Classification model on Cencus Income Dataset

Problem Statement : Prediction task is to determine whether a person makes over 50K a year.

```
from IPython import display
display.Image("income.png")
```

Connect with me

github - https://github.com/saisubhasish

Linked in - https://www.linkedin.com/in/sai-subhasish-rout-655707151/

Steps

1. Data Injection

```
- 1.1 Data Profiling
- 1.2 Basic Operations
- 1.3 Data Cleaning
- 1.4 Analysis of features and Statistical Analysis
```

2. EDA

```
- 2.1 Univariate Analysis
- 2.2 Bivariate Analysis
- 2.3 Multivariate Analysis
```

3. Pre-processing
```
- 3.1 Dropping null values
- 3.2 Saving data to mangoDB
- 3.3 Feature Selection
- 3.4 Feature Encoding
- 3.5 Train-Test split
```

4. Model Building
```
- 4.1 Decision Tree Classifier
- 4.2 HyperParameter Tuning : Decision Tree Classifier
- 4.3 Bagging Classifier
- 4.4 Hyperparameter tuning : Bagging Classifier
- 4.5 Random Forest Classifier
- 4.6 Hyperparameter tuning : Random Forest Classifier
- 4.7 Extra Trees Classifier
- 4.8 HyperParameter Tuning : Extra Tree Classifier
- 4.9 Voting Classifier
    i. hard_voting
    ii. soft_voting
```

5. Evaluation
```
- 5.1 Accuracy Score
- 5.2 Roc-auc score
- 5.3 Confusion_matrix
```

Importing Libraries
```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import scipy.stats as stats

import warnings
warnings.filterwarnings('ignore')
```

1. Data injection
```
column_names = ['age', 'work_class', 'fnlwgt', 'education',
'education_num', 'marital_status', 'occupation', 'relationship',
'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week',
'native_country', 'salary']

data1 =
pd.read_csv('D:/FSDS-iNeuron/3.Resource/Dataset/CencusData/adult_D.dat
a', header=None, names=column_names)
data2 =
pd.read_csv('D:/FSDS-iNeuron/3.Resource/Dataset/CencusData/adult_T.tes
t', header=1, names=column_names)
```

1.1 Data Profiling

```
data1.head()

   age         work_class  fnlwgt   education  education_num  \
0   39          State-gov   77516   Bachelors             13
1   50   Self-emp-not-inc   83311   Bachelors             13
2   38            Private  215646     HS-grad              9
3   53            Private  234721        11th              7
4   28            Private  338409   Bachelors             13

        marital_status          occupation    relationship    race
sex  \
0        Never-married        Adm-clerical   Not-in-family   White
Male
1   Married-civ-spouse     Exec-managerial         Husband   White
Male
2             Divorced   Handlers-cleaners   Not-in-family   White
Male
3   Married-civ-spouse   Handlers-cleaners         Husband   Black
Male
4   Married-civ-spouse      Prof-specialty            Wife   Black
Female

   capital_gain  capital_loss  hours_per_week  native_country  salary

0          2174             0              40   United-States   <=50K

1             0             0              13   United-States   <=50K

2             0             0              40   United-States   <=50K

3             0             0              40   United-States   <=50K

4             0             0              40            Cuba   <=50K

data2.head()

   age  work_class  fnlwgt      education  education_num
marital_status  \
0   38     Private   89814        HS-grad              9   Married-
civ-spouse
1   28   Local-gov  336951     Assoc-acdm             12   Married-
civ-spouse
2   44     Private  160323   Some-college             10   Married-
civ-spouse
3   18           ?  103497   Some-college             10        Never-
married
4   34     Private  198693           10th              6        Never-
married
```

```
           occupation   relationship   race     sex  
capital_gain  \
0     Farming-fishing        Husband  White    Male             0

1     Protective-serv        Husband  White    Male             0

2   Machine-op-inspct        Husband  Black    Male          7688

3                   ?      Own-child  White  Female             0

4       Other-service  Not-in-family  White    Male             0


   capital_loss  hours_per_week  native_country  salary
0             0              50   United-States  <=50K.
1             0              40   United-States   >50K.
2             0              40   United-States   >50K.
3             0              30   United-States  <=50K.
4             0              30   United-States  <=50K.
```

Joining both of the dataset

```
df = pd.concat([data1, data2])

df
```

```
       age         work_class  fnlwgt   education  education_num  \
0       39          State-gov   77516   Bachelors             13   
1       50   Self-emp-not-inc   83311   Bachelors             13   
2       38            Private  215646     HS-grad              9   
3       53            Private  234721        11th              7   
4       28            Private  338409   Bachelors             13   
...    ...                ...     ...         ...            ...   
16275   39            Private  215419   Bachelors             13   
16276   64                  ?  321403     HS-grad              9   
16277   38            Private  374983   Bachelors             13   
16278   44            Private   83891   Bachelors             13   
16279   35       Self-emp-inc  182148   Bachelors             13   

            marital_status          occupation     relationship  \
0            Never-married        Adm-clerical    Not-in-family   
1       Married-civ-spouse     Exec-managerial          Husband   
2                 Divorced   Handlers-cleaners    Not-in-family   
3       Married-civ-spouse   Handlers-cleaners          Husband   
4       Married-civ-spouse      Prof-specialty             Wife   
...                    ...                 ...              ...   
16275             Divorced      Prof-specialty    Not-in-family   
16276              Widowed                   ?   Other-relative   
16277   Married-civ-spouse      Prof-specialty          Husband   
16278             Divorced        Adm-clerical        Own-child   
16279   Married-civ-spouse     Exec-managerial          Husband   
```

```
                     race      sex  capital_gain  capital_loss  \
0                   White     Male          2174             0
1                   White     Male             0             0
2                   White     Male             0             0
3                   Black     Male             0             0
4                   Black   Female             0             0
...                   ...      ...           ...           ...
16275               White   Female             0             0
16276               Black     Male             0             0
16277               White     Male             0             0
16278  Asian-Pac-Islander     Male          5455             0
16279               White     Male             0             0

       hours_per_week  native_country   salary
0                  40   United-States    <=50K
1                  13   United-States    <=50K
2                  40   United-States    <=50K
3                  40   United-States    <=50K
4                  40            Cuba    <=50K
...               ...             ...      ...
16275              36   United-States   <=50K.
16276              40   United-States   <=50K.
16277              50   United-States   <=50K.
16278              40   United-States   <=50K.
16279              60   United-States    >50K.

[48841 rows x 15 columns]
```

Resetting the index

Added a column named 'index' with index value to get data in sequence

```python
df.reset_index(inplace=True)

#### Dropping the index column as it is not required further

df.drop('index', axis=1, inplace=True)
```

1.2 Basic Operations

Getting the shape of the data

```python
df.shape
```

```
(48841, 15)
```

Observation:

- Dataset has 15 columns and 48841 rows.

Columns of the dataset

```python
df.columns
```

```
Index(['age', 'work_class', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week',
'native_country',
       'salary'],
      dtype='object')
```

1.3 Data cleaning

- To categorize person's income >50K, <=50K.
- age: continuous.
- workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- fnlwgt (final weight - In other words, this is the number of people the census believes the entry represents.): continuous.
- education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- education-num: continuous.
- marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex: Female, Male.
- capital-gain: continuous.
- capital-loss: continuous.
- hours-per-week: continuous.
- native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

```
df_cleaned = df.copy()
```

Datatypes of each column

```
df_cleaned.dtypes
```

```
age                int64
work_class        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
salary            object
dtype: object
```

Observation :

- age is a a continuous numeric feature but having data type as "object".
- rest all features have data type as per their properties

To check the duplicate values

```
len(df_cleaned[df_cleaned.duplicated()])
```

```
29
```

Observation :

- There are 29 duplicate records

```
df_cleaned.drop_duplicates(inplace=True)
```

```
df_cleaned[df_cleaned.duplicated()].shape[0]
```

```
0
```

Observation :

- All the duplicate records are dropped.

To check the null values

```
df_cleaned.isna().sum()
```

```
age               0
work_class        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
salary            0
dtype: int64
```

Observation :

- There is no null values in the dataset.

Basic information of the dataset

```
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48812 entries, 0 to 48840
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             48812 non-null  int64 
 1   work_class      48812 non-null  object
 2   fnlwgt          48812 non-null  int64 
 3   education       48812 non-null  object
 4   education_num   48812 non-null  int64 
 5   marital_status  48812 non-null  object
 6   occupation      48812 non-null  object
 7   relationship    48812 non-null  object
 8   race            48812 non-null  object
 9   sex             48812 non-null  object
 10  capital_gain    48812 non-null  int64 
 11  capital_loss    48812 non-null  int64 
 12  hours_per_week  48812 non-null  int64 
 13  native_country  48812 non-null  object
 14  salary          48812 non-null  object
dtypes: int64(6), object(9)
memory usage: 6.0+ MB
```

Observation :

- Memory usage is 6.0+ MB
- There are 1 float, 5 int, 9 object data types.

Checking the unique values in each column

```
for column in df_cleaned.columns:
    print(f"Feature {column} has {df_cleaned[column].unique()} unique 
features\n")
```

```
Feature age has [39 50 38 53 28 37 49 52 31 42 30 23 32 40 34 25 43 54
35 59 56 19 20 45
 22 48 21 24 57 44 41 29 18 47 46 36 79 27 67 33 76 17 55 61 70 64 71
68
 66 51 58 26 60 90 75 65 77 62 63 80 72 74 69 73 81 78 88 82 83 84 85
86
 87 89] unique features

Feature work_class has [' State-gov' ' Self-emp-not-inc' ' Private' '
Federal-gov' ' Local-gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked'] unique features

Feature fnlwgt has [ 77516  83311 215646 ... 173449  89686 350977]
unique features

Feature education has [' Bachelors' ' HS-grad' ' 11th' ' Masters' '
9th' ' Some-college'
 ' Assoc-acdm' ' Assoc-voc' ' 7th-8th' ' Doctorate' ' Prof-school'
 ' 5th-6th' ' 10th' ' 1st-4th' ' Preschool' ' 12th'] unique features

Feature education_num has [13  9  7 14  5 10 12 11  4 16 15  3  6  2
1  8] unique features

Feature marital_status has [' Never-married' ' Married-civ-spouse' '
Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' '
Widowed'] unique features

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

Feature relationship has [' Not-in-family' ' Husband' ' Wife' ' Own-
child' ' Unmarried'
 ' Other-relative'] unique features

Feature race has [' White' ' Black' ' Asian-Pac-Islander' ' Amer-
Indian-Eskimo' ' Other'] unique features

Feature sex has [' Male' ' Female'] unique features

Feature capital_gain has [ 2174     0 14084  5178  5013  2407 14344
15024  7688 34095  4064  4386
  7298  1409  3674  1055  3464  2050  2176   594 20051  6849  4101
1111
  8614  3411  2597 25236  4650  9386  2463  3103 10605  2964  3325
```

```
2580
  3471  4865 99999  6514  1471  2329  2105  2885 25124 10520  2202
2961
 27828  6767  2228  1506 13550  2635  5556  4787  3781  3137  3818
3942
   914   401  2829  2977  4934  2062  2354  5455 15020  1424  3273
22040
  4416  3908 10566   991  4931  1086  7430  6497   114  7896  2346
3418
  3432  2907  1151  2414  2290 15831 41310  4508  2538  3456  6418
1848
  3887  5721  9562  1455  2036  1831 11678  2936  2993  7443  6360
1797
  1173  4687  6723  2009  6097  2653  1639 18481  7978  2387  5060
1264
  7262  1731  6612] unique features

Feature capital_loss has [   0 2042 1408 1902 1573 1887 1719 1762 1564
2179 1816 1980 1977 1876
 1340 2206 1741 1485 2339 2415 1380 1721 2051 2377 1669 2352 1672  653
 2392 1504 2001 1590 1651 1628 1848 1740 2002 1579 2258 1602  419 2547
 2174 2205 1726 2444 1138 2238  625  213 1539  880 1668 1092 1594 3004
 2231 1844  810 2824 2559 2057 1974  974 2149 1825 1735 1258 2129 2603
 2282  323 4356 2246 1617 1648 2489 3770 1755 3683 2267 2080 2457  155
 3900 2201 1944 2467 2163 2754 2472 1411 1429 3175 1510 1870 1911 2465
 1421] unique features

Feature hours_per_week has [40 13 16 45 50 80 30 35 60 20 52 44 15 25
38 43 55 48 58 32 70  2 22 56
 41 28 36 24 46 42 12 65  1 10 34 75 98 33 54  8  6 64 19 18 72  5  9
47
 37 21 26 14  4 59  7 99 53 39 62 57 78 90 66 11 49 84  3 17 68 27 85
31
 51 77 63 23 87 88 73 89 97 94 29 96 67 82 86 91 81 76 92 61 74 95 79
69] unique features

Feature native_country has [' United-States' ' Cuba' ' Jamaica' '
India' ' ?' ' Mexico' ' South'
 ' Puerto-Rico' ' Honduras' ' England' ' Canada' ' Germany' ' Iran'
 ' Philippines' ' Italy' ' Poland' ' Columbia' ' Cambodia' ' Thailand'
 ' Ecuador' ' Laos' ' Taiwan' ' Haiti' ' Portugal' ' Dominican-
Republic'
 ' El-Salvador' ' France' ' Guatemala' ' China' ' Japan' ' Yugoslavia'
 ' Peru' ' Outlying-US(Guam-USVI-etc)' ' Scotland' ' Trinadad&Tobago'
 ' Greece' ' Nicaragua' ' Vietnam' ' Hong' ' Ireland' ' Hungary'
 ' Holand-Netherlands'] unique features

Feature salary has [' <=50K' ' >50K' ' <=50K.' ' >50K.'] unique
features
```

Observation :

- (Work_class,salary, occupation have '?') (education, marital-status, occupation, relationship, race, sex, native_country's values have space)
- We need to change those values

Changing the datatypes of the features

```
df_cleaned = df_cleaned.astype({'age':float, 'hours_per_week':float})
```

1.4 Analysis of features

Analysis of feature : age

Observation :

- "age" data type is changed.

```
df_cleaned.age.min()
```

```
17.0
```

```
df_cleaned.age.max()
```

```
90.0
```

Observation :

- Data set consists records of people agging between 17-90

Analysis of feature : work_class

```
df_cleaned['work_class'].value_counts()
```

```
 Private             33878
 Self-emp-not-inc     3861
 Local-gov            3136
 ?                    2799
 State-gov            1981
 Self-emp-inc         1694
 Federal-gov          1432
 Without-pay            21
 Never-worked           10
Name: work_class, dtype: int64
```

Observation :

- People with "Private" job are more compared to other sectors.
- There are 2799 unknown values.

Analysis of feature : fnlwgt

```
df_cleaned['fnlwgt'].value_counts()
```

```
203488    21
120277    19
190290    19
```

```
126569    18
125892    18
          ..
78170      1
279721     1
390867     1
354075     1
350977     1
Name: fnlwgt, Length: 28522, dtype: int64
```

Analysis of feature : education

```
df_cleaned['education'].value_counts()
```

```
 HS-grad         15777
 Some-college    10869
 Bachelors        8020
 Masters          2656
 Assoc-voc        2060
 11th             1811
 Assoc-acdm       1601
 10th             1389
 7th-8th           954
 Prof-school       834
 9th               756
 12th              656
 Doctorate         594
 5th-6th           508
 1st-4th           245
 Preschool          82
Name: education, dtype: int64
```

Observation :

- People with "HS-grad" are more.

Analysis of feature : education_num

```
df_cleaned['education_num'].value_counts()
```

```
9     15777
10    10869
13     8020
14     2656
11     2060
7      1811
12     1601
6      1389
4       954
15      834
5       756
8       656
16      594
```

```
3       508
2       245
1        82
Name: education_num, dtype: int64
```

Observation :

- People with education level 9 are in larger number followed by 10.

Analysis of feature : marital_status

```
df_cleaned['marital_status'].value_counts()
```

```
 Married-civ-spouse       22372
 Never-married            16097
 Divorced                  6630
 Separated                 1530
 Widowed                   1518
 Married-spouse-absent      628
 Married-AF-spouse           37
Name: marital_status, dtype: int64
```

Observation :

- People who married a Civilian spouse are largest in number
- Preople who married a Armed Force spouse are least

Analysis of feature : marital_status

```
df_cleaned['occupation'].unique()
```

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

```
df_cleaned.groupby(by='marital_status').count()
```

```
                          age  work_class  fnlwgt  education  
education_num  \
marital_status                                                 

 Divorced                6630        6630    6630       6630   
6630
 Married-AF-spouse         37          37      37         37   
37
 Married-civ-spouse     22372       22372   22372      22372   
22372
 Married-spouse-absent    628         628     628        628   
628
 Never-married          16097       16097   16097      16097   
16097
 Separated               1530        1530    1530       1530
```

```
1530
 Widowed                   1518         1518    1518        1518
1518

                        occupation  relationship   race    sex
capital_gain  \
marital_status

 Divorced                     6630          6630   6630   6630
6630
 Married-AF-spouse              37            37     37     37
37
 Married-civ-spouse          22372         22372  22372  22372
22372
 Married-spouse-absent         628           628    628    628
628
 Never-married               16097         16097  16097  16097
16097
 Separated                    1530          1530   1530   1530
1530
 Widowed                      1518          1518   1518   1518
1518

                        capital_loss  hours_per_week  native_country
salary
marital_status

 Divorced                       6630            6630            6630
6630
 Married-AF-spouse                37              37              37
37
 Married-civ-spouse            22372           22372           22372
22372
 Married-spouse-absent           628             628             628
628
 Never-married                 16097           16097           16097
16097
 Separated                      1530            1530            1530
1530
 Widowed                        1518            1518            1518
1518
```

Analysis of feature : occupation

```
df_cleaned['occupation'].value_counts()
```

```
 Prof-specialty       6167
 Craft-repair         6107
 Exec-managerial      6084
 Adm-clerical         5608
 Sales                5504
```

```
 Other-service        4919
 Machine-op-inspct    3018
 ?                    2809
 Transport-moving     2355
 Handlers-cleaners    2071
 Farming-fishing      1487
 Tech-support         1445
 Protective-serv       983
 Priv-house-serv       240
 Armed-Forces           15
Name: occupation, dtype: int64
```

Observation :

- People with occupation "Prof-speciality" are more
- "Armed-forces" people are least
- 2809 records have unknown entries

Analysis of feature : relationship

```
df_cleaned['relationship'].value_counts()
```

```
 Husband           19709
 Not-in-family     12567
 Own-child          7575
 Unmarried          5124
 Wife               2331
 Other-relative     1506
Name: relationship, dtype: int64
```

Analysis of feature : race

```
df_cleaned['race'].value_counts()
```

```
 White                 41736
 Black                  4682
 Asian-Pac-Islander     1518
 Amer-Indian-Eskimo      470
 Other                   406
Name: race, dtype: int64
```

Observation :

- "White" people are the largest as per data followed by "Black" people.

Analysis of feature : sex

```
df_cleaned.groupby('sex').sum()
```

```
               age      fnlwgt  education_num  capital_gain  
capital_loss  \
sex

 Female   597647.0  3001627210         162545       9403120  
995411
```

```
 Male    1288821.0  6256405433         329419      43300701
3278377

         hours_per_week
sex
 Female        589085.0
 Male         1384143.0
```

```
df_cleaned['sex'].value_counts()
```

```
 Male      32630
 Female    16182
Name: sex, dtype: int64
```

Observation :

- There are more male compared to female.

Analysis of feature : capital_gain

```
df_cleaned['capital_gain'].unique()
```

```
array([ 2174,     0, 14084,  5178,  5013,  2407, 14344, 15024,  7688,
       34095,  4064,  4386,  7298,  1409,  3674,  1055,  3464,  2050,
        2176,   594, 20051,  6849,  4101,  1111,  8614,  3411,  2597,
       25236,  4650,  9386,  2463,  3103, 10605,  2964,  3325,  2580,
        3471,  4865, 99999,  6514,  1471,  2329,  2105,  2885, 25124,
       10520,  2202,  2961, 27828,  6767,  2228,  1506, 13550,  2635,
        5556,  4787,  3781,  3137,  3818,  3942,   914,   401,  2829,
        2977,  4934,  2062,  2354,  5455, 15020,  1424,  3273, 22040,
        4416,  3908, 10566,   991,  4931,  1086,  7430,  6497,   114,
        7896,  2346,  3418,  3432,  2907,  1151,  2414,  2290, 15831,
       41310,  4508,  2538,  3456,  6418,  1848,  3887,  5721,  9562,
        1455,  2036,  1831, 11678,  2936,  2993,  7443,  6360,  1797,
        1173,  4687,  6723,  2009,  6097,  2653,  1639, 18481,  7978,
        2387,  5060,  1264,  7262,  1731,  6612], dtype=int64)
```

Analysis of feature : capital_loss

```
df_cleaned['capital_loss'].unique()
```

```
array([   0, 2042, 1408, 1902, 1573, 1887, 1719, 1762, 1564, 2179,
1816,
       1980, 1977, 1876, 1340, 2206, 1741, 1485, 2339, 2415, 1380,
1721,
       2051, 2377, 1669, 2352, 1672,  653, 2392, 1504, 2001, 1590,
1651,
       1628, 1848, 1740, 2002, 1579, 2258, 1602,  419, 2547, 2174,
2205,
       1726, 2444, 1138, 2238,  625,  213, 1539,  880, 1668, 1092,
1594,
       3004, 2231, 1844,  810, 2824, 2559, 2057, 1974,  974, 2149,
1825,
```

```
       1735, 1258, 2129, 2603, 2282,  323, 4356, 2246, 1617, 1648,
2489,
       3770, 1755, 3683, 2267, 2080, 2457,  155, 3900, 2201, 1944,
2467,
       2163, 2754, 2472, 1411, 1429, 3175, 1510, 1870, 1911, 2465,
1421],
      dtype=int64)
```

Analysis of feature : hours_per_week

```
df_cleaned['hours_per_week'].unique()
```

```
array([40., 13., 16., 45., 50., 80., 30., 35., 60., 20., 52., 44.,
15.,
       25., 38., 43., 55., 48., 58., 32., 70.,  2., 22., 56., 41.,
28.,
       36., 24., 46., 42., 12., 65.,  1., 10., 34., 75., 98., 33.,
54.,
        8.,  6., 64., 19., 18., 72.,  5.,  9., 47., 37., 21., 26.,
14.,
        4., 59.,  7., 99., 53., 39., 62., 57., 78., 90., 66., 11.,
49.,
       84.,  3., 17., 68., 27., 85., 31., 51., 77., 63., 23., 87.,
88.,
       73., 89., 97., 94., 29., 96., 67., 82., 86., 91., 81., 76.,
92.,
       61., 74., 95., 79., 69.])
```

Analysis of feature : native_country

```
df_cleaned['native_country'].value_counts()
```

```
 United-States                 43809
 Mexico                          947
 ?                               856
 Philippines                     295
 Germany                         206
 Puerto-Rico                     184
 Canada                          182
 El-Salvador                     155
 India                           151
 Cuba                            138
 England                         127
 China                           122
 South                           115
 Jamaica                         106
 Italy                           105
 Dominican-Republic              103
 Japan                            92
 Poland                           87
 Guatemala                        86
 Vietnam                          86
 Columbia                         85
```

```
 Haiti                            75
 Portugal                         67
 Taiwan                           65
 Iran                             59
 Greece                           49
 Nicaragua                        49
 Peru                             46
 Ecuador                          45
 France                           38
 Ireland                          37
 Hong                             30
 Thailand                         30
 Cambodia                         28
 Trinadad&Tobago                  27
 Laos                             23
 Yugoslavia                       23
 Outlying-US(Guam-USVI-etc)       23
 Scotland                         21
 Honduras                         20
 Hungary                          19
 Holand-Netherlands                1
Name: native_country, dtype: int64
```

Observation :

- People belong to "United-States" are more compared other states.

Analysis of each salary : to check unique values

```
df_cleaned.salary.unique()
```

```
array([' <=50K', ' >50K', ' <=50K.', ' >50K.'], dtype=object)
```

Observation :

- There are . in the values
- We need to replace it

```
#### Repalcing . from salary column

df_cleaned['salary'] = df_cleaned['salary'].replace('<=50K.', '<=50K', regex=True)
df_cleaned['salary'] = df_cleaned['salary'].replace('>50K.', '>50K', regex=True)

df_cleaned['salary'].value_counts()
```

```
 <=50K    37127
 >50K     11685
Name: salary, dtype: int64
```

Observation :

- People with salary more than 50k are more.

Categorizing the categorical and numerical features

```
# For categorical features

categorical_features = [feature for feature in df_cleaned.columns if
df_cleaned[feature].dtype == 'object']
print(categorical_features)
```

```
['work_class', 'education', 'marital_status', 'occupation',
'relationship', 'race', 'sex', 'native_country', 'salary']
```

Getting count of each category from dataframe

```
for feature in categorical_features:
    print(df_cleaned[feature].value_counts())
```

```
 Private              33878
 Self-emp-not-inc      3861
 Local-gov             3136
 ?                     2799
 State-gov             1981
 Self-emp-inc          1694
 Federal-gov           1432
 Without-pay             21
 Never-worked            10
Name: work_class, dtype: int64
 HS-grad         15777
 Some-college    10869
 Bachelors        8020
 Masters          2656
 Assoc-voc        2060
 11th             1811
 Assoc-acdm       1601
 10th             1389
 7th-8th           954
 Prof-school       834
 9th               756
 12th              656
 Doctorate         594
 5th-6th           508
 1st-4th           245
 Preschool          82
Name: education, dtype: int64
 Married-civ-spouse       22372
 Never-married            16097
 Divorced                  6630
 Separated                 1530
 Widowed                   1518
 Married-spouse-absent      628
 Married-AF-spouse           37
Name: marital_status, dtype: int64
 Prof-specialty       6167
```

```
 Craft-repair         6107
 Exec-managerial      6084
 Adm-clerical         5608
 Sales                5504
 Other-service        4919
 Machine-op-inspct    3018
 ?                    2809
 Transport-moving     2355
 Handlers-cleaners    2071
 Farming-fishing      1487
 Tech-support         1445
 Protective-serv       983
 Priv-house-serv       240
 Armed-Forces           15
Name: occupation, dtype: int64
 Husband           19709
 Not-in-family     12567
 Own-child          7575
 Unmarried          5124
 Wife               2331
 Other-relative     1506
Name: relationship, dtype: int64
 White                 41736
 Black                  4682
 Asian-Pac-Islander     1518
 Amer-Indian-Eskimo      470
 Other                   406
Name: race, dtype: int64
 Male      32630
 Female    16182
Name: sex, dtype: int64
 United-States                 43809
 Mexico                          947
 ?                               856
 Philippines                     295
 Germany                         206
 Puerto-Rico                     184
 Canada                          182
 El-Salvador                     155
 India                           151
 Cuba                            138
 England                         127
 China                           122
 South                           115
 Jamaica                         106
 Italy                           105
 Dominican-Republic              103
 Japan                            92
 Poland                           87
 Guatemala                        86
```

```
 Vietnam                          86
 Columbia                         85
 Haiti                            75
 Portugal                         67
 Taiwan                           65
 Iran                             59
 Greece                           49
 Nicaragua                        49
 Peru                             46
 Ecuador                          45
 France                           38
 Ireland                          37
 Hong                             30
 Thailand                         30
 Cambodia                         28
 Trinadad&Tobago                  27
 Laos                             23
 Yugoslavia                       23
 Outlying-US(Guam-USVI-etc)       23
 Scotland                         21
 Honduras                         20
 Hungary                          19
 Holand-Netherlands                1
Name: native_country, dtype: int64
 <=50K    37127
 >50K     11685
Name: salary, dtype: int64
```

Observation :

- In work_class 2799, occupation 2809, Native_country 856 values are '?'

