```
sns.countplot(y_resampled.value_counts(), palette='muted')
```

[53]: <Axes: xlabel='income', ylabel='count'>



[54]:
```
x=x_resampled.values
y=y_resampled.values
```

TRAIN-TEST SPLIT

[55]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪30,random_state=42)
x_train
```

[55]:
```
array([[3.40000000e+01, 3.00000000e+00, 4.00000000e+00, …,
        0.00000000e+00, 5.00000000e+01, 1.00000000e+00],
       [1.80000000e+01, 3.00000000e+00, 4.00000000e+00, …,
        0.00000000e+00, 2.00000000e+01, 1.00000000e+00],
       [3.95022801e+01, 3.00000000e+00, 2.00000000e+00, …,
        0.00000000e+00, 4.00000000e+01, 1.00000000e+00],
       …,
       [4.90000000e+01, 4.00000000e+00, 4.00000000e+00, …,
```

```
        0.00000000e+00, 5.00000000e+01, 1.00000000e+00],
       [5.50000000e+01, 0.00000000e+00, 3.00000000e+00, …,
        1.88700000e+03, 4.00000000e+01, 1.00000000e+00],
       [2.30000000e+01, 3.00000000e+00, 4.00000000e+00, …,
        0.00000000e+00, 2.50000000e+01, 1.00000000e+00]])
```

[56]:
```python
from sklearn.feature_selection import SelectKBest, chi2
X = x_resampled
Y = y_resampled
selector = SelectKBest(chi2, k=10)
selector.fit(X, Y)
X_new = selector.transform(X)
print(X.columns[selector.get_support(indices=True)])
```

```
Index(['age', 'education', 'marital.status', 'occupation', 'relationship',
       'race', 'sex', 'capital.gain', 'capital.loss', 'hours.per.week'],
      dtype='object')
```

[57]:
```python
df.describe()
```

[57]:
```
                age     workclass     education  marital.status    occupation  \
count  32537.000000  32537.000000  32537.000000    32537.000000  32537.000000
mean      38.559855      3.094446      3.102007        0.870670      6.139288
std       13.554847      1.107549      0.919933        0.713894      3.973173
min       17.000000      0.000000      0.000000        0.000000      0.000000
25%       28.000000      3.000000      3.000000        0.000000      3.000000
50%       37.000000      3.000000      3.000000        1.000000      6.000000
75%       48.000000      3.000000      4.000000        1.000000      9.000000
max       78.000000      7.000000      4.000000        2.000000     13.000000

       relationship          race           sex  capital.gain  capital.loss  \
count  32537.000000  32537.000000  32537.000000  32537.000000  32537.000000
mean       1.446538      3.665827      0.669238   1078.443741     87.368227
std        1.607064      0.848847      0.470495   7387.957424    403.101833
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      4.000000      0.000000      0.000000      0.000000
50%        1.000000      4.000000      1.000000      0.000000      0.000000
75%        3.000000      4.000000      1.000000      0.000000      0.000000
max        5.000000      4.000000      1.000000  99999.000000   4356.000000

       hours.per.week  native.country        income
count    32537.000000    32537.000000  32537.000000
mean        40.440329        0.913883      0.240926
std         12.346889        0.280542      0.427652
min          1.000000        0.000000      0.000000
25%         40.000000        1.000000      0.000000
50%         40.000000        1.000000      0.000000
```

```
75%        45.000000      1.000000     0.000000
max        99.000000      1.000000     1.000000
```

[58]:
```
x_test
```

[58]:
```
array([[27.,  1.,  3., …,  0., 40.,  1.],
       [26.,  6.,  4., …,  0., 35.,  1.],
       [77.,  3.,  4., …,  0.,  6.,  1.],
       …,
       [60.,  3.,  3., …,  0., 40.,  1.],
       [38.,  5.,  3., …,  0., 42.,  1.],
       [51.,  3.,  3., …,  0., 50.,  1.]])
```

[59]:
```
y_train
```

[59]:
```
array([1, 0, 1, …, 1, 1, 0])
```

[60]:
```
y_test
```

[60]:
```
array([0, 0, 0, …, 0, 0, 0])
```

NORMALISATION

[61]:
```
#Normalisation
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

