```python
# Importing Libraries

import numpy as np # linear algebra
import pandas as pd # data processing,

# Libraries for data visualization

import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# Importing the dataset

income_data = pd.read_csv("C:/Users/acer/Downloads/adult.csv")
income_data
```

```
       age workclass  fnlwgt     education  education.num
marital.status  \
0       90         ?   77053       HS-grad              9
Widowed
1       82   Private  132870       HS-grad              9
Widowed
2       66         ?  186061  Some-college             10
Widowed
3       54   Private  140359       7th-8th              4
Divorced
4       41   Private  264663  Some-college             10
Separated
...    ...       ...     ...           ...            ...
...
32556   22   Private  310152  Some-college             10       Never-
married
32557   27   Private  257302    Assoc-acdm             12  Married-
civ-spouse
32558   40   Private  154374       HS-grad              9  Married-
civ-spouse
32559   58   Private  151910       HS-grad              9
Widowed
32560   22   Private  201490       HS-grad              9       Never-
married

              occupation   relationship   race     sex
capital.gain  \
0                      ?  Not-in-family  White  Female             0

1        Exec-managerial  Not-in-family  White  Female             0

2                      ?      Unmarried  Black  Female             0
```

```
3      Machine-op-inspct      Unmarried  White  Female             0

4         Prof-specialty      Own-child  White  Female             0

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

32556    Protective-serv  Not-in-family  White    Male             0

32557       Tech-support           Wife  White  Female             0

32558  Machine-op-inspct        Husband  White    Male             0

32559       Adm-clerical      Unmarried  White  Female             0

32560       Adm-clerical      Own-child  White    Male             0


       capital.loss  hours.per.week native.country income
0              4356              40  United-States  <=50K
1              4356              18  United-States  <=50K
2              4356              40  United-States  <=50K
3              3900              40  United-States  <=50K
4              3900              40  United-States  <=50K
...             ...             ...            ...    ...
32556             0              40  United-States  <=50K
32557             0              38  United-States  <=50K
32558             0              40  United-States   >50K
32559             0              40  United-States  <=50K
32560             0              20  United-States  <=50K

[32561 rows x 15 columns]
```

```
#income_data.head(3).to_csv("Character_data.csv")
```

```
# Checking the shape(Number of records and attributes)
income_data.shape
```

```
(32561, 15)
```

```
# Listing dataframe attributes
income_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')
```

```
#replacing columns names having special characters with proper names
income_data.rename(columns={'education.num':'EducationNum',
'capital.gain': 'CapitalGain', 'capital.loss': 'CapitalLoss',
'native.country': 'Country','hours.per.week':
'HoursPerWeek','marital.status': 'MaritalStatus'}, inplace=True)
income_data.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'EducationNum',
       'MaritalStatus', 'occupation', 'relationship', 'race', 'sex',
       'CapitalGain', 'CapitalLoss', 'HoursPerWeek', 'Country',
'income'],
      dtype='object')
```

```
#Listing datatypes of the attributes
income_data.dtypes
```

```
age               int64
workclass        object
fnlwgt            int64
education        object
EducationNum      int64
MaritalStatus    object
occupation       object
relationship     object
race             object
sex              object
CapitalGain       int64
CapitalLoss       int64
HoursPerWeek      int64
Country          object
income           object
dtype: object
```

```
#data cleaning
#Finding the special characters count in the data frame
income_data.isin(['?']).sum(axis=0)
```

```
age                 0
workclass        1836
fnlwgt              0
education           0
EducationNum        0
MaritalStatus       0
occupation       1843
relationship        0
race                0
sex                 0
CapitalGain         0
CapitalLoss         0
HoursPerWeek        0
Country           583
```

```
income              0
dtype: int64
```

```
# replacing the special character to nan and then drop the columns
income_data['Country'] = income_data['Country'].replace('?',np.nan)
income_data['workclass'] =
income_data['workclass'].replace('?',np.nan)
income_data['occupation'] =
income_data['occupation'].replace('?',np.nan)

# viewing data again
income_data
```

```
       age workclass  fnlwgt     education  EducationNum
MaritalStatus  \
0       90       NaN   77053       HS-grad             9
Widowed
1       82   Private  132870       HS-grad             9
Widowed
2       66       NaN  186061  Some-college            10
Widowed
3       54   Private  140359       7th-8th             4
Divorced
4       41   Private  264663  Some-college            10
Separated
...    ...       ...     ...           ...           ...
...
32556   22   Private  310152  Some-college            10       Never-
married
32557   27   Private  257302    Assoc-acdm            12  Married-civ-
spouse
32558   40   Private  154374       HS-grad             9  Married-civ-
spouse
32559   58   Private  151910       HS-grad             9
Widowed
32560   22   Private  201490       HS-grad             9       Never-
married

              occupation   relationship   race     sex  CapitalGain  \
0                    NaN  Not-in-family  White  Female            0
1        Exec-managerial  Not-in-family  White  Female            0
2                    NaN      Unmarried  Black  Female            0
3      Machine-op-inspct      Unmarried  White  Female            0
4         Prof-specialty      Own-child  White  Female            0
...                  ...            ...    ...     ...          ...
32556    Protective-serv  Not-in-family  White    Male            0
32557       Tech-support           Wife  White  Female            0
32558  Machine-op-inspct        Husband  White    Male            0
32559       Adm-clerical      Unmarried  White  Female            0
32560       Adm-clerical      Own-child  White    Male            0
```

```
       CapitalLoss  HoursPerWeek        Country income
0             4356            40  United-States  <=50K
1             4356            18  United-States  <=50K
2             4356            40  United-States  <=50K
3             3900            40  United-States  <=50K
4             3900            40  United-States  <=50K
...            ...           ...            ...    ...
32556            0            40  United-States  <=50K
32557            0            38  United-States  <=50K
32558            0            40  United-States   >50K
32559            0            40  United-States  <=50K
32560            0            20  United-States  <=50K

[32561 rows x 15 columns]
```

```
#Printing top 10 records
income_data.head(10)
```

```
   age    workclass  fnlwgt     education  EducationNum  MaritalStatus  
\
0   90          NaN   77053       HS-grad             9        Widowed   

1   82      Private  132870       HS-grad             9        Widowed   

2   66          NaN  186061  Some-college            10        Widowed   

3   54      Private  140359       7th-8th             4       Divorced   

4   41      Private  264663  Some-college            10      Separated   

5   34      Private  216864       HS-grad             9       Divorced   

6   38      Private  150601          10th             6      Separated   

7   74    State-gov   88638     Doctorate            16  Never-married   

8   68  Federal-gov  422013       HS-grad             9       Divorced   

9   41      Private   70037  Some-college            10  Never-married   


          occupation    relationship   race     sex  CapitalGain  
CapitalLoss  \
0                NaN   Not-in-family  White  Female            0   
4356
1    Exec-managerial   Not-in-family  White  Female            0   
4356
2                NaN       Unmarried  Black  Female            0   
4356
```

```
3  Machine-op-inspct       Unmarried  White  Female            0
3900
4     Prof-specialty       Own-child  White  Female            0
3900
5      Other-service       Unmarried  White  Female            0
3770
6       Adm-clerical       Unmarried  White    Male            0
3770
7     Prof-specialty  Other-relative  White  Female            0
3683
8     Prof-specialty   Not-in-family  White  Female            0
3683
9       Craft-repair       Unmarried  White    Male            0
3004

   HoursPerWeek        Country income
0            40  United-States  <=50K
1            18  United-States  <=50K
2            40  United-States  <=50K
3            40  United-States  <=50K
4            40  United-States  <=50K
5            45  United-States  <=50K
6            40  United-States  <=50K
7            20  United-States   >50K
8            40  United-States  <=50K
9            60            NaN   >50K
```

```
#income_data.head().to_excel("head_income_data.xls")

# income dataset info to find columns and count of the data
income_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      30725 non-null  object
 2   fnlwgt         32561 non-null  int64 
 3   education      32561 non-null  object
 4   EducationNum   32561 non-null  int64 
 5   MaritalStatus  32561 non-null  object
 6   occupation     30718 non-null  object
 7   relationship   32561 non-null  object
 8   race           32561 non-null  object
 9   sex            32561 non-null  object
 10  CapitalGain    32561 non-null  int64 
 11  CapitalLoss    32561 non-null  int64 
 12  HoursPerWeek   32561 non-null  int64 
 13  Country        31978 non-null  object
```

```
 14  income         32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
# Renaming sex column to Gender
income_data.rename(columns = {'sex':'gender'}, inplace=True)
```

```
# counting the duplicate rows in the dataset
income_data.duplicated().sum()
```

```
24
```

```
# removing duplicate values from the dataset
income_data.drop_duplicates(inplace = True)
```

```
income_data.shape
```

```
(32537, 15)
```

```
# Counting total number of null values in the entire dataset
income_data.isnull().sum().sum()
```

```
4261
```

```
#Counting number of NA values for all attributes
income_data.isnull()
income_data.isnull().sum()
income_data.isnull().sum().sort_values(ascending=False)
```

```
occupation       1843
workclass        1836
Country           582
age                 0
fnlwgt              0
education           0
EducationNum        0
MaritalStatus       0
relationship        0
race                0
gender              0
CapitalGain         0
CapitalLoss         0
HoursPerWeek        0
income              0
dtype: int64
```

```
# Filling NA's with mode[0] as only categorical variables have null
values

for col in ['occupation','workclass', 'Country']:
    income_data[col].fillna(income_data[col].mode()[0], inplace=True)
```

```python
# Again checking the nulls

income_data.isnull().sum()
```

```
age              0
workclass        0
fnlwgt           0
education        0
EducationNum     0
MaritalStatus    0
occupation       0
relationship     0
race             0
gender           0
CapitalGain      0
CapitalLoss      0
HoursPerWeek     0
Country          0
income           0
dtype: int64
```

```python
income_data
```

```
       age workclass  fnlwgt     education  EducationNum
MaritalStatus  \
0       90   Private   77053       HS-grad             9
Widowed
1       82   Private  132870       HS-grad             9
Widowed
2       66   Private  186061  Some-college            10
Widowed
3       54   Private  140359       7th-8th             4
Divorced
4       41   Private  264663  Some-college            10
Separated
...    ...       ...     ...           ...           ...
...
32556   22   Private  310152  Some-college            10       Never-
married
32557   27   Private  257302    Assoc-acdm            12  Married-civ-
spouse
32558   40   Private  154374       HS-grad             9  Married-civ-
spouse
32559   58   Private  151910       HS-grad             9
Widowed
32560   22   Private  201490       HS-grad             9       Never-
married

              occupation   relationship   race  gender  CapitalGain  \
0         Prof-specialty  Not-in-family  White  Female            0
1        Exec-managerial  Not-in-family  White  Female            0
```

```
2         Prof-specialty      Unmarried  Black  Female            0
3      Machine-op-inspct      Unmarried  White  Female            0
4         Prof-specialty      Own-child  White  Female            0
...                  ...            ...    ...     ...          ...
32556    Protective-serv  Not-in-family  White    Male            0
32557       Tech-support           Wife  White  Female            0
32558  Machine-op-inspct        Husband  White    Male            0
32559       Adm-clerical      Unmarried  White  Female            0
32560       Adm-clerical      Own-child  White    Male            0

       CapitalLoss  HoursPerWeek        Country income
0             4356            40  United-States  <=50K
1             4356            18  United-States  <=50K
2             4356            40  United-States  <=50K
3             3900            40  United-States  <=50K
4             3900            40  United-States  <=50K
...            ...           ...            ...    ...
32556            0            40  United-States  <=50K
32557            0            38  United-States  <=50K
32558            0            40  United-States   >50K
32559            0            40  United-States  <=50K
32560            0            20  United-States  <=50K

[32537 rows x 15 columns]
```

```
# Descriptive Analytics of quantitative variables in the dataframe

Numeric_data = round(income_data.describe(),0)  # rounding off the decimals
Numeric_data # transposing the data

#Numeric_data.T.to_csv("Numeric_data.csv")
```

```
           age     fnlwgt  EducationNum  CapitalGain  CapitalLoss  \
count  32537.0    32537.0       32537.0      32537.0      32537.0
mean      39.0   189781.0          10.0       1078.0         87.0
std       14.0   105556.0           3.0       7388.0        403.0
min       17.0    12285.0           1.0          0.0          0.0
25%       28.0   117827.0           9.0          0.0          0.0
50%       37.0   178356.0          10.0          0.0          0.0
75%       48.0   236993.0          12.0          0.0          0.0
max       90.0  1484705.0          16.0      99999.0       4356.0

       HoursPerWeek
count       32537.0
mean           40.0
std            12.0
min             1.0
25%            40.0
50%            40.0
```

```
75%            45.0
max            99.0

#Checking unique values for all Categorical
columns(workclass,education,MaritalStatus,occupation,
#relationship,race,gender,Country,income )

print("workclass:", income_data.workclass.unique())
print("education:", income_data.education.unique())
print("MaritalStatus :", income_data.MaritalStatus.unique())
print("occupation :", income_data.occupation.unique())
print("relationship :", income_data.relationship.unique())
print("race :", income_data.race.unique())
print("gender :", income_data.gender.unique())
print("Country :", income_data.Country.unique())
print("income :", income_data.income.unique())

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