Creating a function for trimming the space from each values in columns and repalcing the '?' value from each feature

```
def feature_cleaning(dataframe, features):
    for feature in features:
        dataframe[feature] = dataframe[feature].str.strip()

feature_cleaning(df_cleaned, categorical_features)

df_cleaned = df_cleaned.replace('?', np.nan)

#for feature in categorical_features:
#    pd.df_cleaned.replace('?', np.nan)
#    df_cleaned[feature] = df_cleaned[feature].replace('?', np.nan, 
regex=True)

for feature in categorical_features:
    print(df_cleaned[feature].unique())
```

```
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov'
nan
 'Self-emp-inc' 'Without-pay' 'Never-worked']
['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-
acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']
['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-
absent'
 'Separated' 'Married-AF-spouse' 'Widowed']
['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-specialty'
 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' nan
 'Protective-serv' 'Armed-Forces' 'Priv-house-serv']
['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-
relative']
['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']
['Male' 'Female']
['United-States' 'Cuba' 'Jamaica' 'India' nan 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand'
'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-
Netherlands']
['<=50K' '>50K']
```

Observation :

- "?" values are replaced with nan
- All the white space before and after the values are removed.

```
df_cleaned.shape
```

```
(48812, 15)
```

3.1 Dropping the nan values

```
df_cleaned.dropna(inplace=True)
```

```
df_cleaned.shape
```

```
(45193, 15)
```

Observation:

- records with na value got removed

Numerical features

```
numerical_features = [feature for feature in df_cleaned.columns if
feature not in categorical_features]
print(numerical_features)
```

```
['age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss',
'hours_per_week']
```

Finding the number of unique values

```
for feature in numerical_features:
    print('Feature "{}" has {} no of unique values'.format(feature,
df_cleaned[feature].nunique()))
```

```
Feature "age" has 74 no of unique values
Feature "fnlwgt" has 26740 no of unique values
Feature "education_num" has 16 no of unique values
Feature "capital_gain" has 121 no of unique values
Feature "capital_loss" has 97 no of unique values
Feature "hours_per_week" has 96 no of unique values
```

```
# To get the discrete features

#descrete_features = [feature for feature in df_cleaned.columns if
df_cleaned[feature].dtype == 'int64']
descrete_features = [feature for feature in numerical_features if
df_cleaned[feature].nunique()<20]
descrete_features
```

```
['education_num']
```

Segrigating the continuous numerical features

```
continuous_features = [feature for feature in numerical_features if
feature not in descrete_features]
print(continuous_features)
```

```
['age', 'fnlwgt', 'capital_gain', 'capital_loss', 'hours_per_week']
```

Statistical Analysis

Covariance

```
df_cleaned.cov()
```

```
                          age        fnlwgt  education_num
capital_gain  \
age                174.657324 -1.055561e+05       1.261498
7.906633e+03
fnlwgt         -105556.113482  1.116016e+10  -11329.864895 -
3.264128e+06
education_num        1.261498 -1.132986e+04       6.512925
2.432688e+03
capital_gain      7906.632876 -3.264128e+06    2432.687676
5.638187e+07
capital_loss       317.522494 -1.863604e+05      84.464128 -
9.770912e+04
hours_per_week      16.142885 -2.369313e+04       4.485939
7.561748e+03
```

```
                 capital_loss  hours_per_week
age                317.522494       16.142885
fnlwgt         -186360.438586   -23693.126340
education_num       84.464128        4.485939
capital_gain    -97709.118917     7561.747858
capital_loss    164089.629501      263.407537
hours_per_week     263.407537      144.157990
```

Correlation

```
df_cleaned.corr()
```

```
                     age    fnlwgt  education_num  capital_gain  
capital_loss  \
age             1.000000 -0.075606       0.037403      0.079676      
0.059312
fnlwgt         -0.075606  1.000000      -0.042024     -0.004115     -
0.004355
education_num   0.037403 -0.042024       1.000000      0.126949      
0.081704
capital_gain    0.079676 -0.004115       0.126949      1.000000     -
0.032124
capital_loss    0.059312 -0.004355       0.081704     -0.032124      
1.000000
hours_per_week  0.101735 -0.018680       0.146402      0.083875      
0.054159

                hours_per_week  
age                   0.101735  
fnlwgt               -0.018680  
education_num         0.146402  
capital_gain          0.083875  
capital_loss          0.054159  
hours_per_week        1.000000
```

Observation :

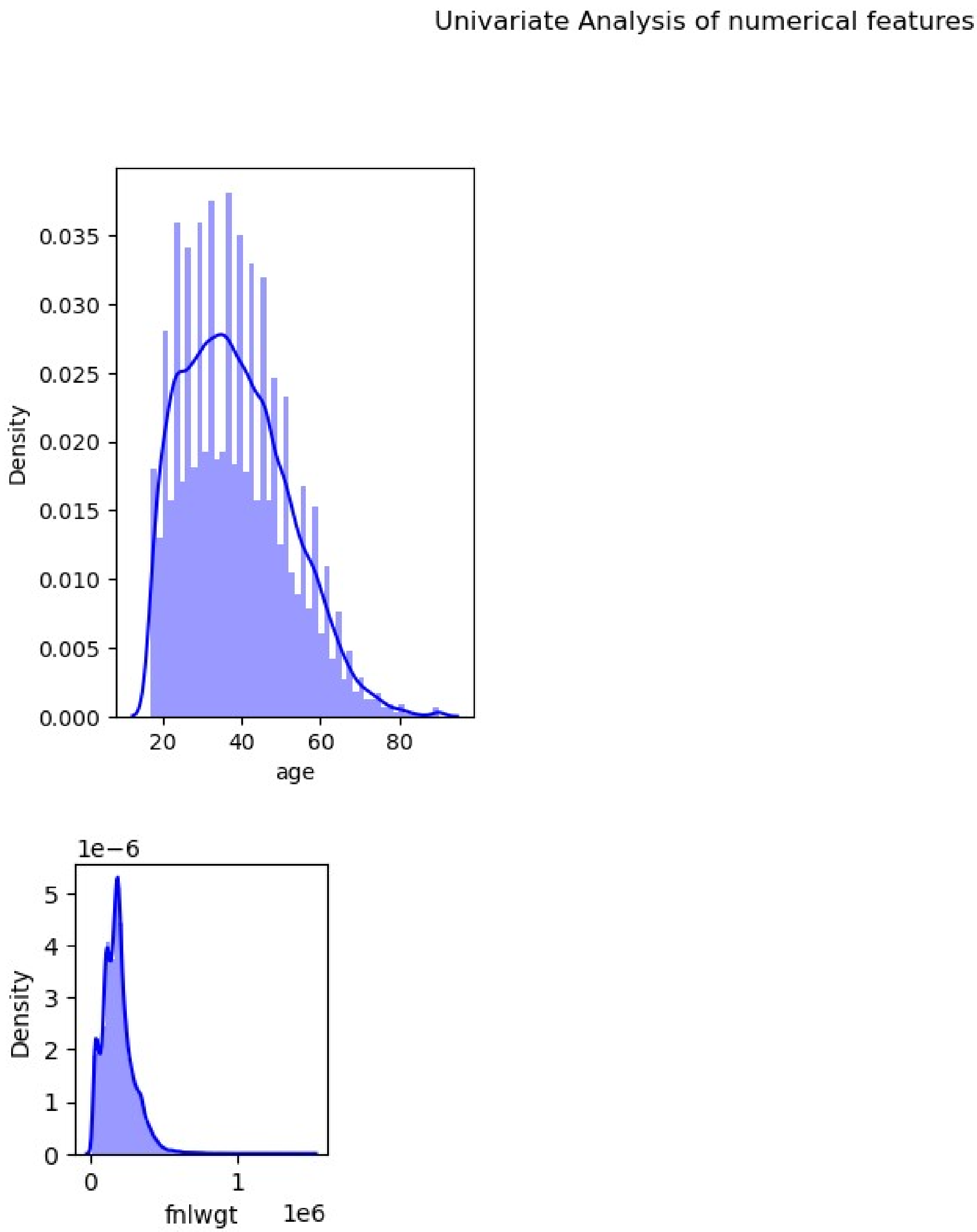
- There is no such correltion between the features

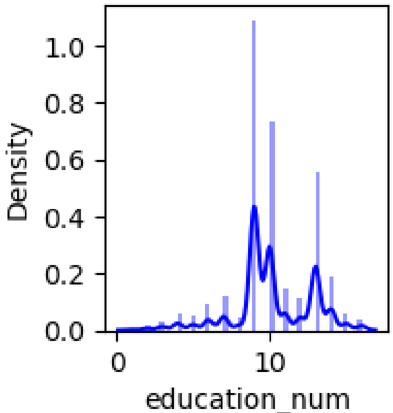
2. EDA

2.1 Univariate Analysis numerical feature

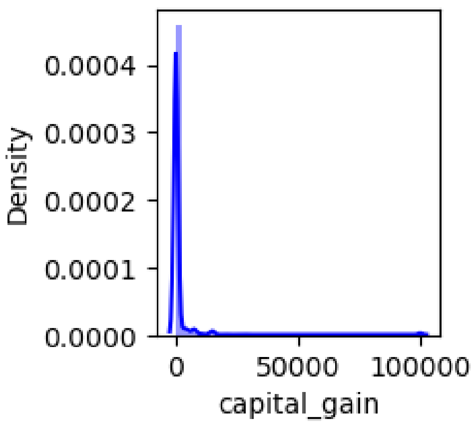
```
df_eda = df_cleaned.copy()

plt.figure(figsize=(10,10))
plt.suptitle('Univariate Analysis of numerical features')

for i in range(0,len(numerical_features)):
    plt.subplot(2,3,i+1)
    sns.distplot(x=df_eda[numerical_features[i]], kde=True, color='b')
```

```
plt.xlabel(numerical_features[i])
plt.show()
```

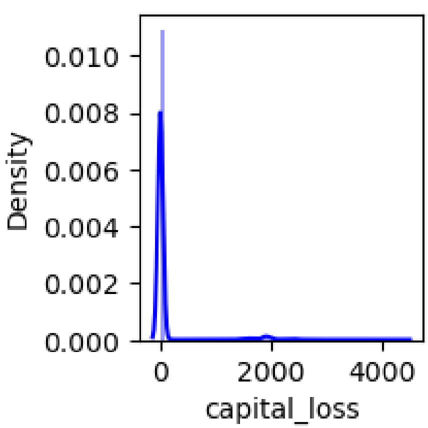




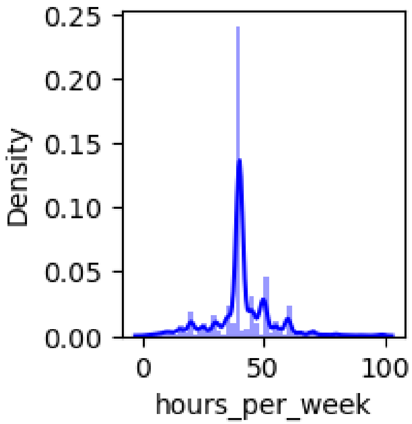
Density
1.0
0.8
0.6
0.4
0.2
0.0
0
10
education_num



Density
0.0004
0.0003
0.0002
0.0001
0.0000
0
50000
100000
capital_gain

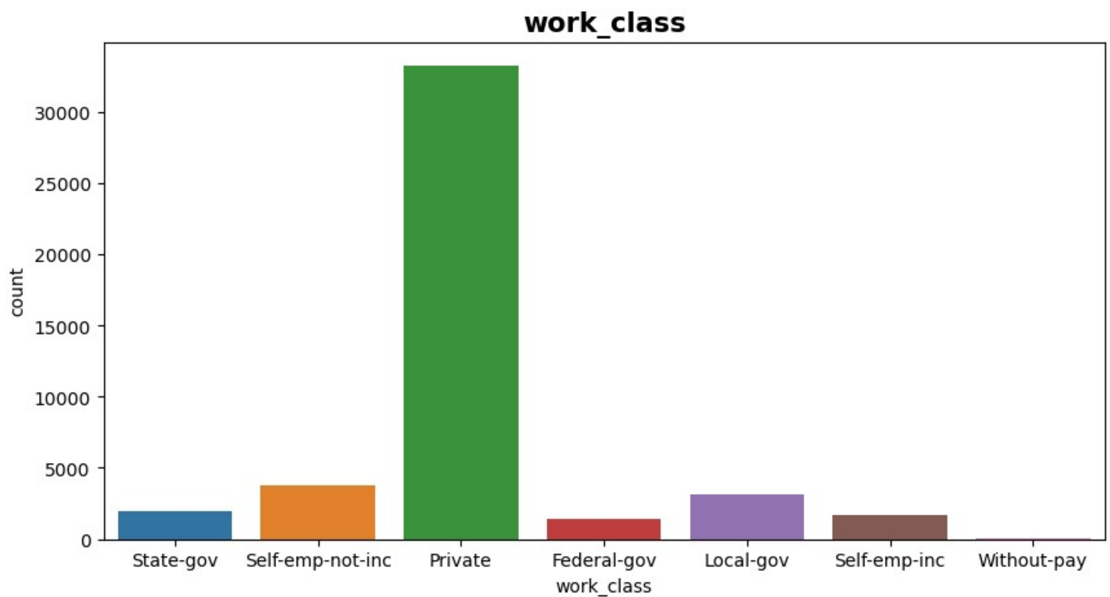


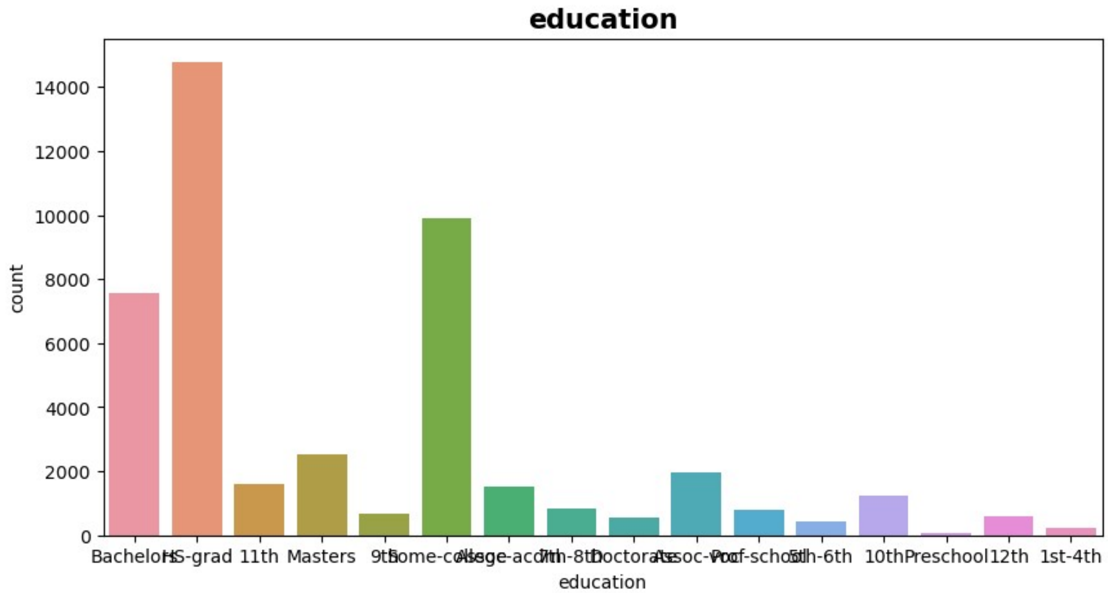
Density
0.010
0.008
0.006
0.004
0.002
0.000
0
2000
4000
capital_loss

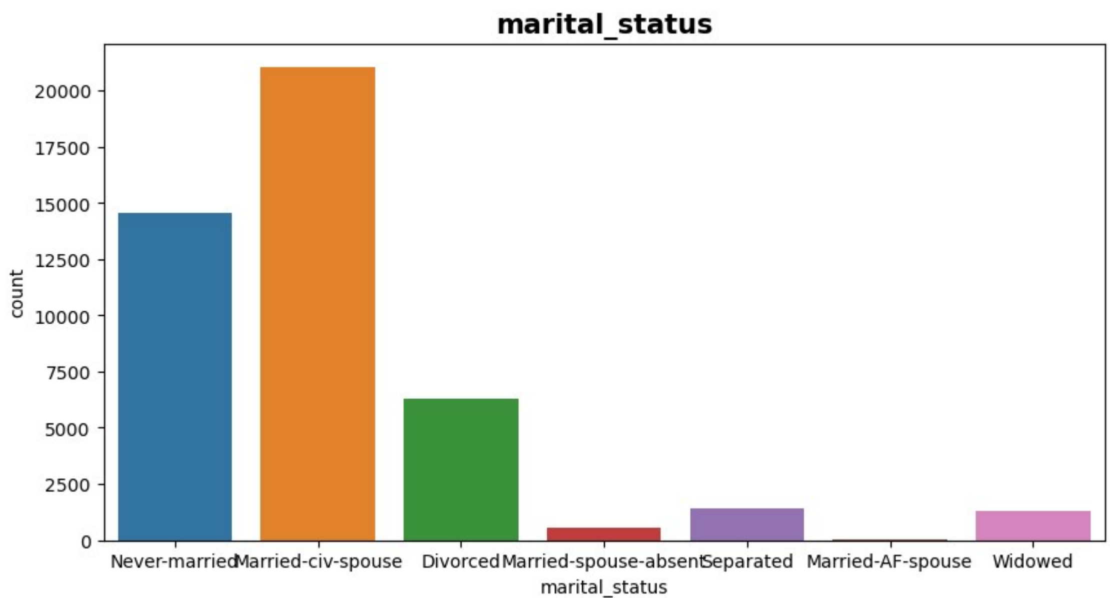


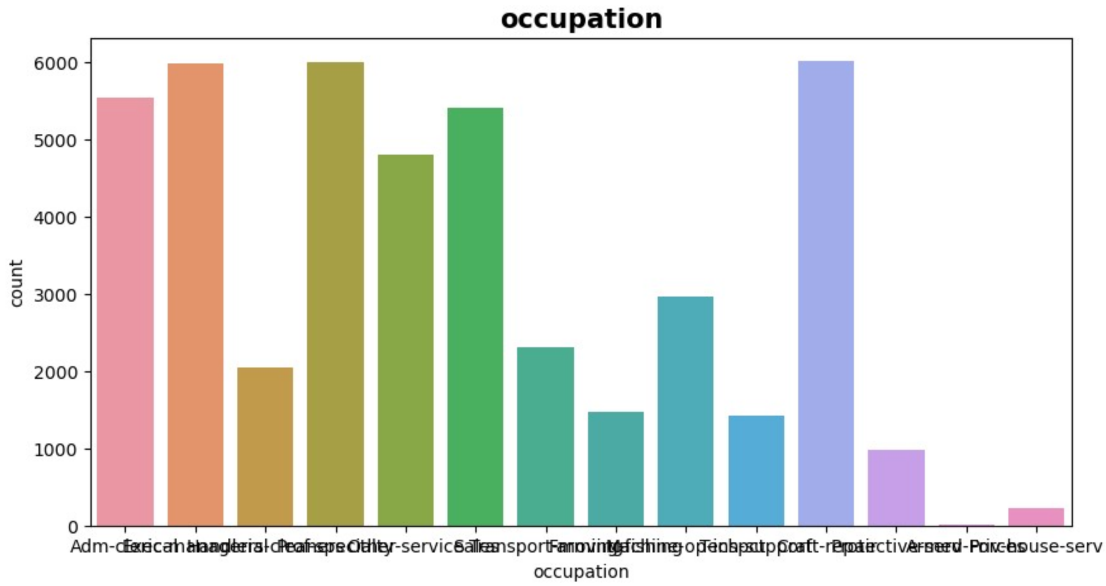
Countplot : To visualize the count of each value in a category

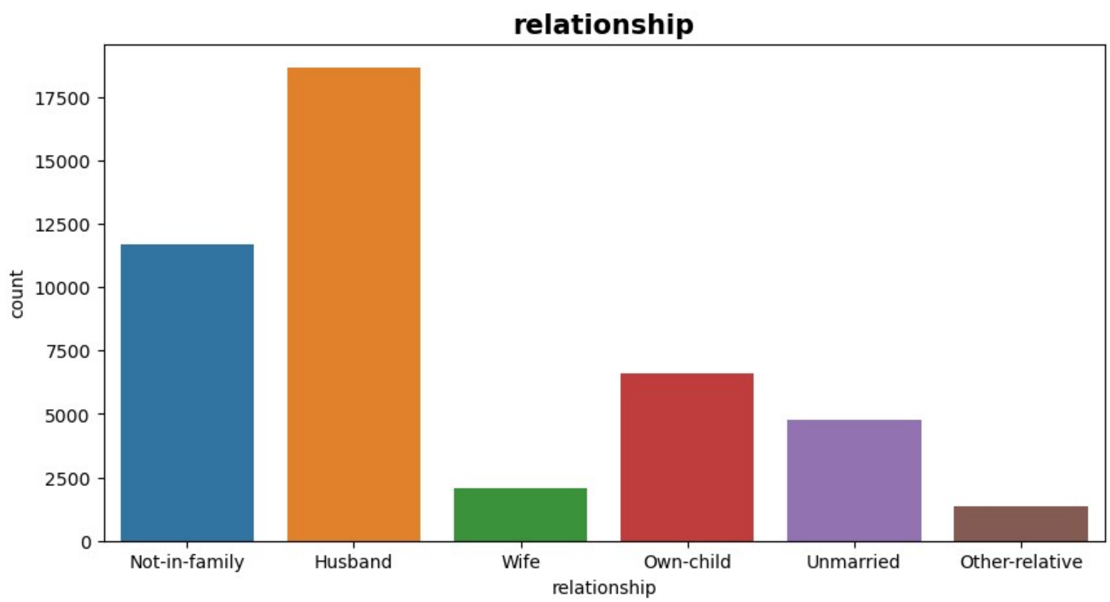
```
for feature in [feature for feature in categorical_features if feature
not in ['native_country']]:
    plt.figure(figsize=(10,5))
    sns.countplot(data=df_eda, x=feature)
    plt.title(feature, fontsize=15, weight='bold')
    plt.show()
```

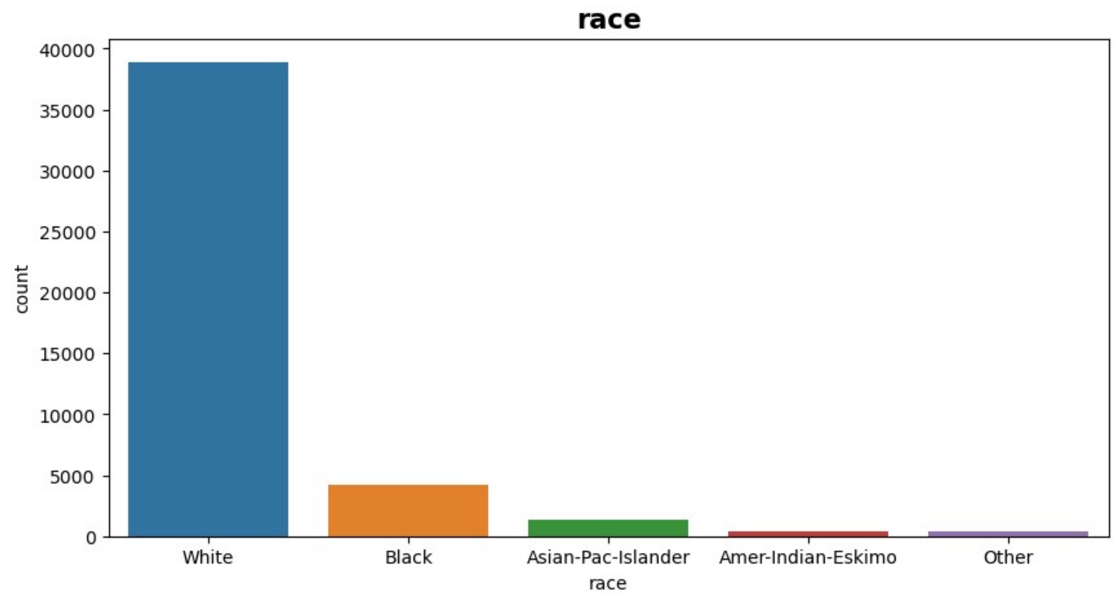


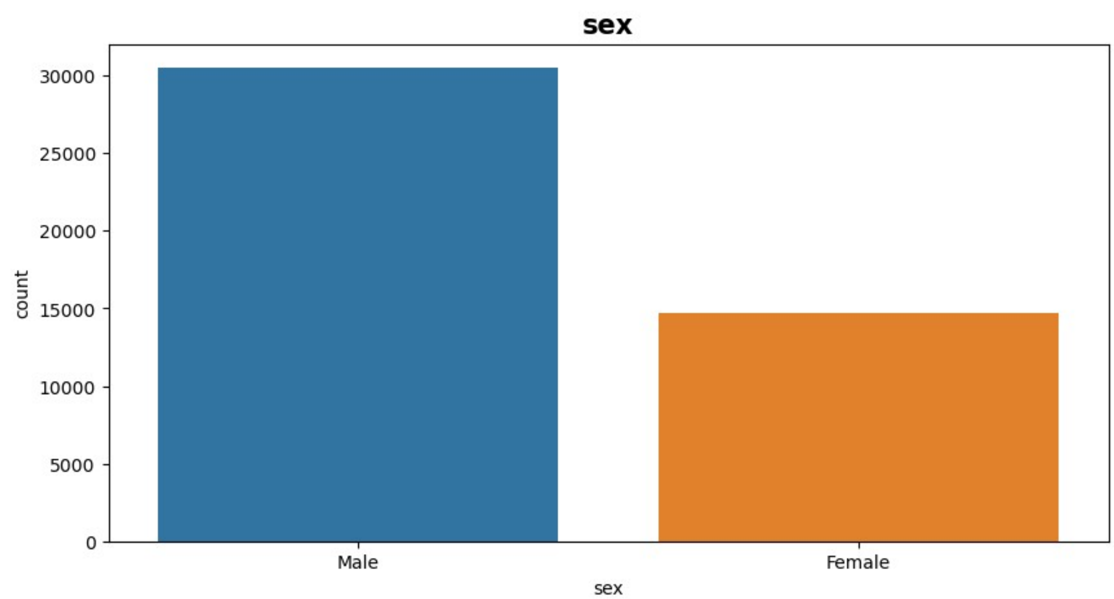

education
count
0
2000
4000
6000
8000
10000
12000
14000
Bachelors
HS-grad
11th
Masters
10th
Preschool
12th
1st-4th
education