MODEL CREATION AND PERFORMANCE EVALUATION

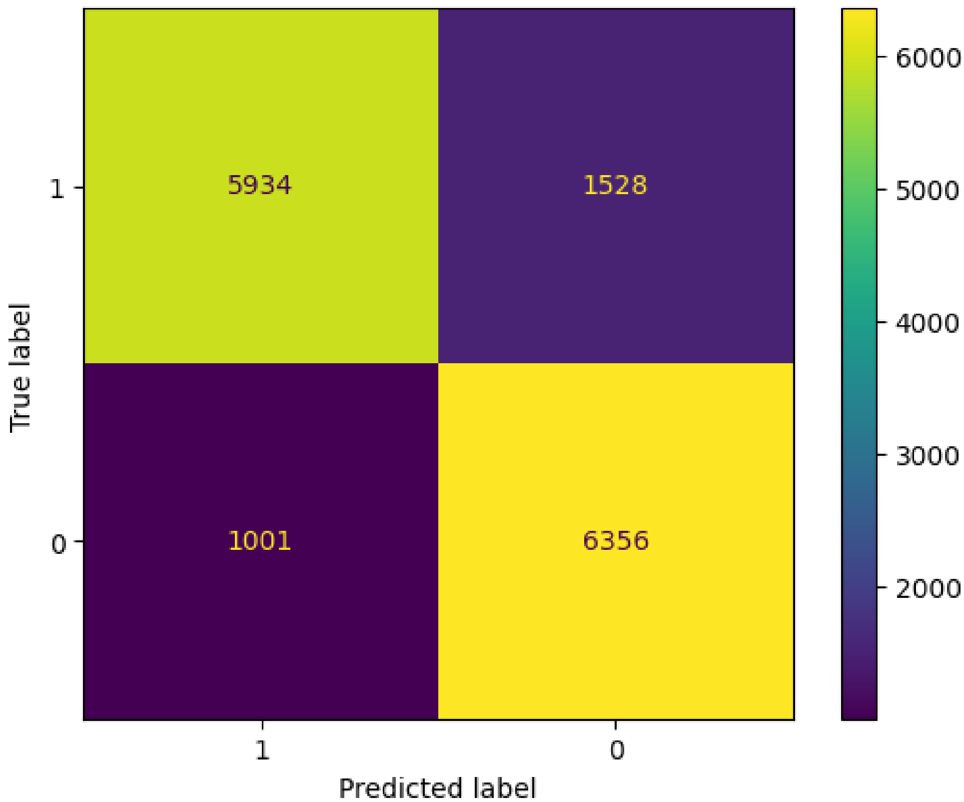
[62]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7)
from sklearn.naive_bayes import BernoulliNB
base=BernoulliNB()
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='entropy')
from sklearn.ensemble import RandomForestClassifier
rand=RandomForestClassifier()
list=[knn,base,model,rand]
from sklearn.metrics import
 ↪accuracy_score,confusion_matrix,classification_report,ConfusionMatrixDisplay
```