```python
# Counting the occurence of values for workclass
income_data["workclass"].value_counts()
```

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

```python
# Counting the occurence of values for education
income_data["education"].value_counts()
```

```
HS-grad         10494
Some-college     7282
Bachelors        5353
Masters          1722
Assoc-voc        1382
11th             1175
Assoc-acdm       1067
10th              933
7th-8th           645
Prof-school       576
9th               514
12th              433
Doctorate         413
5th-6th           332
1st-4th           166
Preschool          50
Name: education, dtype: int64
```

```python
# Counting the occurence of values for MaritalStatus
income_data["MaritalStatus"].value_counts()
```

```
Married-civ-spouse       14970
Never-married            10667
Divorced                  4441
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: MaritalStatus, dtype: int64
```

```python
# Counting the occurence of values for occupation
income_data["occupation"].value_counts()
```

```
Prof-specialty       5979
Craft-repair         4094
Exec-managerial      4065
```

```
Adm-clerical         3768
Sales                3650
Other-service        3291
Machine-op-inspct    2000
Transport-moving     1597
Handlers-cleaners    1369
Farming-fishing       992
Tech-support          927
Protective-serv       649
Priv-house-serv       147
Armed-Forces            9
Name: occupation, dtype: int64
```

```
# Counting the occurence of values for relationship
income_data["relationship"].value_counts()
```

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

```
# Counting the occurence of values for race
income_data["race"].value_counts()
```

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

```
# Counting the occurence of values for gender
income_data["gender"].value_counts()
```

```
Male      21775
Female    10762
Name: gender, dtype: int64
```

```
# Counting the occurence of values for Country
income_data["Country"].value_counts()
```

```
United-States                 29735
Mexico                          639
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
```

```
Cuba                              95
England                           90
Jamaica                           81
South                             80
China                             75
Italy                             73
Dominican-Republic                70
Vietnam                           67
Japan                             62
Guatemala                         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
Trinadad&Tobago                   19
Cambodia                          19
Thailand                          18
Laos                              18
Yugoslavia                        16
Outlying-US(Guam-USVI-etc)        14
Hungary                           13
Honduras                          13
Scotland                          12
Holand-Netherlands                 1
Name: Country, dtype: int64
```

```
# Counting the occurence of values for income
income_data["income"].value_counts()
```

```
<=50K    24698
>50K      7839
Name: income, dtype: int64
```

```
income_data.dtypes
```

```
age               int64
workclass        object
fnlwgt            int64
education        object
EducationNum      int64
MaritalStatus    object
occupation       object
```

