Problem Statement:

- **Classification problem**
  - Collect dataset from here https://archive.ics.uci.edu/ml/datasets/census+income
  - Here we have missing values also.
  - So here perform EDA, Data wrangling, Data Pre processing
  - Now make a Classification model to find how many people are `>50k` and how many are `<=50k`.
  - Here create `Logistic Regression`, `SVM`.
- **Steps to be followed**
  - Data ingestion.
  - EDA (end to end).
  - Preprocessing of the data.
  - Use pickle to store the scaling of the data for later use.
  - Store the final processed data inside MongoDB.
  - Again load the data from MongoDB.
  - Model building.
  - Use `GridSearchCV` for hyper parameter tuning.
  - Evaluation.
    - `Confusion Matrix`, `ROC` and `AUC` for classification model.

Attribute Information:

1. **age:** continuous.
2. **workclass:** Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. **fnlwgt:** continuous.
4. **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.
5. **education-num:** continuous.
6. **marital-status:** Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. **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.
8. **relationship:** Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. **race:** White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. **sex:** Female, Male.
11. **capital-gain:** continuous.
12. **capital-loss:** continuous.
13. **hours-per-week:** continuous.

14. **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.

1. Data Ingestion:

1.1 Import modules and data to create dataframe

```
# Importing the required libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import pymongo

sns.set()
%matplotlib inline
warnings.filterwarnings('ignore')

# Creating final csv file

df = pd.read_csv('dataset/adult.csv', header=None)
df.rename(columns={0:'age', 1:'workclass', 2:'fnlwgt', 3:'education',
4:'education-num', 5:'marital-status',
                  6:'occupation', 7:'relationship', 8:'race',
9:'sex', 10:'capital-gain', 11:'capital-loss',
                  12:'hours-per-week', 13:'native-country',
14:'income'}, inplace=True)
df.to_csv('dataset/adult_final.csv', index=False)
```

1.2 Creating dataframe

```
data = pd.read_csv('dataset/adult_final.csv')
data.head()

   age         workclass  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  income

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
```

```
data.shape
```

```
(32561, 15)
```

```
# Checking the data types

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  native-country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Observations:

- There are 32561 rows with 15 columns.

2. Data Cleaning

```
# Name of the columns

data.columns
```

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

2.1 Checking all the unique values in each columns

```
for column in data.columns:
    print(f"The unique values in column {column}:")
    print(data[column].unique())
    print(f"\nThe number of unique values in {column} is:
{len(data[column].unique())}")
    print("----------------------------------\n")
```

```
The unique values in column age:
[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]

The number of unique values in age is: 73
----------------------------------

The unique values in column workclass:
[' State-gov' ' Self-emp-not-inc' ' Private' ' Federal-gov' ' Local-
gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked']

The number of unique values in workclass is: 9
----------------------------------

The unique values in column fnlwgt:
[ 77516  83311 215646 ...  34066  84661 257302]

The number of unique values in fnlwgt is: 21648
----------------------------------
```

```
The unique values in column education:
[' Bachelors' ' HS-grad' ' 11th' ' Masters' ' 9th' ' Some-college'
 ' Assoc-acdm' ' Assoc-voc' ' 7th-8th' ' Doctorate' ' Prof-school'
 ' 5th-6th' ' 10th' ' 1st-4th' ' Preschool' ' 12th']

The number of unique values in education is: 16
----------------------------------

The unique values in column education-num:
[13  9  7 14  5 10 12 11  4 16 15  3  6  2  1  8]

The number of unique values in education-num is: 16
----------------------------------

The unique values in column marital-status:
[' Never-married' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' '
Widowed']

The number of unique values in marital-status is: 7
----------------------------------

The unique values in column occupation:
[' 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']

The number of unique values in occupation is: 15
----------------------------------

The unique values in column relationship:
[' Not-in-family' ' Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']

The number of unique values in relationship is: 6
----------------------------------

The unique values in column race:
[' White' ' Black' ' Asian-Pac-Islander' ' Amer-Indian-Eskimo' '
Other']

The number of unique values in race is: 5
----------------------------------

The unique values in column sex:
[' Male' ' Female']
```

```
The number of unique values in sex is: 2
----------------------------------

The unique values in column capital-gain:
[ 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]

The number of unique values in capital-gain is: 119
----------------------------------

The unique values in column capital-loss:
[   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]

The number of unique values in capital-loss is: 92
----------------------------------

The unique values in column hours-per-week:
[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]
```

```
The number of unique values in hours-per-week is: 94
----------------------------------

The unique values in column native-country:
[' 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']

The number of unique values in native-country is: 42
----------------------------------

The unique values in column income:
[' <=50K' ' >50K']

The number of unique values in income is: 2
----------------------------------
```

Checking number of occurence of each unique values

```
for column in data.columns:
    print(f"The number of occurence of each values in column {column}
is:")
    print(data[column].value_counts())
    print("----------------------------------\n\n")
```

```
The number of occurence of each values in column age is:
36    898
31    888
34    886
23    877
35    876
     ...
83      6
88      3
85      3
86      1
87      1
Name: age, Length: 73, dtype: int64
----------------------------------


The number of occurence of each values in column workclass is:
 Private             22696
```

```
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
----------------------------------


The number of occurence of each values in column fnlwgt is:
164190    13
203488    13
123011    13
148995    12
121124    12
          ..
232784     1
325573     1
140176     1
318264     1
257302     1
Name: fnlwgt, Length: 21648, dtype: int64
----------------------------------


The number of occurence of each values in column education is:
 HS-grad         10501
 Some-college     7291
 Bachelors        5355
 Masters          1723
 Assoc-voc        1382
 11th             1175
 Assoc-acdm       1067
 10th              933
 7th-8th           646
 Prof-school       576
 9th               514
 12th              433
 Doctorate         413
 5th-6th           333
 1st-4th           168
 Preschool          51
Name: education, dtype: int64
----------------------------------


The number of occurence of each values in column education-num is:
```

```
9     10501
10     7291
13     5355
14     1723
11     1382
7      1175
12     1067
6       933
4       646
15      576
5       514
8       433
16      413
3       333
2       168
1        51
Name: education-num, dtype: int64
----------------------------------


The number of occurence of each values in column marital-status is:
 Married-civ-spouse       14976
 Never-married            10683
 Divorced                  4443
 Separated                 1025
 Widowed                    993
 Married-spouse-absent      418
 Married-AF-spouse           23
Name: marital-status, dtype: int64
----------------------------------


The number of occurence of each values in column occupation is:
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
 ?                    1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64
----------------------------------
```

```
The number of occurence of each values in column relationship is:
 Husband           13193
 Not-in-family      8305
 Own-child          5068
 Unmarried          3446
 Wife               1568
 Other-relative      981
Name: relationship, dtype: int64
----------------------------------


The number of occurence of each values in column race is:
 White                 27816
 Black                  3124
 Asian-Pac-Islander     1039
 Amer-Indian-Eskimo      311
 Other                   271
Name: race, dtype: int64
----------------------------------


The number of occurence of each values in column sex is:
 Male      21790
 Female    10771
Name: sex, dtype: int64
----------------------------------


The number of occurence of each values in column capital-gain is:
0        29849
15024      347
7688       284
7298       246
99999      159
         ...
1111         1
2538         1
22040        1
4931         1
5060         1
Name: capital-gain, Length: 119, dtype: int64
----------------------------------


The number of occurence of each values in column capital-loss is:
0       31042
1902      202
```

```
1977      168
1887      159
1848       51
         ...
2080        1
1539        1
1844        1
2489        1
1411        1
Name: capital-loss, Length: 92, dtype: int64
----------------------------------


The number of occurence of each values in column hours-per-week is:
40    15217
50     2819
45     1824
60     1475
35     1297
      ...
82        1
92        1
87        1
74        1
94        1
Name: hours-per-week, Length: 94, dtype: int64
----------------------------------


The number of occurence of each values in column native-country is:
 United-States                 29170
 Mexico                          643
 ?                               583
 Philippines                     198
 Germany                         137
 Canada                          121
 Puerto-Rico                     114
 El-Salvador                     106
 India                           100
 Cuba                             95
 England                          90
 Jamaica                          81
 South                            80
 China                            75
 Italy                            73
 Dominican-Republic               70
 Vietnam                          67
 Guatemala                        64
 Japan                            62
 Poland                           60
```

```
 Columbia                         59
 Taiwan                           51
 Haiti                            44
 Iran                             43
 Portugal                         37
 Nicaragua                        34
 Peru                             31
 France                           29
 Greece                           29
 Ecuador                          28
 Ireland                          24
 Hong                             20
 Cambodia                         19
 Trinadad&Tobago                  19
 Laos                             18
 Thailand                         18
 Yugoslavia                       16
 Outlying-US(Guam-USVI-etc)       14
 Honduras                         13
 Hungary                          13
 Scotland                         12
 Holand-Netherlands                1
Name: native-country, dtype: int64
----------------------------------


The number of occurence of each values in column income is:
 <=50K    24720
 >50K      7841
Name: income, dtype: int64
----------------------------------
```

Observations:

- We have special character ? in columns `workclass`, `occupation`, `native-country`.
- Also the column `fnlwgt` has more than 1000 unique values. So we need to check for special characters in that as well.

```
# Searching for special character in 'fnlwgt' column

data.loc[data['fnlwgt'] == ' ?', :]
```

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

Observation:

- So there is no special character ? in the column `fnlwgt`.

Replacing the special character ? with most appeared value in that column

```
data['workclass'] = data['workclass'].str.replace('?','Private')
data['occupation'] = data['occupation'].str.replace('?','Prof-
specialty')
data['native-country'] = data['native-
country'].str.replace('?','United-States')
data.head()
```

```
   age         workclass  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  income

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
```

```
# Verifying the result

data.loc[data['workclass'] == '?', :]
```

```
Empty DataFrame
Columns: [age, workclass, fnlwgt, education, education-num, marital-
```

```
status, occupation, relationship, race, sex, capital-gain, capital-
loss, hours-per-week, native-country, income]
Index: []
```

```
data.loc[data['occupation'] == '?', :]
```

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

```
data.loc[data['native-country'] == '?', :]
```

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

Checking for null and duplicate values

```
# checking for null values

data.isnull().sum()
```

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

```
# checking for duplicate values

df[data.duplicated()]
```

```
       age          workclass  fnlwgt      education  education-num  \
4881    25            Private  308144      Bachelors             13
5104    90            Private   52386   Some-college             10
9171    21            Private  250051   Some-college             10
11631   20            Private  107658   Some-college             10
13084   25            Private  195994        1st-4th              2
```