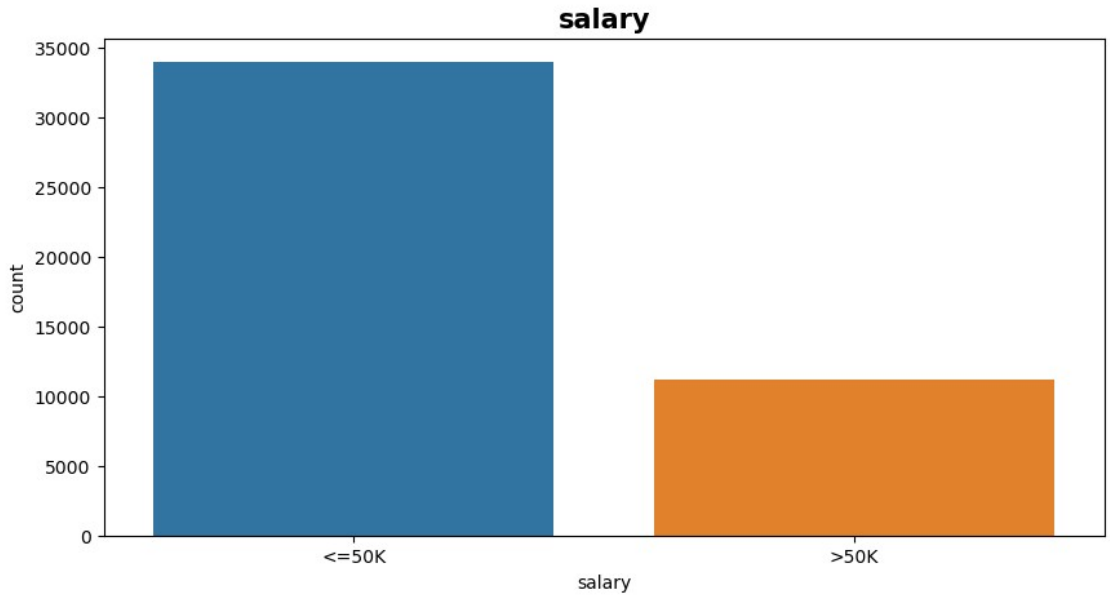

marital_status
count
0
2500
5000
7500
10000
12500
15000
17500
20000
Never-married
Married-civ-spouse
Divorced
Married-spouse-absent
Separated
Married-AF-spouse
Widowed
marital_status


occupation
count
6000
5000
4000
3000
2000
1000
0
occupation


relationship
count
17500
15000
12500
10000
7500
5000
2500
0
Not-in-family
Husband
Wife
Own-child
Unmarried
Other-relative
relationship


race
count
40000
35000
30000
25000
20000
15000
10000
5000
0
White
Black
Asian-Pac-Islander
Amer-Indian-Eskimo
Other
race

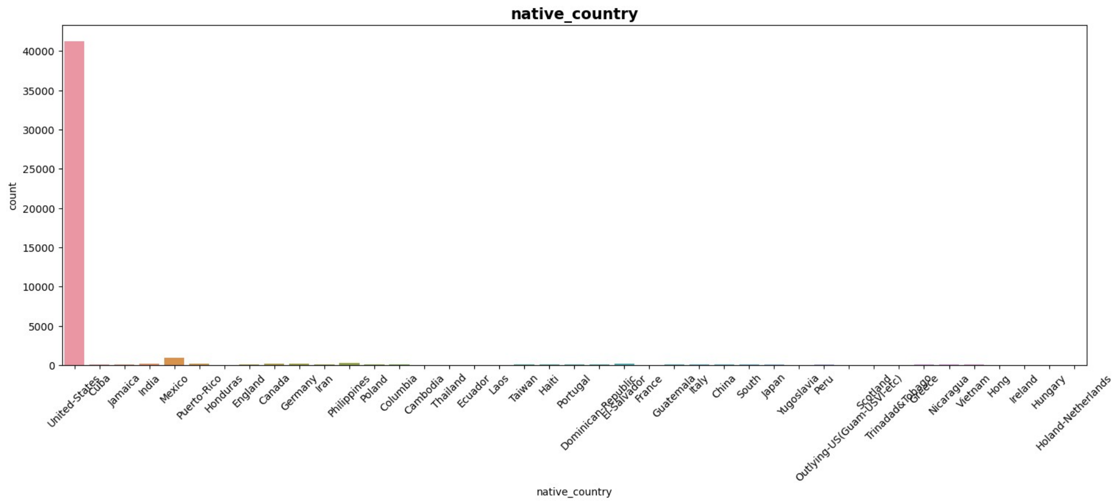

sex
count
30000
25000
20000
15000
10000
5000
0
Male
Female
sex



Observation :

- More people are doing private job
- People with Local-gov job are more than State-gov
- Most of the population are with edcation "HighSchool-graduate"
- Never married people are second highest after married.
- People with occupation as "Armed-Forces" are the least
- "Handlers-cleaners" are the most in the population.
- People who are not having family are the second highest in the population.
- White race people are the most followed by Black.
- Male are larger in numbers compared to female
- People with more than 50k income are more

```
plt.figure(figsize=(18,6))
sns.countplot(df_eda['native_country'])
plt.title('native_country', fontsize=15, weight='bold')
plt.xticks(rotation=45)
plt.show()
```

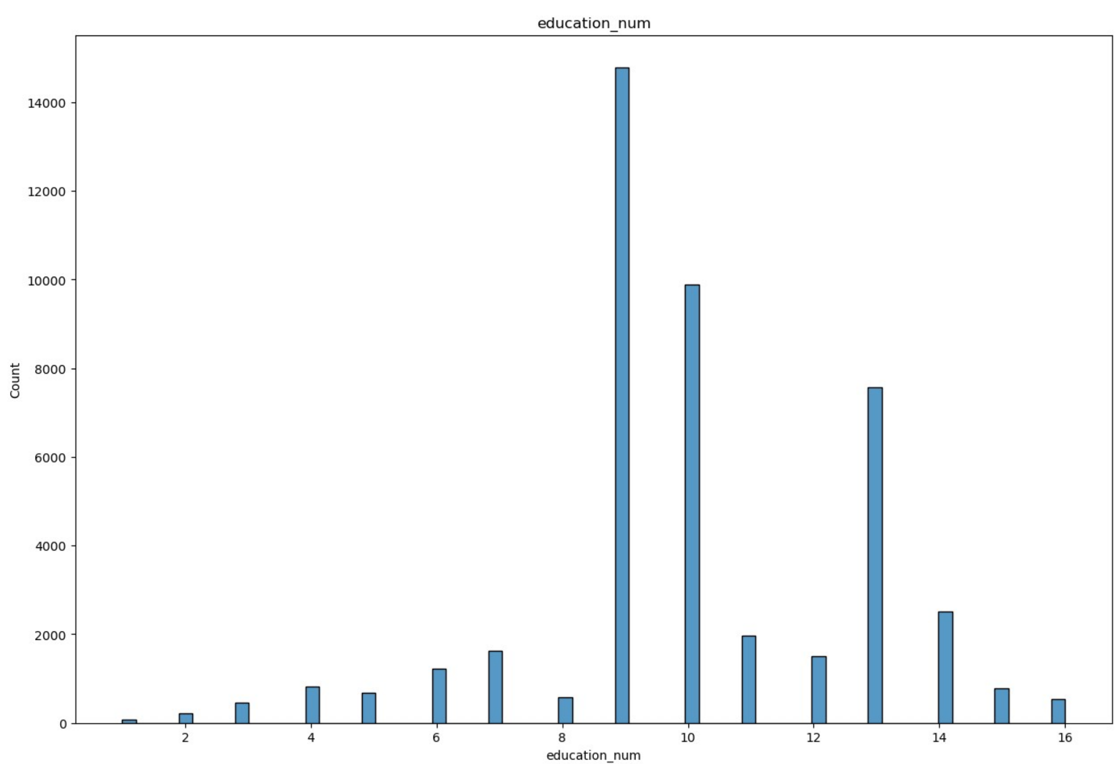


Observation :

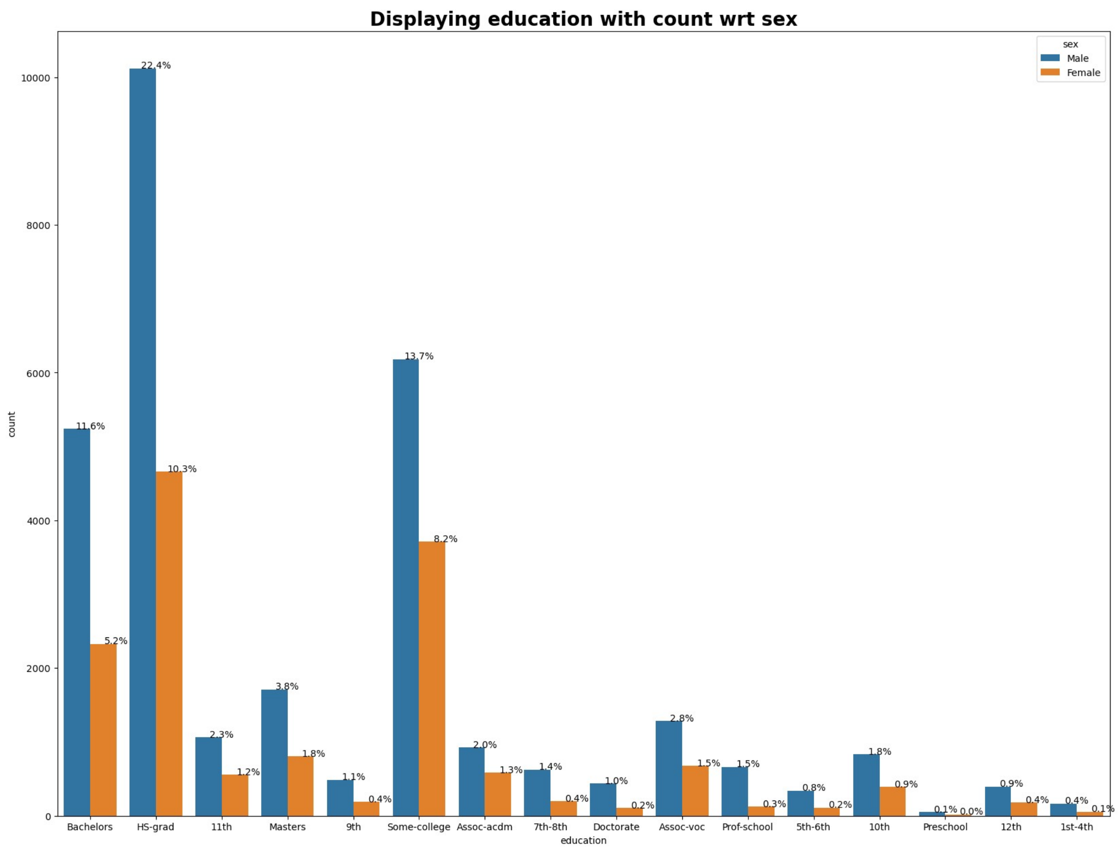
- Most people are staying in United States compared to other states.

Histogram for educational_num

```
for feature in descrete_features:
    plt.figure(figsize=(15,10))
    sns.histplot(data=df_eda, x=feature)
    plt.title(feature)
    plt.xticks=45
    plt.show()
```



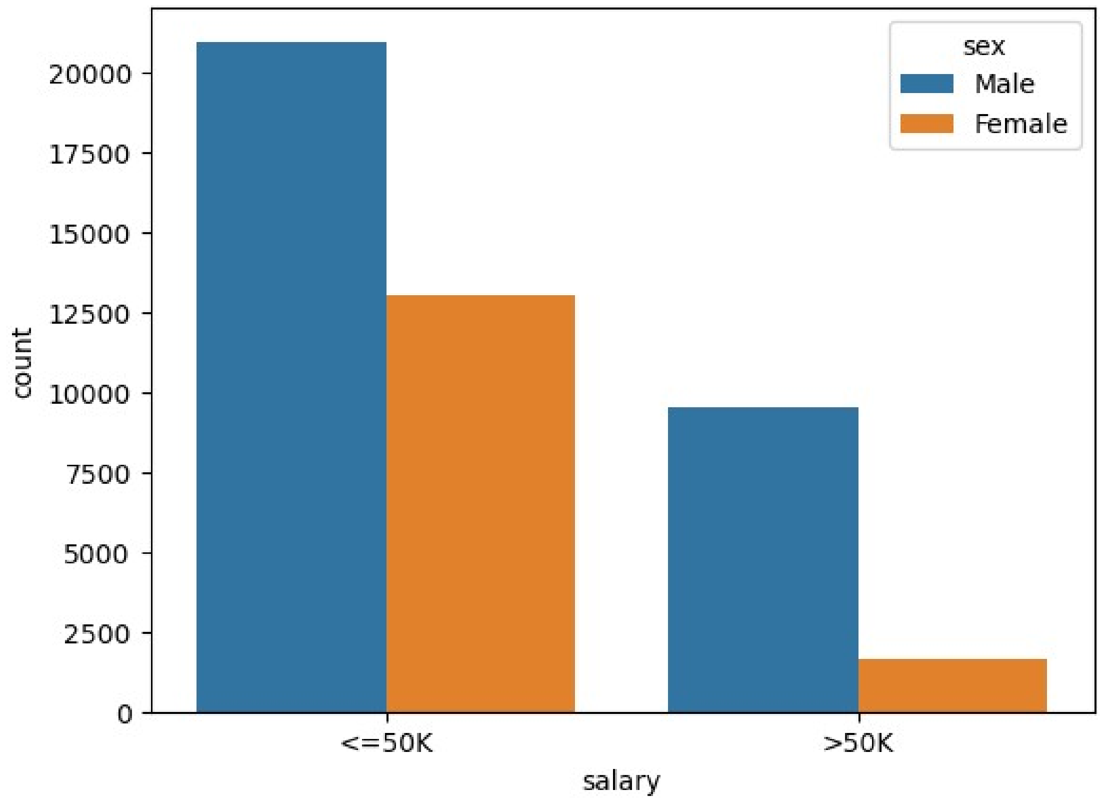
```
total = float(len(df_cleaned))
plt.figure(figsize=(20,15))
ax = sns.countplot(data=df_eda, x='education', hue='sex')
plt.title("Displaying education with count wrt sex", weight='bold',
fontsize=20)
for i in ax.patches:
    percentage = "{:.1f}%".format(100 * i.get_height()/total)
    x = i.get_x()+i.get_width()
    y = i.get_height()
    ax.annotate(percentage, (x,y), ha='center')
plt.show()
```



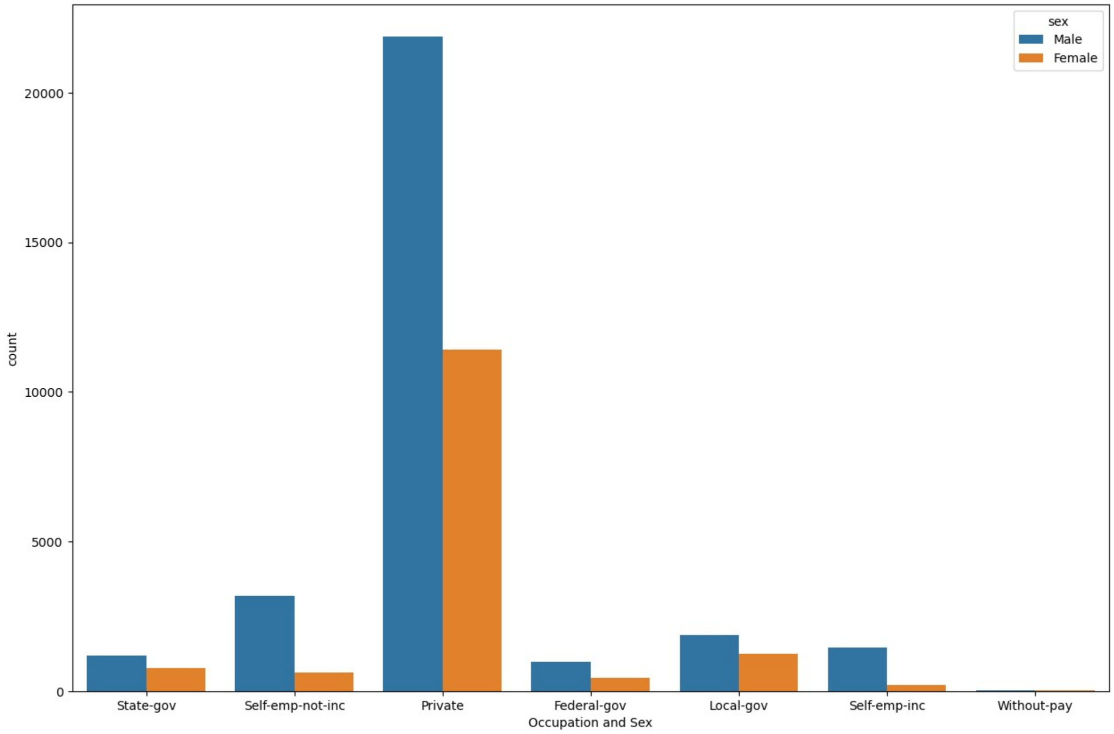
2.2 Bivariate Analysis

```
sns.countplot(data= df_eda, x='salary', hue='sex')
```

```
<AxesSubplot:xlabel='salary', ylabel='count'>
```




```
plt.figure(figsize=(15,10))
sns.countplot(data=df_eda, x='work_class', hue='sex')
plt.xlabel('Occupation and Sex')
plt.show()
```

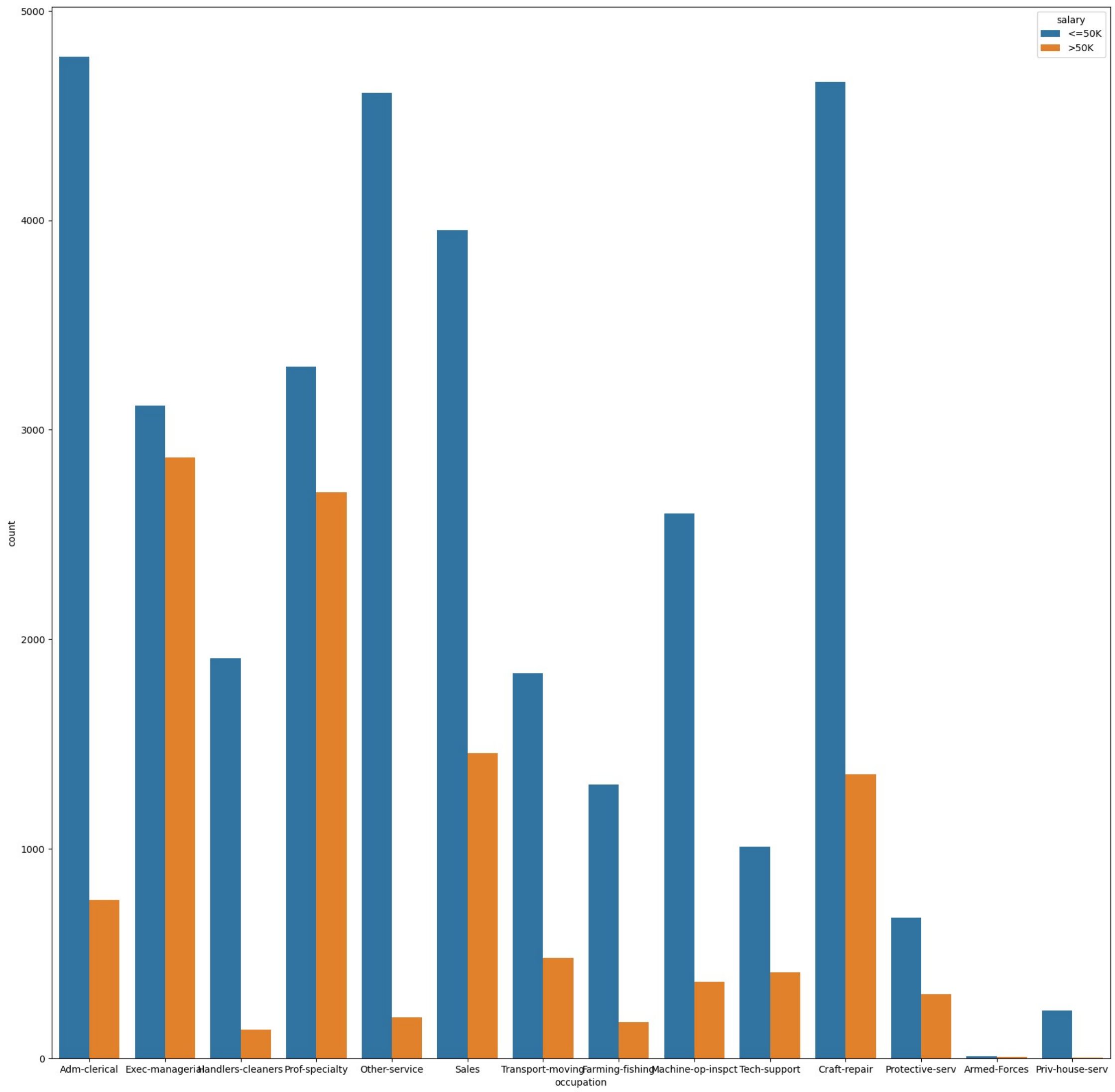


Observation :

- More male are into private jobs.

```
plt.figure(figsize=(20,20))
sns.countplot(data=df_eda, x='occupation', hue='salary')
```

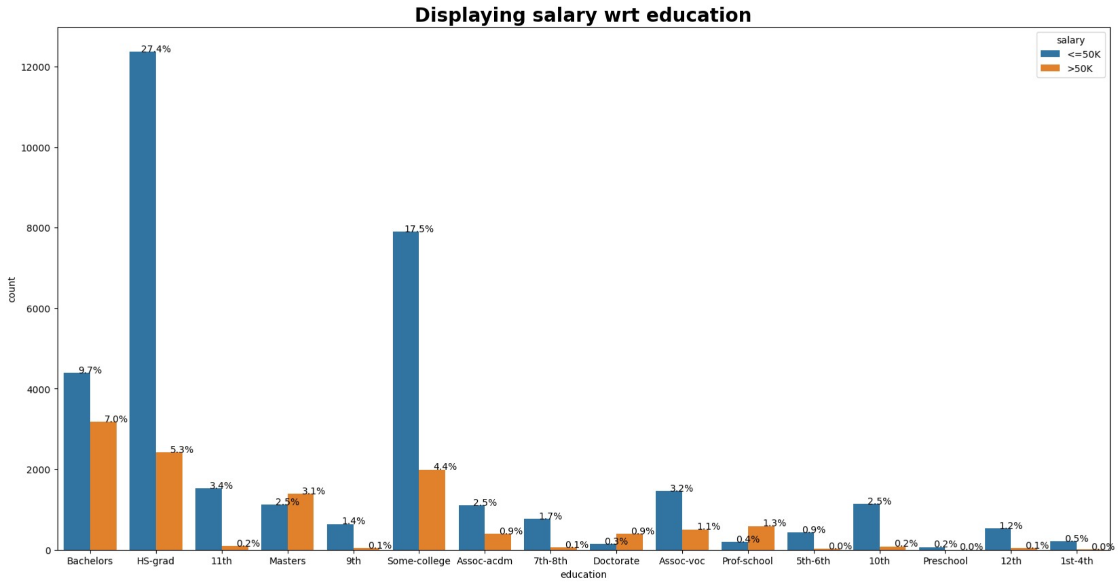
```
<AxesSubplot:xlabel='occupation', ylabel='count'>
```



Observation :

- "Exec-managers" are highest paid compared to other occupation.