```
relationship     object
race             object
gender           object
CapitalGain       int64
CapitalLoss       int64
HoursPerWeek      int64
Country          object
income           object
dtype: object
```

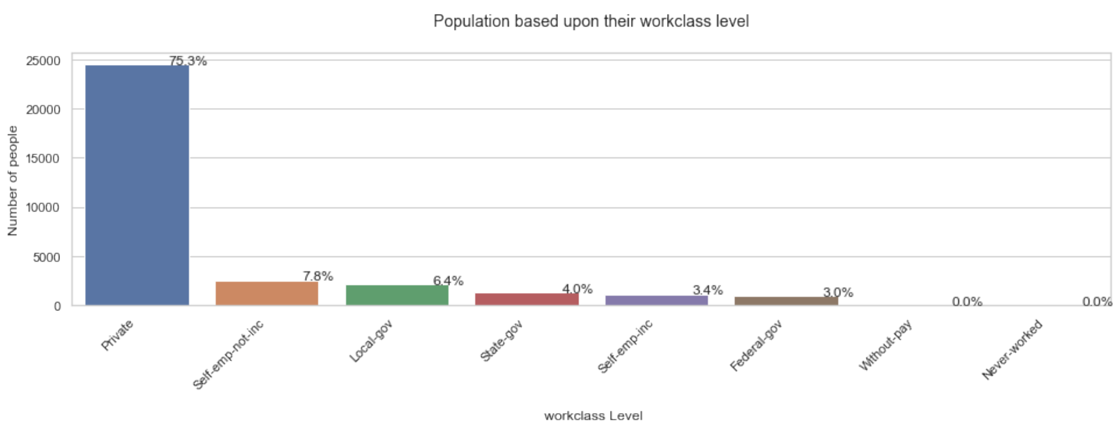
```
#converting the object datatypes to categorical data types
income_data["workclass"]=income_data["workclass"].astype("category")
income_data["education"]=income_data["education"].astype("category")
income_data["MaritalStatus"]=
income_data["MaritalStatus"].astype("category")
income_data["occupation"]=
income_data["occupation"].astype("category")
income_data["relationship"]=
income_data["relationship"].astype("category")
income_data["race"]= income_data["race"].astype("category")
income_data["gender"]= income_data["gender"].astype("category")
income_data["Country"]= income_data["Country"].astype("category")
income_data["income"]= income_data["income"].astype("category")
```

```
income_data.dtypes
```

```
age                 int64
workclass        category
fnlwgt              int64
education        category
EducationNum        int64
MaritalStatus    category
occupation       category
relationship     category
race             category
gender           category
CapitalGain         int64
CapitalLoss         int64
HoursPerWeek        int64
Country          category
income           category
dtype: object
```

```
#Creating countplot for workclass
# Counting the occurence for values for workclass
#data["workclass"].value_counts()
#Creating countplot for workclass
plt.figure(figsize=(16,4))
sns.set(style = 'whitegrid')
total = float(len(income_data))
top_1 = income_data['workclass'].value_counts()[:10].index
```

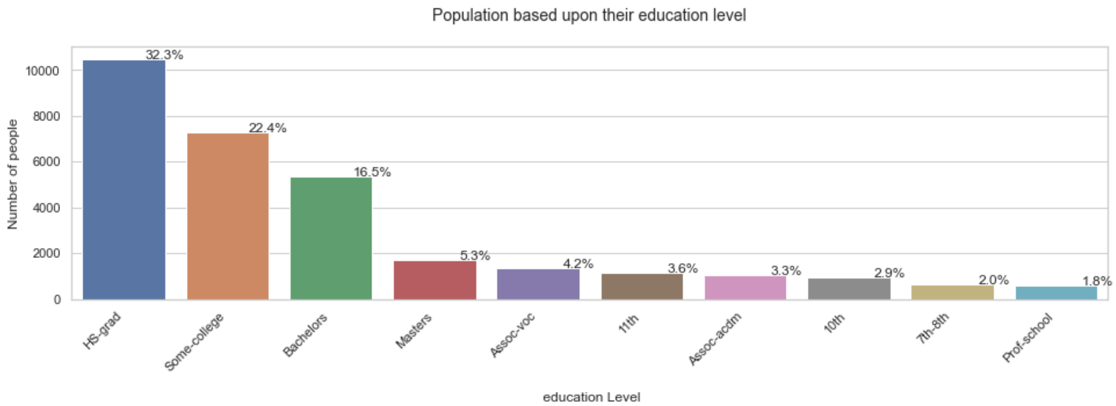
```
ax=sns.countplot(data=income_data, x='workclass', order=top_1)


plt.title("Population based upon their workclass level\n",size=14)
plt.ylabel("Number of people")
plt.xlabel(" \n workclass Level")

ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
horizontalalignment='right')


for p in (ax.patches):
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
#plt.savefig("bar chart of workclass and population.png")
plt.show()
```



The countplot shows that Private Occupation have the maximum number of the people.

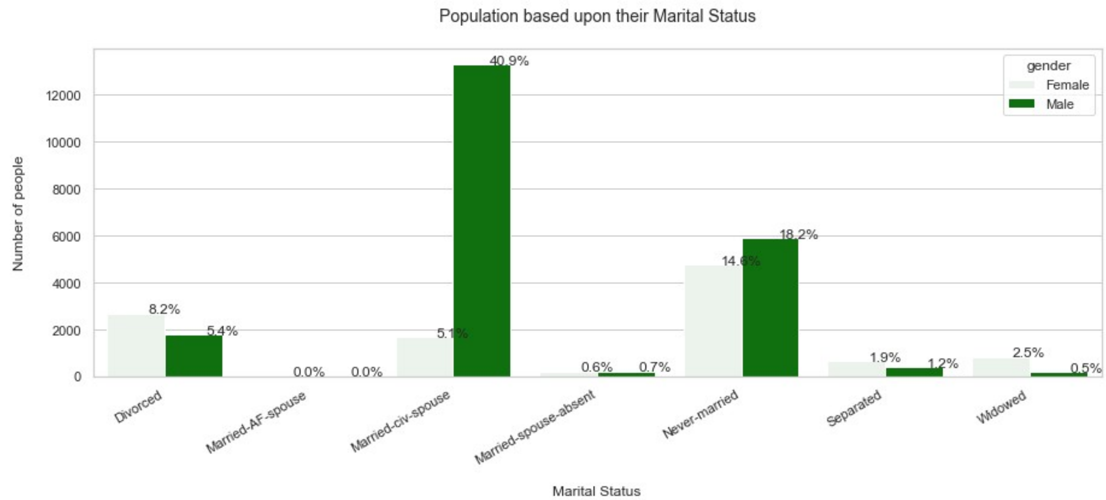
```
#Representing number of people with different education level

plt.figure(figsize=(16,4))
sns.set(style = 'whitegrid')
total = float(len(income_data))
top_1 = income_data['education'].value_counts()[:10].index
ax=sns.countplot(data=income_data, x='education', order=top_1)


plt.title("Population based upon their education level\n",size=14)
plt.ylabel("Number of people")
plt.xlabel(" \n education Level")

ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
horizontalalignment='right')
```