```
15059   21            Private  243368      Preschool              1
17040   46            Private  173243        HS-grad              9
18555   30            Private  144593        HS-grad              9
18698   19            Private   97261        HS-grad              9
21318   19            Private  138153   Some-college             10
21490   19            Private  146679   Some-college             10
21875   49            Private   31267        7th-8th              4
22300   25            Private  195994        1st-4th              2
22367   44            Private  367749      Bachelors             13
22494   49   Self-emp-not-inc   43479   Some-college             10
25872   23            Private  240137        5th-6th              3
26313   28            Private  274679        Masters             14
28230   27            Private  255582        HS-grad              9
28522   42            Private  204235   Some-college             10
28846   39            Private   30916        HS-grad              9
29157   38            Private  207202        HS-grad              9
30845   46            Private  133616   Some-college             10
31993   19            Private  251579   Some-college             10
32404   35            Private  379959        HS-grad              9

            marital-status          occupation    relationship  \
4881         Never-married        Craft-repair   Not-in-family
5104         Never-married       Other-service   Not-in-family
9171         Never-married      Prof-specialty       Own-child
11631        Never-married        Tech-support   Not-in-family
13084        Never-married     Priv-house-serv   Not-in-family
15059        Never-married     Farming-fishing   Not-in-family
17040   Married-civ-spouse        Craft-repair         Husband
18555        Never-married       Other-service   Not-in-family
18698        Never-married     Farming-fishing   Not-in-family
21318        Never-married        Adm-clerical       Own-child
21490        Never-married     Exec-managerial       Own-child
21875   Married-civ-spouse        Craft-repair         Husband
22300        Never-married     Priv-house-serv   Not-in-family
22367        Never-married      Prof-specialty   Not-in-family
22494   Married-civ-spouse        Craft-repair         Husband
25872        Never-married   Handlers-cleaners   Not-in-family
26313        Never-married      Prof-specialty   Not-in-family
28230        Never-married   Machine-op-inspct   Not-in-family
28522   Married-civ-spouse      Prof-specialty         Husband
28846   Married-civ-spouse        Craft-repair         Husband
29157   Married-civ-spouse   Machine-op-inspct         Husband
30845             Divorced        Adm-clerical       Unmarried
31993        Never-married       Other-service       Own-child
32404             Divorced       Other-service   Not-in-family

                      race      sex  capital-gain  capital-loss  \
4881                 White     Male             0             0
5104    Asian-Pac-Islander     Male             0             0
9171                 White   Female             0             0
```

11631	White	Female	0	0
13084	White	Female	0	0
15059	White	Male	0	0
17040	White	Male	0	0
18555	Black	Male	0	0
18698	White	Male	0	0
21318	White	Female	0	0
21490	Black	Male	0	0
21875	White	Male	0	0
22300	White	Female	0	0
22367	White	Female	0	0
22494	White	Male	0	0
25872	White	Male	0	0
26313	White	Male	0	0
28230	White	Female	0	0
28522	White	Male	0	0
28846	White	Male	0	0
29157	White	Male	0	0
30845	White	Female	0	0
31993	White	Male	0	0
32404	White	Female	0	0

	hours-per-week	native-country	income
4881	40	Mexico	<=50K
5104	35	United-States	<=50K
9171	10	United-States	<=50K
11631	10	United-States	<=50K
13084	40	Guatemala	<=50K
15059	50	Mexico	<=50K
17040	40	United-States	<=50K
18555	40	?	<=50K
18698	40	United-States	<=50K
21318	10	United-States	<=50K
21490	30	United-States	<=50K
21875	40	United-States	<=50K
22300	40	Guatemala	<=50K
22367	45	Mexico	<=50K
22494	40	United-States	<=50K
25872	55	Mexico	<=50K
26313	50	United-States	<=50K
28230	40	United-States	<=50K
28522	40	United-States	>50K
28846	40	United-States	<=50K
29157	48	United-States	>50K
30845	40	United-States	<=50K
31993	14	United-States	<=50K
32404	40	United-States	<=50K

```
# Number of duplicated values

df[data.duplicated()].count()
```

```
age               24
workclass         24
fnlwgt            24
education         24
education-num     24
marital-status    24
occupation        24
relationship      24
race              24
sex               24
capital-gain      24
capital-loss      24
hours-per-week    24
native-country    24
income            24
dtype: int64
```

```
# Dropping the duplicated values as their numbers are minimal in
respect of the dataset

data.drop_duplicates(inplace=True)

data.shape
```

```
(32537, 15)
```

Observations

- So now we have 32537 rows with no null or duplicated values.

Let's save this clean dataset for future use

```
try:
    data.to_csv("dataset/adult_cleaned.csv", index=False)
except Exception as err:
    print("Error is: ", err)
else:
    print("Clean csv file created successfully.")
```

```
Clean csv file created successfully.
```

3. EDA:

Using the clean data

```
df = pd.read_csv('dataset/adult_cleaned.csv')
df.head()
```

```
   age          workclass  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  income

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
```

3.1 Differentiating numerical and categorical columns

```
numerical_features = [feature for feature in df.columns if
df[feature].dtypes != 'O']
categorical_features = [feature for feature in df.columns if
df[feature].dtypes == 'O']


print(f"The number of Numerical features are:
{len(numerical_features)}, and the column names are:\
n{numerical_features}")
print(f"\nThe number of Categorical features are:
{len(categorical_features)}, and the column names are:\
n{categorical_features}")
```

```
The number of Numerical features are: 6, and the column names are:
['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss',
'hours-per-week']
```

```
The number of Categorical features are: 9, and the column names are:
['workclass', 'education', 'marital-status', 'occupation',
'relationship', 'race', 'sex', 'native-country', 'income']
```

```
# proportion of count data on categorical columns
for col in categorical_features:
    print(df[col].value_counts(normalize=True) * 100)
    print('---------------------------')
```

```
 Private             75.326551
 Self-emp-not-inc     7.806497
 Local-gov            6.432677
 State-gov            3.989304
 Self-emp-inc         3.429941
 Federal-gov          2.950487
 Without-pay          0.043028
 Never-worked         0.021514
Name: workclass, dtype: float64
---------------------------
 HS-grad         32.252513
 Some-college    22.380674
 Bachelors       16.452039
 Masters          5.292436
 Assoc-voc        4.247472
 11th             3.611273
 Assoc-acdm       3.279344
 10th             2.867505
 7th-8th          1.982359
 Prof-school      1.770292
 9th              1.579740
 12th             1.330793
 Doctorate        1.269324
 5th-6th          1.020377
 1st-4th          0.510188
 Preschool        0.153671
Name: education, dtype: float64
---------------------------
 Married-civ-spouse       46.009159
 Never-married            32.784215
 Divorced                 13.649076
 Separated                 3.150260
 Widowed                   3.051910
 Married-spouse-absent     1.284691
 Married-AF-spouse         0.070689
Name: marital-status, dtype: float64
---------------------------
 Prof-specialty       18.376003
 Craft-repair         12.582598
 Exec-managerial      12.493469
```

```
 Adm-clerical         11.580662
 Sales                11.217998
 Other-service        10.114639
 Machine-op-inspct     6.146848
 Transport-moving      4.908258
 Handlers-cleaners     4.207518
 Farming-fishing       3.048837
 Tech-support          2.849064
 Protective-serv       1.994652
 Priv-house-serv       0.451793
 Armed-Forces          0.027661
Name: occupation, dtype: float64
---------------------------
 Husband           40.529244
 Not-in-family     25.484833
 Own-child         15.563820
 Unmarried         10.587946
 Wife               4.819129
 Other-relative     3.015029
Name: relationship, dtype: float64
---------------------------
 White                 85.425823
 Black                  9.595230
 Asian-Pac-Islander     3.190214
 Amer-Indian-Eskimo     0.955835
 Other                  0.832898
Name: race, dtype: float64
---------------------------
 Male      66.92381
 Female    33.07619
Name: sex, dtype: float64
---------------------------
 United-States                 91.388266
 Mexico                         1.963918
 Philippines                    0.608538
 Germany                        0.421059
 Canada                         0.371884
 Puerto-Rico                    0.350370
 El-Salvador                    0.325783
 India                          0.307342
 Cuba                           0.291975
 England                        0.276608
 Jamaica                        0.248947
 South                          0.245874
 China                          0.230507
 Italy                          0.224360
 Dominican-Republic             0.215140
 Vietnam                        0.205919
 Japan                          0.190552
 Guatemala                      0.190552
```

```
 Poland                          0.184405
 Columbia                        0.181332
 Taiwan                          0.156745
 Haiti                           0.135231
 Iran                            0.132157
 Portugal                        0.113717
 Nicaragua                       0.104496
 Peru                            0.095276
 France                          0.089129
 Greece                          0.089129
 Ecuador                         0.086056
 Ireland                         0.073762
 Hong                            0.061468
 Cambodia                        0.058395
 Trinadad&Tobago                 0.058395
 Laos                            0.055322
 Thailand                        0.055322
 Yugoslavia                      0.049175
 Outlying-US(Guam-USVI-etc)      0.043028
 Honduras                        0.039955
 Hungary                         0.039955
 Scotland                        0.036881
 Holand-Netherlands              0.003073
Name: native-country, dtype: float64
---------------------------
 <=50K    75.907428
 >50K     24.092572
Name: income, dtype: float64
---------------------------
```

3.2 Statistical Analysis of the data

```
# summary of the dataset

df.describe().T

                  count           mean            std      min
25%  \
age             32537.0      38.585549      13.637984     17.0
28.0
fnlwgt          32537.0  189780.848511  105556.471009  12285.0
117827.0
education-num   32537.0      10.081815       2.571633      1.0
9.0
capital-gain    32537.0    1078.443741    7387.957424      0.0
0.0
capital-loss    32537.0      87.368227     403.101833      0.0
0.0
hours-per-week  32537.0      40.440329      12.346889      1.0
40.0
```