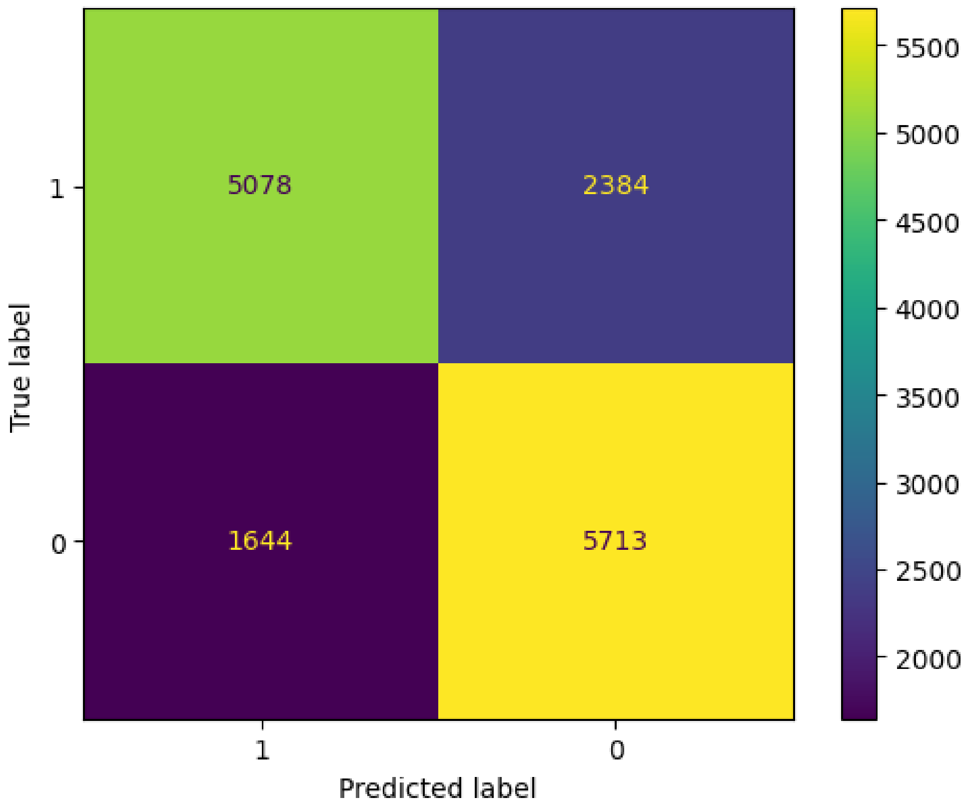
[63]:
```
for i in list:
  print(i)
  i.fit(x_train,y_train)
  y_pred=i.predict(x_test)
```

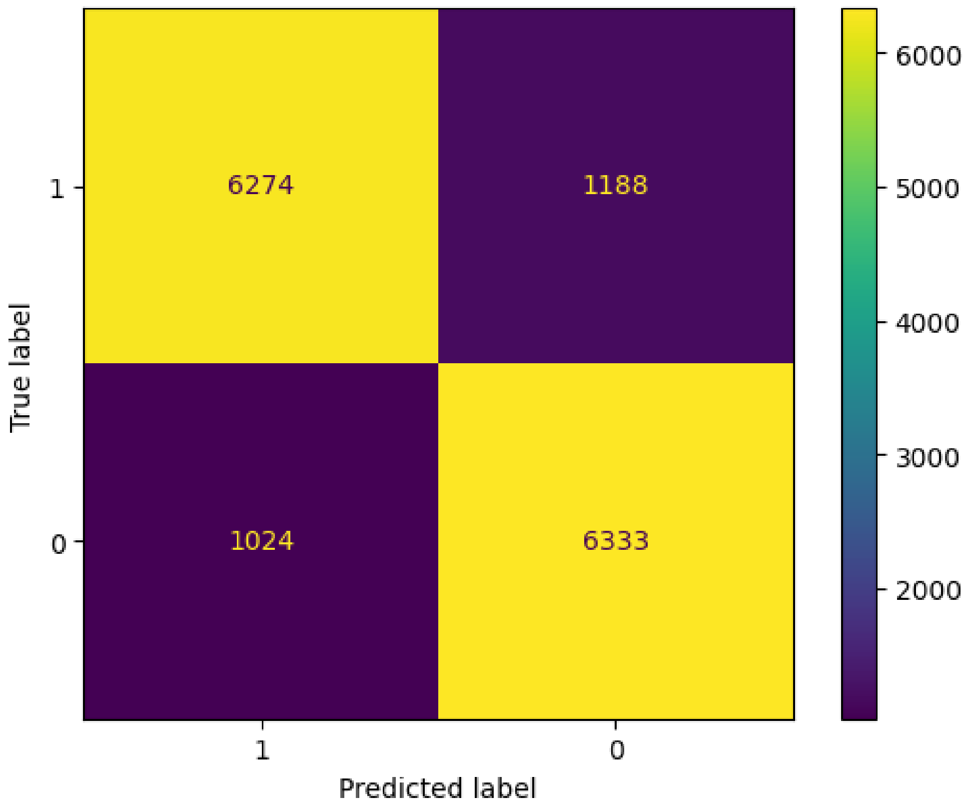
```
print(y_pred)
print('*'*100)
cm=confusion_matrix(y_test,y_pred)
print('confusion matrix is',cm)
print('*'*100)
print('accuracy score is',accuracy_score(y_test,y_pred))
print('*'*100)
print('classification report is',classification_report(y_test,y_pred))
label=[1,0]
cmd=ConfusionMatrixDisplay(cm,display_labels=label)
cmd.plot()
```