```
for p in (ax.patches):
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
#plt.savefig("bar chart of education and population.png")
plt.show()
```



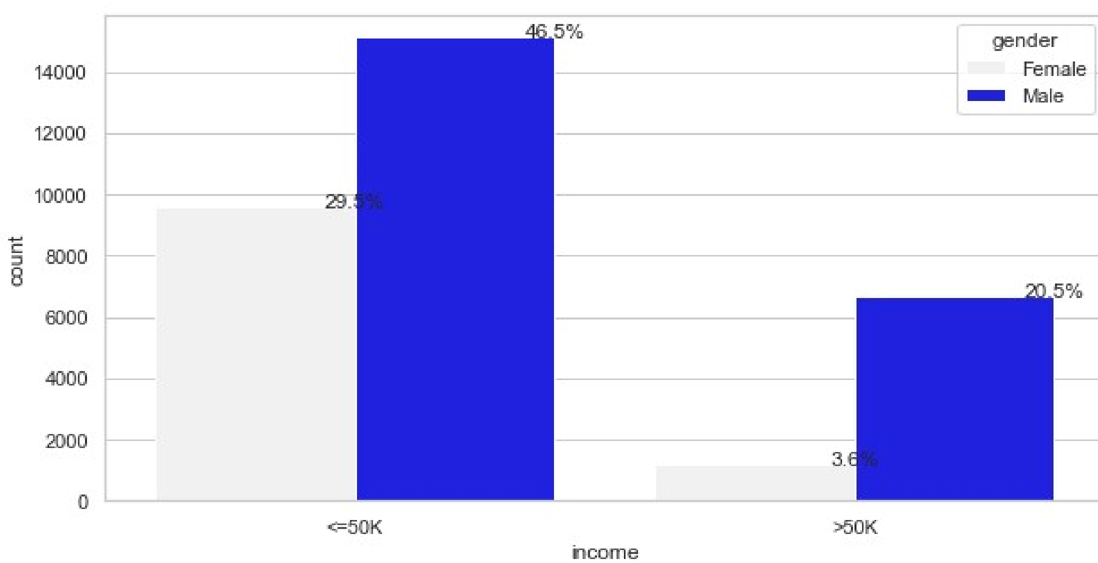
The bargraph illustrates that the majority participants in the dataset are High School Graduates, followed by some college, then Bachelors degree holders.

```
#Graph representing marital_status of people in the dataset:

plt.figure(figsize=(15,5))
ax =
sns.countplot("MaritalStatus",data=income_data,hue="gender",color="gre
en")

plt.title("\nPopulation based upon their Marital Status\n",size=14)
plt.ylabel("\nNumber of people\n")
plt.xlabel(" \nMarital Status\n")

ax.set_xticklabels(ax.get_xticklabels(), rotation=30,
horizontalalignment='right')


for p in (ax.patches):
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
#plt.savefig("bar chart of Marital Status and population.png")
plt.show()
```

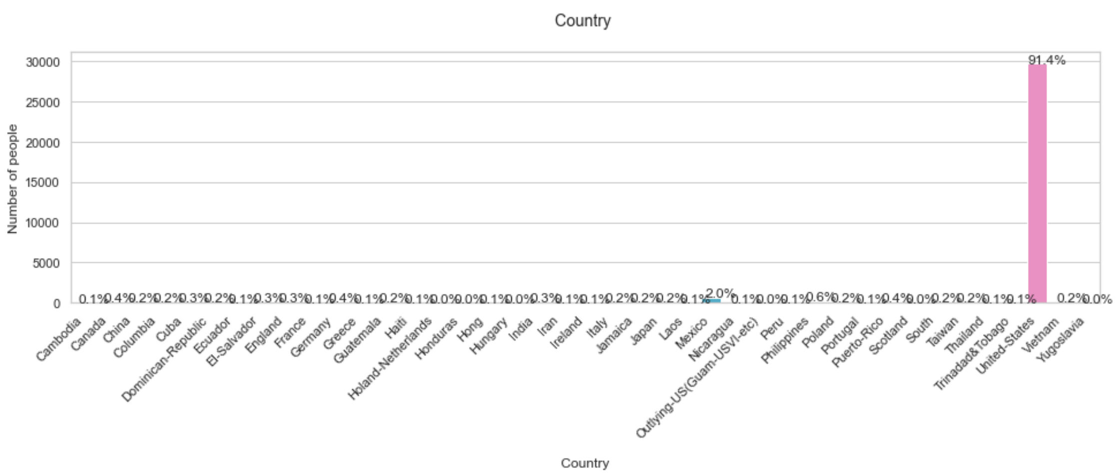


The majority of working males are married, followed by singles. However, the majority of working females are single, followed by divorced and married women.

```
#Graph representing number of people based upon income:

plt.figure(figsize=(10,5))
ax =
sns.countplot("income",data=income_data,hue="gender",color="blue")

for p in (ax.patches):
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
#plt.savefig("bar chart of income and gender count.png")
plt.show()
```

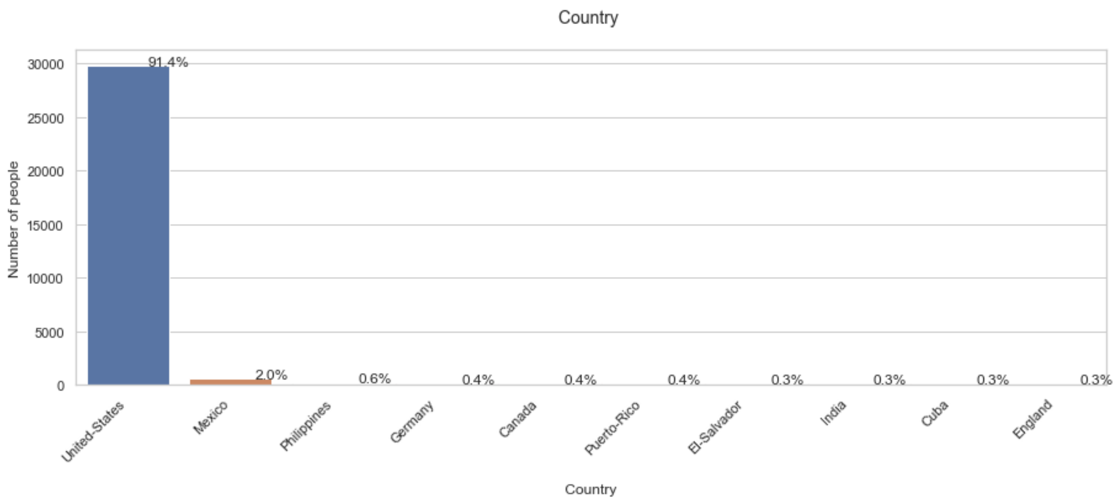


```
# Counting the occurence for values for Country
#data["Country"].value_counts()

plt.figure(figsize=(16,4))
sns.set(style = 'whitegrid')
ax = sns.countplot(x= 'Country', data = income_data)
total = float(len(income_data))
#ax = sns.countplot(x = 'Country', data = top_10)
plt.title("Country \n",size=14)
plt.ylabel("Number of people")
plt.xlabel(" \n Country ")

ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
horizontalalignment='right')


for p in (ax.patches):
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
#plt.savefig("bar chart of Country .png")
plt.show()
```




```
# To get only top 10 countries
plt.figure(figsize=(15, 5))
sns.set(style = 'whitegrid')
top_10 = income_data['Country'].value_counts()[:10].index
ax=sns.countplot(data=income_data, x='Country', order=top_10)
total = float(len(income_data))