```
                     50%       75%        max
age                 37.0      48.0       90.0
fnlwgt          178356.0  236993.0  1484705.0
education-num       10.0      12.0       16.0
capital-gain         0.0       0.0    99999.0
capital-loss         0.0       0.0     4356.0
hours-per-week      40.0      45.0       99.0
```

Observations:

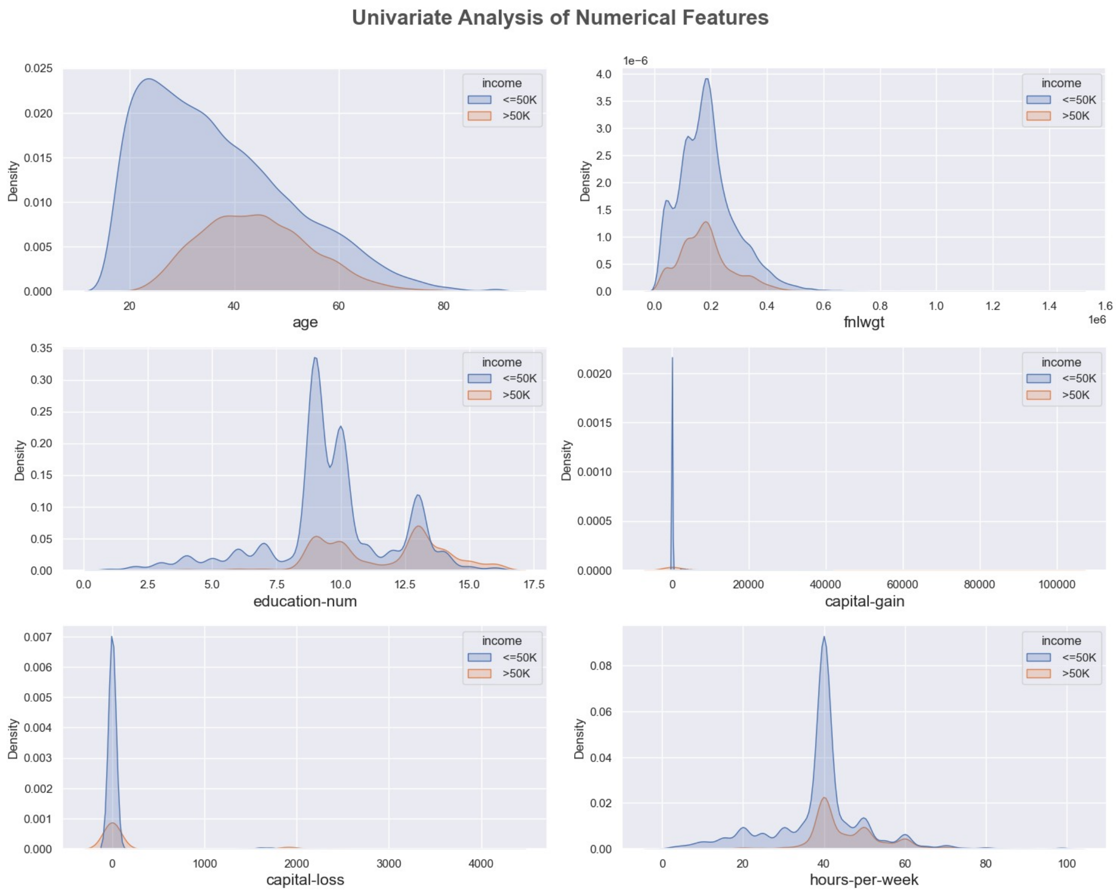
- There are outliers in all the numerical columns except `education-num`.

3.3 Graphical Analysis of the data

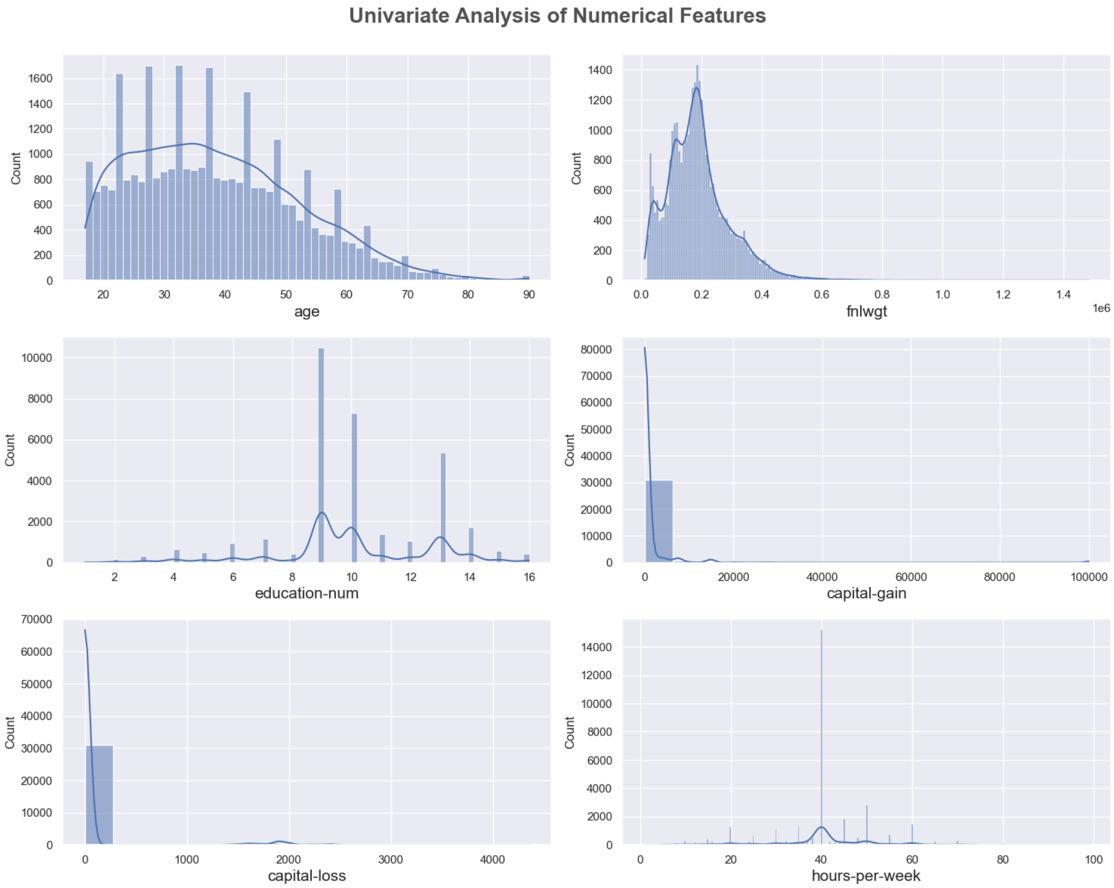
3.3.1 Univariate Analysis

Numerical Features

```
# Kernal Density plots

plt.figure(figsize=(15, 15))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20,
fontweight='bold', alpha=0.8, y=1.)

for i in range(0, len(numerical_features)):
    plt.subplot(4, 2, i+1)
    sns.kdeplot(x=df[numerical_features[i]], shade=True,
hue=df['income'], color='r')
    plt.xlabel(numerical_features[i], fontsize=15)
    plt.tight_layout()
```




```
# Histograms

plt.figure(figsize=(15, 15))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20,
fontweight='bold', alpha=0.8, y=1.)

for i in range(0, len(numerical_features)):
    plt.subplot(4, 2, i+1)
    sns.histplot(x=df[numerical_features[i]], kde=True, color='b')
    plt.xlabel(numerical_features[i], fontsize=15)
    plt.tight_layout()
```

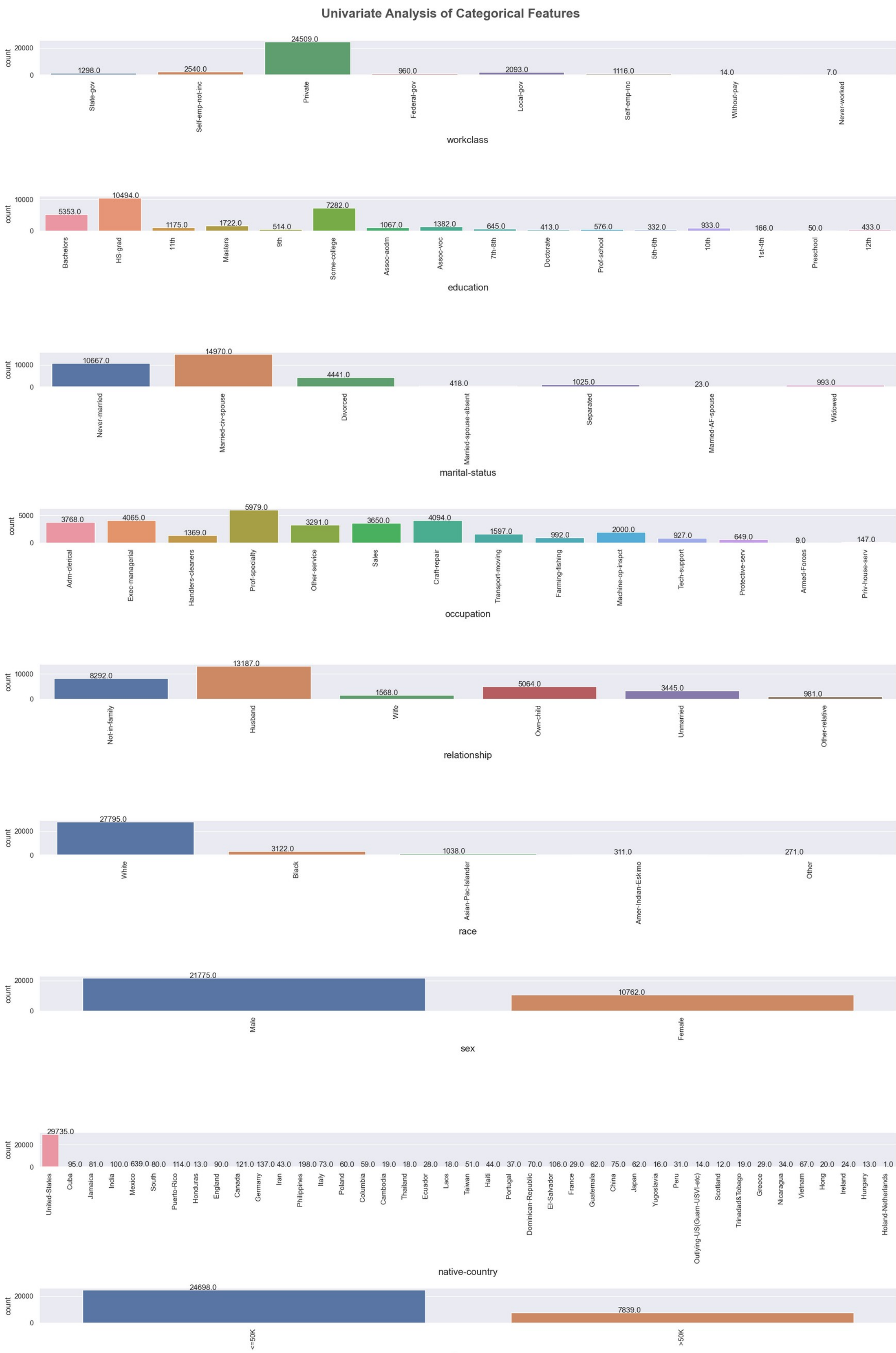


Observations

- age, `fnlwgt` is rightly skewed.
- Generally people work for 30 to 40 hours per week.
- There are outliers in all the columns.

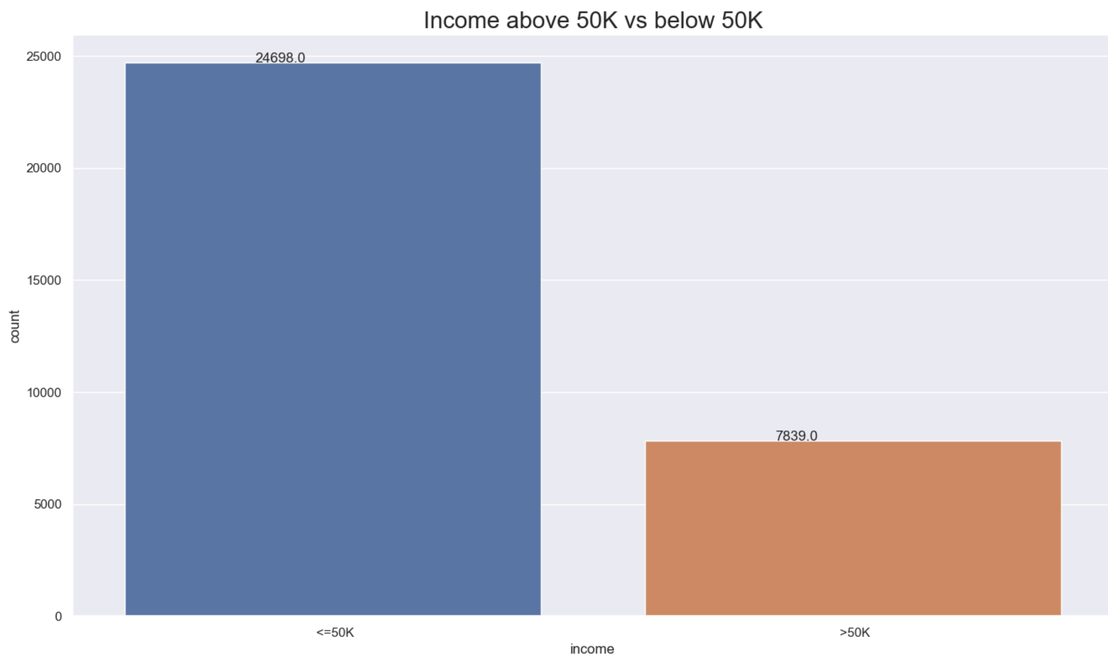
Categorical Features