```
KNeighborsClassifier(n_neighbors=7)
[0 0 0 … 0 0 1]
********************************************************************************
********************
confusion matrix is [[5934 1528]
 [1001 6356]]
********************************************************************************
********************
accuracy score is 0.8293407112490722
********************************************************************************
********************
classification report is               precision    recall  f1-score   support

           0       0.86      0.80      0.82      7462
           1       0.81      0.86      0.83      7357

    accuracy                           0.83     14819
   macro avg       0.83      0.83      0.83     14819
weighted avg       0.83      0.83      0.83     14819

BernoulliNB()
[0 0 1 … 1 0 1]
********************************************************************************
********************
confusion matrix is [[5078 2384]
 [1644 5713]]
********************************************************************************
********************
accuracy score is 0.7281867872326068
********************************************************************************
********************
classification report is               precision    recall  f1-score   support

           0       0.76      0.68      0.72      7462
           1       0.71      0.78      0.74      7357
```

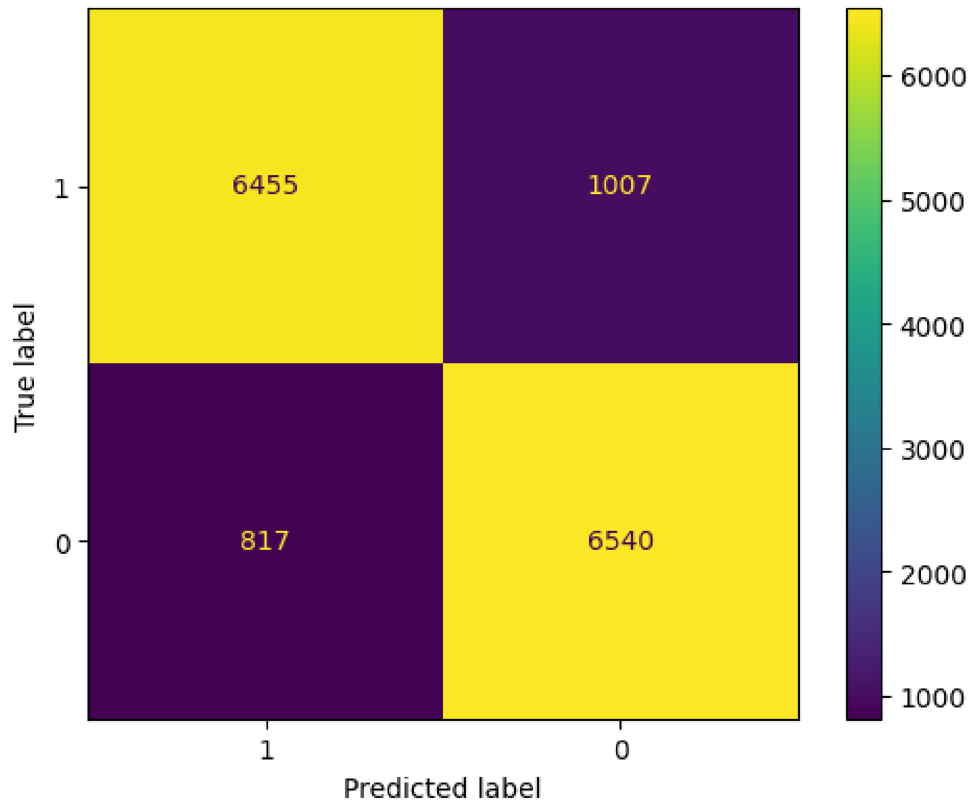
```
    accuracy                           0.73     14819
   macro avg       0.73      0.73      0.73     14819
weighted avg       0.73      0.73      0.73     14819

DecisionTreeClassifier(criterion='entropy')
[0 0 0 … 0 0 0]
********************************************************************************
********************
confusion matrix is [[6274 1188]
 [1024 6333]]
********************************************************************************
********************
accuracy score is 0.8507321681624941
********************************************************************************
********************
classification report is               precision    recall  f1-score   support