plt.title("Country \n",size=14)
plt.ylabel("Number of people")
plt.xlabel(" \n Country ")
```

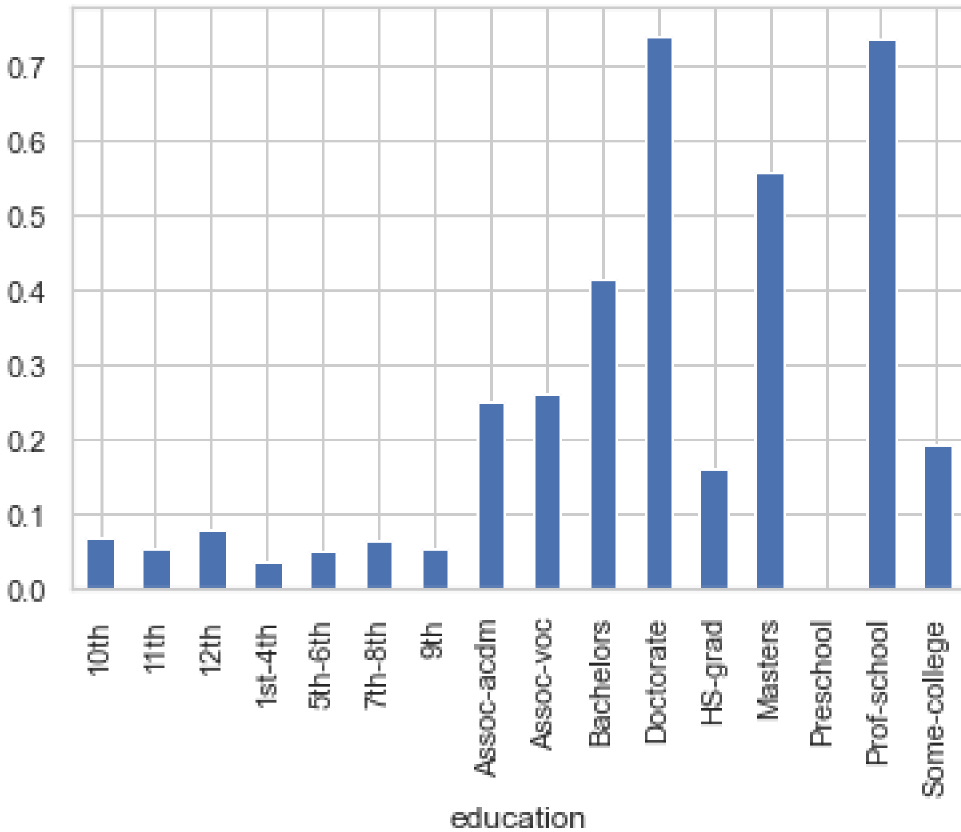
```
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
horizontalalignment='right')


for p in (ax.patches):
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + p.get_width()
    y = p.get_height()
    ax.annotate(percentage, (x, y),ha='center')
#plt.savefig("bar chart of Country .png")
plt.show()
```



As demonstrated in the graph above, the majority of people earn less than or equal to $50,000.

```
#Using the map function, convert the data type of income variable to
numerical data

income_data['income'] = income_data['income'].map({'<=50K': 0, '>50K':
1}).astype(int)

#Creating a bar graph for Education vs. Income to illustrate how these
two columns are related
income_data.groupby('education').income.mean().plot(kind='bar')
# plt.savefig("bar chart of income and education.png")
```

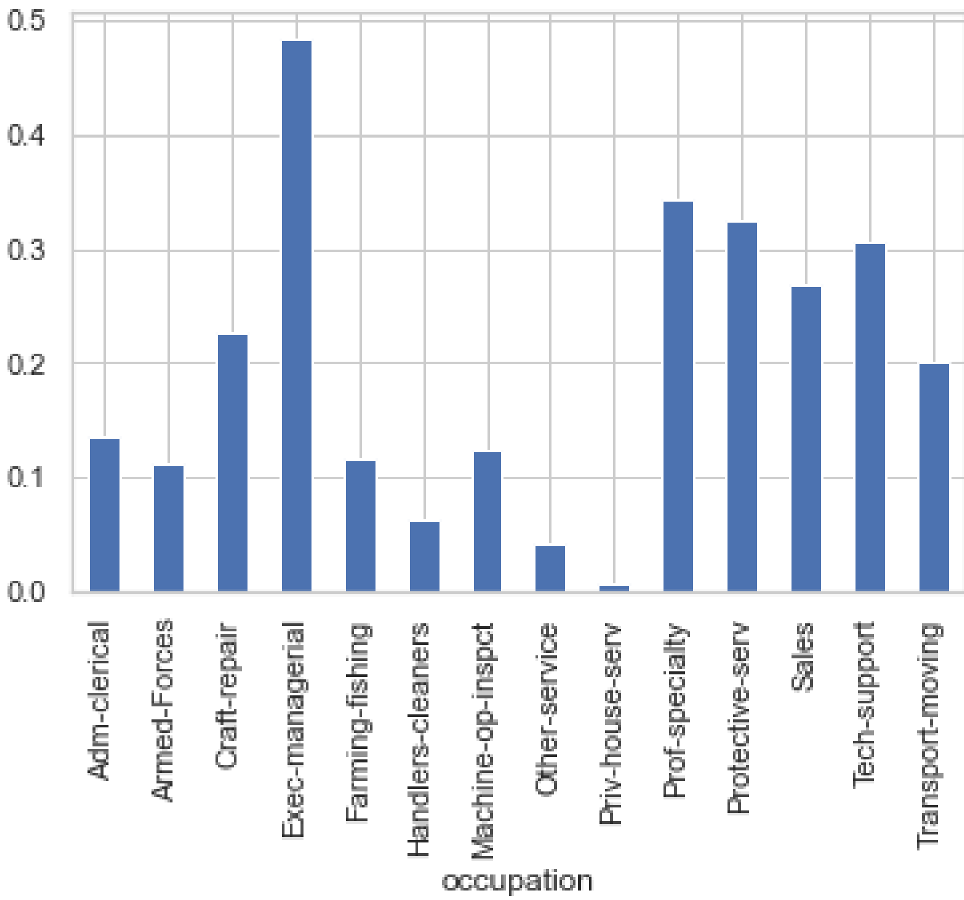
```
<AxesSubplot:xlabel='education'>
```



Adults with a Prof-school and Doctorate educational background will have a greater salary, and it is likely that they will earn more over $50,000.

```
#To demonstrate the relationship between these columns, make a bar graph of Occupation vs. Income.
income_data.groupby('occupation').income.mean().plot(kind='bar')
```

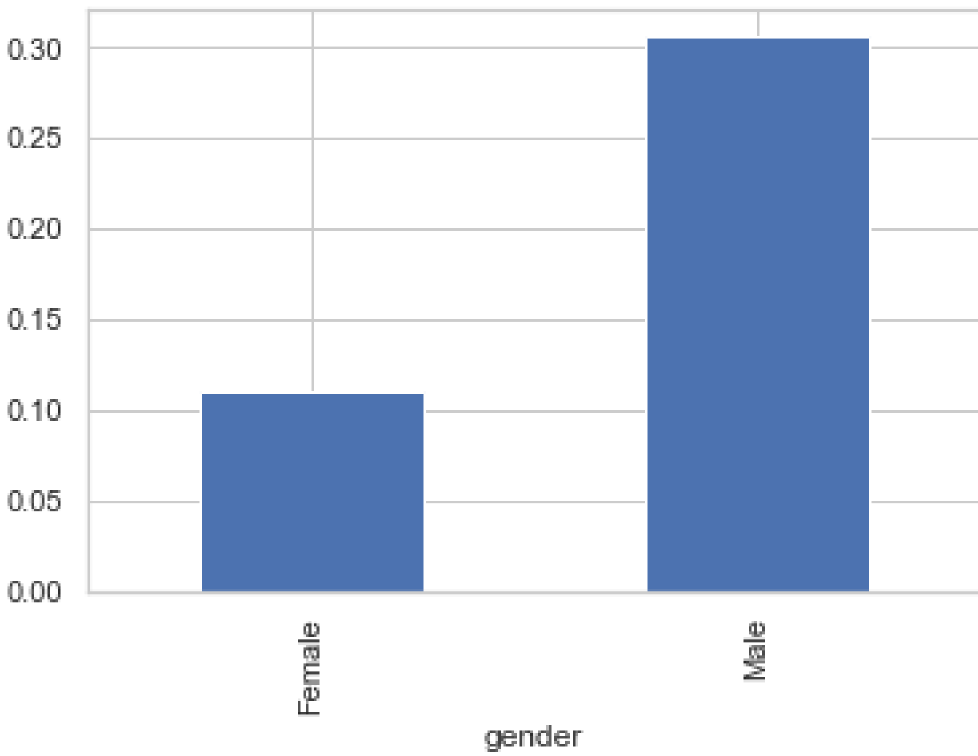
```
<AxesSubplot:xlabel='occupation'>
```



According to our statistics, people with the jobs Prof-specialty and Exec-managerial have a better chance of earning more than $50,000.