```
plt.figure(figsize=(20, 30))
plt.suptitle('Univariate Analysis of Categorical Features',
fontsize=20, fontweight='bold', alpha=0.8, y=1.)
cat = ['workclass', 'education', 'marital-status', 'occupation',
'relationship', 'race', 'sex', 'native-country', 'income']

for i in range(0, len(cat)):
    plt.subplot(9, 1, i+1)
    ax = sns.countplot(x=df[cat[i]])
    for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))
    plt.xlabel(cat[i], fontsize=15)
    plt.xticks(rotation=90)
    plt.tight_layout()
```


Univariate Analysis of Categorical Features
count
workclass
State-gov 1298.0
Self-emp-not-inc 2540.0
Private 24509.0
Federal-gov 960.0
Local-gov 2093.0
Self-emp-inc 1116.0
Without-pay 14.0
Never-worked 7.0
education
Bachelors 5353.0
HS-grad 10494.0
11th 1175.0
Masters 1722.0
9th 514.0
Some-college 7282.0
Assoc-acdm 1067.0
Assoc-voc 1382.0
7th-8th 645.0
Doctorate 413.0
Prof-school 576.0
5th-6th 332.0
10th 933.0
1st-4th 166.0
Preschool 50.0
12th 433.0
marital-status
Never-married 10667.0
Married-civ-spouse 14970.0
Divorced 4441.0
Married-spouse-absent 418.0
Separated 1025.0
Married-AF-spouse 23.0
Widowed 993.0
occupation
Adm-clerical 3768.0
Exec-managerial 4065.0
Handlers-cleaners 1369.0
Prof-specialty 5979.0
Other-service 3291.0
Sales 3650.0
Craft-repair 4094.0
Transport-moving 1597.0
Farming-fishing 992.0
Machine-op-inspct 2000.0
Tech-support 927.0
Protective-serv 649.0
Armed-Forces 9.0
Priv-house-serv 147.0
relationship
Not-in-family 8292.0
Husband 13187.0
Wife 1568.0
Own-child 5064.0
Unmarried 3445.0
Other-relative 981.0
race
White 27795.0
Black 3122.0
Asian-Pac-Islander 1038.0
Amer-Indian-Eskimo 311.0
Other 271.0
sex
Male 21775.0
Female 10762.0
native-country
United-States 29735.0
Cuba 95.0
Jamaica 81.0
India 100.0
Mexico 639.0
South 80.0
Puerto-Rico 114.0
Honduras 13.0
England 90.0
Canada 121.0
Germany 137.0
Iran 43.0
Philippines 198.0
Italy 73.0
Poland 60.0
Columbia 59.0
Cambodia 19.0
Thailand 18.0
Ecuador 28.0
Laos 18.0
Taiwan 51.0
Haiti 44.0
Portugal 37.0
Dominican-Republic 70.0
El-Salvador 106.0
France 29.0
Guatemala 62.0
China 75.0
Japan 62.0
Yugoslavia 16.0
Peru 31.0
Outlying-US(Guam-USVI-etc) 14.0
Scotland 12.0
Trinadad&Tobago 19.0
Greece 29.0
Nicaragua 34.0
Vietnam 67.0
Hong 20.0
Ireland 24.0
Hungary 13.0
Holland-Netherlands 1.0
income
<=50K 24698.0
>50K 7839.0

Countplot of income column

```
plt.figure(figsize=(16, 9))
plt.title("Income above 50K vs below 50K", fontsize=20)
ax = sns.countplot(x='income', data=df)
for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))
plt.show()
```



Observation:

- Most people earn less than 50K.

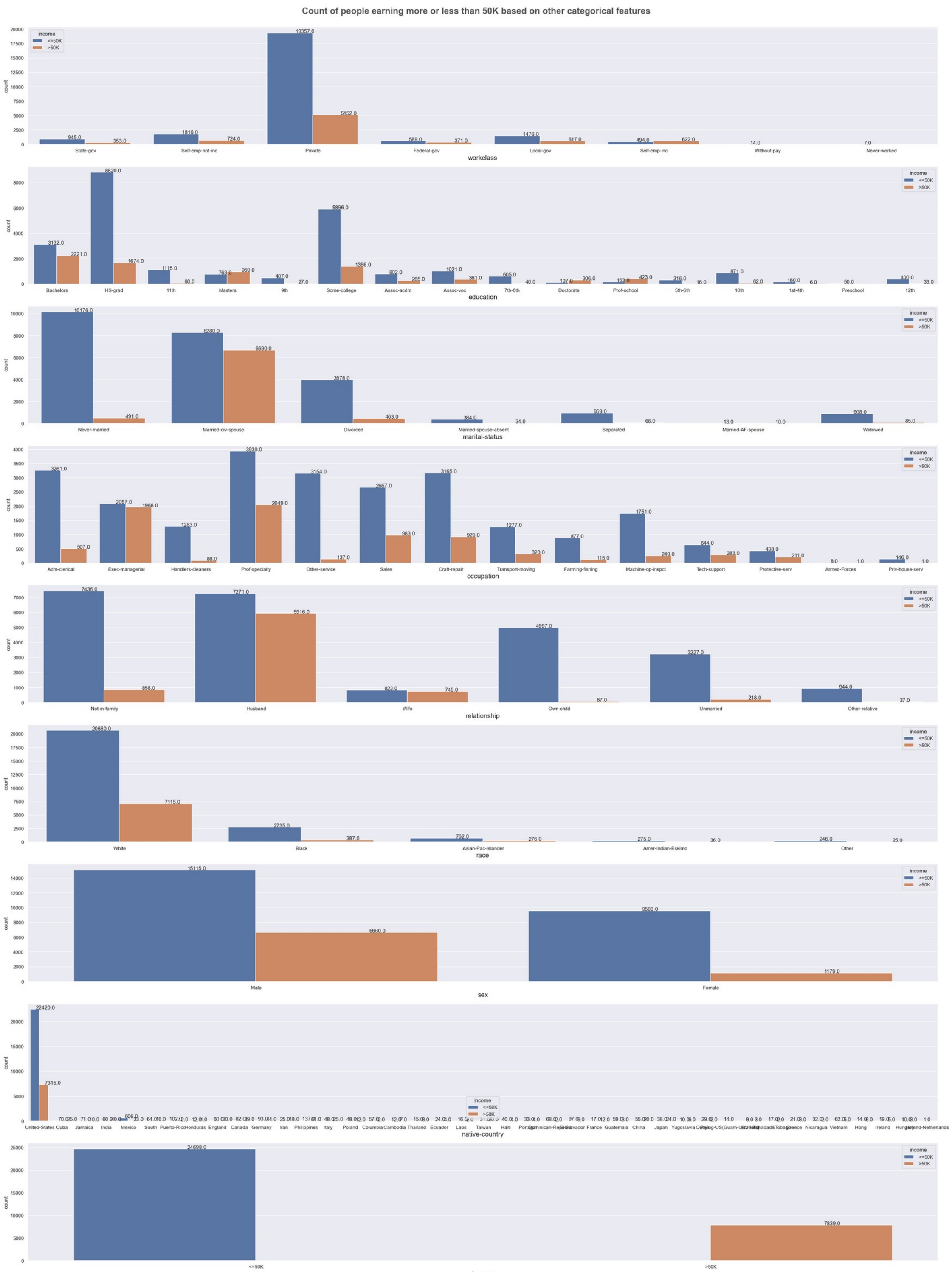
3.3.2 Biivariate Analysis

Categorical features

```
# Creating a dataframe of categorical columns

df_categoric = df[categorical_features]
df_categoric.head()

           workclass   education      marital-status
occupation  \
0          State-gov   Bachelors       Never-married        Adm-
clerical
1   Self-emp-not-inc   Bachelors  Married-civ-spouse     Exec-
managerial
2            Private     HS-grad            Divorced   Handlers-
cleaners
3            Private        11th  Married-civ-spouse   Handlers-
```

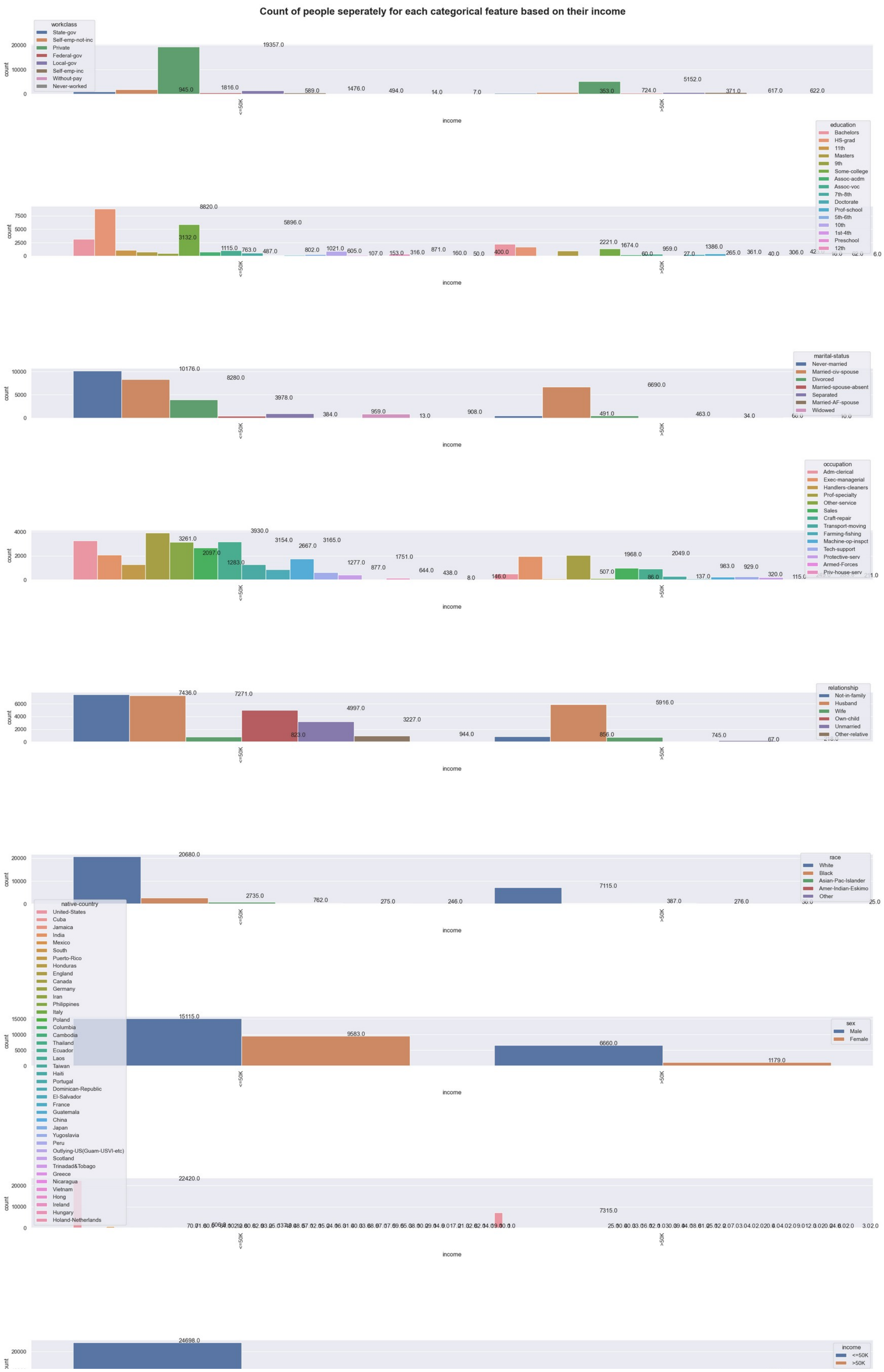
```
cleaners
4            Private   Bachelors   Married-civ-spouse      Prof-
specialty