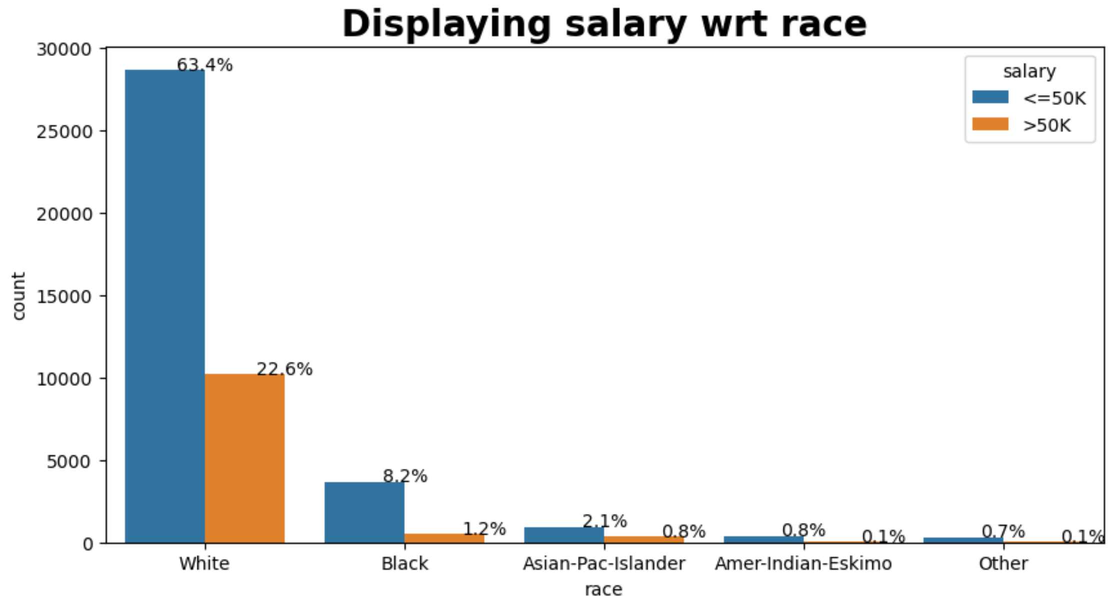
```
total = float(len(df_cleaned))
plt.figure(figsize=(20,10))
ax = sns.countplot(data=df_eda, x='education', hue='salary')
plt.title("Displaying salary wrt education", weight='bold',
fontsize=20)
for i in ax.patches:
    percentage = "{:.1f}%".format(100 * i.get_height()/total)
    x = i.get_x()+i.get_width()
    y = i.get_height()
    ax.annotate(percentage, (x,y), ha='center')
plt.show()
```



Observation :

- Persons with education of 'Batchelors' are largest wrt population count to have earning more than 50K.
- Persons with education as "Prof-school" have a higher earning ratio as more than 50K compared to less than 50K.

```
total = float(len(df_cleaned))
plt.figure(figsize=(10,5))
ax = sns.countplot(data=df_eda, x='race', hue='salary')
plt.title("Displaying salary wrt race", weight='bold', fontsize=20)
for i in ax.patches:
    percentage = "{:.1f}%".format(100 * i.get_height()/total)
    x = i.get_x()+i.get_width()
    y = i.get_height()
    ax.annotate(percentage, (x,y), ha='center')
plt.show()
```

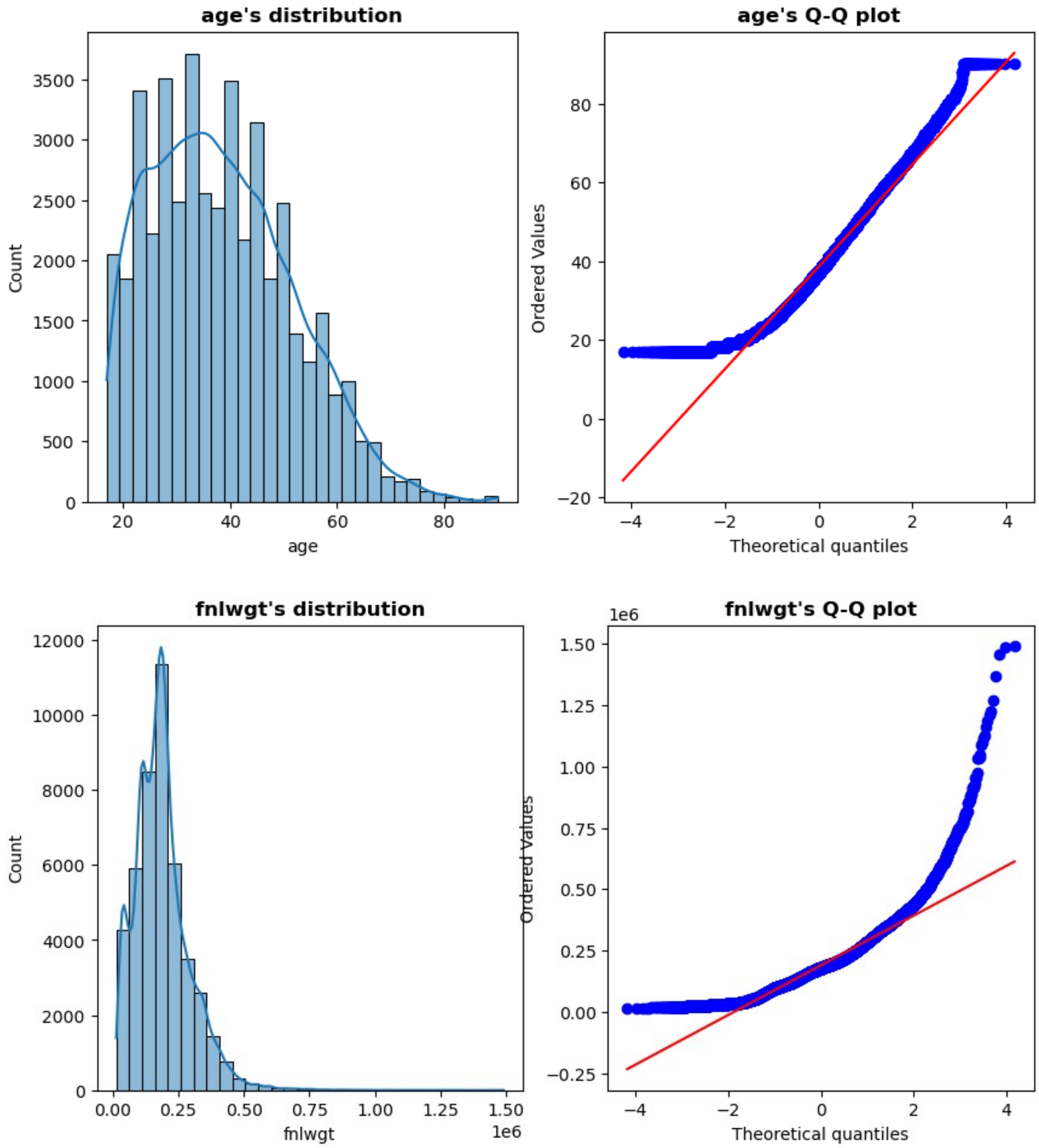


Observation :

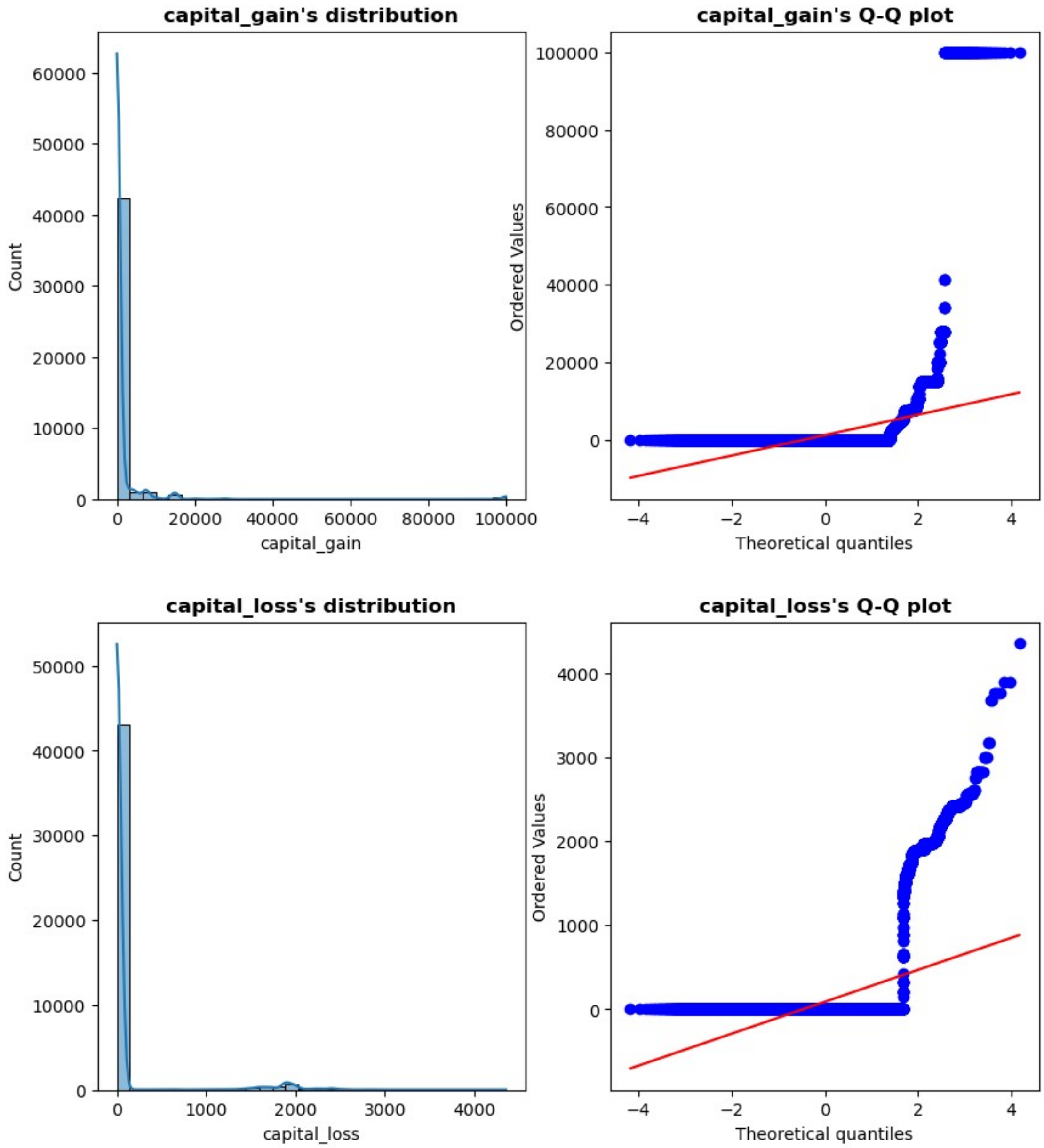
- White people have a greater ratio of income more than 50K to less than 50k (63:22) compared to other race.

histogram and Q-Q plot

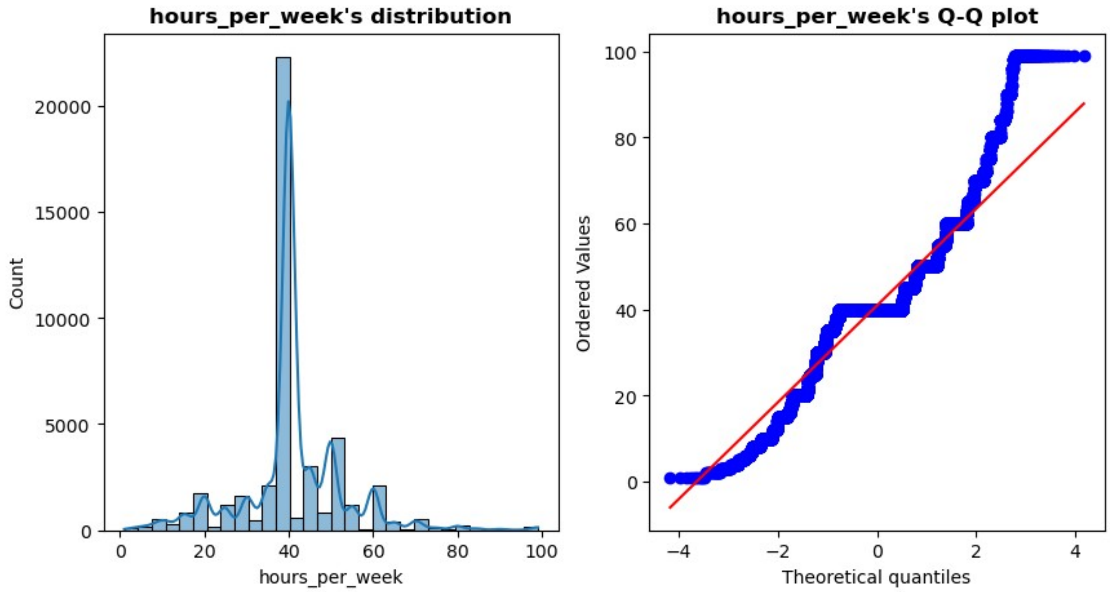
```
for feature in continuous_features:
    plt.figure(figsize=(10,5))
    plt.subplot(121)
    sns.histplot(data=df_eda, x=feature, kde=True, bins=30)
    plt.title(f"{feature}'s distribution", fontweight='bold')
    plt.subplot(122)
    stats.probplot(df_eda[feature], dist='norm', plot=plt)
    plt.title(f"{feature}'s Q-Q plot", fontweight='bold')
    plt.show()
```



age's distribution
Count
age
age's Q-Q plot
Ordered Values
Theoretical quantiles
fnlwgt's distribution
Count
fnlwgt
1e6
fnlwgt's Q-Q plot
1e6
Ordered Values
Theoretical quantiles



capital_gain's distribution
Count
capital_gain
capital_gain's Q-Q plot
Ordered Values
Theoretical quantiles
capital_loss's distribution
Count
capital_loss
capital_loss's Q-Q plot
Ordered Values
Theoretical quantiles

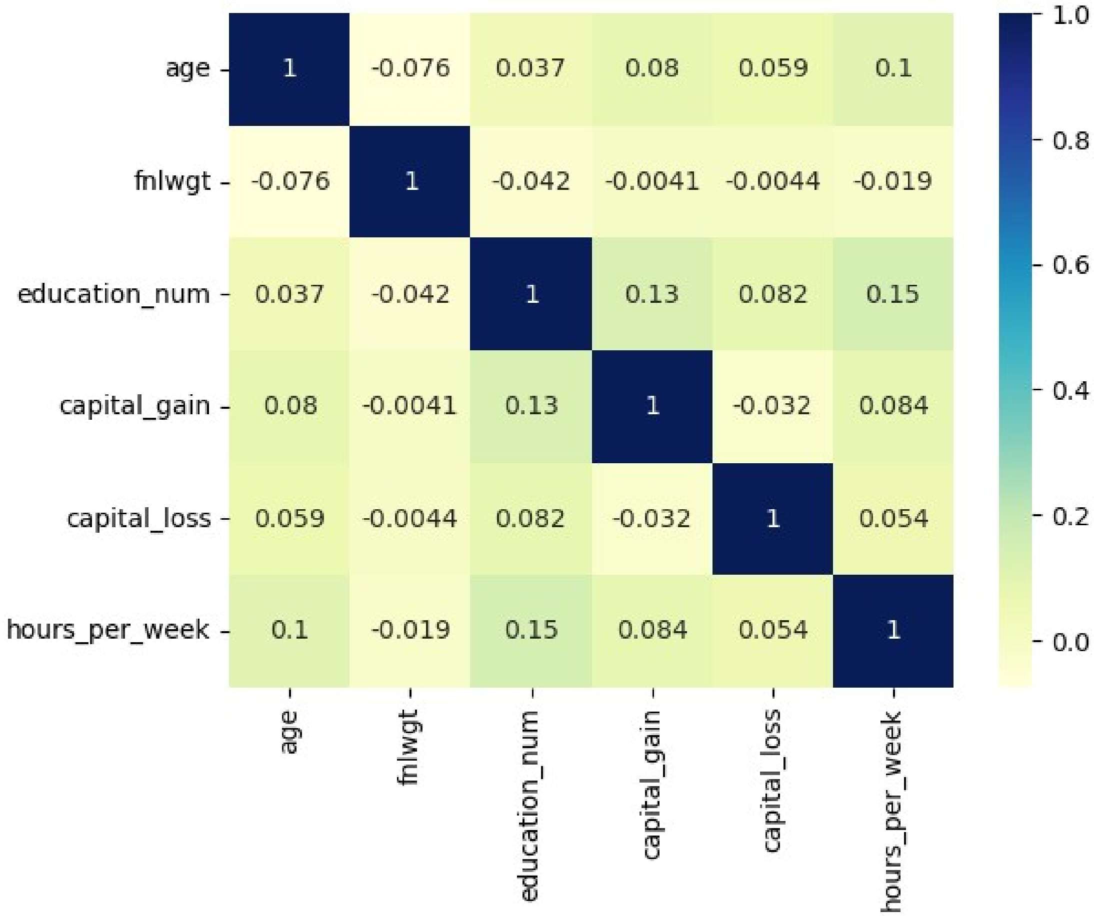


2.3 Multivariate Analysis

Heatmap : To check the correlation

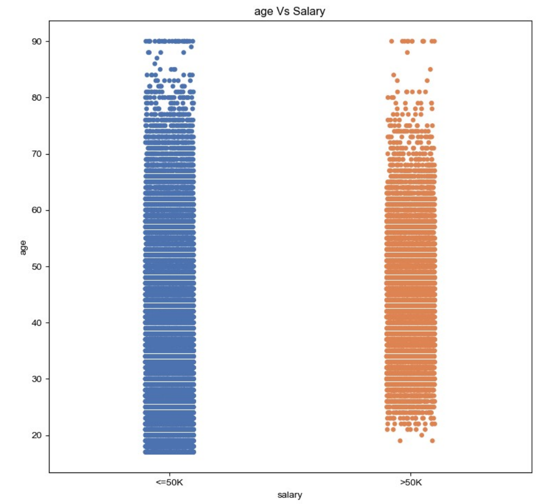
```
sns.heatmap(data=df_eda.corr(), annot=True, cmap='YlGnBu')
```

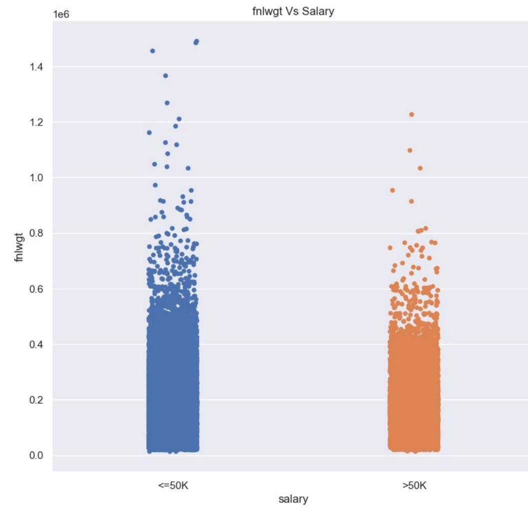
```
<AxesSubplot:>
```

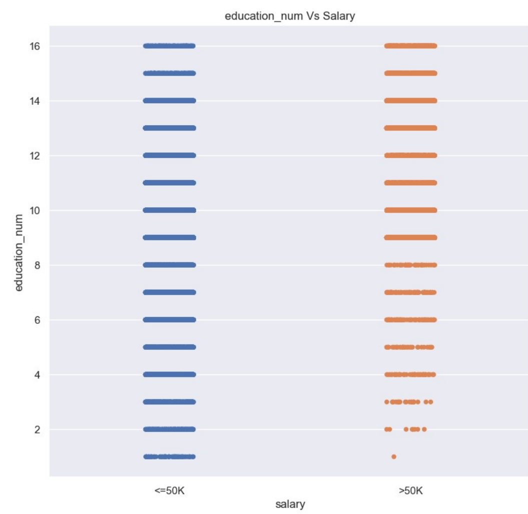


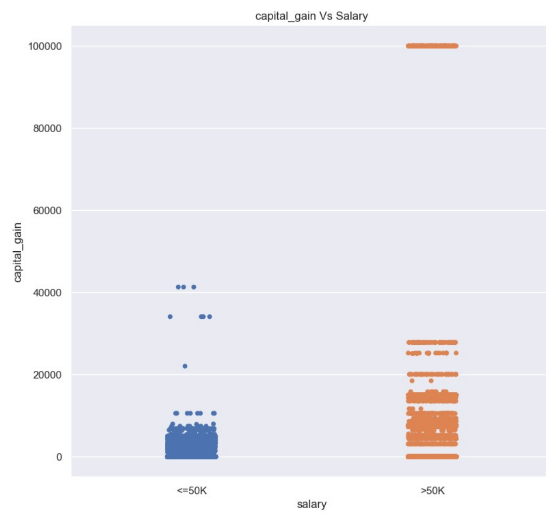
Continuous numerical feature vs Output(Dependent) features

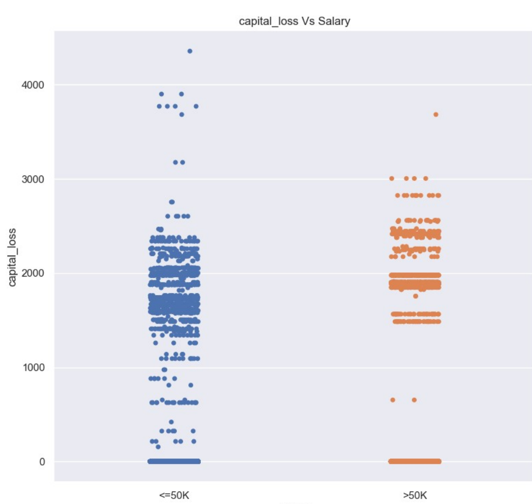
```
plt.figure(figsize=(20,60))
for i in enumerate(numerical_features):
    plt.subplot(6, 2, i[0]+1)
    sns.set(rc={'figure.figsize':(8,10)})
    sns.stripplot(data=df_eda, y=i[1], x='salary')
    plt.title("{} Vs Salary".format(i[1], fontsize=15,
fontweight='bold'))
```


age Vs Salary
age
20
30
40
50
60
70
80
90
<=50K
>50K
salary


fnlwgt Vs Salary
1e6
fnlwgt
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
<=50K
>50K
salary


education_num Vs Salary
education_num
2
4
6
8
10
12
14
16
<=50K
>50K
salary


capital_gain Vs Salary
capital_gain
0
20000
40000
60000
80000
100000
<=50K
>50K
salary


capital_loss Vs Salary
capital_loss
0
1000
2000
3000
4000
<=50K
>50K
salary


hours_per_week Vs Salary
hours_per_week
0
20
40
60
80
100
<=50K
>50K
salary

3. Preprocessing

3.2 Saving Data to MongoDB

Converting the data to key value pair to upload to mangoDB

```
## To reset the indexes of the records
## And dropping the index column

df_DB = df_eda.reset_index()
df_DB.drop('index', axis=1, inplace=True)
df_DB
```

```
        age        work_class  fnlwgt  education  education_num  \
0      39.0         State-gov   77516  Bachelors             13
1      50.0  Self-emp-not-inc   83311  Bachelors             13
2      38.0           Private  215646    HS-grad              9
3      53.0           Private  234721       11th              7
4      28.0           Private  338409  Bachelors             13
...     ...               ...     ...        ...            ...
45188  33.0           Private  245211  Bachelors             13
45189  39.0           Private  215419  Bachelors             13
45190  38.0           Private  374983  Bachelors             13
45191  44.0           Private   83891  Bachelors             13
45192  35.0      Self-emp-inc  182148  Bachelors             13

           marital_status         occupation   relationship  \
0           Never-married       Adm-clerical  Not-in-family
1      Married-civ-spouse    Exec-managerial        Husband
2                Divorced  Handlers-cleaners  Not-in-family
3      Married-civ-spouse  Handlers-cleaners        Husband
4      Married-civ-spouse     Prof-specialty           Wife
...                   ...                ...            ...
45188       Never-married     Prof-specialty      Own-child
45189            Divorced     Prof-specialty  Not-in-family
45190  Married-civ-spouse     Prof-specialty        Husband
45191            Divorced       Adm-clerical      Own-child
45192  Married-civ-spouse    Exec-managerial        Husband

                     race     sex  capital_gain  capital_loss
hours_per_week  \
0                   White    Male          2174             0
40.0
1                   White    Male             0             0
13.0
2                   White    Male             0             0
40.0
3                   Black    Male             0             0
40.0
4                   Black  Female             0             0
```

```
40.0
...                   ...     ...           ...           ...
...
45188               White    Male             0             0
40.0
45189               White  Female             0             0
36.0
45190               White    Male             0             0
50.0
45191  Asian-Pac-Islander    Male          5455             0
40.0
45192               White    Male             0             0
60.0

      native_country salary
0      United-States  <=50K
1      United-States  <=50K
2      United-States  <=50K
3      United-States  <=50K
4               Cuba  <=50K
...              ...    ...
45188  United-States  <=50K
45189  United-States  <=50K
45190  United-States  <=50K
45191  United-States  <=50K
45192  United-States   >50K

[45193 rows x 15 columns]
```

```
# Creating connection

import pymongo


client = pymongo.MongoClient("mongodb://MongoDB:MongoDB@ac-ialq2ju-shard-00-00.i7o85x8.mongodb.net:27017,ac-ialq2ju-shard-00-01.i7o85x8.mongodb.net:27017,ac-ialq2ju-shard-00-02.i7o85x8.mongodb.net:27017/?ssl=true&replicaSet=atlas-8t92h8-shard-0&authSource=admin&retryWrites=true&w=majority")


database = client['CencusIncome']
collection = database['Income']
```

```
# Convering the data to json type

import json

data = df_DB.to_json(orient="records")
json_data = json.loads(data)
```

```
# Inserting the data to mongoDB

collection.insert_many(json_data)
```

```
<pymongo.results.InsertManyResult at 0x24be9ed77c0>
```