```
#To demonstrate the relationship between these columns, make a bar
graph of gender vs. Income.
income_data.groupby('gender').income.mean().plot(kind='bar')
```

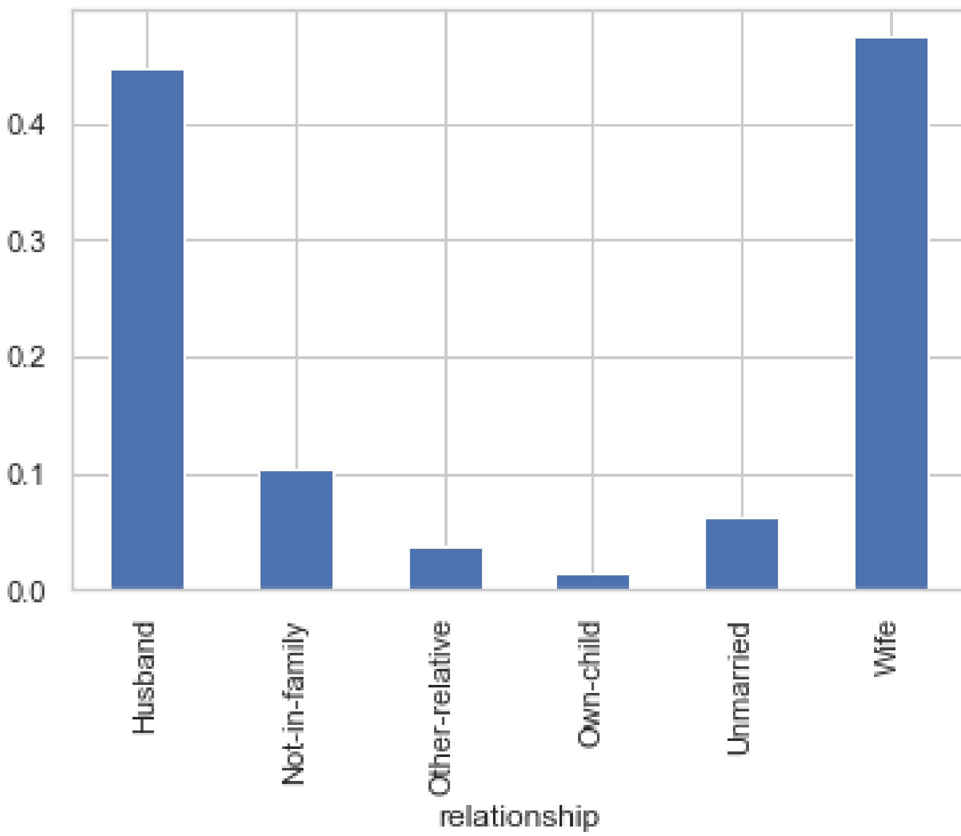
```
<AxesSubplot:xlabel='gender'>
```



The gender bar chart shows that men are more likely to have a greater salary than women.

```
#Creating a bar graph for relationship vs. Income to illustrate how
these two columns are related
income_data.groupby('relationship').income.mean().plot(kind='bar')
```

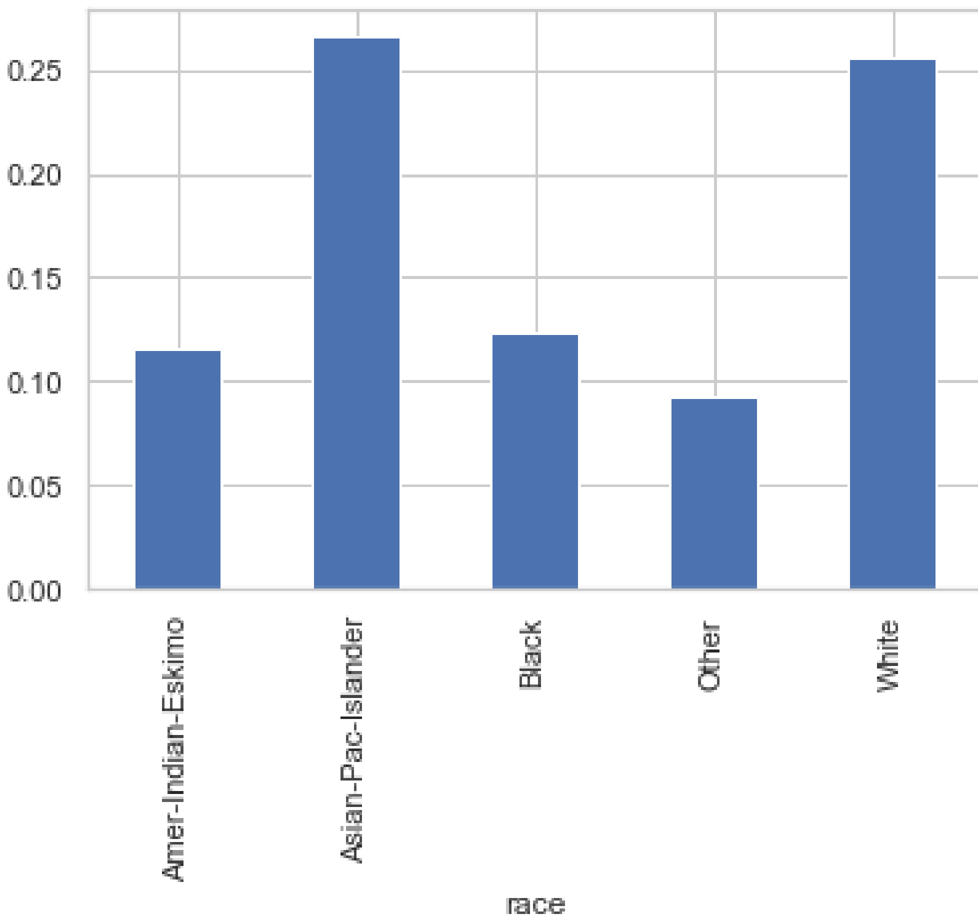
```
<AxesSubplot:xlabel='relationship'>
```



We can see from the relationship chart that the wife and husband have a larger income. A married pair would very certainly earn more than $50,000.