           0       0.86      0.84      0.85      7462
           1       0.84      0.86      0.85      7357

    accuracy                           0.85     14819
   macro avg       0.85      0.85      0.85     14819
weighted avg       0.85      0.85      0.85     14819

RandomForestClassifier()
[0 0 0 … 0 0 1]
********************************************************************************
********************
confusion matrix is [[6455 1007]
 [ 817 6540]]
********************************************************************************
********************
accuracy score is 0.8769147715770295
********************************************************************************
********************
classification report is               precision    recall  f1-score   support

           0       0.89      0.87      0.88      7462
           1       0.87      0.89      0.88      7357

    accuracy                           0.88     14819
   macro avg       0.88      0.88      0.88     14819
weighted avg       0.88      0.88      0.88     14819
```


5934
1528
1001
6356
True label
1
0
1
0
Predicted label
6000
5000
4000
3000
2000


5078
2384
1644
5713
True label
Predicted label
1
0
1
0
5500
5000
4500
4000
3500
3000
2500
2000



6274
1188
1024
6333
True label
1
0
Predicted label
6000
5000
4000
3000
2000



HYPER PARAMETER TUNING

[64]:
```
#KNN
from sklearn.model_selection import GridSearchCV
knn_1=KNeighborsClassifier()
para_knn={'n_neighbors':[5,7,9,11,13,15],'weights':['uniform','distance']}
clf1=GridSearchCV(knn_1,para_knn,cv=5,scoring='accuracy')
clf1.fit(x_train,y_train)
print(clf1.best_params_)
```

```
{'n_neighbors': 15, 'weights': 'distance'}
```

[65]:
```
#KNN after hyper parameter tuning
knn2=KNeighborsClassifier(n_neighbors=15,weights='distance')
knn2.fit(x_train,y_train)
y_pred_knn=knn2.predict(x_test)
print(classification_report(y_test,y_pred_knn))
```

```
              precision    recall  f1-score   support

           0       0.87      0.81      0.84      7462
           1       0.82      0.87      0.85      7357
```

```
    accuracy                           0.84     14819
   macro avg       0.84      0.84      0.84     14819
weighted avg       0.84      0.84      0.84     14819
```

[66]:
```
#NAIVE BAYES
naiv1=BernoulliNB()
para_nb={'alpha':[0.1,0.5,1.0,1.5],'binarize':[0.0,0.1,0.2],'fit_prior':
 ↪[True,False]}
clf2=GridSearchCV(naiv1,para_nb,cv=5,scoring='accuracy')
clf2.fit(x_train,y_train)
print(clf2.best_params_)
```

```
{'alpha': 1.5, 'binarize': 0.2, 'fit_prior': False}
```

[67]:
```
#NAIVE BAYES after hyperparameter tuning
naiv2=BernoulliNB(alpha=0.1,binarize=0.2,fit_prior=False)
naiv2.fit(x_train,y_train)
y_pred_n=naiv2.predict(x_test)
print(classification_report(y_test,y_pred_n))
```

```
              precision    recall  f1-score   support

           0       0.74      0.73      0.74      7462
           1       0.73      0.75      0.74      7357

    accuracy                           0.74     14819
   macro avg       0.74      0.74      0.74     14819
weighted avg       0.74      0.74      0.74     14819
```

[68]:
```
#DECISION TREE
model1=DecisionTreeClassifier()
para_dec={'criterion':['entropy'],'splitter':['best','random']}
clf4=GridSearchCV(model1,para_dec,cv=5,scoring='accuracy')
clf4.fit(x_train,y_train)
print(clf4.best_params_)
```

```
{'criterion': 'entropy', 'splitter': 'best'}
```

[69]:
```
#DECISION TREE after hyper parameter tuning
model2=DecisionTreeClassifier(criterion='entropy',splitter='best',random_state=3)
model2.fit(x_train,y_train)
y_pred_m=model2.predict(x_test)
print(classification_report(y_test,y_pred_m))
```