Retriving data from mongoDB

```
data_mongoDB = collection.find()
data_mongo = pd.DataFrame(data_mongoDB)
data_mongo
```

```
                            _id   age        work_class  fnlwgt
education  \
0      637629b8517e19b4193c64a7  39.0         State-gov   77516
Bachelors
1      637629b8517e19b4193c64a8  50.0  Self-emp-not-inc   83311
Bachelors
2      637629b8517e19b4193c64a9  38.0           Private  215646    HS-
grad
3      637629b8517e19b4193c64aa  53.0           Private  234721
11th
4      637629b8517e19b4193c64ab  28.0           Private  338409
Bachelors
...                         ...   ...               ...     ...
...
90381  6378dbba5508bddd2142698b  33.0           Private  245211
Bachelors
90382  6378dbba5508bddd2142698c  39.0           Private  215419
Bachelors
90383  6378dbba5508bddd2142698d  38.0           Private  374983
Bachelors
90384  6378dbba5508bddd2142698e  44.0           Private   83891
Bachelors
90385  6378dbba5508bddd2142698f  35.0      Self-emp-inc  182148
Bachelors

       education_num      marital_status         occupation
relationship  \
0                 13       Never-married       Adm-clerical  Not-in-
family
1                 13  Married-civ-spouse    Exec-managerial
Husband
2                  9            Divorced  Handlers-cleaners  Not-in-
family
3                  7  Married-civ-spouse  Handlers-cleaners
Husband
4                 13  Married-civ-spouse     Prof-specialty
Wife
...              ...                 ...                ...
...
```

```
90381             13       Never-married     Prof-specialty      Own-
child
90382             13            Divorced     Prof-specialty  Not-in-
family
90383             13  Married-civ-spouse     Prof-specialty
Husband
90384             13            Divorced       Adm-clerical      Own-
child
90385             13  Married-civ-spouse    Exec-managerial
Husband

                      race     sex  capital_gain  capital_loss
hours_per_week  \
0                    White    Male          2174             0
40.0
1                    White    Male             0             0
13.0
2                    White    Male             0             0
40.0
3                    Black    Male             0             0
40.0
4                    Black  Female             0             0
40.0
...                    ...     ...           ...           ...
...
90381                White    Male             0             0
40.0
90382                White  Female             0             0
36.0
90383                White    Male             0             0
50.0
90384   Asian-Pac-Islander    Male          5455             0
40.0
90385                White    Male             0             0
60.0

      native_country salary
0      United-States  <=50K
1      United-States  <=50K
2      United-States  <=50K
3      United-States  <=50K
4               Cuba  <=50K
...              ...    ...
90381  United-States  <=50K
90382  United-States  <=50K
90383  United-States  <=50K
90384  United-States  <=50K
90385  United-States   >50K

[90386 rows x 16 columns]
```

```python
### Making a copy of data

data_db = data_mongo.copy()

data_db.drop(['_id'], axis=1, inplace=True)
data_db.head()
```

```
    age        work_class  fnlwgt  education  education_num  \
0  39.0         State-gov   77516  Bachelors             13
1  50.0  Self-emp-not-inc   83311  Bachelors             13
2  38.0           Private  215646    HS-grad              9
3  53.0           Private  234721       11th              7
4  28.0           Private  338409  Bachelors             13

       marital_status         occupation   relationship   race     sex
\
0       Never-married       Adm-clerical  Not-in-family  White    Male

1  Married-civ-spouse    Exec-managerial        Husband  White    Male

2            Divorced  Handlers-cleaners  Not-in-family  White    Male

3  Married-civ-spouse  Handlers-cleaners        Husband  Black    Male

4  Married-civ-spouse     Prof-specialty           Wife  Black  Female


   capital_gain  capital_loss  hours_per_week native_country salary
0          2174             0            40.0  United-States  <=50K
1             0             0            13.0  United-States  <=50K
2             0             0            40.0  United-States  <=50K
3             0             0            40.0  United-States  <=50K
4             0             0            40.0           Cuba  <=50K
```

3.3 Feature Selection

```python
# Dropping unnecessary columns

data_db.drop(['marital_status', 'relationship', 'race'], axis=1,
inplace=True)
```

3.4 Feature Encoding

```python
data_db = pd.get_dummies(data_db)

data_db.head()
```

```
    age  fnlwgt  education_num  capital_gain  capital_loss
hours_per_week  \
0  39.0   77516             13          2174             0
40.0
1  50.0   83311             13             0             0
```

```
13.0
2  38.0  215646              9             0             0
40.0
3  53.0  234721              7             0             0
40.0
4  28.0  338409             13             0             0
40.0

   work_class_Federal-gov  work_class_Local-gov  work_class_Private  \
0                       0                     0                   0
1                       0                     0                   0
2                       0                     0                   1
3                       0                     0                   1
4                       0                     0                   1

   work_class_Self-emp-inc  ...  native_country_Scotland  \
0                        0  ...                        0
1                        0  ...                        0
2                        0  ...                        0
3                        0  ...                        0
4                        0  ...                        0

   native_country_South  native_country_Taiwan
native_country_Thailand  \
0                     0                      0
0
1                     0                      0
0
2                     0                      0
0
3                     0                      0
0
4                     0                      0
0

   native_country_Trinadad&Tobago  native_country_United-States  \
0                               0                             1
1                               0                             1
2                               0                             1
3                               0                             1
4                               0                             0

   native_country_Vietnam  native_country_Yugoslavia  salary_<=50K  \
0                       0                          0             1
1                       0                          0             1
2                       0                          0             1
3                       0                          0             1
4                       0                          0             1

   salary_>50K
```

```
0            0
1            0
2            0
3            0
4            0

[5 rows x 88 columns]
```

Segragating the features(Independent variables) and Labels (Dependent Variables)

```
#X = data_db.iloc[:,:-2]
X = data_db[['age', 'fnlwgt', 'education_num', 'capital_gain',
'capital_loss', 'hours_per_week']]

X

        age  fnlwgt  education_num  capital_gain  capital_loss
hours_per_week
0      39.0   77516             13          2174             0
40.0
1      50.0   83311             13             0             0
13.0
2      38.0  215646              9             0             0
40.0
3      53.0  234721              7             0             0
40.0
4      28.0  338409             13             0             0
40.0
...     ...     ...            ...           ...           ...
...
90381  33.0  245211             13             0             0
40.0
90382  39.0  215419             13             0             0
36.0
90383  38.0  374983             13             0             0
50.0
90384  44.0   83891             13          5455             0
40.0
90385  35.0  182148             13             0             0
60.0

[90386 rows x 6 columns]

y = data_db['>50K']
y

0        0
1        0
2        0
3        0
4        0
        ..
```

```
90381    0
90382    0
90383    0
90384    0
90385    1
Name: >50K, Length: 90386, dtype: uint8
```

3.5 Train-Test split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=66)

print(X_train.shape, X_test.shape)

(60558, 6) (29828, 6)

print(y_train.shape, y_test.shape)

(60558,) (29828,)
```

4. Model Building

4.1 Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

dtc.fit(X_train, y_train)

DecisionTreeClassifier()
```

Pickling

```
## Saving model to pickle file

import pickle

with open("Cencus_Income_DTC.pkl", "wb") as f:
    pickle.dump(dtc, f)

## Loading data from pickle file

model_dtc = pickle.load(open('Cencus_Income_DTC.pkl', 'rb'))

## Model training

model_dtc.fit(X_train, y_train)

DecisionTreeClassifier()
```

5.1 Accuracy Score

```
from sklearn.metrics import accuracy_score

print("Decision Tree Classifier training accuracy score is : {}
%".format(round(model_dtc.score(X_train, y_train)*100)))
y_predict_dtc = model_dtc.predict(X_test)
print("Decision Tree Classifier model's accuracy score is : {}
%".format(round(accuracy_score(y_test, y_predict_dtc)*100)))
```

```
Decision Tree Classifier training accuracy score is : 100%
Decision Tree Classifier model's accuracy score is : 92%
```

Observation :

- It is an overfitted model

5.2 Roc-auc score

```
from sklearn.metrics import roc_auc_score

y_train_predict_roc = model_dtc.predict_proba(X_train)
print("Decision Tree Classifier model's training roc-auc score is : {}
%".format(round(roc_auc_score(y_train,
y_train_predict_roc[:,1])*100)))
y_test_predict_roc = model_dtc.predict_proba(X_test)
print("Decision Tree Classifier model's roc-auc accuracy score is : {}
%".format(round(roc_auc_score(y_test, y_test_predict_roc[:,1])*100)))
```

```
Decision Tree Classifier model's training roc-auc score is : 100%
Decision Tree Classifier model's roc-auc accuracy score is : 90%
```

5.3 Confusion_matrix

```
from sklearn.metrics import confusion_matrix

conf_mat = confusion_matrix(y_test, y_predict_dtc)
conf_mat
```

```
array([[21184,  1228],
       [ 1143,  6273]], dtype=int64)
```

```
true_positive = conf_mat[0][0]
false_positive = conf_mat[0][1]
false_negative = conf_mat[1][0]
true_negative = conf_mat[1][0]
print('True Positive:',true_positive, '\nTrue
Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse
Positive:',false_positive)
```

```
True Positive: 21184
True Negative: 1143
False Negative: 1143
False Positive: 1228
```

Classification Report

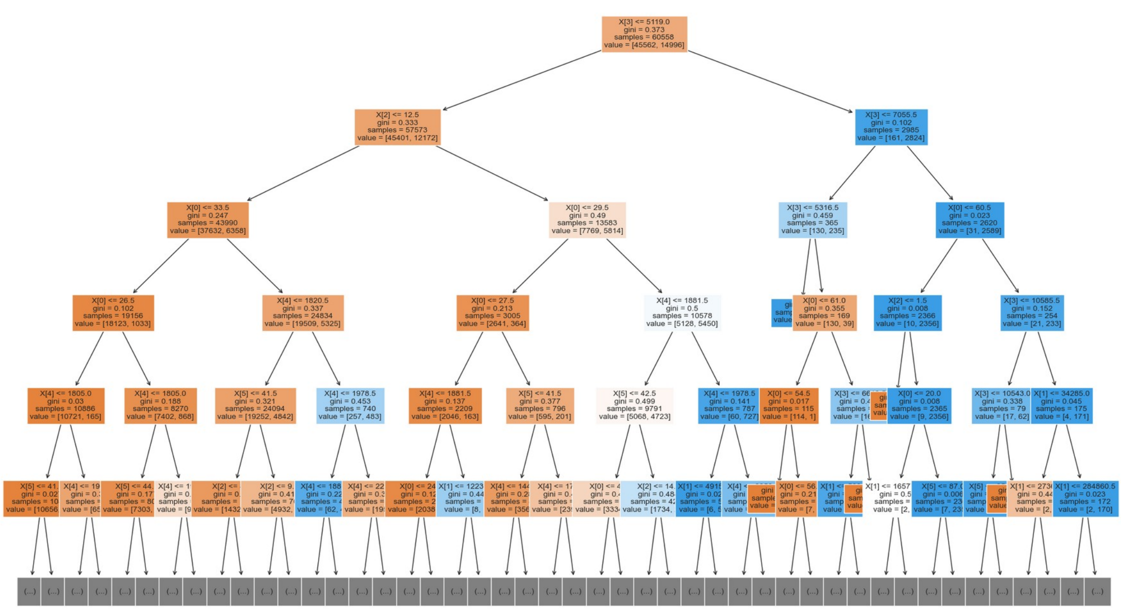
```
from sklearn.metrics import classification_report

class_reprt_log_reg = classification_report(y_test, y_predict_dtc)
print(class_reprt_log_reg)

              precision    recall  f1-score   support

           0       0.95      0.95      0.95     22412
           1       0.84      0.85      0.84      7416

    accuracy                           0.92     29828
   macro avg       0.89      0.90      0.89     29828
weighted avg       0.92      0.92      0.92     29828
```

```
# Plotting Decision Tree

from sklearn import tree
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(25,15))
tree.plot_tree(dtc, max_depth=5, filled=True, fontsize=10)

[Text(0.5740489130434783, 0.9285714285714286, 'X[3] <= 5119.0\ngini = 0.373\nsamples = 60558\nvalue = [45562, 14996]'),
 Text(0.34782608695652173, 0.7857142857142857, 'X[2] <= 12.5\ngini = 0.333\nsamples = 57573\nvalue = [45401, 12172]'),
 Text(0.17391304347826086, 0.6428571428571429, 'X[0] <= 33.5\ngini = 0.247\nsamples = 43990\nvalue = [37632, 6358]'),
 Text(0.08695652173913043, 0.5, 'X[0] <= 26.5\ngini = 0.102\nsamples = 19156\nvalue = [18123, 1033]'),
 Text(0.043478260869565216, 0.35714285714285715, 'X[4] <= 1805.0\ngini = 0.03\nsamples = 10886\nvalue = [10721, 165]'),
 Text(0.021739130434782608, 0.21428571428571427, 'X[5] <= 41.5\ngini = 0.027\nsamples = 10803\nvalue = [10656, 147]'),
 Text(0.010869565217391304, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.03260869565217391, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.06521739130434782, 0.21428571428571427, 'X[4] <= 1938.0\ngini = 0.34\nsamples = 83\nvalue = [65, 18]'),
 Text(0.05434782608695652, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.07608695652173914, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.13043478260869565, 0.35714285714285715, 'X[4] <= 1805.0\ngini = 0.188\nsamples = 8270\nvalue = [7402, 868]'),
 Text(0.10869565217391304, 0.21428571428571427, 'X[5] <= 44.5\ngini = 0.177\nsamples = 8098\nvalue = [7303, 795]'),
 Text(0.09782608695652174, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.11956521739130435, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.15217391304347827, 0.21428571428571427, 'X[4] <= 1978.5\ngini = 0.489\nsamples = 172\nvalue = [99, 73]'),
```

```
 Text(0.14130434782608695, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.16304347826086957, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.2608695652173913, 0.5, 'X[4] <= 1820.5\ngini = 0.337\nsamples
= 24834\nvalue = [19509, 5325]'),
 Text(0.21739130434782608, 0.35714285714285715, 'X[5] <= 41.5\ngini =
0.321\nsamples = 24094\nvalue = [19252, 4842]'),
 Text(0.1956521739130435, 0.21428571428571427, 'X[2] <= 8.5\ngini =
0.27\nsamples = 17064\nvalue = [14320, 2744]'),
 Text(0.18478260869565216, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.20652173913043478, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.2391304347826087, 0.21428571428571427, 'X[2] <= 9.5\ngini =
0.419\nsamples = 7030\nvalue = [4932, 2098]'),
 Text(0.22826086956521738, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.25, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.30434782608695654, 0.35714285714285715, 'X[4] <= 1978.5\ngini
= 0.453\nsamples = 740\nvalue = [257, 483]'),
 Text(0.2826086956521739, 0.21428571428571427, 'X[4] <= 1881.5\ngini =
0.227\nsamples = 474\nvalue = [62, 412]'),
 Text(0.2717391304347826, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.29347826086956524, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.32608695652173914, 0.21428571428571427, 'X[4] <= 2218.5\ngini
= 0.391\nsamples = 266\nvalue = [195, 71]'),
 Text(0.31521739130434784, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.33695652173913043, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.5217391304347826, 0.6428571428571429, 'X[0] <= 29.5\ngini =
0.49\nsamples = 13583\nvalue = [7769, 5814]'),
 Text(0.43478260869565216, 0.5, 'X[0] <= 27.5\ngini = 0.213\nsamples =
3005\nvalue = [2641, 364]'),
 Text(0.391304347826087, 0.35714285714285715, 'X[4] <= 1881.5\ngini =
0.137\nsamples = 2209\nvalue = [2046, 163]'),
 Text(0.3695652173913043, 0.21428571428571427, 'X[0] <= 24.5\ngini =
0.126\nsamples = 2185\nvalue = [2038, 147]'),
 Text(0.358695652173913, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.3804347826086957, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.41304347826086957, 0.21428571428571427, 'X[1] <= 122304.5\
ngini = 0.444\nsamples = 24\nvalue = [8, 16]'),
 Text(0.40217391304347827, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.42391304347826086, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.4782608695652174, 0.35714285714285715, 'X[5] <= 41.5\ngini =
0.377\nsamples = 796\nvalue = [595, 201]'),
 Text(0.45652173913043476, 0.21428571428571427, 'X[4] <= 1446.5\ngini
= 0.287\nsamples = 431\nvalue = [356, 75]'),
 Text(0.44565217391304346, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.4673913043478261, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.5, 0.21428571428571427, 'X[4] <= 1794.0\ngini = 0.452\nsamples
= 365\nvalue = [239, 126]'),
 Text(0.4891304347826087, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.5108695652173914, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6086956521739131, 0.5, 'X[4] <= 1881.5\ngini = 0.5\nsamples =
10578\nvalue = [5128, 5450]'),
```

```
 Text(0.5652173913043478, 0.35714285714285715, 'X[5] <= 42.5\ngini =
0.499\nsamples = 9791\nvalue = [5068, 4723]'),
 Text(0.5434782608695652, 0.21428571428571427, 'X[0] <= 40.5\ngini =
0.481\nsamples = 5569\nvalue = [3334, 2235]'),
 Text(0.532608695652174, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.5543478260869565, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.5869565217391305, 0.21428571428571427, 'X[2] <= 14.5\ngini =
0.484\nsamples = 4222\nvalue = [1734, 2488]'),
 Text(0.5760869565217391, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.5978260869565217, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6521739130434783, 0.35714285714285715, 'X[4] <= 1978.5\ngini =
0.141\nsamples = 787\nvalue = [60, 727]'),
 Text(0.6304347826086957, 0.21428571428571427, 'X[1] <= 49159.0\ngini
= 0.021\nsamples = 572\nvalue = [6, 566]'),
 Text(0.6195652173913043, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6413043478260869, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6739130434782609, 0.21428571428571427, 'X[4] <= 2358.0\ngini =
0.376\nsamples = 215\nvalue = [54, 161]'),
 Text(0.6630434782608695, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.6847826086956522, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.8002717391304348, 0.7857142857142857, 'X[3] <= 7055.5\ngini =
0.102\nsamples = 2985\nvalue = [161, 2824]'),
 Text(0.7282608695652174, 0.6428571428571429, 'X[3] <= 5316.5\ngini =
0.459\nsamples = 365\nvalue = [130, 235]'),
 Text(0.717391304347826, 0.5, 'gini = 0.0\nsamples = 196\nvalue = [0,
196]'),
 Text(0.7391304347826086, 0.5, 'X[0] <= 61.0\ngini = 0.355\nsamples =
169\nvalue = [130, 39]'),
 Text(0.7065217391304348, 0.35714285714285715, 'X[0] <= 54.5\ngini =
0.017\nsamples = 115\nvalue = [114, 1]'),
 Text(0.6956521739130435, 0.21428571428571427, 'gini = 0.0\nsamples =
107\nvalue = [107, 0]'),
 Text(0.717391304347826, 0.21428571428571427, 'X[0] <= 56.0\ngini =
0.219\nsamples = 8\nvalue = [7, 1]'),
 Text(0.7065217391304348, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.7282608695652174, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.7717391304347826, 0.35714285714285715, 'X[3] <= 6618.5\ngini =
0.417\nsamples = 54\nvalue = [16, 38]'),
 Text(0.7608695652173914, 0.21428571428571427, 'X[1] <= 50818.5\ngini
= 0.172\nsamples = 42\nvalue = [4, 38]'),
 Text(0.75, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.7717391304347826, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.782608695652174, 0.21428571428571427, 'gini = 0.0\nsamples =
12\nvalue = [12, 0]'),
 Text(0.8722826086956522, 0.6428571428571429, 'X[0] <= 60.5\ngini =
0.023\nsamples = 2620\nvalue = [31, 2589]'),
 Text(0.8152173913043478, 0.5, 'X[2] <= 1.5\ngini = 0.008\nsamples =
2366\nvalue = [10, 2356]'),
 Text(0.8043478260869565, 0.35714285714285715, 'gini = 0.0\nsamples =
1\nvalue = [1, 0]'),
```

```
 Text(0.8260869565217391, 0.35714285714285715, 'X[0] <= 20.0\ngini =
0.008\nsamples = 2365\nvalue = [9, 2356]'),
 Text(0.8043478260869565, 0.21428571428571427, 'X[1] <= 165776.5\ngini
= 0.5\nsamples = 4\nvalue = [2, 2]'),
 Text(0.7934782608695652, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.8152173913043478, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.8478260869565217, 0.21428571428571427, 'X[5] <= 87.0\ngini =
0.006\nsamples = 2361\nvalue = [7, 2354]'),
 Text(0.8369565217391305, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.8586956521739131, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.9293478260869565, 0.5, 'X[3] <= 10585.5\ngini = 0.152\nsamples
= 254\nvalue = [21, 233]'),
 Text(0.9021739130434783, 0.35714285714285715, 'X[3] <= 10543.0\ngini
= 0.338\nsamples = 79\nvalue = [17, 62]'),
 Text(0.8913043478260869, 0.21428571428571427, 'X[5] <= 35.5\ngini =
0.114\nsamples = 66\nvalue = [4, 62]'),
 Text(0.8804347826086957, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.9021739130434783, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.9130434782608695, 0.21428571428571427, 'gini = 0.0\nsamples =
13\nvalue = [13, 0]'),
 Text(0.9565217391304348, 0.35714285714285715, 'X[1] <= 34285.0\ngini
= 0.045\nsamples = 175\nvalue = [4, 171]'),
 Text(0.9347826086956522, 0.21428571428571427, 'X[1] <= 27368.0\ngini
= 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(0.9239130434782609, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.9456521739130435, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.9782608695652174, 0.21428571428571427, 'X[1] <= 284860.5\ngini
= 0.023\nsamples = 172\nvalue = [2, 170]'),
 Text(0.967391304347826, 0.07142857142857142, '\n  (...)  \n'),
 Text(0.9891304347826086, 0.07142857142857142, '\n  (...)  \n')]
```



4.2 Hyper paramer Tuning

Decision Tree Classifier

```
grid_params = {
    'criterion' : ['gini', 'entropy', 'log_loss'],
    'splitter' : ['best', 'random'],
    'max_depth' : range(1,10,1),
    'min_samples_split' : range(2,10,2),
    'min_samples_leaf' : range(1,5,1),
    'max_features' : ['auto', 'sqrt', 'log2']
}

grid_search_dtc = GridSearchCV(estimator= DecisionTreeClassifier(),
param_grid=grid_params, verbose=2, n_jobs=-1, cv=3)

grid_search_dtc.fit(X_train, y_train)

Fitting 3 folds for each of 2592 candidates, totalling 7776 fits

GridSearchCV(cv=3, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_depth': range(1, 10),
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': range(1, 5),
                         'min_samples_split': range(2, 10, 2),
                         'splitter': ['best', 'random']},
             verbose=2)

grid_search.best_params_

{'criterion': 'gini',
 'max_depth': 9,
 'min_samples_leaf': 1,
 'min_samples_split': 2}

model_with_bst_prm_dtc = DecisionTreeClassifier(criterion = 'gini',
max_depth = 9, min_samples_leaf = 1, min_samples_split = 2)

model_with_bst_prm_dtc.fit(X_train, y_train)

DecisionTreeClassifier(max_depth=9)

y_pred_bst_prm_dtc = model_with_bst_prm_dtc.predict(X_test)
```

5.1 Accuracy Score

```
# Decision Tree Regressor Model with best params accuracy score

print("Decision Tree Regressor Model with best params training
accuracy score is : {}
%".format(round(model_with_bst_prm_dtc.score(X_train, y_train)*100,
2)))
print("Decision Tree Regressor Model with best params accuracy score
```

```
is : {}%".format(round(accuracy_score(y_test, y_pred_bst_prm_dtc)*100,
2)))
```

```
Decision Tree Regressor Model with best params training accuracy score
is : 83.74%
Decision Tree Regressor Model with best params accuracy score is :
83.09%
```

5.2 Roc-auc score

```
y_train_predict_roc_dtc_bst_prm =
model_with_bst_prm_dtc.predict_proba(X_train)

print("Decision Tree Regressor Model with best params training roc-auc
score is : {}%".format(round(roc_auc_score(y_train,
y_train_predict_roc_dtc_bst_prm[:,1])*100)))

y_test_predict_roc_dtc_bst_prm =
model_with_bst_prm_dtc.predict_proba(X_test)

print("Decision Tree Regressor Model with best params roc-auc accuracy
score is : {}%".format(round(roc_auc_score(y_test,
y_test_predict_roc_dtc_bst_prm[:,1])*100)))
```

```
Decision Tree Regressor Model with best params training roc-auc score
is : 87%
Decision Tree Regressor Model with best params roc-auc accuracy score
is : 85%
```

5.3 Confusion_matrix

```
conf_mat_dtc_bst_prm = confusion_matrix(y_test, y_pred_bc)
conf_mat_dtc_bst_prm
```

```
array([[21572,   840],
       [ 1608,  5808]], dtype=int64)
```

```
true_positive = conf_mat_dtc_bst_prm[0][0]
false_positive = conf_mat_dtc_bst_prm[0][1]
false_negative = conf_mat_dtc_bst_prm[1][0]
true_negative = conf_mat_dtc_bst_prm[1][0]
print('True Positive:',true_positive, '\nTrue
Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse
Positive:',false_positive)
```

```
True Positive: 21572
True Negative: 1608
False Negative: 1608
False Positive: 840
```