     relationship    race      sex  native-country  income
0   Not-in-family   White     Male   United-States   <=50K
1         Husband   White     Male   United-States   <=50K
2   Not-in-family   White     Male   United-States   <=50K
3         Husband   Black     Male   United-States   <=50K
4            Wife   Black   Female            Cuba   <=50K
```

```
plt.figure(figsize=(30, 40))
plt.suptitle('Count of people earning more or less than 50K based on
other categorical features',
             fontsize=20, fontweight='bold', alpha=0.8, y=1.)

column_names = df_categoric.columns

for i in range(0, len(column_names)):
    plt.subplot(9, 1, i+1)
    ax = sns.countplot(x=df_categoric[column_names[i]],
hue=df_categoric['income'])
    for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))
    plt.xlabel(column_names[i], fontsize=15)
    plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(25, 39))
plt.suptitle('Count of people seperately for each categorical feature
based on their income',
                fontsize=20, fontweight='bold', alpha=1, y=1)
```

```
column_names = df_categoric.columns

for i in range(0, len(column_names)):
    plt.subplot(9, 1, i+1)
    ax = sns.countplot(x='income', hue=column_names[i],
data=df_categoric)
    for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))
    plt.xticks(rotation = 90)
    plt.tight_layout()
```



Count of people seperately for each categorical feature based on their income
workclass
State-gov
Self-emp-not-inc
Private
Federal-gov
Local-gov
Self-emp-inc
Without-pay
Never-worked
count
income
<=50K
>50K
education
Bachelors
HS-grad
11th
Masters
9th
Some-college
Assoc-acdm
Assoc-voc
7th-8th
Doctorate
Prof-school
5th-6th
10th
1st-4th
Preschool
12th
marital-status
Never-married
Married-civ-spouse
Divorced
Married-spouse-absent
Separated
Married-AF-spouse
Widowed
occupation
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
relationship
Not-in-family
Husband
Wife
Own-child
Unmarried
Other-relative
race
White
Black
Asian-Pac-Islander
Amer-Indian-Eskimo
Other
sex
Male
Female
native-country
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
Holland-Netherlands

Numerical Features

```
# Creating a dataframe leaving the two columns 'Classes' and 'Region'

df_numeric = df[numerical_features]
df_numeric.head()

   age  fnlwgt  education-num  capital-gain  capital-loss  hours-per-
week
0   39   77516             13          2174             0
40
1   50   83311             13             0             0
13
2   38  215646              9             0             0
40
3   53  234721              7             0             0
40
4   28  338409             13             0             0
40

# Adding the categorical column 'income' to the numeric dataframe

df_numeric['income'] = df_categoric['income']
df_numeric.head()

   age  fnlwgt  education-num  capital-gain  capital-loss  hours-per-
week  \
0   39   77516             13          2174             0
40
1   50   83311             13             0             0
13
2   38  215646              9             0             0
40
3   53  234721              7             0             0
40
4   28  338409             13             0             0
40

   income
0   <=50K
1   <=50K
2   <=50K
3   <=50K
4   <=50K
```

Creating a numeric column on the basis of the categorical column `income`

```
# In this column the value ' <=50K' will be represented as '0' and '
>50K' as '1'

df_numeric['earning_class'] = df_categoric['income'].map({' <=50K':0,
```

```
' >50K':1})
df_numeric.head()
```

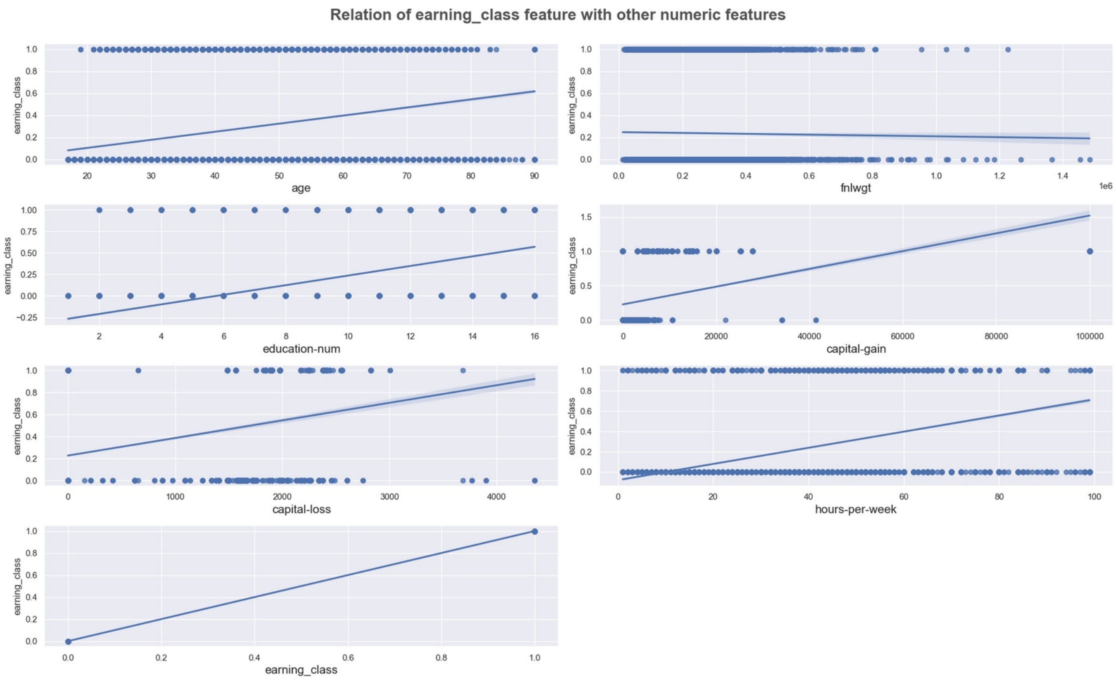
```
   age  fnlwgt  education-num  capital-gain  capital-loss  hours-per-
week  \
0   39   77516             13          2174             0
40
1   50   83311             13             0             0
13
2   38  215646              9             0             0
40
3   53  234721              7             0             0
40
4   28  338409             13             0             0
40

   income  earning_class
0   <=50K              0
1   <=50K              0
2   <=50K              0
3   <=50K              0
4   <=50K              0
```

```
# Now dropping the categorical class 'income'

df_numeric.drop(columns=['income'], axis=1, inplace=True)
df_numeric.head()
```

```
   age  fnlwgt  education-num  capital-gain  capital-loss  hours-per-
week  \
0   39   77516             13          2174             0
40
1   50   83311             13             0             0
13
2   38  215646              9             0             0
40
3   53  234721              7             0             0
40
4   28  338409             13             0             0
40

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

```
plt.figure(figsize=(20, 20))
plt.suptitle('Relation of earning_class feature with other numeric
features', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
```

```
column_names = df_numeric.columns

for i in range(0, len(column_names)):
    plt.subplot(7, 2, i+1)
    sns.regplot(x=df_numeric[column_names[i]],
y=df_numeric['earning_class'])
    plt.xlabel(column_names[i], fontsize=15)
    plt.tight_layout()
plt.show()
```



Transforming `income` from categorical to numeric in the original dataset

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

```
   age         workclass  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  income

0          2174             0              40   United-States       0

1             0             0              13   United-States       0

2             0             0              40   United-States       0

3             0             0              40   United-States       0

4             0             0              40            Cuba       0
```

```python
# Checking numeric features again

numeric_features = [feature for feature in df.columns if 
df[feature].dtype != 'O']
print(f"The number of Numerical features are: {len(numeric_features)}, 
and the column names are:\n{numeric_features}")
```

```
The number of Numerical features are: 7, and the column names are:
['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 
'hours-per-week', 'income']
```