```
              precision    recall  f1-score   support
```

```
              0       0.86      0.84      0.85      7462
              1       0.84      0.86      0.85      7357

       accuracy                           0.85     14819
      macro avg       0.85      0.85      0.85     14819
   weighted avg       0.85      0.85      0.85     14819
```

[70]:
```
#RANDOM FOREST
rand1=RandomForestClassifier()
para_rand={'n_estimators':[50,75,100,125,150],'criterion':['entropy']}
clf5=GridSearchCV(rand1,para_rand,cv=5,scoring='accuracy')
clf5.fit(x_train,y_train)
print(clf5.best_params_)
```

```
{'criterion': 'entropy', 'n_estimators': 75}
```

[71]:
```
#RANDOM FOREST after hyper parameter tuning
rand2=RandomForestClassifier(criterion='entropy',n_estimators=100,random_state=1)
rand2.fit(x_train,y_train)
y_pred_rand=rand2.predict(x_test)
print(classification_report(y_test,y_pred_rand))
```

```
              precision    recall  f1-score   support

              0       0.89      0.86      0.88      7462
              1       0.87      0.89      0.88      7357

       accuracy                           0.88     14819
      macro avg       0.88      0.88      0.88     14819
   weighted avg       0.88      0.88      0.88     14819
```

PERFOMANCE EVALUATION

[72]:
```
#KNN
cm_k=confusion_matrix(y_test,y_pred_knn)
cm_k
```

[72]:
```
array([[6041, 1421],
       [ 930, 6427]])
```

[73]:
```
score_k=accuracy_score(y_test,y_pred_knn)
score_k
```

[73]:
```
0.8413523179701734
```

[74]:
```
#NAIVE BAYES
cm_n=confusion_matrix(y_test,y_pred_n)
cm_n
```

[74]:
```
array([[5428, 2034],
       [1875, 5482]])
```

[75]:
```
score_n=accuracy_score(y_test,y_pred_n)
score_n
```

[75]:
```
0.7362170186922194
```

[76]:
```
#DECISION TREE
cm_d=confusion_matrix(y_test,y_pred_m)
cm_d
```

[76]:
```
array([[6272, 1190],
       [1027, 6330]])
```

[77]:
```
score_d=accuracy_score(y_test,y_pred_m)
score_d
```

[77]:
```
0.8503947634793171
```

[78]:
```
#RANDOM FOREST
cm_r=confusion_matrix(y_test,y_pred_rand)
cm_r
```

[78]:
```
array([[6450, 1012],
       [ 810, 6547]])
```

[79]:
```
score_r=accuracy_score(y_test,y_pred_rand)
score_r
```

[79]:
```
0.8770497334503002
```

[80]:
```
output_df=pd.DataFrame({'Before hyper parameter tuning':[82.9,72.8,85.1,87.
 ↪7],'After hyper parameter tuning':
 ↪[score_k*100,score_n*100,score_d*100,score_r*100]},index=['KNN','NAIVE␣
 ↪BAYES','DECISION TREE','RANDOM FOREST'])
output_df
```

[80]:

	Before hyper parameter tuning	After hyper parameter tuning
KNN	82.9	84.135232
NAIVE BAYES	72.8	73.621702
DECISION TREE	85.1	85.039476
RANDOM FOREST	87.7	87.704973

Streamlit

[81]:
```
import pickle
pickle.dump(rand2, open('/content/random_model.pkl', 'wb'))
```

[82]:
```
! pip install streamlit -q
```

```
                                   8.6/8.6 MB
17.3 MB/s eta 0:00:00
                                   207.3/207.3
kB 19.9 MB/s eta 0:00:00
                                   6.9/6.9 MB
36.5 MB/s eta 0:00:00
                                   83.0/83.0 kB
7.4 MB/s eta 0:00:00
                                   62.7/62.7 kB
5.0 MB/s eta 0:00:00
```

[83]:
```
! wget -q -O - ipv4.icanhazip.com
```

```
34.125.71.127
```

[ ]:
```
! streamlit run project.py & npx localtunnel --port 8501
```

```
Usage: streamlit run [OPTIONS] TARGET [ARGS]...
Try 'streamlit run --help' for help.

Error: Invalid value: File does not exist: project.py
npx: installed 22 in 6.574s
your url is: https://dark-ducks-wink.loca.lt
```