Classification Report

```
class_reprt_log_reg = classification_report(y_test, 
y_pred_bst_prm_dtc)
print(class_reprt_log_reg)
```

```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     22412
           1       0.79      0.43      0.56      7416

    accuracy                           0.83     29828
   macro avg       0.81      0.70      0.73     29828
weighted avg       0.83      0.83      0.81     29828
```

```
# Plotting Decision Tree

from sklearn import tree
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(25,15))
tree.plot_tree(model_with_bst_prm_dtc, max_depth=9, filled=True, 
fontsize=10)
```

```
[Text(0.7040466392318244, 0.95, 'X[3] <= 5119.0\ngini = 0.373\nsamples 
= 60558\nvalue = [45562, 14996]'),
 Text(0.474022633744856, 0.85, 'X[2] <= 12.5\ngini = 0.333\nsamples = 
57573\nvalue = [45401, 12172]'),
 Text(0.25685871056241427, 0.75, 'X[0] <= 33.5\ngini = 0.247\nsamples 
= 43990\nvalue = [37632, 6358]'),
 Text(0.12808641975308643, 0.65, 'X[0] <= 26.5\ngini = 0.102\nsamples 
= 19156\nvalue = [18123, 1033]'),
 Text(0.06927297668038408, 0.55, 'X[4] <= 1805.0\ngini = 0.03\nsamples 
= 10886\nvalue = [10721, 165]'),
 Text(0.0438957475994513, 0.45, 'X[5] <= 41.5\ngini = 0.027\nsamples = 
10803\nvalue = [10656, 147]'),
 Text(0.02194787379972565, 0.35, 'X[0] <= 23.5\ngini = 0.014\nsamples 
= 9344\nvalue = [9277, 67]'),
 Text(0.010973936899862825, 0.25, 'X[0] <= 21.5\ngini = 0.004\nsamples 
= 6873\nvalue = [6859, 14]'),
 Text(0.0054869684499314125, 0.15, 'X[1] <= 546197.0\ngini = 0.001\
nsamples = 4759\nvalue = [4757, 2]'),
 Text(0.0027434842249657062, 0.05, 'gini = 0.0\nsamples = 4719\nvalue 
= [4718, 1]'),
 Text(0.00823045267489712, 0.05, 'gini = 0.049\nsamples = 40\nvalue = 
[39, 1]'),
 Text(0.01646090534979424, 0.15, 'X[1] <= 65514.0\ngini = 0.011\
nsamples = 2114\nvalue = [2102, 12]'),
 Text(0.013717421124828532, 0.05, 'gini = 0.041\nsamples = 190\nvalue 
= [186, 4]'),
```

```
 Text(0.019204389574759947, 0.05, 'gini = 0.008\nsamples = 1924\nvalue
= [1916, 8]'),
 Text(0.03292181069958848, 0.25, 'X[3] <= 3005.0\ngini = 0.042\
nsamples = 2471\nvalue = [2418, 53]'),
 Text(0.027434842249657063, 0.15, 'X[1] <= 31320.0\ngini = 0.037\
nsamples = 2434\nvalue = [2388, 46]'),
 Text(0.024691358024691357, 0.05, 'gini = 0.213\nsamples = 33\nvalue =
[29, 4]'),
 Text(0.03017832647462277, 0.05, 'gini = 0.034\nsamples = 2401\nvalue
= [2359, 42]'),
 Text(0.038408779149519894, 0.15, 'X[3] <= 3120.0\ngini = 0.307\
nsamples = 37\nvalue = [30, 7]'),
 Text(0.03566529492455418, 0.05, 'gini = 0.278\nsamples = 6\nvalue =
[1, 5]'),
 Text(0.0411522633744856, 0.05, 'gini = 0.121\nsamples = 31\nvalue =
[29, 2]'),
 Text(0.06584362139917696, 0.35, 'X[0] <= 23.5\ngini = 0.104\nsamples
= 1459\nvalue = [1379, 80]'),
 Text(0.05486968449931413, 0.25, 'X[1] <= 41895.0\ngini = 0.034\
nsamples = 746\nvalue = [733, 13]'),
 Text(0.04938271604938271, 0.15, 'X[2] <= 10.5\ngini = 0.162\nsamples
= 45\nvalue = [41, 4]'),
 Text(0.04663923182441701, 0.05, 'gini = 0.089\nsamples = 43\nvalue =
[41, 2]'),
 Text(0.05212620027434842, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(0.06035665294924554, 0.15, 'X[2] <= 11.5\ngini = 0.025\nsamples
= 701\nvalue = [692, 9]'),
 Text(0.05761316872427984, 0.05, 'gini = 0.02\nsamples = 683\nvalue =
[676, 7]'),
 Text(0.06310013717421124, 0.05, 'gini = 0.198\nsamples = 18\nvalue =
[16, 2]'),
 Text(0.07681755829903979, 0.25, 'X[5] <= 68.0\ngini = 0.17\nsamples =
713\nvalue = [646, 67]'),
 Text(0.07133058984910837, 0.15, 'X[3] <= 3005.0\ngini = 0.155\
nsamples = 660\nvalue = [604, 56]'),
 Text(0.06858710562414266, 0.05, 'gini = 0.148\nsamples = 647\nvalue =
[595, 52]'),
 Text(0.07407407407407407, 0.05, 'gini = 0.426\nsamples = 13\nvalue =
[9, 4]'),
 Text(0.0823045267489712, 0.15, 'X[1] <= 196155.5\ngini = 0.329\
nsamples = 53\nvalue = [42, 11]'),
 Text(0.07956104252400549, 0.05, 'gini = 0.437\nsamples = 31\nvalue =
[21, 10]'),
 Text(0.0850480109739369, 0.05, 'gini = 0.087\nsamples = 22\nvalue =
[21, 1]'),
 Text(0.09465020576131687, 0.45, 'X[4] <= 1938.0\ngini = 0.34\nsamples
= 83\nvalue = [65, 18]'),
 Text(0.0877914951989026, 0.35, 'X[1] <= 200349.0\ngini = 0.36\
nsamples = 17\nvalue = [4, 13]'),
```

```
 Text(0.0850480109739369, 0.25, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.09053497942386832, 0.25, 'gini = 0.0\nsamples = 13\nvalue =
[0, 13]'),
 Text(0.10150891632373114, 0.35, 'X[4] <= 2424.5\ngini = 0.14\nsamples
= 66\nvalue = [61, 5]'),
 Text(0.09602194787379972, 0.25, 'X[4] <= 1978.5\ngini = 0.062\
nsamples = 62\nvalue = [60, 2]'),
 Text(0.09327846364883402, 0.15, 'X[4] <= 1975.5\ngini = 0.375\
nsamples = 8\nvalue = [6, 2]'),
 Text(0.09053497942386832, 0.05, 'gini = 0.0\nsamples = 6\nvalue = [6,
0]'),
 Text(0.09602194787379972, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(0.09876543209876543, 0.15, 'gini = 0.0\nsamples = 54\nvalue =
[54, 0]'),
 Text(0.10699588477366255, 0.25, 'X[4] <= 2581.0\ngini = 0.375\
nsamples = 4\nvalue = [1, 3]'),
 Text(0.10425240054869685, 0.15, 'gini = 0.0\nsamples = 3\nvalue = [0,
3]'),
 Text(0.10973936899862825, 0.15, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.18689986282578874, 0.55, 'X[4] <= 1805.0\ngini = 0.188\
nsamples = 8270\nvalue = [7402, 868]'),
 Text(0.15363511659807957, 0.45, 'X[5] <= 44.5\ngini = 0.177\nsamples
= 8098\nvalue = [7303, 795]'),
 Text(0.13168724279835392, 0.35, 'X[2] <= 7.5\ngini = 0.142\nsamples =
5972\nvalue = [5512, 460]'),
 Text(0.12071330589849108, 0.25, 'X[1] <= 236688.5\ngini = 0.036\
nsamples = 824\nvalue = [809, 15]'),
 Text(0.11522633744855967, 0.15, 'X[1] <= 236461.0\ngini = 0.052\
nsamples = 527\nvalue = [513, 14]'),
 Text(0.11248285322359397, 0.05, 'gini = 0.045\nsamples = 524\nvalue =
[512, 12]'),
 Text(0.11796982167352538, 0.05, 'gini = 0.444\nsamples = 3\nvalue =
[1, 2]'),
 Text(0.1262002743484225, 0.15, 'X[1] <= 380856.5\ngini = 0.007\
nsamples = 297\nvalue = [296, 1]'),
 Text(0.12345679012345678, 0.05, 'gini = 0.0\nsamples = 232\nvalue =
[232, 0]'),
 Text(0.1289437585733882, 0.05, 'gini = 0.03\nsamples = 65\nvalue =
[64, 1]'),
 Text(0.14266117969821673, 0.25, 'X[5] <= 35.5\ngini = 0.158\nsamples
= 5148\nvalue = [4703, 445]'),
 Text(0.13717421124828533, 0.15, 'X[1] <= 29862.5\ngini = 0.068\
nsamples = 913\nvalue = [881, 32]'),
 Text(0.13443072702331962, 0.05, 'gini = 0.444\nsamples = 9\nvalue =
[6, 3]'),
 Text(0.13991769547325103, 0.05, 'gini = 0.062\nsamples = 904\nvalue =
[875, 29]'),
```

```
 Text(0.14814814814814814, 0.15, 'X[0] <= 28.5\ngini = 0.176\nsamples
= 4235\nvalue = [3822, 413]'),
 Text(0.14540466392318244, 0.05, 'gini = 0.127\nsamples = 1186\nvalue
= [1105, 81]'),
 Text(0.15089163237311384, 0.05, 'gini = 0.194\nsamples = 3049\nvalue
= [2717, 332]'),
 Text(0.1755829903978052, 0.35, 'X[0] <= 29.5\ngini = 0.265\nsamples =
2126\nvalue = [1791, 335]'),
 Text(0.1646090534979424, 0.25, 'X[3] <= 4225.0\ngini = 0.198\nsamples
= 869\nvalue = [772, 97]'),
 Text(0.15912208504801098, 0.15, 'X[1] <= 387946.5\ngini = 0.19\
nsamples = 858\nvalue = [767, 91]'),
 Text(0.15637860082304528, 0.05, 'gini = 0.173\nsamples = 814\nvalue =
[736, 78]'),
 Text(0.16186556927297668, 0.05, 'gini = 0.416\nsamples = 44\nvalue =
[31, 13]'),
 Text(0.1700960219478738, 0.15, 'X[3] <= 4625.5\ngini = 0.496\nsamples
= 11\nvalue = [5, 6]'),
 Text(0.1673525377229081, 0.05, 'gini = 0.0\nsamples = 6\nvalue = [0,
6]'),
 Text(0.1728395061728395, 0.05, 'gini = 0.0\nsamples = 5\nvalue = [5,
0]'),
 Text(0.18655692729766804, 0.25, 'X[2] <= 9.5\ngini = 0.307\nsamples =
1257\nvalue = [1019, 238]'),
 Text(0.18106995884773663, 0.15, 'X[2] <= 7.5\ngini = 0.268\nsamples =
720\nvalue = [605, 115]'),
 Text(0.17832647462277093, 0.05, 'gini = 0.15\nsamples = 110\nvalue =
[101, 9]'),
 Text(0.18381344307270234, 0.05, 'gini = 0.287\nsamples = 610\nvalue =
[504, 106]'),
 Text(0.19204389574759945, 0.15, 'X[1] <= 116069.5\ngini = 0.353\
nsamples = 537\nvalue = [414, 123]'),
 Text(0.18930041152263374, 0.05, 'gini = 0.248\nsamples = 117\nvalue =
[100, 17]'),
 Text(0.19478737997256515, 0.05, 'gini = 0.377\nsamples = 420\nvalue =
[314, 106]'),
 Text(0.22016460905349794, 0.45, 'X[4] <= 1978.5\ngini = 0.489\
nsamples = 172\nvalue = [99, 73]'),
 Text(0.20713305898491083, 0.35, 'X[1] <= 47712.0\ngini = 0.383\
nsamples = 89\nvalue = [23, 66]'),
 Text(0.20027434842249658, 0.25, 'X[5] <= 28.0\ngini = 0.278\nsamples
= 6\nvalue = [5, 1]'),
 Text(0.19753086419753085, 0.15, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.2030178326474623, 0.15, 'gini = 0.0\nsamples = 5\nvalue = [5,
0]'),
 Text(0.2139917695473251, 0.25, 'X[4] <= 1938.0\ngini = 0.34\nsamples
= 83\nvalue = [18, 65]'),
 Text(0.2085048010973937, 0.15, 'X[2] <= 9.5\ngini = 0.271\nsamples =
68\nvalue = [11, 57]'),
```

```
 Text(0.205761316872428, 0.05, 'gini = 0.351\nsamples = 44\nvalue =
[10, 34]'),
 Text(0.2112482853223594, 0.05, 'gini = 0.08\nsamples = 24\nvalue =
[1, 23]'),
 Text(0.2194787379972565, 0.15, 'X[4] <= 1975.5\ngini = 0.498\nsamples
= 15\nvalue = [7, 8]'),
 Text(0.2167352537722908, 0.05, 'gini = 0.0\nsamples = 7\nvalue = [7,
0]'),
 Text(0.2222222222222222, 0.05, 'gini = 0.0\nsamples = 8\nvalue = [0,
8]'),
 Text(0.23319615912208505, 0.35, 'X[4] <= 2396.0\ngini = 0.154\
nsamples = 83\nvalue = [76, 7]'),
 Text(0.22770919067215364, 0.25, 'X[4] <= 2248.0\ngini = 0.027\
nsamples = 73\nvalue = [72, 1]'),
 Text(0.22496570644718794, 0.15, 'gini = 0.0\nsamples = 63\nvalue =
[63, 0]'),
 Text(0.23045267489711935, 0.15, 'X[5] <= 43.5\ngini = 0.18\nsamples =
10\nvalue = [9, 1]'),
 Text(0.22770919067215364, 0.05, 'gini = 0.0\nsamples = 6\nvalue = [6,
0]'),
 Text(0.23319615912208505, 0.05, 'gini = 0.375\nsamples = 4\nvalue =
[3, 1]'),
 Text(0.23868312757201646, 0.25, 'X[5] <= 38.5\ngini = 0.48\nsamples =
10\nvalue = [4, 6]'),
 Text(0.23593964334705075, 0.15, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.24142661179698216, 0.15, 'gini = 0.0\nsamples = 6\nvalue = [0,
6]'),
 Text(0.3856310013717421, 0.65, 'X[4] <= 1820.5\ngini = 0.337\nsamples
= 24834\nvalue = [19509, 5325]'),
 Text(0.32887517146776407, 0.55, 'X[5] <= 41.5\ngini = 0.321\nsamples
= 24094\nvalue = [19252, 4842]'),
 Text(0.28532235939643347, 0.45, 'X[2] <= 8.5\ngini = 0.27\nsamples =
17064\nvalue = [14320, 2744]'),
 Text(0.26337448559670784, 0.35, 'X[0] <= 37.5\ngini = 0.103\nsamples
= 3112\nvalue = [2943, 169]'),
 Text(0.252400548696845, 0.25, 'X[0] <= 34.5\ngini = 0.013\nsamples =
451\nvalue = [448, 3]'),
 Text(0.24691358024691357, 0.15, 'X[1] <= 238660.0\ngini = 0.037\
nsamples = 105\nvalue = [103, 2]'),
 Text(0.24417009602194786, 0.05, 'gini = 0.0\nsamples = 72\nvalue =
[72, 0]'),
 Text(0.2496570644718793, 0.05, 'gini = 0.114\nsamples = 33\nvalue =
[31, 2]'),
 Text(0.2578875171467764, 0.15, 'X[1] <= 112391.5\ngini = 0.006\
nsamples = 346\nvalue = [345, 1]'),
 Text(0.2551440329218107, 0.05, 'gini = 0.027\nsamples = 74\nvalue =
[73, 1]'),
 Text(0.2606310013717421, 0.05, 'gini = 0.0\nsamples = 272\nvalue =
[272, 0]'),
```

```
 Text(0.27434842249657065, 0.25, 'X[2] <= 5.5\ngini = 0.117\nsamples =
2661\nvalue = [2495, 166]'),
 Text(0.26886145404663925, 0.15, 'X[1] <= 34337.5\ngini = 0.076\
nsamples = 1444\nvalue = [1387, 57]'),
 Text(0.2661179698216735, 0.05, 'gini = 0.298\nsamples = 44\nvalue =
[36, 8]'),
 Text(0.2716049382716049, 0.05, 'gini = 0.068\nsamples = 1400\nvalue =
[1351, 49]'),
 Text(0.27983539094650206, 0.15, 'X[3] <= 4243.5\ngini = 0.163\
nsamples = 1217\nvalue = [1108, 109]'),
 Text(0.27709190672153633, 0.05, 'gini = 0.158\nsamples = 1211\nvalue
= [1106, 105]'),
 Text(0.2825788751714678, 0.05, 'gini = 0.444\nsamples = 6\nvalue =
[2, 4]'),
 Text(0.30727023319615915, 0.35, 'X[5] <= 35.5\ngini = 0.301\nsamples
= 13952\nvalue = [11377, 2575]'),
 Text(0.2962962962962963, 0.25, 'X[2] <= 9.5\ngini = 0.158\nsamples =
2963\nvalue = [2706, 257]'),
 Text(0.2908093278463649, 0.15, 'X[0] <= 64.5\ngini = 0.111\nsamples =
1771\nvalue = [1667, 104]'),
 Text(0.2880658436213992, 0.05, 'gini = 0.133\nsamples = 1367\nvalue =
[1269, 98]'),
 Text(0.2935528120713306, 0.05, 'gini = 0.029\nsamples = 404\nvalue =
[398, 6]'),
 Text(0.3017832647462277, 0.15, 'X[1] <= 268726.0\ngini = 0.224\
nsamples = 1192\nvalue = [1039, 153]'),
 Text(0.299039780521262, 0.05, 'gini = 0.189\nsamples = 993\nvalue =
[888, 105]'),
 Text(0.3045267489711934, 0.05, 'gini = 0.366\nsamples = 199\nvalue =
[151, 48]'),
 Text(0.31824417009602196, 0.25, 'X[2] <= 9.5\ngini = 0.333\nsamples =
10989\nvalue = [8671, 2318]'),
 Text(0.31275720164609055, 0.15, 'X[0] <= 41.5\ngini = 0.294\nsamples
= 6230\nvalue = [5114, 1116]'),
 Text(0.3100137174211248, 0.05, 'gini = 0.236\nsamples = 2397\nvalue =
[2069, 328]'),
 Text(0.31550068587105623, 0.05, 'gini = 0.327\nsamples = 3833\nvalue
= [3045, 788]'),
 Text(0.32373113854595337, 0.15, 'X[0] <= 39.5\ngini = 0.378\nsamples
= 4759\nvalue = [3557, 1202]'),
 Text(0.32098765432098764, 0.05, 'gini = 0.311\nsamples = 1445\nvalue
= [1167, 278]'),
 Text(0.32647462277091904, 0.05, 'gini = 0.402\nsamples = 3314\nvalue
= [2390, 924]'),
 Text(0.3724279835390947, 0.45, 'X[2] <= 9.5\ngini = 0.419\nsamples =
7030\nvalue = [4932, 2098]'),
 Text(0.3511659807956104, 0.35, 'X[2] <= 7.5\ngini = 0.366\nsamples =
4004\nvalue = [3038, 966]'),
 Text(0.3401920438957476, 0.25, 'X[0] <= 46.5\ngini = 0.241\nsamples =
821\nvalue = [706, 115]'),
```

```
 Text(0.3347050754458162, 0.15, 'X[4] <= 1432.5\ngini = 0.16\nsamples
= 331\nvalue = [302, 29]'),
 Text(0.3319615912208505, 0.05, 'gini = 0.151\nsamples = 329\nvalue =
[302, 27]'),
 Text(0.3374485596707819, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(0.345679012345679, 0.15, 'X[1] <= 341747.0\ngini = 0.289\
nsamples = 490\nvalue = [404, 86]'),
 Text(0.3429355281207133, 0.05, 'gini = 0.276\nsamples = 465\nvalue =
[388, 77]'),
 Text(0.3484224965706447, 0.05, 'gini = 0.461\nsamples = 25\nvalue =
[16, 9]'),
 Text(0.36213991769547327, 0.25, 'X[0] <= 44.5\ngini = 0.392\nsamples
= 3183\nvalue = [2332, 851]'),
 Text(0.35665294924554186, 0.15, 'X[1] <= 140655.5\ngini = 0.345\
nsamples = 1671\nvalue = [1301, 370]'),
 Text(0.35390946502057613, 0.05, 'gini = 0.291\nsamples = 611\nvalue =
[503, 108]'),
 Text(0.35939643347050754, 0.05, 'gini = 0.372\nsamples = 1060\nvalue
= [798, 262]'),
 Text(0.3676268861454047, 0.15, 'X[4] <= 1529.0\ngini = 0.434\nsamples
= 1512\nvalue = [1031, 481]'),
 Text(0.36488340192043894, 0.05, 'gini = 0.438\nsamples = 1486\nvalue
= [1005, 481]'),
 Text(0.37037037037037035, 0.05, 'gini = 0.0\nsamples = 26\nvalue =
[26, 0]'),
 Text(0.3936899862825789, 0.35, 'X[1] <= 75898.5\ngini = 0.468\
nsamples = 3026\nvalue = [1894, 1132]'),
 Text(0.3840877914951989, 0.25, 'X[0] <= 49.5\ngini = 0.389\nsamples =
424\nvalue = [312, 112]'),
 Text(0.3786008230452675, 0.15, 'X[1] <= 37698.0\ngini = 0.338\
nsamples = 325\nvalue = [255, 70]'),
 Text(0.37585733882030176, 0.05, 'gini = 0.428\nsamples = 132\nvalue =
[91, 41]'),
 Text(0.3813443072702332, 0.05, 'gini = 0.255\nsamples = 193\nvalue =
[164, 29]'),
 Text(0.3895747599451303, 0.15, 'X[1] <= 27462.0\ngini = 0.489\
nsamples = 99\nvalue = [57, 42]'),
 Text(0.3868312757201646, 0.05, 'gini = 0.305\nsamples = 16\nvalue =
[3, 13]'),
 Text(0.39231824417009603, 0.05, 'gini = 0.455\nsamples = 83\nvalue =
[54, 29]'),
 Text(0.40329218106995884, 0.25, 'X[4] <= 1577.0\ngini = 0.477\
nsamples = 2602\nvalue = [1582, 1020]'),
 Text(0.40054869684499317, 0.15, 'X[0] <= 39.5\ngini = 0.479\nsamples
= 2573\nvalue = [1553, 1020]'),
 Text(0.39780521262002744, 0.05, 'gini = 0.453\nsamples = 868\nvalue =
[567, 301]'),
 Text(0.40329218106995884, 0.05, 'gini = 0.488\nsamples = 1705\nvalue
= [986, 719]'),
```

```
 Text(0.4060356652949246, 0.15, 'gini = 0.0\nsamples = 29\nvalue =
[29, 0]'),
 Text(0.44238683127572015, 0.55, 'X[4] <= 1978.5\ngini = 0.453\
nsamples = 740\nvalue = [257, 483]'),
 Text(0.42661179698216734, 0.45, 'X[4] <= 1881.5\ngini = 0.227\
nsamples = 474\nvalue = [62, 412]'),
 Text(0.4170096021947874, 0.35, 'X[4] <= 1859.0\ngini = 0.486\nsamples
= 79\nvalue = [33, 46]'),
 Text(0.41426611796982166, 0.25, 'X[0] <= 66.5\ngini = 0.115\nsamples
= 49\nvalue = [3, 46]'),
 Text(0.411522633744856, 0.15, 'gini = 0.0\nsamples = 45\nvalue = [0,
45]'),
 Text(0.4170096021947874, 0.15, 'X[0] <= 79.0\ngini = 0.375\nsamples =
4\nvalue = [3, 1]'),
 Text(0.41426611796982166, 0.05, 'gini = 0.0\nsamples = 3\nvalue = [3,
0]'),
 Text(0.41975308641975306, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.41975308641975306, 0.25, 'gini = 0.0\nsamples = 30\nvalue =
[30, 0]'),
 Text(0.43621399176954734, 0.35, 'X[0] <= 64.5\ngini = 0.136\nsamples
= 395\nvalue = [29, 366]'),
 Text(0.4334705075445816, 0.25, 'X[2] <= 5.5\ngini = 0.116\nsamples =
390\nvalue = [24, 366]'),
 Text(0.4279835390946502, 0.15, 'X[1] <= 128296.5\ngini = 0.32\
nsamples = 5\nvalue = [4, 1]'),
 Text(0.4252400548696845, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.43072702331961593, 0.05, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.438957475994513, 0.15, 'X[0] <= 35.5\ngini = 0.098\nsamples =
385\nvalue = [20, 365]'),
 Text(0.43621399176954734, 0.05, 'gini = 0.426\nsamples = 26\nvalue =
[8, 18]'),
 Text(0.44170096021947874, 0.05, 'gini = 0.065\nsamples = 359\nvalue =
[12, 347]'),
 Text(0.438957475994513, 0.25, 'gini = 0.0\nsamples = 5\nvalue = [5,
0]'),
 Text(0.45816186556927296, 0.45, 'X[4] <= 2218.5\ngini = 0.391\
nsamples = 266\nvalue = [195, 71]'),
 Text(0.4499314128943759, 0.35, 'X[0] <= 64.5\ngini = 0.065\nsamples =
148\nvalue = [143, 5]'),
 Text(0.44718792866941015, 0.25, 'gini = 0.0\nsamples = 140\nvalue =
[140, 0]'),
 Text(0.45267489711934156, 0.25, 'X[4] <= 2190.0\ngini = 0.469\
nsamples = 8\nvalue = [3, 5]'),
 Text(0.4499314128943759, 0.15, 'X[4] <= 2161.5\ngini = 0.278\nsamples
= 6\nvalue = [1, 5]'),
 Text(0.44718792866941015, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
```

```
 Text(0.45267489711934156, 0.05, 'gini = 0.0\nsamples = 5\nvalue = [0,
5]'),
 Text(0.4554183813443073, 0.15, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.4663923182441701, 0.35, 'X[4] <= 3089.5\ngini = 0.493\nsamples
= 118\nvalue = [52, 66]'),
 Text(0.46364883401920437, 0.25, 'X[2] <= 4.0\ngini = 0.47\nsamples =
106\nvalue = [40, 66]'),
 Text(0.4609053497942387, 0.15, 'gini = 0.0\nsamples = 7\nvalue = [7,
0]'),
 Text(0.4663923182441701, 0.15, 'X[4] <= 2384.5\ngini = 0.444\nsamples
= 99\nvalue = [33, 66]'),
 Text(0.46364883401920437, 0.05, 'gini = 0.5\nsamples = 54\nvalue =
[27, 27]'),
 Text(0.4691358024691358, 0.05, 'gini = 0.231\nsamples = 45\nvalue =
[6, 39]'),
 Text(0.4691358024691358, 0.25, 'gini = 0.0\nsamples = 12\nvalue =
[12, 0]'),
 Text(0.6911865569272977, 0.75, 'X[0] <= 29.5\ngini = 0.49\nsamples =
13583\nvalue = [7769, 5814]'),
 Text(0.5799039780521262, 0.65, 'X[0] <= 27.5\ngini = 0.213\nsamples =
3005\nvalue = [2641, 364]'),
 Text(0.5363511659807956, 0.55, 'X[4] <= 1881.5\ngini = 0.137\nsamples
= 2209\nvalue = [2046, 163]'),
 Text(0.5157750342935528, 0.45, 'X[0] <= 24.5\ngini = 0.126\nsamples =
2185\nvalue = [2038, 147]'),
 Text(0.49382716049382713, 0.35, 'X[5] <= 53.5\ngini = 0.034\nsamples
= 971\nvalue = [954, 17]'),
 Text(0.4828532235939643, 0.25, 'X[0] <= 21.5\ngini = 0.026\nsamples =
916\nvalue = [904, 12]'),
 Text(0.4773662551440329, 0.15, 'X[5] <= 39.0\ngini = 0.188\nsamples =
19\nvalue = [17, 2]'),
 Text(0.47462277091906724, 0.05, 'gini = 0.0\nsamples = 13\nvalue =
[13, 0]'),
 Text(0.48010973936899864, 0.05, 'gini = 0.444\nsamples = 6\nvalue =
[4, 2]'),
 Text(0.4883401920438957, 0.15, 'X[0] <= 23.5\ngini = 0.022\nsamples =
897\nvalue = [887, 10]'),
 Text(0.48559670781893005, 0.05, 'gini = 0.0\nsamples = 512\nvalue =
[512, 0]'),
 Text(0.49108367626886146, 0.05, 'gini = 0.051\nsamples = 385\nvalue =
[375, 10]'),
 Text(0.50480109739369, 0.25, 'X[1] <= 49301.0\ngini = 0.165\nsamples
= 55\nvalue = [50, 5]'),
 Text(0.4993141289437586, 0.15, 'X[1] <= 39596.0\ngini = 0.444\
nsamples = 3\nvalue = [1, 2]'),
 Text(0.49657064471879286, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.5020576131687243, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
```

```
 Text(0.5102880658436214, 0.15, 'X[3] <= 2393.5\ngini = 0.109\nsamples
= 52\nvalue = [49, 3]'),
 Text(0.5075445816186557, 0.05, 'gini = 0.075\nsamples = 51\nvalue =
[49, 2]'),
 Text(0.5130315500685871, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.5377229080932785, 0.35, 'X[5] <= 44.5\ngini = 0.191\nsamples =
1214\nvalue = [1084, 130]'),
 Text(0.5267489711934157, 0.25, 'X[5] <= 37.5\ngini = 0.13\nsamples =
872\nvalue = [811, 61]'),
 Text(0.5212620027434842, 0.15, 'X[1] <= 33627.5\ngini = 0.017\
nsamples = 234\nvalue = [232, 2]'),
 Text(0.5185185185185185, 0.05, 'gini = 0.278\nsamples = 6\nvalue =
[5, 1]'),
 Text(0.52400548696845, 0.05, 'gini = 0.009\nsamples = 228\nvalue =
[227, 1]'),
 Text(0.532235939643347, 0.15, 'X[1] <= 23912.5\ngini = 0.168\nsamples
= 638\nvalue = [579, 59]'),
 Text(0.5294924554183813, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.5349794238683128, 0.05, 'gini = 0.166\nsamples = 637\nvalue =
[579, 58]'),
 Text(0.5486968449931413, 0.25, 'X[5] <= 85.0\ngini = 0.322\nsamples =
342\nvalue = [273, 69]'),
 Text(0.5432098765432098, 0.15, 'X[5] <= 45.5\ngini = 0.311\nsamples =
337\nvalue = [272, 65]'),
 Text(0.5404663923182441, 0.05, 'gini = 0.408\nsamples = 91\nvalue =
[65, 26]'),
 Text(0.5459533607681756, 0.05, 'gini = 0.267\nsamples = 246\nvalue =
[207, 39]'),
 Text(0.5541838134430727, 0.15, 'X[0] <= 26.5\ngini = 0.32\nsamples =
5\nvalue = [1, 4]'),
 Text(0.551440329218107, 0.05, 'gini = 0.0\nsamples = 3\nvalue = [0,
3]'),
 Text(0.5569272976680384, 0.05, 'gini = 0.5\nsamples = 2\nvalue = [1,
1]'),
 Text(0.5569272976680384, 0.45, 'X[1] <= 122304.5\ngini = 0.444\
nsamples = 24\nvalue = [8, 16]'),
 Text(0.5541838134430727, 0.35, 'gini = 0.0\nsamples = 9\nvalue = [0,
9]'),
 Text(0.5596707818930041, 0.35, 'X[4] <= 1930.5\ngini = 0.498\nsamples
= 15\nvalue = [8, 7]'),
 Text(0.5569272976680384, 0.25, 'gini = 0.0\nsamples = 4\nvalue = [0,
4]'),
 Text(0.5624142661179699, 0.25, 'X[1] <= 236841.0\ngini = 0.397\
nsamples = 11\nvalue = [8, 3]'),
 Text(0.5596707818930041, 0.15, 'gini = 0.0\nsamples = 8\nvalue = [8,
0]'),
 Text(0.5651577503429356, 0.15, 'gini = 0.0\nsamples = 3\nvalue = [0,
3]'),
```

```
 Text(0.6234567901234568, 0.55, 'X[5] <= 41.5\ngini = 0.377\nsamples =
796\nvalue = [595, 201]'),
 Text(0.6035665294924554, 0.45, 'X[4] <= 1446.5\ngini = 0.287\nsamples
= 431\nvalue = [356, 75]'),
 Text(0.5871056241426612, 0.35, 'X[1] <= 178448.0\ngini = 0.265\
nsamples = 407\nvalue = [343, 64]'),
 Text(0.5761316872427984, 0.25, 'X[1] <= 168101.5\ngini = 0.342\
nsamples = 201\nvalue = [157, 44]'),
 Text(0.5706447187928669, 0.15, 'X[3] <= 3855.5\ngini = 0.286\nsamples
= 179\nvalue = [148, 31]'),
 Text(0.5679012345679012, 0.05, 'gini = 0.274\nsamples = 177\nvalue =
[148, 29]'),
 Text(0.5733882030178327, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(0.5816186556927297, 0.15, 'X[2] <= 13.5\ngini = 0.483\nsamples =
22\nvalue = [9, 13]'),
 Text(0.578875171467764, 0.05, 'gini = 0.496\nsamples = 11\nvalue =
[6, 5]'),
 Text(0.5843621399176955, 0.05, 'gini = 0.397\nsamples = 11\nvalue =
[3, 8]'),
 Text(0.598079561042524, 0.25, 'X[2] <= 15.5\ngini = 0.175\nsamples =
206\nvalue = [186, 20]'),
 Text(0.5925925925925926, 0.15, 'X[1] <= 249860.5\ngini = 0.162\
nsamples = 203\nvalue = [185, 18]'),
 Text(0.5898491083676269, 0.05, 'gini = 0.215\nsamples = 106\nvalue =
[93, 13]'),
 Text(0.5953360768175583, 0.05, 'gini = 0.098\nsamples = 97\nvalue =
[92, 5]'),
 Text(0.6035665294924554, 0.15, 'X[0] <= 28.5\ngini = 0.444\nsamples =
3\nvalue = [1, 2]'),
 Text(0.6008230452674898, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.6063100137174211, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(0.6200274348422496, 0.35, 'X[4] <= 1978.5\ngini = 0.497\nsamples
= 24\nvalue = [13, 11]'),
 Text(0.6172839506172839, 0.25, 'X[4] <= 1758.5\ngini = 0.457\nsamples
= 17\nvalue = [6, 11]'),
 Text(0.6145404663923183, 0.15, 'X[1] <= 188906.5\ngini = 0.48\
nsamples = 10\nvalue = [6, 4]'),
 Text(0.6117969821673526, 0.05, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.6172839506172839, 0.05, 'gini = 0.444\nsamples = 6\nvalue =
[2, 4]'),
 Text(0.6200274348422496, 0.15, 'gini = 0.0\nsamples = 7\nvalue = [0,
7]'),
 Text(0.6227709190672154, 0.25, 'gini = 0.0\nsamples = 7\nvalue = [7,
0]'),
 Text(0.6433470507544582, 0.45, 'X[4] <= 1794.0\ngini = 0.452\nsamples
= 365\nvalue = [239, 126]'),
```

```
 Text(0.6337448559670782, 0.35, 'X[2] <= 15.5\ngini = 0.442\nsamples =
349\nvalue = [234, 115]'),
 Text(0.6282578875171467, 0.25, 'X[3] <= 3214.0\ngini = 0.433\nsamples
= 335\nvalue = [229, 106]'),
 Text(0.6255144032921811, 0.15, 'X[5] <= 67.5\ngini = 0.439\nsamples =
326\nvalue = [220, 106]'),
 Text(0.6227709190672154, 0.05, 'gini = 0.448\nsamples = 307\nvalue =
[203, 104]'),
 Text(0.6282578875171467, 0.05, 'gini = 0.188\nsamples = 19\nvalue =
[17, 2]'),
 Text(0.6310013717421125, 0.15, 'gini = 0.0\nsamples = 9\nvalue = [9,
0]'),
 Text(0.6392318244170097, 0.25, 'X[1] <= 188374.5\ngini = 0.459\
nsamples = 14\nvalue = [5, 9]'),
 Text(0.6364883401920439, 0.15, 'X[0] <= 28.5\ngini = 0.408\nsamples =
7\nvalue = [5, 2]'),
 Text(0.6337448559670782, 0.05, 'gini = 0.444\nsamples = 3\nvalue =
[1, 2]'),
 Text(0.6392318244170097, 0.05, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.6419753086419753, 0.15, 'gini = 0.0\nsamples = 7\nvalue = [0,
7]'),
 Text(0.6529492455418381, 0.35, 'X[0] <= 28.5\ngini = 0.43\nsamples =
16\nvalue = [5, 11]'),
 Text(0.6502057613168725, 0.25, 'X[4] <= 1938.0\ngini = 0.494\nsamples
= 9\nvalue = [5, 4]'),
 Text(0.6474622770919067, 0.15, 'X[2] <= 13.5\ngini = 0.444\nsamples =
6\nvalue = [2, 4]'),
 Text(0.644718792866941, 0.05, 'gini = 0.0\nsamples = 4\nvalue = [0,
4]'),
 Text(0.6502057613168725, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.6529492455418381, 0.15, 'gini = 0.0\nsamples = 3\nvalue = [3,
0]'),
 Text(0.6556927297668038, 0.25, 'gini = 0.0\nsamples = 7\nvalue = [0,
7]'),
 Text(0.8024691358024691, 0.65, 'X[4] <= 1881.5\ngini = 0.5\nsamples =
10578\nvalue = [5128, 5450]'),
 Text(0.7407407407407407, 0.55, 'X[5] <= 42.5\ngini = 0.499\nsamples =
9791\nvalue = [5068, 4723]'),
 Text(0.6968449931412894, 0.45, 'X[0] <= 40.5\ngini = 0.481\nsamples =
5569\nvalue = [3334, 2235]'),
 Text(0.6748971193415638, 0.35, 'X[2] <= 14.5\ngini = 0.439\nsamples =
2293\nvalue = [1546, 747]'),
 Text(0.663923182441701, 0.25, 'X[5] <= 39.5\ngini = 0.43\nsamples =
2137\nvalue = [1469, 668]'),
 Text(0.6584362139917695, 0.15, 'X[1] <= 85966.5\ngini = 0.361\
nsamples = 469\nvalue = [358, 111]'),
 Text(0.6556927297668038, 0.05, 'gini = 0.14\nsamples = 66\nvalue =
[61, 5]'),
```

```
 Text(0.6611796982167353, 0.05, 'gini = 0.388\nsamples = 403\nvalue =
[297, 106]'),
 Text(0.6694101508916324, 0.15, 'X[0] <= 33.5\ngini = 0.445\nsamples =
1668\nvalue = [1111, 557]'),
 Text(0.6666666666666666, 0.05, 'gini = 0.413\nsamples = 668\nvalue =
[473, 195]'),
 Text(0.6721536351165981, 0.05, 'gini = 0.462\nsamples = 1000\nvalue =
[638, 362]'),
 Text(0.6858710562414266, 0.25, 'X[0] <= 31.5\ngini = 0.5\nsamples =
156\nvalue = [77, 79]'),
 Text(0.6803840877914952, 0.15, 'X[1] <= 177832.5\ngini = 0.32\
nsamples = 25\nvalue = [20, 5]'),
 Text(0.6776406035665294, 0.05, 'gini = 0.117\nsamples = 16\nvalue =
[15, 1]'),
 Text(0.6831275720164609, 0.05, 'gini = 0.494\nsamples = 9\nvalue =
[5, 4]'),
 Text(0.691358024691358, 0.15, 'X[1] <= 210189.0\ngini = 0.492\
nsamples = 131\nvalue = [57, 74]'),
 Text(0.6886145404663924, 0.05, 'gini = 0.499\nsamples = 94\nvalue =
[49, 45]'),
 Text(0.6941015089163237, 0.05, 'gini = 0.339\nsamples = 37\nvalue =
[8, 29]'),
 Text(0.7187928669410151, 0.35, 'X[5] <= 35.5\ngini = 0.496\nsamples =
3276\nvalue = [1788, 1488]'),
 Text(0.7078189300411523, 0.25, 'X[2] <= 14.5\ngini = 0.421\nsamples =
818\nvalue = [572, 246]'),
 Text(0.7023319615912208, 0.15, 'X[0] <= 48.5\ngini = 0.392\nsamples =
691\nvalue = [506, 185]'),
 Text(0.6995884773662552, 0.05, 'gini = 0.462\nsamples = 273\nvalue =
[174, 99]'),
 Text(0.7050754458161865, 0.05, 'gini = 0.327\nsamples = 418\nvalue =
[332, 86]'),
 Text(0.7133058984910837, 0.15, 'X[5] <= 23.5\ngini = 0.499\nsamples =
127\nvalue = [66, 61]'),
 Text(0.710562414266118, 0.05, 'gini = 0.395\nsamples = 48\nvalue =
[35, 13]'),
 Text(0.7160493827160493, 0.05, 'gini = 0.477\nsamples = 79\nvalue =
[31, 48]'),
 Text(0.7297668038408779, 0.25, 'X[2] <= 14.5\ngini = 0.5\nsamples =
2458\nvalue = [1216, 1242]'),
 Text(0.7242798353909465, 0.15, 'X[4] <= 312.5\ngini = 0.5\nsamples =
2207\nvalue = [1137, 1070]'),
 Text(0.7215363511659808, 0.05, 'gini = 0.5\nsamples = 2138\nvalue =
[1081, 1057]'),
 Text(0.7270233196159122, 0.05, 'gini = 0.306\nsamples = 69\nvalue =
[56, 13]'),
 Text(0.7352537722908093, 0.15, 'X[4] <= 546.0\ngini = 0.431\nsamples
= 251\nvalue = [79, 172]'),
 Text(0.7325102880658436, 0.05, 'gini = 0.41\nsamples = 240\nvalue =
[69, 171]'),
```

```
 Text(0.7379972565157751, 0.05, 'gini = 0.165\nsamples = 11\nvalue =
[10, 1]'),
 Text(0.7846364883401921, 0.45, 'X[2] <= 14.5\ngini = 0.484\nsamples =
4222\nvalue = [1734, 2488]'),
 Text(0.7626886145404664, 0.35, 'X[0] <= 33.5\ngini = 0.492\nsamples =
3652\nvalue = [1595, 2057]'),
 Text(0.7517146776406035, 0.25, 'X[1] <= 429230.5\ngini = 0.494\
nsamples = 561\nvalue = [311, 250]'),
 Text(0.7462277091906722, 0.15, 'X[5] <= 49.0\ngini = 0.492\nsamples =
548\nvalue = [309, 239]'),
 Text(0.7434842249657064, 0.05, 'gini = 0.448\nsamples = 177\nvalue =
[117, 60]'),
 Text(0.7489711934156379, 0.05, 'gini = 0.499\nsamples = 371\nvalue =
[192, 179]'),
 Text(0.757201646090535, 0.15, 'X[1] <= 640071.0\ngini = 0.26\nsamples
= 13\nvalue = [2, 11]'),
 Text(0.7544581618655692, 0.05, 'gini = 0.153\nsamples = 12\nvalue =
[1, 11]'),
 Text(0.7599451303155007, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.7736625514403292, 0.25, 'X[1] <= 107864.0\ngini = 0.486\
nsamples = 3091\nvalue = [1284, 1807]'),
 Text(0.7681755829903978, 0.15, 'X[3] <= 1087.0\ngini = 0.5\nsamples =
762\nvalue = [379, 383]'),
 Text(0.7654320987654321, 0.05, 'gini = 0.5\nsamples = 732\nvalue =
[356, 376]'),
 Text(0.7709190672153635, 0.05, 'gini = 0.358\nsamples = 30\nvalue =
[23, 7]'),
 Text(0.7791495198902606, 0.15, 'X[3] <= 4973.5\ngini = 0.475\nsamples
= 2329\nvalue = [905, 1424]'),
 Text(0.7764060356652949, 0.05, 'gini = 0.474\nsamples = 2316\nvalue =
[892, 1424]'),
 Text(0.7818930041152263, 0.05, 'gini = 0.0\nsamples = 13\nvalue =
[13, 0]'),
 Text(0.8065843621399177, 0.35, 'X[0] <= 32.5\ngini = 0.369\nsamples =
570\nvalue = [139, 431]'),
 Text(0.7956104252400549, 0.25, 'X[2] <= 15.5\ngini = 0.482\nsamples =
42\nvalue = [25, 17]'),
 Text(0.7901234567901234, 0.15, 'X[1] <= 340889.5\ngini = 0.499\
nsamples = 29\nvalue = [14, 15]'),
 Text(0.7873799725651578, 0.05, 'gini = 0.48\nsamples = 25\nvalue =
[10, 15]'),
 Text(0.7928669410150891, 0.05, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.8010973936899863, 0.15, 'X[1] <= 401751.5\ngini = 0.26\
nsamples = 13\nvalue = [11, 2]'),
 Text(0.7983539094650206, 0.05, 'gini = 0.153\nsamples = 12\nvalue =
[11, 1]'),
 Text(0.803840877914952, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
```