3.3.3 Multivariate Analysis

Checking Multicollinearity in the numerical features

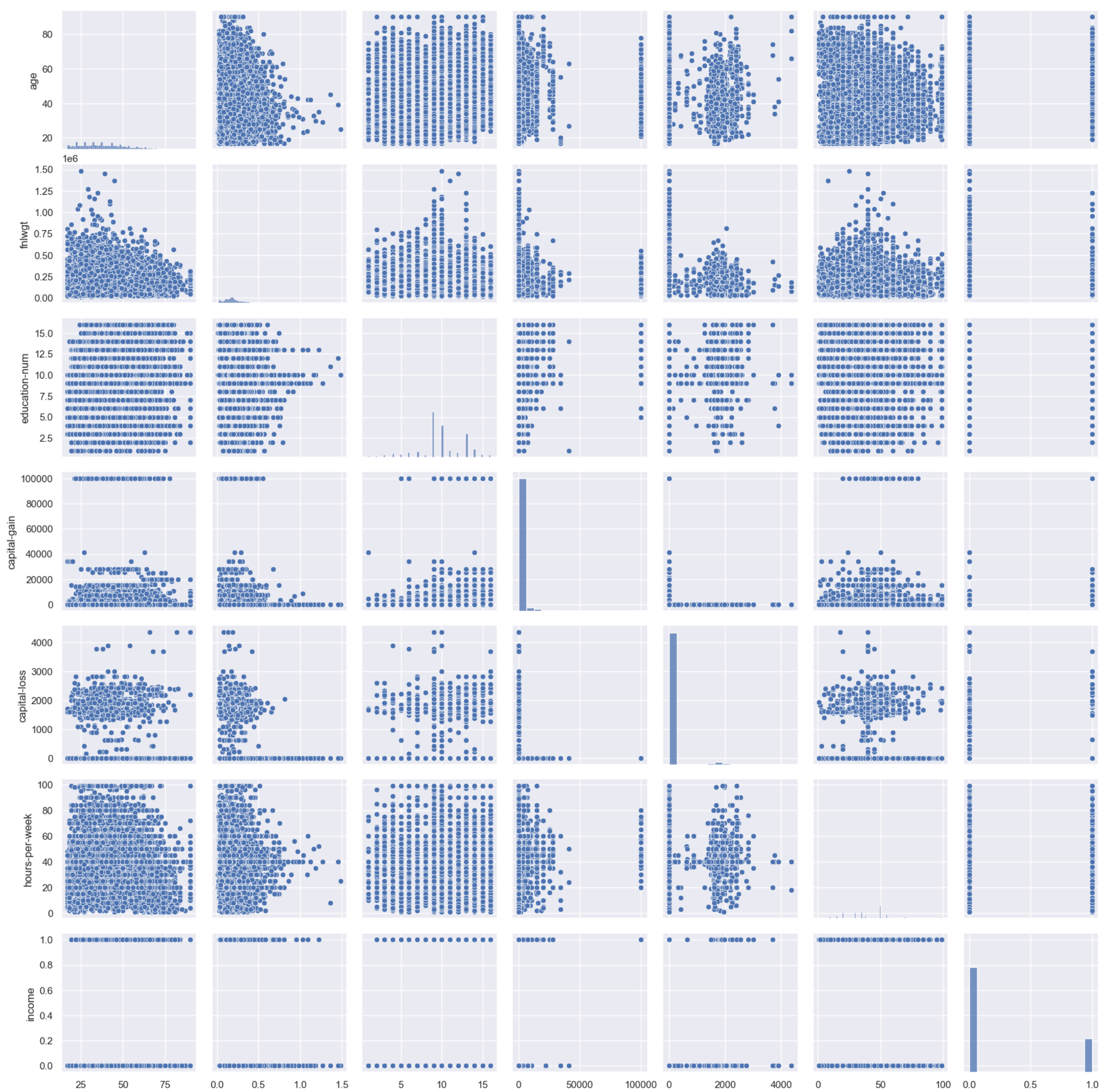
```python
df[numeric_features].corr()
```

```
                     age    fnlwgt  education-num  capital-gain  
capital-loss  \
age             1.000000 -0.076447       0.036224      0.077676      
0.057745   
fnlwgt         -0.076447  1.000000      -0.043388      0.000429     -
0.010260   
education-num   0.036224 -0.043388       1.000000      0.122664      
0.079892   
capital-gain    0.077676  0.000429       0.122664      1.000000     -
0.031639   
capital-loss    0.057745 -0.010260       0.079892     -0.031639      
1.000000   
hours-per-week  0.068515 -0.018898       0.148422      0.078408      
0.054229
```

```
income           0.234037 -0.009502       0.335272      0.223336
0.150501

                 hours-per-week    income
age                    0.068515  0.234037
fnlwgt                -0.018898 -0.009502
education-num          0.148422  0.335272
capital-gain           0.078408  0.223336
capital-loss           0.054229  0.150501
hours-per-week         1.000000  0.229658
income                 0.229658  1.000000
```

Graphical Representation

```
sns.pairplot(df[numeric_features])
plt.show()
```



```
plt.figure(figsize = (15,10))
sns.heatmap(df[numeric_features].corr(), cmap="CMRmap", annot=True)
plt.show()
```

Checking Multicollinearity in the categorical features

- **Using Chi-squared Test**

```
categorical_features

['workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native-country',
 'income']

from scipy.stats import chi2_contingency

chi2_test = []

for feature in categorical_features:
    if chi2_contingency(pd.crosstab(df['income'], df[feature]))[1] <
0.05:
```

```
        chi2_test.append('Reject Null Hypothesis')
    else:
        chi2_test.append('Fail to Reject Null Hypothesis')

result = pd.DataFrame(data=[categorical_features, chi2_test]).T
result.columns = ['Column', 'Hypothesis Result']
result
```

```
           Column       Hypothesis Result
0       workclass  Reject Null Hypothesis
1       education  Reject Null Hypothesis
2  marital-status  Reject Null Hypothesis
3      occupation  Reject Null Hypothesis
4    relationship  Reject Null Hypothesis
5            race  Reject Null Hypothesis
6             sex  Reject Null Hypothesis
7  native-country  Reject Null Hypothesis
8          income  Reject Null Hypothesis
```

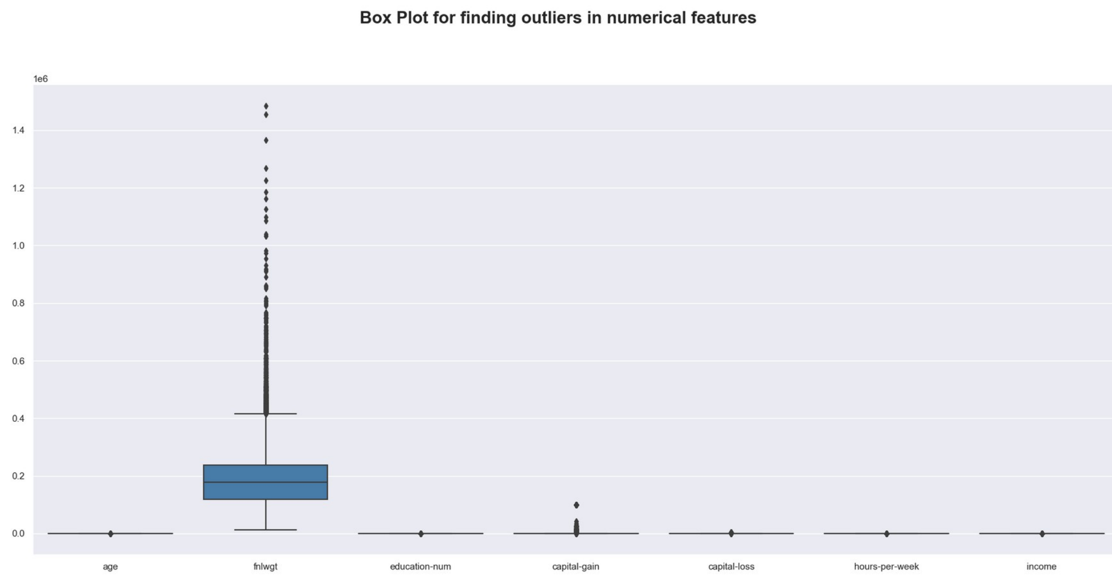
Observations:

- Distribution of Numerical columns `age`, `fnlwgt` are rightly skewed.
- In the columns `capital-gain` and `capital-loss` most of the values are **powerlaw distributed**.
- Most of the employees work in the **Private** sector.
- Most of the people participated in the census are **white**, **male**, **Married-civ-spouse** and their native country is **USA**.
- All Numerical features have outliers.
- Also we can see the dataset is imbalanced as the target column `income` has 75% datapoints in the **<50K** class, so we need to balance it.

4. Data Pre-Processing

Checking Outliers

```
fig = plt.figure( figsize=(23, 10))
plt.suptitle('Box Plot for finding outliers in numerical features',
fontsize=20, fontweight='bold', alpha=1, y=1)
stud_bplt = sns.boxplot(orient='v', data=df[numeric_features],
palette="Set1")
stud_bplt.plot()
plt.show()
```



There are outliers

Checking for multicollinearity in numeric features

```
# importing library

from statsmodels.stats.outliers_influence import
variance_inflation_factor

# Creating function

def calc_vif(X):
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]

    return(vif)

df1 = df[numeric_features]
calc_vif(df1)
```

```
        variables        VIF
0             age   7.293626
1          fnlwgt   3.716829
2   education-num  11.205498
3    capital-gain   1.081154
4    capital-loss   1.078271
5  hours-per-week   9.776523
6          income   1.549287
```

Observation:

- Only `education-num` has a value of >10, so it has multicollinearity.

```python
# Deleting 'education num'


df = df.drop(['education-num'], axis = 1)
df.head()
```

```
   age          workclass  fnlwgt   education       marital-status  \
0   39          State-gov   77516   Bachelors        Never-married
1   50   Self-emp-not-inc   83311   Bachelors   Married-civ-spouse
2   38            Private  215646     HS-grad             Divorced
3   53            Private  234721        11th   Married-civ-spouse
4   28            Private  338409   Bachelors   Married-civ-spouse

          occupation    relationship    race      sex  capital-
gain  \
0       Adm-clerical   Not-in-family   White     Male          2174

1    Exec-managerial         Husband   White     Male             0

2  Handlers-cleaners   Not-in-family   White     Male             0

3  Handlers-cleaners         Husband   Black     Male             0

4     Prof-specialty            Wife   Black   Female             0


   capital-loss  hours-per-week  native-country  income
0             0              40   United-States       0
1             0              13   United-States       0
2             0              40   United-States       0
3             0              40   United-States       0
4             0              40            Cuba       0
```

Storing this data 1st in folder then in MongoDB for later use

```python
try:
    df.to_csv("dataset/adult_processed.csv", index=None)
except Exception as err:
    print("Error is: ", err)
else:
    print("Processed csv file created successfully.")
```

```
Processed csv file created successfully.
```

MongoDB part

```python
income_dict = df.to_dict('records')
```

```python
# connecting with the server
try:
    client =
```

```
pymongo.MongoClient("mongodb+srv://ineuron:Project1@cluster0.rp4qzrr.m
ongodb.net/?retryWrites=true&w=majority")
    db = client
except Exception as e:
    print(e)
else:
    print("Connection to MongoDB server is successful.")
finally:
    print(db)
```

```
Connection to MongoDB server is successful.
MongoClient(host=['ac-rnik5cy-shard-00-00.rp4qzrr.mongodb.net:27017',
'ac-rnik5cy-shard-00-02.rp4qzrr.mongodb.net:27017', 'ac-rnik5cy-shard-
00-01.rp4qzrr.mongodb.net:27017'], document_class=dict,
tz_aware=False, connect=True, retrywrites=True, w='majority',
authsource='admin', replicaset='atlas-jltif8-shard-0', tls=True)
```

```
# Creating database and collection

database = client["income_census_data"]
collection = database['income_census']

try:
    collection.insert_many(income_dict)
except Exception as e:
    print(e)
else:
    print("Records inserted successfully.")
```

```
Records inserted successfully.
```