```
#Creating a bar graph for race vs. Income to illustrate how these two columns are related
income_data.groupby('race').income.mean().plot(kind='bar')
```

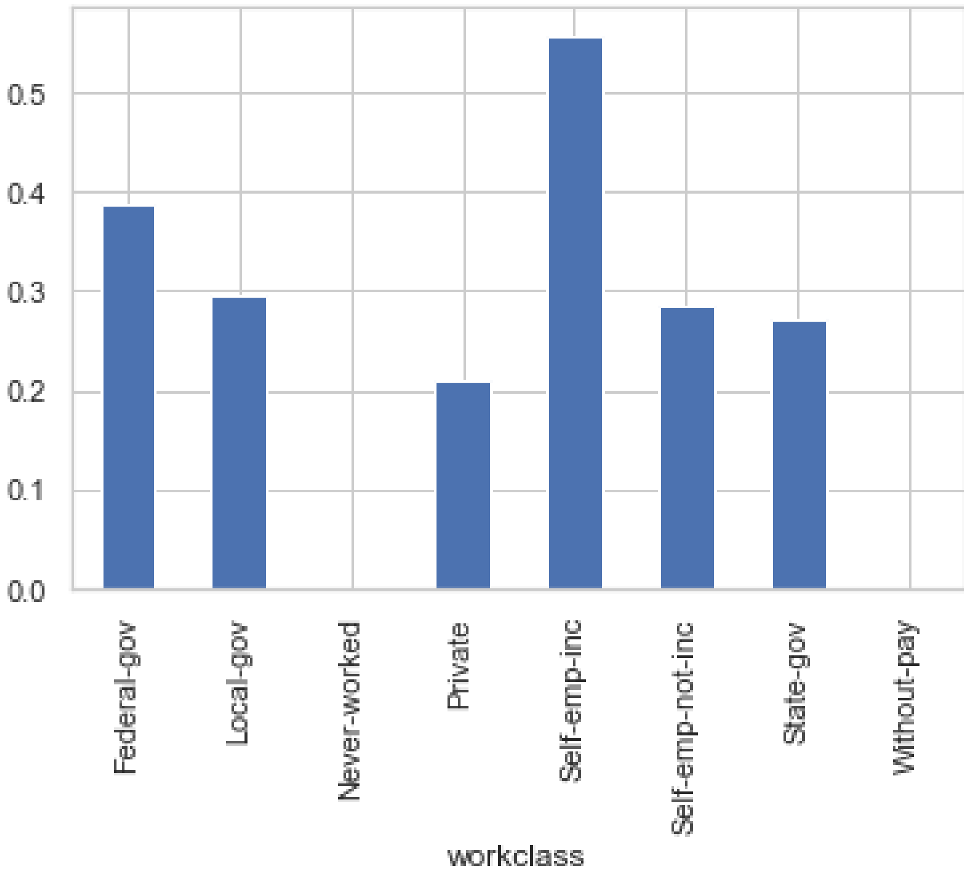
```
<AxesSubplot:xlabel='race'>
```



According to the data, an Asian-Pacific Islander or a white person had a better probability of earning more than $50,000.

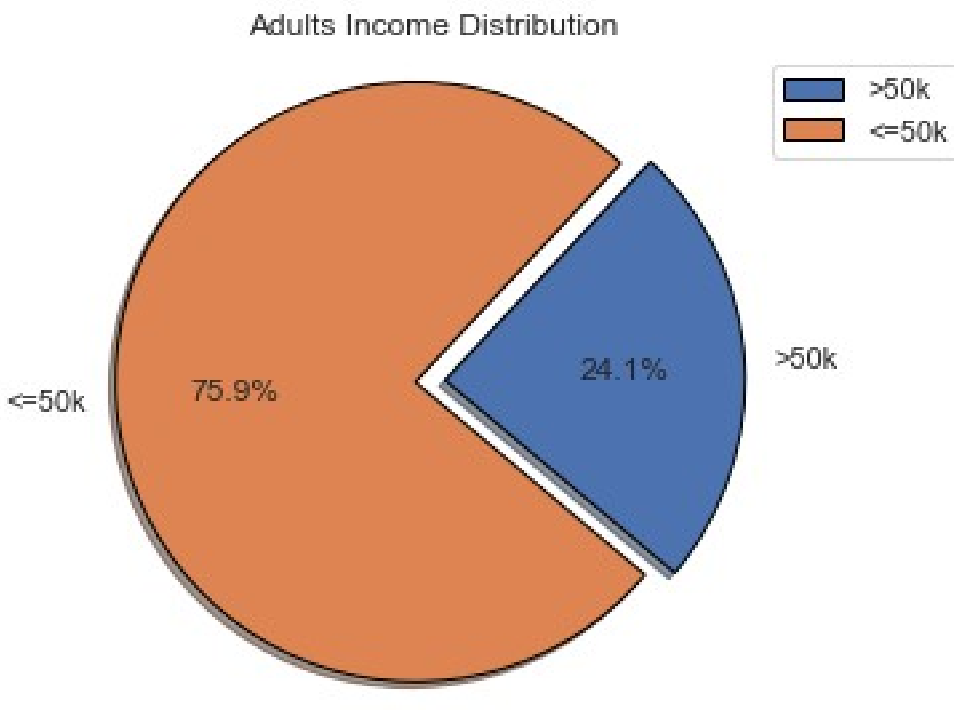
```
#Creating a bar graph for workclass vs. Income to illustrate how these two columns are related
income_data.groupby('workclass').income.mean().plot(kind='bar')
```

```
<AxesSubplot:xlabel='workclass'>
```



Self-employees in Federal government workclasses have a better probability of earning more than $50,000.

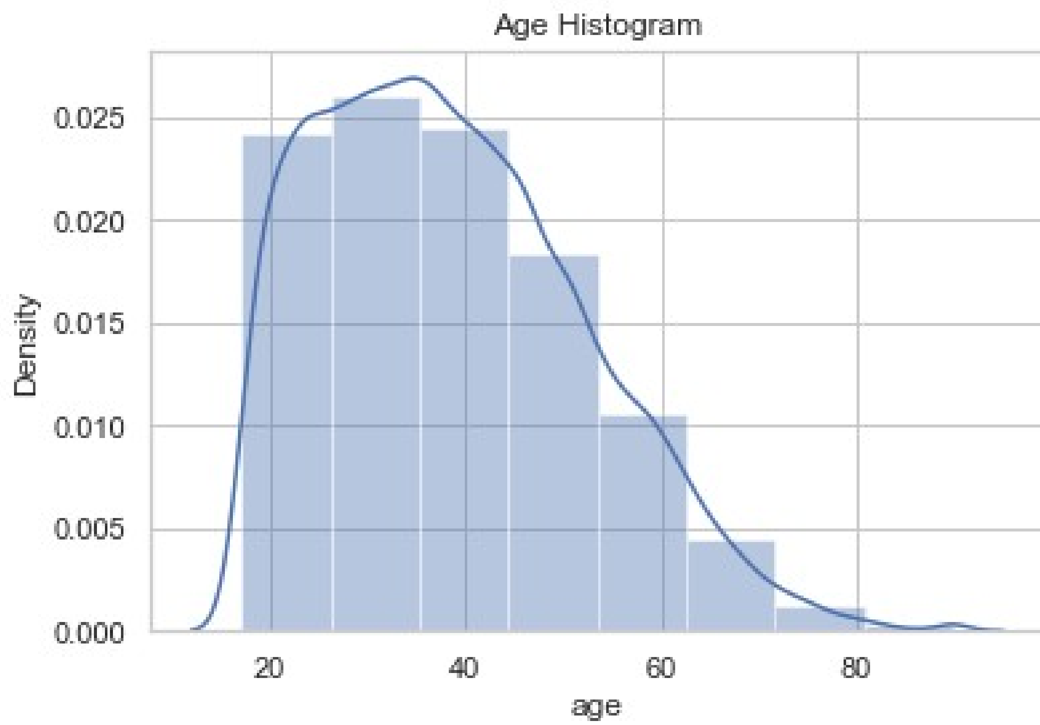
```
# plotting pie chart for percentage of incomes of adults

countG50k = income_data.income[income_data['income']==1].count()
countL50k = income_data.income[income_data['income']==0].count()

labels = ['>50k', '<=50k']
slices = [countG50k, countL50k]
explode = [0, 0.1]

plt.pie(slices, labels=labels, explode=explode, shadow=True,
       startangle = -40, autopct='%1.1f%%',
       wedgeprops ={'edgecolor' : 'black'})
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a
circle
plt.title("Adults Income Distribution")
plt.legend()
plt.tight_layout() # Used for padding
plt.savefig('Adults Income distribution.jpg')
plt.show()
```



Visualisation of Numerical Attributes

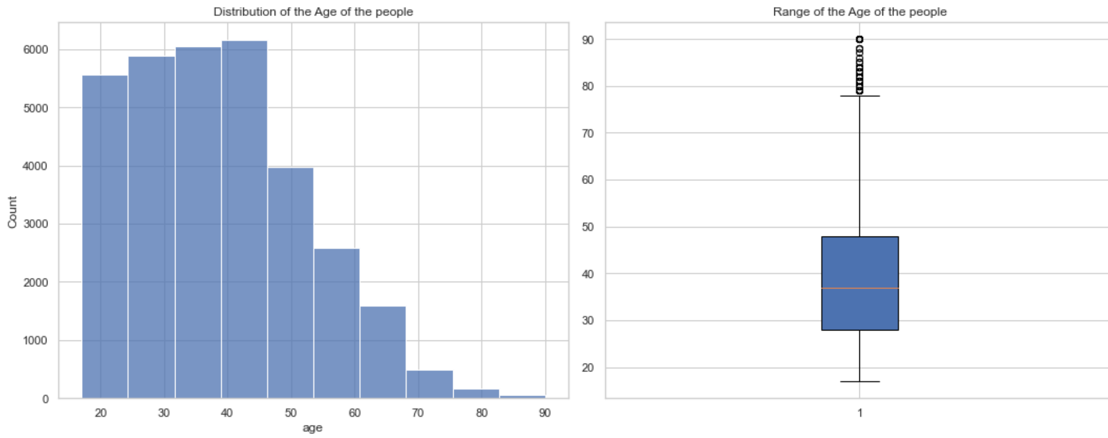
```
#Histogram representing age
sns.distplot(income_data['age'],bins=8)
plt.title("Age Histogram")
```

```
Text(0.5, 1.0, 'Age Histogram')
```



We can see from the above histogram that the majority of the people are between the ages of 20 and 40, with the age group beyond 80 having the least number of people.