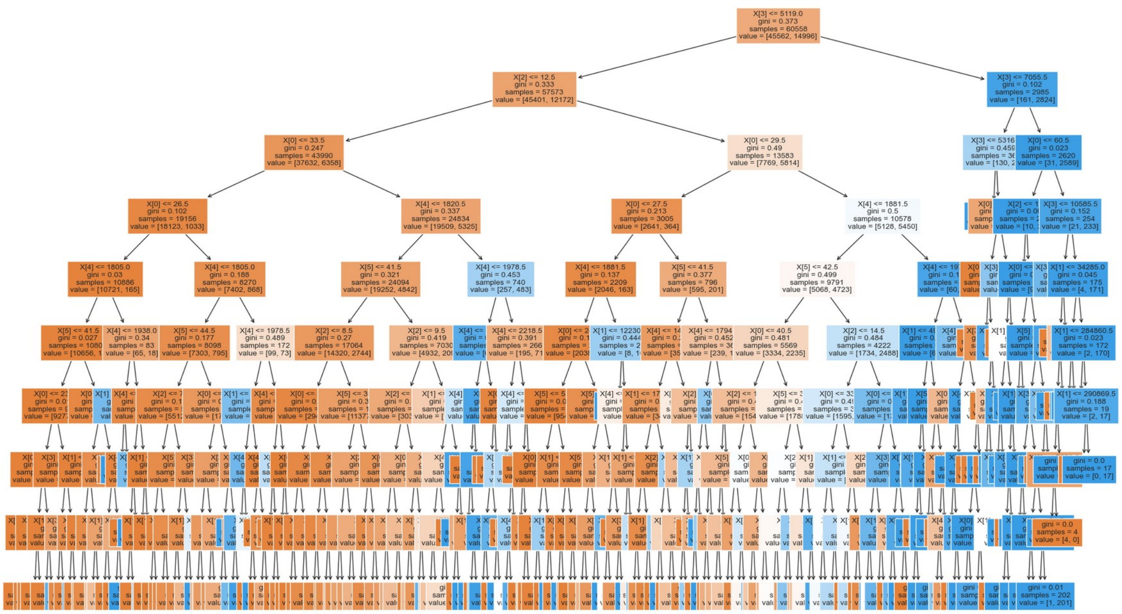
```
 Text(0.8175582990397805, 0.25, 'X[3] <= 543.0\ngini = 0.339\nsamples
= 528\nvalue = [114, 414]'),
 Text(0.8120713305898491, 0.15, 'X[1] <= 39267.5\ngini = 0.326\
nsamples = 516\nvalue = [106, 410]'),
 Text(0.8093278463648834, 0.05, 'gini = 0.5\nsamples = 20\nvalue =
[10, 10]'),
 Text(0.8148148148148148, 0.05, 'gini = 0.312\nsamples = 496\nvalue =
[96, 400]'),
 Text(0.823045267489712, 0.15, 'X[3] <= 3370.0\ngini = 0.444\nsamples
= 12\nvalue = [8, 4]'),
 Text(0.8203017832647462, 0.05, 'gini = 0.0\nsamples = 7\nvalue = [7,
0]'),
 Text(0.8257887517146777, 0.05, 'gini = 0.32\nsamples = 5\nvalue = [1,
4]'),
 Text(0.8641975308641975, 0.55, 'X[4] <= 1978.5\ngini = 0.141\nsamples
= 787\nvalue = [60, 727]'),
 Text(0.8463648834019204, 0.45, 'X[1] <= 49159.0\ngini = 0.021\
nsamples = 572\nvalue = [6, 566]'),
 Text(0.8395061728395061, 0.35, 'X[1] <= 48181.5\ngini = 0.172\
nsamples = 21\nvalue = [2, 19]'),
 Text(0.8367626886145405, 0.25, 'X[1] <= 33199.5\ngini = 0.095\
nsamples = 20\nvalue = [1, 19]'),
 Text(0.8340192043895748, 0.15, 'X[1] <= 32922.0\ngini = 0.198\
nsamples = 9\nvalue = [1, 8]'),
 Text(0.831275720164609, 0.05, 'gini = 0.0\nsamples = 8\nvalue = [0,
8]'),
 Text(0.8367626886145405, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.8395061728395061, 0.15, 'gini = 0.0\nsamples = 11\nvalue = [0,
11]'),
 Text(0.8422496570644719, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.8532235939643347, 0.35, 'X[5] <= 31.0\ngini = 0.014\nsamples =
551\nvalue = [4, 547]'),
 Text(0.8477366255144033, 0.25, 'X[1] <= 96699.0\ngini = 0.219\
nsamples = 8\nvalue = [1, 7]'),
 Text(0.8449931412894376, 0.15, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.850480109739369, 0.15, 'gini = 0.0\nsamples = 7\nvalue = [0,
7]'),
 Text(0.8587105624142661, 0.25, 'X[5] <= 41.0\ngini = 0.011\nsamples =
543\nvalue = [3, 540]'),
 Text(0.8559670781893004, 0.15, 'X[0] <= 50.5\ngini = 0.027\nsamples =
218\nvalue = [3, 215]'),
 Text(0.8532235939643347, 0.05, 'gini = 0.012\nsamples = 167\nvalue =
[1, 166]'),
 Text(0.8587105624142661, 0.05, 'gini = 0.075\nsamples = 51\nvalue =
[2, 49]'),
 Text(0.8614540466392319, 0.15, 'gini = 0.0\nsamples = 325\nvalue =
[0, 325]'),
```

```
 Text(0.8820301783264746, 0.45, 'X[4] <= 2358.0\ngini = 0.376\nsamples
= 215\nvalue = [54, 161]'),
 Text(0.8724279835390947, 0.35, 'X[0] <= 68.0\ngini = 0.424\nsamples =
72\nvalue = [50, 22]'),
 Text(0.869684499314129, 0.25, 'X[4] <= 2151.5\ngini = 0.342\nsamples
= 64\nvalue = [50, 14]'),
 Text(0.8669410150891632, 0.15, 'gini = 0.0\nsamples = 23\nvalue =
[23, 0]'),
 Text(0.8724279835390947, 0.15, 'X[4] <= 2252.0\ngini = 0.45\nsamples
= 41\nvalue = [27, 14]'),
 Text(0.869684499314129, 0.05, 'gini = 0.426\nsamples = 13\nvalue =
[4, 9]'),
 Text(0.8751714677640604, 0.05, 'gini = 0.293\nsamples = 28\nvalue =
[23, 5]'),
 Text(0.8751714677640604, 0.25, 'gini = 0.0\nsamples = 8\nvalue = [0,
8]'),
 Text(0.8916323731138546, 0.35, 'X[4] <= 3726.5\ngini = 0.054\nsamples
= 143\nvalue = [4, 139]'),
 Text(0.8888888888888888, 0.25, 'X[4] <= 2384.5\ngini = 0.028\nsamples
= 141\nvalue = [2, 139]'),
 Text(0.8834019204389575, 0.15, 'X[0] <= 54.5\ngini = 0.198\nsamples =
9\nvalue = [1, 8]'),
 Text(0.8806584362139918, 0.05, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.8861454046639232, 0.05, 'gini = 0.0\nsamples = 8\nvalue = [0,
8]'),
 Text(0.8943758573388203, 0.15, 'X[0] <= 65.5\ngini = 0.015\nsamples =
132\nvalue = [1, 131]'),
 Text(0.8916323731138546, 0.05, 'gini = 0.0\nsamples = 121\nvalue =
[0, 121]'),
 Text(0.897119341563786, 0.05, 'gini = 0.165\nsamples = 11\nvalue =
[1, 10]'),
 Text(0.8943758573388203, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.9340706447187929, 0.85, 'X[3] <= 7055.5\ngini = 0.102\nsamples
= 2985\nvalue = [161, 2824]'),
 Text(0.9094650205761317, 0.75, 'X[3] <= 5316.5\ngini = 0.459\nsamples
= 365\nvalue = [130, 235]'),
 Text(0.906721536351166, 0.65, 'gini = 0.0\nsamples = 196\nvalue = [0,
196]'),
 Text(0.9122085048010974, 0.65, 'X[0] <= 61.0\ngini = 0.355\nsamples =
169\nvalue = [130, 39]'),
 Text(0.9026063100137174, 0.55, 'X[0] <= 54.5\ngini = 0.017\nsamples =
115\nvalue = [114, 1]'),
 Text(0.8998628257887518, 0.45, 'gini = 0.0\nsamples = 107\nvalue =
[107, 0]'),
 Text(0.9053497942386831, 0.45, 'X[3] <= 6457.5\ngini = 0.219\nsamples
= 8\nvalue = [7, 1]'),
 Text(0.9026063100137174, 0.35, 'X[3] <= 5936.5\ngini = 0.5\nsamples =
2\nvalue = [1, 1]'),
```

```
 Text(0.8998628257887518, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.9053497942386831, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.9080932784636488, 0.35, 'gini = 0.0\nsamples = 6\nvalue = [6,
0]'),
 Text(0.9218106995884774, 0.55, 'X[3] <= 6618.5\ngini = 0.417\nsamples
= 54\nvalue = [16, 38]'),
 Text(0.9190672153635117, 0.45, 'X[1] <= 50818.5\ngini = 0.172\
nsamples = 42\nvalue = [4, 38]'),
 Text(0.9135802469135802, 0.35, 'X[5] <= 30.0\ngini = 0.444\nsamples =
3\nvalue = [2, 1]'),
 Text(0.9108367626886146, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.9163237311385459, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.9245541838134431, 0.35, 'X[5] <= 37.5\ngini = 0.097\nsamples =
39\nvalue = [2, 37]'),
 Text(0.9218106995884774, 0.25, 'X[3] <= 6389.0\ngini = 0.32\nsamples
= 10\nvalue = [2, 8]'),
 Text(0.9190672153635117, 0.15, 'X[1] <= 220421.5\ngini = 0.5\nsamples
= 4\nvalue = [2, 2]'),
 Text(0.9163237311385459, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.9218106995884774, 0.05, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(0.9245541838134431, 0.15, 'gini = 0.0\nsamples = 6\nvalue = [0,
6]'),
 Text(0.9272976680384087, 0.25, 'gini = 0.0\nsamples = 29\nvalue = [0,
29]'),
 Text(0.9245541838134431, 0.45, 'gini = 0.0\nsamples = 12\nvalue =
[12, 0]'),
 Text(0.9586762688614541, 0.75, 'X[0] <= 60.5\ngini = 0.023\nsamples =
2620\nvalue = [31, 2589]'),
 Text(0.9379286694101509, 0.65, 'X[2] <= 1.5\ngini = 0.008\nsamples =
2366\nvalue = [10, 2356]'),
 Text(0.9351851851851852, 0.55, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.9406721536351166, 0.55, 'X[0] <= 20.0\ngini = 0.008\nsamples =
2365\nvalue = [9, 2356]'),
 Text(0.9327846364883402, 0.45, 'X[1] <= 165776.5\ngini = 0.5\nsamples
= 4\nvalue = [2, 2]'),
 Text(0.9300411522633745, 0.35, 'gini = 0.0\nsamples = 2\nvalue = [0,
2]'),
 Text(0.9355281207133059, 0.35, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.948559670781893, 0.45, 'X[5] <= 87.0\ngini = 0.006\nsamples =
2361\nvalue = [7, 2354]'),
 Text(0.9410150891632373, 0.35, 'X[1] <= 22618.5\ngini = 0.005\
nsamples = 2352\nvalue = [6, 2346]'),
```

```
 Text(0.934156378600823, 0.25, 'X[3] <= 7565.5\ngini = 0.153\nsamples
= 12\nvalue = [1, 11]'),
 Text(0.9314128943758574, 0.15, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.9368998628257887, 0.15, 'gini = 0.0\nsamples = 11\nvalue = [0,
11]'),
 Text(0.9478737997256516, 0.25, 'X[3] <= 7565.5\ngini = 0.004\nsamples
= 2340\nvalue = [5, 2335]'),
 Text(0.9423868312757202, 0.15, 'X[3] <= 7436.5\ngini = 0.017\nsamples
= 477\nvalue = [4, 473]'),
 Text(0.9396433470507545, 0.05, 'gini = 0.0\nsamples = 473\nvalue =
[0, 473]'),
 Text(0.9451303155006858, 0.05, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.953360768175583, 0.15, 'X[0] <= 54.5\ngini = 0.001\nsamples =
1863\nvalue = [1, 1862]'),
 Text(0.9506172839506173, 0.05, 'gini = 0.0\nsamples = 1661\nvalue =
[0, 1661]'),
 Text(0.9561042524005487, 0.05, 'gini = 0.01\nsamples = 202\nvalue =
[1, 201]'),
 Text(0.9561042524005487, 0.35, 'X[3] <= 28167.0\ngini = 0.198\
nsamples = 9\nvalue = [1, 8]'),
 Text(0.953360768175583, 0.25, 'gini = 0.0\nsamples = 8\nvalue = [0,
8]'),
 Text(0.9588477366255144, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.9794238683127572, 0.65, 'X[3] <= 10585.5\ngini = 0.152\
nsamples = 254\nvalue = [21, 233]'),
 Text(0.9725651577503429, 0.55, 'X[3] <= 10543.0\ngini = 0.338\
nsamples = 79\nvalue = [17, 62]'),
 Text(0.9698216735253772, 0.45, 'X[5] <= 35.5\ngini = 0.114\nsamples =
66\nvalue = [4, 62]'),
 Text(0.9670781893004116, 0.35, 'X[3] <= 8682.0\ngini = 0.375\nsamples
= 16\nvalue = [4, 12]'),
 Text(0.9643347050754458, 0.25, 'X[3] <= 7579.0\ngini = 0.32\nsamples
= 5\nvalue = [4, 1]'),
 Text(0.9615912208504801, 0.15, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.9670781893004116, 0.15, 'gini = 0.0\nsamples = 4\nvalue = [4,
0]'),
 Text(0.9698216735253772, 0.25, 'gini = 0.0\nsamples = 11\nvalue = [0,
11]'),
 Text(0.9725651577503429, 0.35, 'gini = 0.0\nsamples = 50\nvalue = [0,
50]'),
 Text(0.9753086419753086, 0.45, 'gini = 0.0\nsamples = 13\nvalue =
[13, 0]'),
 Text(0.9862825788751715, 0.55, 'X[1] <= 34285.0\ngini = 0.045\
nsamples = 175\nvalue = [4, 171]'),
 Text(0.9807956104252401, 0.45, 'X[2] <= 9.5\ngini = 0.444\nsamples =
3\nvalue = [2, 1]'),
```

```
 Text(0.9780521262002744, 0.35, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.9835390946502057, 0.35, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(0.9917695473251029, 0.45, 'X[1] <= 284860.5\ngini = 0.023\
nsamples = 172\nvalue = [2, 170]'),
 Text(0.9890260631001372, 0.35, 'gini = 0.0\nsamples = 153\nvalue =
[0, 153]'),
 Text(0.9945130315500685, 0.35, 'X[1] <= 290869.5\ngini = 0.188\
nsamples = 19\nvalue = [2, 17]'),
 Text(0.9917695473251029, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [2,
0]'),
 Text(0.9972565157750343, 0.25, 'gini = 0.0\nsamples = 17\nvalue = [0,
17]')]
```