Loading the data from MongoDB

```
db = client.income_census_data
collect_names = db.list_collection_names()
collect_names
```

```
['income_census']
```

```
final_df = pd.DataFrame(list(db.income_census.find()))
final_df
```

```
                            _id  age         workclass  fnlwgt
education  \
0      63651ee3a1b4514aba84746d   39         State-gov   77516
Bachelors
1      63651ee3a1b4514aba84746e   50  Self-emp-not-inc   83311
Bachelors
2      63651ee3a1b4514aba84746f   38           Private  215646
HS-grad
3      63651ee3a1b4514aba847470   53           Private  234721
11th
```

```
4      63651ee3a1b4514aba847471   28           Private  338409
Bachelors
...                           ...  ...               ...     ...
...
32532  63651ee4a1b4514aba84f381   27           Private  257302
Assoc-acdm
32533  63651ee4a1b4514aba84f382   40           Private  154374
HS-grad
32534  63651ee4a1b4514aba84f383   58           Private  151910
HS-grad
32535  63651ee4a1b4514aba84f384   22           Private  201490
HS-grad
32536  63651ee4a1b4514aba84f385   52      Self-emp-inc  287927
HS-grad

            marital-status          occupation    relationship    race
\
0            Never-married        Adm-clerical   Not-in-family   White

1       Married-civ-spouse     Exec-managerial         Husband   White

2                 Divorced   Handlers-cleaners   Not-in-family   White

3       Married-civ-spouse   Handlers-cleaners         Husband   Black

4       Married-civ-spouse      Prof-specialty            Wife   Black

...                    ...                 ...             ...     ...

32532   Married-civ-spouse        Tech-support            Wife   White

32533   Married-civ-spouse   Machine-op-inspct         Husband   White

32534              Widowed        Adm-clerical       Unmarried   White

32535        Never-married        Adm-clerical       Own-child   White

32536   Married-civ-spouse     Exec-managerial            Wife   White


           sex  capital-gain  capital-loss  hours-per-week  native-
country  \
0         Male          2174             0              40   United-
States
1         Male             0             0              13   United-
States
2         Male             0             0              40   United-
States
3         Male             0             0              40   United-
```

```
States
4       Female              0              0              40
Cuba
...         ...           ...            ...             ...
...
32532   Female              0              0              38   United-
States
32533     Male              0              0              40   United-
States
32534   Female              0              0              40   United-
States
32535     Male              0              0              20   United-
States
32536   Female          15024              0              40   United-
States

       income
0           0
1           0
2           0
3           0
4           0
...       ...
32532       0
32533       1
32534       0
32535       0
32536       1

[32537 rows x 15 columns]
```

```
# Dropping the '_id' column

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

```
       age          workclass  fnlwgt    education       marital-
status  \
0       39          State-gov   77516    Bachelors        Never-
married
1       50   Self-emp-not-inc   83311    Bachelors   Married-civ-
spouse
2       38            Private  215646      HS-grad
Divorced
3       53            Private  234721         11th   Married-civ-
spouse
4       28            Private  338409    Bachelors   Married-civ-
spouse
...    ...                ...     ...          ...                ..
.
```

```
32532   27            Private  257302   Assoc-acdm   Married-civ-
spouse
32533   40            Private  154374      HS-grad   Married-civ-
spouse
32534   58            Private  151910      HS-grad
Widowed
32535   22            Private  201490      HS-grad        Never-
married
32536   52       Self-emp-inc  287927      HS-grad   Married-civ-
spouse

               occupation    relationship    race      sex  capital-
gain  \
0            Adm-clerical   Not-in-family   White     Male
2174
1         Exec-managerial         Husband   White     Male
0
2       Handlers-cleaners   Not-in-family   White     Male
0
3       Handlers-cleaners         Husband   Black     Male
0
4          Prof-specialty            Wife   Black   Female
0
...                   ...             ...     ...      ...           .
..
32532        Tech-support            Wife   White   Female
0
32533   Machine-op-inspct         Husband   White     Male
0
32534        Adm-clerical       Unmarried   White   Female
0
32535        Adm-clerical       Own-child   White     Male
0
32536     Exec-managerial            Wife   White   Female
15024

       capital-loss  hours-per-week  native-country  income
0                 0              40   United-States       0
1                 0              13   United-States       0
2                 0              40   United-States       0
3                 0              40   United-States       0
4                 0              40            Cuba       0
...             ...             ...             ...     ...
32532             0              38   United-States       0
32533             0              40   United-States       1
32534             0              40   United-States       0
32535             0              20   United-States       0
32536             0              40   United-States       1

[32537 rows x 14 columns]
```

Creating independent and dependent variables

```
X = df.drop('income', axis = 1)
y = df['income']
```

```
# Doing Test Train 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=42)
```

```
X_train.shape
```

```
(21799, 13)
```

```
X_test.shape
```

```
(10738, 13)
```

```
category_var = [col for col in X.columns if X[col].dtypes == object]
category_var
```

```
['workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native-country']
```

```
numeric_var = [col for col in X.columns if X[col].dtypes != object]
numeric_var
```

```
['age', 'fnlwgt', 'capital-gain', 'capital-loss', 'hours-per-week']
```

Encoding

To do encoding

- https://www.kaggle.com/code/subinium/11-categorical-encoders-and-benchmark

```
# importing library

import category_encoders as ce
```

```
one_hot = ce.OneHotEncoder(cols = category_var, handle_unknown =
'ignore')
```

```
# Creating dataframe for categorical variables which converted to one
hot encoded variables.
X_train_one_hot = pd.DataFrame(one_hot.fit_transform(X_train))
X_test_one_hot = pd.DataFrame(one_hot.transform(X_test))
```

```
X_train_one_hot.index = X_train.index
X_test_one_hot.index = X_test.index

num_X_train = X_train[numeric_var]
num_X_test = X_test[numeric_var]

# Joining numerical and one hot encoded variables to create our final X_train and X_test.
X_train_new = pd.concat([num_X_train, X_train_one_hot], axis = 1)
X_test_new = pd.concat([num_X_test, X_test_one_hot], axis = 1)
```

Scaling the data

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler

StandardScaler()

scale = scaler.fit(X_train_new)
scale

StandardScaler()

# Printing the mean

print(scale.mean_)

[3.85290151e+01 1.90201068e+05 1.02077577e+03 9.01778063e+01
 4.04248360e+01 3.85290151e+01 7.54392403e-01 6.33515299e-02
 7.74347447e-02 3.46346163e-02 2.96802606e-02 3.99559613e-02
 3.66989311e-04 1.83494656e-04 1.90201068e+05 2.25102069e-01
 3.22858847e-01 3.26620487e-02 2.78911877e-02 1.01380797e-02
 1.58722877e-02 1.64778201e-01 1.36244782e-02 3.71576678e-02
 5.24794715e-02 5.00022937e-03 1.27528786e-02 1.70191293e-02
 4.20202762e-02 1.90375705e-02 1.60557824e-03 3.29969265e-01
 4.57681545e-01 1.35464930e-01 3.10105968e-02 3.22033121e-02
 1.28446259e-02 8.25725951e-04 4.21120235e-02 1.16335612e-01
 1.96798018e-02 1.00371577e-01 4.77544842e-02 1.12849213e-01
 1.22895546e-01 1.84595624e-01 1.26978302e-01 2.81664297e-02
 6.20211936e-02 3.10105968e-02 4.95435570e-03 2.75241984e-04
 1.58906372e-01 4.92683151e-02 4.01899170e-01 2.53130878e-01
 1.05968164e-01 3.08271022e-02 8.52974907e-01 9.59677049e-02
 3.27996697e-02 9.63346943e-03 8.62424882e-03 6.68195789e-01
 3.31804211e-01 1.02077577e+03 9.01778063e+01 4.04248360e+01
 9.11417955e-01 2.00009175e-02 4.49561907e-03 3.44052479e-03
 1.65145190e-03 2.24780953e-03 2.47717785e-03 1.78907289e-03
 6.65168127e-03 3.66989311e-03 3.48639846e-03 2.01844121e-03
 1.37620992e-03 3.80751411e-03 4.12862975e-04 2.38543052e-03
```

```
 1.28446259e-03 2.24780953e-03 2.01844121e-03 7.33978623e-04
 5.96357631e-04 1.05509427e-03 1.51383091e-03 1.10096793e-03
 2.93591449e-03 2.15606220e-03 4.58736639e-04 2.70654617e-03
 1.97256755e-03 4.58736639e-04 2.11018854e-03 7.33978623e-04
 5.04610303e-04 6.88104959e-04 7.79852287e-04 3.66989311e-04
 3.66989311e-04 5.96357631e-04 9.17473279e-04 3.21115648e-04
 4.58736639e-05]
```

Saving the scale to use it later to transform the data and predict the values

```python
# To save a Standard scaler object
import pickle

with open('scaled.pkl', 'wb') as f:
    pickle.dump(scale, f)

# Loading the scaled object to transform the data

with open('scaled.pkl', 'rb') as f:
    scaled = pickle.load(f)

# Now transforming the train and test dataset

X_train_tf = scaled.transform(X_train_new)
X_test_tf = scaled.transform(X_test_new)

# checking the transformed data

X_train_tf
```

```
array([[-1.36146381, -0.76094509, -0.14229344, ..., -0.03030373,
        -0.01792258, -0.00677317],
       [ 0.62242593, -0.2277653 , -0.14229344, ..., -0.03030373,
        -0.01792258, -0.00677317],
       [-0.25930285,  1.04760296,  0.87502844, ..., -0.03030373,
        -0.01792258, -0.00677317],
       ...,
       [-1.50841861,  0.24732597, -0.14229344, ..., -0.03030373,
        -0.01792258, -0.00677317],
       [-0.33278024,  0.50027559, -0.14229344, ..., -0.03030373,
        -0.01792258, -0.00677317],
       [-1.14103162,  1.3249031 , -0.14229344, ..., -0.03030373,
        -0.01792258, -0.00677317]])
```

```python
X_test_tf
```

```
array([[-0.55321244, -1.48770587, -0.14229344, ..., -0.03030373,
        -0.01792258, -0.00677317],
       [-1.06755422, -0.56578056, -0.14229344, ..., -0.03030373,
        -0.01792258, -0.00677317],
       [ 1.5776321 , -0.87095864, -0.14229344, ..., -0.03030373,
```

```
         -0.01792258, -0.00677317],
        ...,
        [-0.84712203, -0.0192497 , -0.14229344, ..., -0.03030373,
         -0.01792258, -0.00677317],
        [-0.03887065,  0.14500256, -0.14229344, ..., -0.03030373,
         -0.01792258, -0.00677317],
        [-0.62668984,  1.42913245, -0.14229344, ..., -0.03030373,
         -0.01792258, -0.00677317]])
```

5. Model Building

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, roc_curve, roc_auc_score
```

Logistic Regression

```
lgr = LogisticRegression()
lgr.fit(X_train_new, y_train)
lgr_pred = lgr.predict(X_test_new)
ac_lr = accuracy_score(y_test, lgr_pred)
roc_logr = roc_auc_score(y_test, lgr_pred)
print('Logistic Regression accuracy score:{0:0.2f}%'.
format(ac_lr*100))
print('Logistic Regression ROC score:{0:0.2f}%'. format(roc_logr*100))
```

```
Logistic Regression accuracy score:79.74%
Logistic Regression ROC score:62.02%
```

SVC

```
classifier = SVC(random_state = 0, kernel = 'rbf')
classifier.fit(X_train_new, y_train)
y_pred = classifier.predict(X_test_new)
ac_svc = accuracy_score(y_test,y_pred)
roc_svc = roc_auc_score(y_test,y_pred)
print('SVC accuracy score:{0:0.2f}%'. format(ac_svc*100))
print('SVC ROC score:{0:0.2f}%'. format(roc_svc*100))
```