4.3 Bagging Classifier

```
from sklearn.ensemble import BaggingClassifier

bagg_cls = BaggingClassifier()

# Model training

bagg_cls.fit(X_train, y_train)

BaggingClassifier()

# Predicting values

y_pred_bc = bagg_cls.predict(X_test)
```

5.1 Accuracy Score

```
# Bagging Classifier Model's accuracy score

print("Bagging Classifier training accuracy score is : {}
%".format(round(bagg_cls.score(X_train, y_train)*100, 2)))
print("Bagging Classifier model's accuracy score is : {}
%".format(round(accuracy_score(y_test, y_pred_bc)*100, 2)))
```

```
Bagging Classifier training accuracy score is : 99.14%
Bagging Classifier model's accuracy score is : 91.79%
```

Observation :

- This is an over-fitted model

5.2 Roc-auc score

```
y_train_predict_roc_bc = bagg_cls.predict_proba(X_train)

print("Bagging Classifier model's training roc-auc score is : {}
%".format(round(roc_auc_score(y_train,
y_train_predict_roc_bc[:,1])*100)))

y_test_predict_roc_bc = bagg_cls.predict_proba(X_test)

print("Bagging Classifier model's roc-auc accuracy score is : {}
%".format(round(roc_auc_score(y_test,
y_test_predict_roc_bc[:,1])*100)))
```

```
Bagging Classifier model's training roc-auc score is : 100%
Bagging Classifier model's roc-auc accuracy score is : 94%
```

5.3 Confusion_matrix

```
conf_mat_bc = confusion_matrix(y_test, y_pred_bc)
conf_mat_bc
```

```
array([[21572,   840],
       [ 1608,  5808]], dtype=int64)
```

```
true_positive = conf_mat_bc[0][0]
false_positive = conf_mat_bc[0][1]
false_negative = conf_mat_bc[1][0]
true_negative = conf_mat_bc[1][0]
print('True Positive:',true_positive, '\nTrue
Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse
Positive:',false_positive)
```

```
True Positive: 21572
True Negative: 1608
False Negative: 1608
False Positive: 840
```

Classification Report

```
class_reprt_log_reg = classification_report(y_test, y_pred_bc)
print(class_reprt_log_reg)
```

```
              precision    recall  f1-score   support

           0       0.93      0.96      0.95     22412
           1       0.87      0.78      0.83      7416

    accuracy                           0.92     29828
   macro avg       0.90      0.87      0.89     29828
weighted avg       0.92      0.92      0.92     29828
```

4.4 Grid Search CV : Hyperparameter tuning

Bagging Classifier

```
from sklearn.model_selection import GridSearchCV

# Defining parameters for hyper parameters

grid_params = {'n_estimators' : [5, 10, 15],
        'max_samples' : range(2, 10, 1),
        'max_features' : range(2, 10, 3)
        }

grid_search = GridSearchCV(estimator=bagg_cls, param_grid=grid_params,
verbose=2, n_jobs=-1, cv=3)

# Hyper parameter tuning

grid_search_bc = grid_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
# Finding the best parameters

grid_search_bc.best_params_
```

```
{'max_features': 5, 'max_samples': 8, 'n_estimators': 15}
```

```
model_with_best_params_bc = BaggingClassifier(max_features = 5,
max_samples = 7, n_estimators = 10, oob_score=True)

model_with_best_params_bc.fit(X_train, y_train)
```

```
BaggingClassifier(max_features=5, max_samples=7, oob_score=True)
```

```
y_pred_bst_est_bc = model_with_best_params_bc.predict(X_test)
```

5.1 Accuracy Score

```
# Bagging Classifier Model's accuracy score after hyper parameter
tuning

print("Bagging Classifier with best parameter training accuracy score
is : {}%".format(round(model_with_best_params_bc.score(X_train,
y_train)*100, 2)))
print("Bagging Classifier with best parameter model's accuracy score
is : {}%".format(round(accuracy_score(y_test, y_pred_bst_est_bc)*100,
2)))
```

```
Bagging Classifier with best parameter training accuracy score is :
77.09%
Bagging Classifier with best parameter model's accuracy score is :
76.99%
```

5.2 Roc-auc score

```
y_train_predict_bc_bst_prm =
model_with_best_params_bc.predict_proba(X_train)

print("Bagging Classifier model with best parameter training training
roc-auc score is : {}%".format(round(roc_auc_score(y_train,
y_train_predict_bc_bst_prm[:,1])*100)))

y_test_predict_roc_bc_bst_prm =
model_with_best_params_bc.predict_proba(X_test)

print("Bagging Classifier model with best parameter training roc-auc
accuracy score is : {}%".format(round(roc_auc_score(y_test,
y_test_predict_roc_bc_bst_prm[:,1])*100)))
```

```
Bagging Classifier model with best parameter training training roc-auc
score is : 73%
Bagging Classifier model with best parameter training roc-auc accuracy
score is : 72%
```

5.3 Confusion_matrix

```
conf_mat_bc_bst_prm = confusion_matrix(y_test, y_pred_bc)
conf_mat_bc_bst_prm
```

```
array([[21572,   840],
       [ 1608,  5808]], dtype=int64)
```

```
true_positive = conf_mat_bc_bst_prm[0][0]
false_positive = conf_mat_bc_bst_prm[0][1]
false_negative = conf_mat_bc_bst_prm[1][0]
true_negative = conf_mat_bc_bst_prm[1][0]
print('True Positive:',true_positive, '\nTrue
Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse
Positive:',false_positive)
```

```
True Positive: 21572
True Negative: 1608
False Negative: 1608
False Positive: 840
```

Classification Report

```
class_reprt_log_reg = classification_report(y_test, y_pred_bst_est_bc)
print(class_reprt_log_reg)
```

```
              precision    recall  f1-score   support

           0       0.77      1.00      0.87     22412
           1       0.92      0.08      0.15      7416

    accuracy                           0.77     29828
   macro avg       0.85      0.54      0.51     29828
weighted avg       0.81      0.77      0.69     29828
```

4.5 Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

rfc.fit(X_train, y_train)

RandomForestClassifier()

y_pred_rfc = rfc.predict(X_test)
```

5.1 Accuracy Score

```
# Random Forest Classifier Model's accuracy score

print("Random Forest Classifier training accuracy score is : {}
%".format(round(rfc.score(X_train, y_train)*100, 2)))
print("Random Forest model's accuracy score is : {}
%".format(round(accuracy_score(y_test, y_pred_rfc)*100, 2)))

Random Forest Classifier training accuracy score is : 99.85%
Random Forest model's accuracy score is : 92.99%
```

Observation :

- Over-fitted model

5.2 Roc-auc score

```
y_train_predict_roc_rfc = rfc.predict_proba(X_train)

print("Random Forest Classifier model with best parameter training
training roc-auc score is : {}%".format(round(roc_auc_score(y_train,
y_train_predict_roc_rfc[:,1])*100)))
```

```
y_test_predict_roc_rfc = rfc.predict_proba(X_test)

print("Random Forest Classifier model with best parameter training roc-auc accuracy score is : {}%".format(round(roc_auc_score(y_test, y_test_predict_roc_rfc[:,1])*100)))
```

```
Random Forest Classifier model with best parameter training training roc-auc score is : 100%
Random Forest Classifier model with best parameter training roc-auc accuracy score is : 95%
```

5.3 Confusion_matrix

```
conf_mat_rfc = confusion_matrix(y_test, y_pred_rfc)
conf_mat_rfc
```

```
array([[21484,   928],
       [ 1164,  6252]], dtype=int64)
```

```
true_positive = conf_mat_rfc[0][0]
false_positive = conf_mat_rfc[0][1]
false_negative = conf_mat_rfc[1][0]
true_negative = conf_mat_rfc[1][0]
print('True Positive:',true_positive, '\nTrue Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse Positive:',false_positive)
```

```
True Positive: 21484
True Negative: 1164
False Negative: 1164
False Positive: 928
```

Classification Report

```
class_reprt_rfc = classification_report(y_test, y_pred_rfc)
print(class_reprt_rfc)
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     22412
           1       0.87      0.84      0.86      7416

    accuracy                           0.93     29828
   macro avg       0.91      0.90      0.91     29828
weighted avg       0.93      0.93      0.93     29828
```

4.6 Grid Search CV : Hyperparameter tuning

Random Forest Classifier

```
grid_params = { 'criterion' : ['gini', 'entropy', 'log_loss'],
                'max_depth' : range(1, 10, 1),
                'min_samples_split' : range(2, 10, 2),
                'min_samples_leaf' : range(1, 10, 1),
}

grid_search = GridSearchCV(estimator=rfc, param_grid=grid_params,
n_jobs=-1, verbose=2, cv=3)

grid_search_rfc = grid_search.fit(X_train, y_train)

Fitting 3 folds for each of 972 candidates, totalling 2916 fits

## Finding the best parameters

grid_search_rfc.best_params_

{'criterion': 'gini',
 'max_depth': 9,
 'min_samples_leaf': 1,
 'min_samples_split': 2}

# default n-estimators value is 100

model_with_bst_est_rfc = RandomForestClassifier(criterion = 'gini',
max_depth = 9, min_samples_leaf = 1, min_samples_split = 4, verbose=1,
n_jobs=2)

model_with_bst_est_rfc.fit(X_train, y_train)

[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent
workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    1.1s
[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    2.4s finished

RandomForestClassifier(max_depth=9, min_samples_split=4, n_jobs=2,
verbose=1)

y_pred_bst_est_rfc = model_with_bst_est_rfc.predict(X_test)

[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent
workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.1s finished
```

5.1 Accuracy Score

```
# Random Forest Classifier model accuracy after hyper-parameter tuning

print("Random Forest Classifier best parameter training accuracy score
```

```
is : {}%".format(round(model_with_bst_est_rfc.score(X_train,
y_train)*100, 2)))
print("Random Forest Classifier best parameter model's accuracy score
is : {}%".format(round(accuracy_score(y_test, y_pred_bst_est_rfc)*100,
2)))
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent
workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.1s

Random Forest Classifier best parameter training accuracy score is :
83.98%
Random Forest Classifier best parameter model's accuracy score is :
83.23%

[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.3s finished
```

5.2 Roc-auc score

```
y_train_predict_roc_rfc_bst_est =
model_with_bst_est_rfc.predict_proba(X_train)

print("Random Forest Classifier model with best parameter training
training roc-auc score is : {}%".format(round(roc_auc_score(y_train,
y_train_predict_roc_rfc_bst_est[:,1])*100)))

y_test_predict_roc_rfc_bst_est =
model_with_bst_est_rfc.predict_proba(X_test)

print("Random Forest Classifier model with best parameter training
roc-auc accuracy score is : {}%".format(round(roc_auc_score(y_test,
y_test_predict_roc_rfc_bst_est[:,1])*100)))
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent
workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.3s finished
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent
workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:    0.0s

Random Forest Classifier model with best parameter training training
roc-auc score is : 87%
Random Forest Classifier model with best parameter training roc-auc
accuracy score is : 86%

[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed:    0.1s finished
```

5.3 Confusion_matrix

```
conf_mat_rfc_bst_est = confusion_matrix(y_test, y_pred_bst_est_rfc)
conf_mat_rfc_bst_est
```

```
array([[21569,   843],
       [ 4158,  3258]], dtype=int64)
```

```
true_positive = conf_mat_rfc_bst_est[0][0]
false_positive = conf_mat_rfc_bst_est[0][1]
false_negative = conf_mat_rfc_bst_est[1][0]
true_negative = conf_mat_rfc_bst_est[1][0]
print('True Positive:',true_positive, '\nTrue
Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse
Positive:',false_positive)
```

```
True Positive: 21569
True Negative: 4158
False Negative: 4158
False Positive: 843
```

Classification Report

```
class_reprt_rfc_bst_prm = classification_report(y_test,
y_pred_bst_est_rfc)
print(class_reprt_rfc_bst_prm)
```

```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90     22412
           1       0.79      0.44      0.57      7416

    accuracy                           0.83     29828
   macro avg       0.82      0.70      0.73     29828
weighted avg       0.83      0.83      0.81     29828
```

4.7 Extra Trees Classifier

```
from sklearn.ensemble import ExtraTreesClassifier
```

```
etc = ExtraTreesClassifier()
```

```
etc.fit(X_train, y_train)
```

```
ExtraTreesClassifier()
```

```
y_pred_etc = etc.predict(X_test)
```

5.1 Accuracy Score

```
# Random Forest Classifier model accuracy after hyper-parameter tuning

print("Extra Trees Classifier training accuracy score is : {}
%".format(round(etc.score(X_train, y_train)*100, 2)))
print("Extra Trees Classifier model's accuracy score is : {}
%".format(round(accuracy_score(y_test, y_pred_etc)*100, 2)))
```

```
Extra Trees Classifier training accuracy score is : 99.86%
Extra Trees Classifier model's accuracy score is : 92.53%
```

5.2 Roc-auc score

```
y_train_predict_roc_etc = etc.predict_proba(X_train)

print("Extra Trees Classifier model with best parameter training
training roc-auc score is : {}%".format(round(roc_auc_score(y_train,
y_train_predict_roc_etc[:,1])*100)))

y_test_predict_roc_etc = etc.predict_proba(X_test)

print("Extra Trees Classifier model with best parameter training roc-
auc accuracy score is : {}%".format(round(roc_auc_score(y_test,
y_test_predict_roc_etc[:,1])*100)))
```

```
Extra Trees Classifier model with best parameter training training
roc-auc score is : 100%
Extra Trees Classifier model with best parameter training roc-auc
accuracy score is : 97%
```

5.3 Confusion_matrix

```
conf_mat_etc = confusion_matrix(y_test, y_pred_etc)
conf_mat_etc
```

```
array([[21392,  1020],
       [ 1209,  6207]], dtype=int64)
```

```
true_positive = conf_mat_etc[0][0]
false_positive = conf_mat_etc[0][1]
false_negative = conf_mat_etc[1][0]
true_negative = conf_mat_etc[1][0]
print('True Positive:',true_positive, '\nTrue
Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse
Positive:',false_positive)
```

```
True Positive: 21392
True Negative: 1209
False Negative: 1209
False Positive: 1020
```

Classification Report

```
class_reprt_etc= classification_report(y_test, y_pred_etc)
print(class_reprt_etc)
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95     22412
           1       0.86      0.84      0.85      7416

    accuracy                           0.93     29828
```

```
   macro avg       0.90      0.90      0.90     29828
weighted avg       0.92      0.93      0.92     29828
```

4.8 HyperParameter Tuning

Extra Tree Classifier

```
grid_params = {
    'n_estimators' : [10,20,30],
    'criterion' : ['gini', 'entropy', 'log_loss'],
    'max_depth' : range(2,10,1),
    'min_samples_split' : range(2,10,2),
    'min_samples_leaf' : range(1,5,1),
    'max_features' : ['sqrt', 'log2']
}
```

```
grid_search = GridSearchCV(estimator=ExtraTreesClassifier(),
param_grid=grid_params, n_jobs=4, verbose=3, cv=3)
```

```
grid_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 2304 candidates, totalling 6912 fits
```

```
GridSearchCV(cv=3, estimator=ExtraTreesClassifier(), n_jobs=4,
             param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_depth': range(2, 10),
                         'max_features': ['sqrt', 'log2'],
                         'min_samples_leaf': range(1, 5),
                         'min_samples_split': range(2, 10, 2),
                         'n_estimators': [10, 20, 30]},
             verbose=3)
```

```
grid_search.best_params_
```

```
{'criterion': 'log_loss',
 'max_depth': 9,
 'max_features': 'sqrt',
 'min_samples_leaf': 2,
 'min_samples_split': 8,
 'n_estimators': 10}
```

```
model_with_bst_prm_etc = ExtraTreesClassifier(criterion = 'log_loss',
 max_depth = 9,
 max_features = 'sqrt',
 min_samples_leaf = 2,
 min_samples_split = 8,
 n_estimators = 10)
```

```
model_with_bst_prm_etc.fit(X_train, y_train)
```

```
ExtraTreesClassifier(criterion='log_loss', max_depth=9,
min_samples_leaf=2,
                     min_samples_split=8, n_estimators=10)
```

```
y_pred_etc_bst_prm = model_with_bst_prm_etc.predict(X_test)
```

5.1 Accuracy Score

```
# Random Forest Classifier model accuracy after hyper-parameter tuning

print("Extra Trees Classifier best param training accuracy score is :
{}%".format(round(model_with_bst_prm_etc.score(X_train, y_train)*100,
2)))
print("Extra Trees Classifier best param model's accuracy score is :
{}%".format(round(accuracy_score(y_test, y_pred_etc_bst_prm)*100, 2)))
```

```
Extra Trees Classifier best param training accuracy score is : 81.18%
Extra Trees Classifier best param model's accuracy score is : 80.94%
```

5.2 Roc-auc score

```
y_train_predict_roc_etc_bst_prm =
model_with_bst_prm_etc.predict_proba(X_train)

print("Extra Trees Classifier model with best parameter training
training roc-auc score is : {}%".format(round(roc_auc_score(y_train,
y_train_predict_roc_etc_bst_prm[:,1])*100)))

y_test_predict_roc_etc_bst_prm =
model_with_bst_prm_etc.predict_proba(X_test)

print("Extra Trees Classifier model with best parameter training roc-
auc accuracy score is : {}%".format(round(roc_auc_score(y_test,
y_test_predict_roc_etc_bst_prm[:,1])*100)))
```

```
Extra Trees Classifier model with best parameter training training
roc-auc score is : 84%
Extra Trees Classifier model with best parameter training roc-auc
accuracy score is : 83%
```

5.3 Confusion_matrix

```
conf_mat_etc_bst_prm = confusion_matrix(y_test, y_pred_etc_bst_prm)
conf_mat_etc_bst_prm
```

```
array([[22105,   307],
       [ 5378,  2038]], dtype=int64)
```

```
true_positive = conf_mat_etc_bst_prm[0][0]
false_positive = conf_mat_etc_bst_prm[0][1]
false_negative = conf_mat_etc_bst_prm[1][0]
true_negative = conf_mat_etc_bst_prm[1][0]
print('True Positive:',true_positive, '\nTrue
```

```
Negative:',true_negative, '\nFalse Negative:',false_negative, '\nFalse 
Positive:',false_positive)
```

```
True Positive: 22105
True Negative: 5378
False Negative: 5378
False Positive: 307
```

Classification Report

```
class_reprt_etc_bst_prm = classification_report(y_test, 
y_pred_etc_bst_prm)
print(class_reprt_etc_bst_prm)
```

```
              precision    recall  f1-score   support

           0       0.80      0.99      0.89     22412
           1       0.87      0.27      0.42      7416

    accuracy                           0.81     29828
   macro avg       0.84      0.63      0.65     29828
weighted avg       0.82      0.81      0.77     29828
```

4.9 Voting Classifier

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

lr = LogisticRegression(multi_class='multinomial', random_state=7)
rfc = RandomForestClassifier(n_estimators=50, random_state=7)
svc = SVC(probability=True, random_state=7)

# estimators take list of decision makers  (Boosting)

vc_hard = VotingClassifier(estimators = [('lr', lr), ('rfc', rfc), 
('svc', svc)], voting='hard')
vc_soft = VotingClassifier(estimators = [('lr', lr), ('rfc', rfc), 
('svc', svc)], voting='soft')

model_vc_hard = vc_hard.fit(X_train, y_train)

model_vc_soft = vc_soft.fit(X_train, y_train)

y_pred_vc_hard = model_vc_hard.predict(X_test)
y_pred_vc_soft = model_vc_soft.predict(X_test)

## Accuracy wrt hard_voting

print(f'Accuracy score in hard voting classifier model is :
{round(accuracy_score(y_test, y_pred_vc_hard)*100, 2)}%')
```

```
Accuracy score in hard voting classifier model is : 81.16%
```

```python
## Accuracy wrt soft_voting

print(f'Accuracy score in soft voting classifier model is :
{round(accuracy_score(y_test, y_pred_vc_soft)*100, 2)}%')
```

```
Accuracy score in soft voting classifier model is : 82.11%
```