```
SVC accuracy score:79.55%
SVC ROC score:57.83%
```

```
# Saving the models

with open('lgr.pkl', 'wb') as f:
    pickle.dump(lgr, f)

with open('svc.pkl', 'wb') as f:
    pickle.dump(classifier, f)

# Loading the models
```

```
with open('lgr.pkl', 'rb') as f:
    clf_logreg = pickle.load(f)

with open('svc.pkl', 'rb') as f:
    clf_SVC = pickle.load(f)
```

Evaluation of the models

```
models = {clf_logreg:'LogisticRegression',
          clf_SVC: 'SVC',
          }

def train(algo, name, X_train, y_train, X_test, y_test):
    algo.fit(X_train,y_train)
    y_pred = algo.predict(X_test)
    score = accuracy_score(y_test,y_pred)
    print(f"--------------------------------------------
{name}---------------------------------------------------")
    print(f"Accuracy Score for {name}: {score*100:.4f}%")
    return y_test,y_pred,score

# acc_res function calculates confusion matrix
def acc_res(y_test,y_pred):
    null_accuracy = y_test.value_counts()[0]/len(y_test)
    print(f"Null Accuracy: {null_accuracy*100:.4f}%")
    print("Confusion Matrix")
    matrix = confusion_matrix(y_test,y_pred)
    print(matrix)
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++")
    TN = matrix[0,0]
    FP = matrix[0,1]
    FN = matrix[1,0]
    TP = matrix[1,1]
    accuracy_score=(TN+TP) / float(TP+TN+FP+FN)
    recall_score = (TP)/ float(TP+FN)
    specificity = TN / float(TN+FP)
    FPR = FP / float(FP+TN)
    precision_score = TP / float(TP+FP)
    print(f"Accuracy Score: {accuracy_score*100:.4f}%")
    print(f"Recall Score: {recall_score*100:.4f}%")
    print(f"Specificity Score: {specificity*100:.4f}%")
    print(f"False Positive Rate: {FPR*100:.4f}%")
    print(f"Precision Score: {precision_score*100:.4f}%")
    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++")
    print("Classification Report")
    print(classification_report(y_test,y_pred))

def main(models):
    accuracy_scores = []
    for algo,name in models.items():
```

```
        y_test_train,y_pred,acc_score =
train(algo,name,X_train_new,y_train,X_test_new,y_test)
        acc_res(y_test_train,y_pred)
        accuracy_scores.append(acc_score)
    return accuracy_scores

accuracy_scores = main(models)
```

```
-------------------------------------------
LogisticRegression--------------------------------------------------
Accuracy Score for LogisticRegression: 79.7448%
Null Accuracy: 75.7962%
Confusion Matrix
[[7844  295]
 [1880  719]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Accuracy Score: 79.7448%
Recall Score: 27.6645%
Specificity Score: 96.3755%
False Positive Rate: 3.6245%
Precision Score: 70.9073%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Classification Report
              precision    recall  f1-score   support

           0       0.81      0.96      0.88      8139
           1       0.71      0.28      0.40      2599

    accuracy                           0.80     10738
   macro avg       0.76      0.62      0.64     10738
weighted avg       0.78      0.80      0.76     10738

-------------------------------------------
SVC--------------------------------------------------
Accuracy Score for SVC: 79.5493%
Null Accuracy: 75.7962%
Confusion Matrix
[[8133    6]
 [2190  409]]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Accuracy Score: 79.5493%
Recall Score: 15.7368%
Specificity Score: 99.9263%
False Positive Rate: 0.0737%
Precision Score: 98.5542%
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.79      1.00      0.88      8139
           1       0.99      0.16      0.27      2599

    accuracy                           0.80     10738
   macro avg       0.89      0.58      0.58     10738
weighted avg       0.84      0.80      0.73     10738
```

As both the models are giving almost same accuracy so we go with logistic regression

Hyperparameter Tuning using GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

Logistic Regression

```
parameters = {
    'penalty' : ['l1','l2'],
    'C'       : np.logspace(-3,3,7),
    'solver'  : ['newton-cg', 'lbfgs', 'liblinear'],
}

clf = GridSearchCV(clf_logreg,                    # model
                   param_grid = parameters,   # hyperparameters
                   scoring='accuracy',        # metric for scoring
                   verbose=2,
                   cv=10)
clf.fit(X_train_new,y_train)
print("Tuned Hyperparameters :", clf.best_params_)
print("Accuracy :", clf.best_score_)
```

```
Fitting 10 folds for each of 42 candidates, totalling 420 fits
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
```

```
[CV] END ..............C=0.001, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=0.001, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   6.3s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   6.7s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   8.2s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   5.3s
```

```
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   5.6s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   5.5s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   5.4s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   5.5s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   6.1s
[CV] END ..............C=0.001, penalty=l2, solver=newton-cg; total
time=   5.0s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.4s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.1s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ..................C=0.001, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.1s
```

```
[CV] END ..............C=0.001, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.01, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.0s
```

```
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   5.1s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   5.0s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   5.5s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   4.6s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   5.5s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   5.0s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   5.3s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   4.8s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   4.7s
[CV] END ...............C=0.01, penalty=l2, solver=newton-cg; total
time=   4.8s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.2s

[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.1s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
```

```
time=   0.3s
[CV] END ...................C=0.01, penalty=l2, solver=lbfgs; total
time=   0.4s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.01, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=0.1, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ....................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=0.1, penalty=l1, solver=lbfgs; total
```

```
time=   0.0s
[CV] END ...................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=0.1, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.8s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.8s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   1.1s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.9s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.1, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   4.6s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   5.0s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   4.7s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   4.5s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   5.4s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   4.6s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   4.4s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
time=   4.3s
[CV] END ...............C=0.1, penalty=l2, solver=newton-cg; total
```

```
time=   4.8s
[CV] END ................C=0.1, penalty=l2, solver=newton-cg; total
time=   4.4s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.4s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.1s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ....................C=0.1, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ................C=0.1, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
```

```
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ....................C=1.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.1s

[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.3s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.0s
```

```
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.3s
[CV] END ................C=1.0, penalty=l1, solver=liblinear; total
time=   0.5s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   4.8s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   4.8s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   6.1s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   5.3s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   4.7s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   4.4s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   5.0s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   4.9s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   4.5s
[CV] END ................C=1.0, penalty=l2, solver=newton-cg; total
time=   4.7s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.5s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.4s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.1s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ....................C=1.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
```

```
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ................C=1.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
```

```
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...................C=10.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.2s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   4.9s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   4.4s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   4.7s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   4.5s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   4.6s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   5.2s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   5.2s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   5.1s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   5.0s
[CV] END ...............C=10.0, penalty=l2, solver=newton-cg; total
time=   5.5s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.6s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.3s
```

```
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.4s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.4s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.9s
[CV] END ...................C=10.0, penalty=l2, solver=lbfgs; total
time=   0.6s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.2s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.2s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.2s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ...............C=10.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
```

```
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s

[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..................C=100.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   5.8s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   6.1s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
```

```
time=   4.5s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   5.0s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   4.4s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   4.5s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   5.5s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   5.2s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   4.6s
[CV] END ..............C=100.0, penalty=l2, solver=newton-cg; total
time=   4.6s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.5s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.1s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END ..................C=100.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
```

```
time=   0.1s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END ..............C=100.0, penalty=l2, solver=liblinear; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=newton-cg; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .................C=1000.0, penalty=l1, solver=lbfgs; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
```

```
time=   0.1s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.1s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l1, solver=liblinear; total
time=   0.0s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   4.9s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   4.8s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   4.9s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   4.5s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   4.6s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   5.2s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   4.5s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   4.9s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   5.7s
[CV] END .............C=1000.0, penalty=l2, solver=newton-cg; total
time=   6.0s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.6s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.4s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.1s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.2s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
```

```
time=   0.2s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END .................C=1000.0, penalty=l2, solver=lbfgs; total
time=   0.3s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.2s

[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.2s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.3s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.2s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.1s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.2s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.4s
[CV] END .............C=1000.0, penalty=l2, solver=liblinear; total
time=   0.1s
Tuned Hyperparameters : {'C': 100.0, 'penalty': 'l2', 'solver':
'newton-cg'}
Accuracy : 0.8506351074266034
```

```
# With tuned parameters

model_logr = LogisticRegression(solver="newton-cg", C=100.0,
penalty='l2')
model_logr.fit(X_train_new, y_train)
pred_logr = model_logr.predict(X_test_new)
ac_lr_tuned=accuracy_score(y_test, pred_logr)
roc_logr_tuned =roc_auc_score(y_test,pred_logr)
print('Logistic Regression accuracy score:{0:0.2f}%'.
format(ac_lr_tuned*100))
print('Logistic Regression ROC score:{0:0.2f}%'.
format(roc_logr_tuned*100))
```

```
Logistic Regression accuracy score:85.17%
Logistic Regression ROC score:76.69%
```

Roc_curve for Logistic Regression Model

```
clf_logreg = LogisticRegression(solver="newton-cg", C=100.0,
penalty='l2')

clf_logreg.fit(X_train_new, y_train)
```

```
LogisticRegression(C=100.0, solver='newton-cg')
```

```
ytrain_pred = clf_logreg.predict_proba(X_train_new)
print('Logistic train roc-auc: {}'.format(roc_auc_score(y_train,
ytrain_pred[:,1])))
ytest_pred = clf_logreg.predict_proba(X_test_new)
print('Logistic test roc-auc: {}'.format(roc_auc_score(y_test,
ytest_pred[:,1])))
```

```
Logistic train roc-auc: 0.9068401203837859
Logistic test roc-auc: 0.9031225965585163
```

```
pred=[]
for model in [clf_logreg]:
    pred.append(pd.Series(model.predict_proba(X_test_new)[:,1]))
final_prediction=pd.concat(pred,axis=1).mean(axis=1)
print('Ensemble test roc-auc:
{}'.format(roc_auc_score(y_test,final_prediction)))
```

```
Ensemble test roc-auc: 0.9031225965585163
```

```
fpr, tpr, thresholds = roc_curve(y_test, final_prediction)
thresholds
```

```
array([2.00000000e+00, 1.00000000e+00, 1.00000000e+00, ...,
       1.28511109e-03, 1.27719252e-03, 2.41553571e-04])
```

```
from sklearn.metrics import accuracy_score
accuracy_ls = []
for thres in thresholds:
    y_pred = np.where(final_prediction>thres,1,0)
    accuracy_ls.append(accuracy_score(y_test, y_pred, normalize=True))

accuracy_ls = pd.concat([pd.Series(thresholds),
pd.Series(accuracy_ls)],
                        axis=1)
accuracy_ls.columns = ['thresholds', 'accuracy']
accuracy_ls.sort_values(by='accuracy', ascending=False, inplace=True)
accuracy_ls.head()
```

	thresholds	accuracy
747	0.498183	0.852300
748	0.498142	0.852207
746	0.498286	0.852207
745	0.499805	0.851835
749	0.496549	0.851835

```
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
```

```
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()

plot_roc_curve(fpr,tpr)
```