```
# Histogram and Boxplot for Age

fig, axes = plt.subplots(1,2,figsize=(15,6))

sns.histplot(income_data['age'],bins=10,ax=axes[0])
plt.boxplot(income_data['age'], patch_artist = True)

axes[0].set_title('Distribution of the Age of the people')
axes[1].set_title('Range of the Age of the people')
plt.tight_layout()
#plt.savefig("Figure3.png")
plt.show()
```




```
income_data.describe()
```

```
                  age         fnlwgt  EducationNum   CapitalGain  
CapitalLoss  \
count  32537.000000  3.253700e+04  32537.000000  32537.000000  
32537.000000
mean      38.585549  1.897808e+05     10.081815   1078.443741  
87.368227
std       13.637984  1.055565e+05      2.571633   7387.957424  
403.101833
min       17.000000  1.228500e+04      1.000000      0.000000  
0.000000
25%       28.000000  1.178270e+05      9.000000      0.000000  
0.000000
50%       37.000000  1.783560e+05     10.000000      0.000000  
0.000000
75%       48.000000  2.369930e+05     12.000000      0.000000  
0.000000
max       90.000000  1.484705e+06     16.000000  99999.000000  
4356.000000

       HoursPerWeek        income
count  32537.000000  32537.000000
mean      40.440329      0.240926
std       12.346889      0.427652
min        1.000000      0.000000
25%       40.000000      0.000000
50%       40.000000      0.000000
75%       45.000000      0.000000
max       99.000000      1.000000
```

Detect Outliers Based on Age Attribute

```
# Calculating Q1 and Q3 for age attribute

Q1 = income_data.age.quantile(0.25)
```

```
Q3 = income_data.age.quantile(0.75)
Q1, Q3
```

```
(28.0, 48.0)
```

```
# Calculating interquartile range for age attribute
IQR = Q3-Q1
IQR
```

```
20.0
```

```
#Calculate lower limit for age attribute
lower_limit = Q1 - 1.5*IQR

#Calculate upper limit for age attribute
upper_limit = Q3 + 1.5*IQR

lower_limit, upper_limit
```

```
(-2.0, 78.0)
```

```
income_data.age.describe()
```

```
count    32537.000000
mean        38.585549
std         13.637984
min         17.000000
25%         28.000000
50%         37.000000
75%         48.000000
max         90.000000
Name: age, dtype: float64
```

```
# removing the outlier
index = income_data.loc[~(income_data.age>upper_limit),"age"].index
index
```

```
Int64Index([    2,     3,     4,     5,     6,     7,     8,     9,
10,
               11,
            ...
            32551, 32552, 32553, 32554, 32555, 32556, 32557, 32558,
32559,
            32560],
           dtype='int64', length=32395)
```

```
income_data = income_data.loc[index]
income_data
```

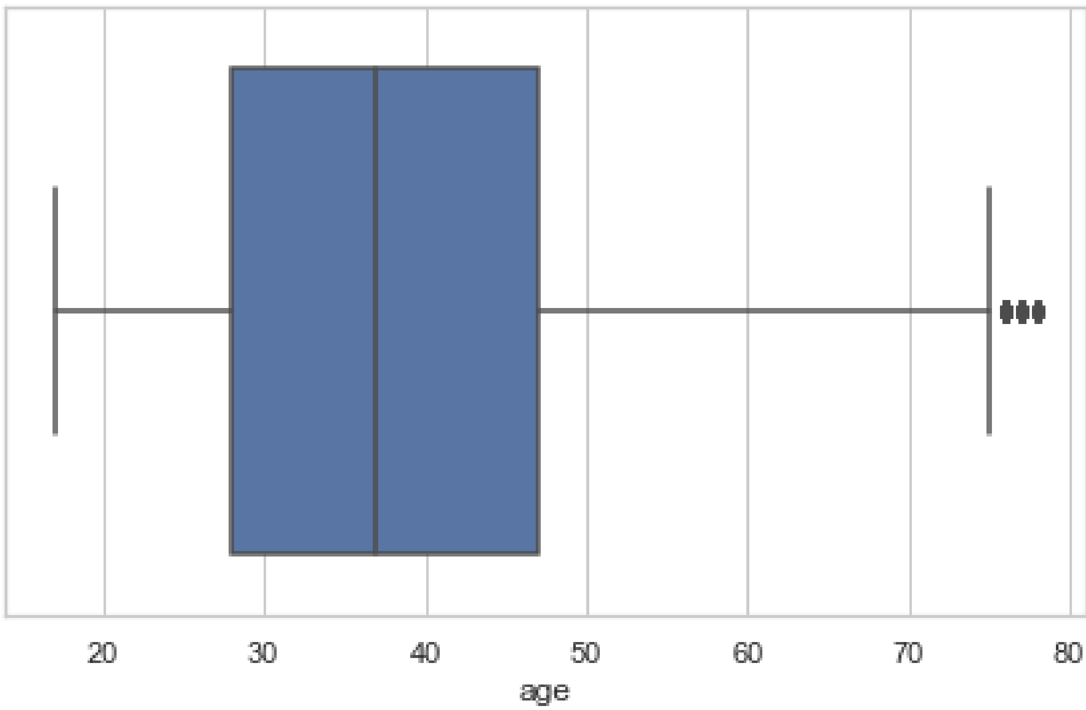
```
       age workclass  fnlwgt     education  EducationNum
MaritalStatus  \
2       66   Private  186061  Some-college            10
```

```
Widowed
3       54   Private  140359       7th-8th            4
Divorced
4       41   Private  264663  Some-college           10
Separated
5       34   Private  216864       HS-grad            9
Divorced
6       38   Private  150601          10th            6
Separated
...    ...       ...     ...           ...          ...
...
32556   22   Private  310152  Some-college           10       Never-
married
32557   27   Private  257302    Assoc-acdm           12  Married-civ-
spouse
32558   40   Private  154374       HS-grad            9  Married-civ-
spouse
32559   58   Private  151910       HS-grad            9
Widowed
32560   22   Private  201490       HS-grad            9       Never-
married

              occupation   relationship   race  gender  CapitalGain  \
2         Prof-specialty      Unmarried  Black  Female            0
3      Machine-op-inspct      Unmarried  White  Female            0
4         Prof-specialty      Own-child  White  Female            0
5          Other-service      Unmarried  White  Female            0
6           Adm-clerical      Unmarried  White    Male            0
...                  ...            ...    ...     ...          ...
32556    Protective-serv  Not-in-family  White    Male            0
32557       Tech-support           Wife  White  Female            0
32558  Machine-op-inspct        Husband  White    Male            0
32559       Adm-clerical      Unmarried  White  Female            0
32560       Adm-clerical      Own-child  White    Male            0

       CapitalLoss  HoursPerWeek        Country  income
2             4356            40  United-States       0
3             3900            40  United-States       0
4             3900            40  United-States       0
5             3770            45  United-States       0
6             3770            40  United-States       0
...            ...           ...            ...     ...
32556            0            40  United-States       0
32557            0            38  United-States       0
32558            0            40  United-States       1
32559            0            40  United-States       0
32560            0            20  United-States       0

[32395 rows x 15 columns]
```

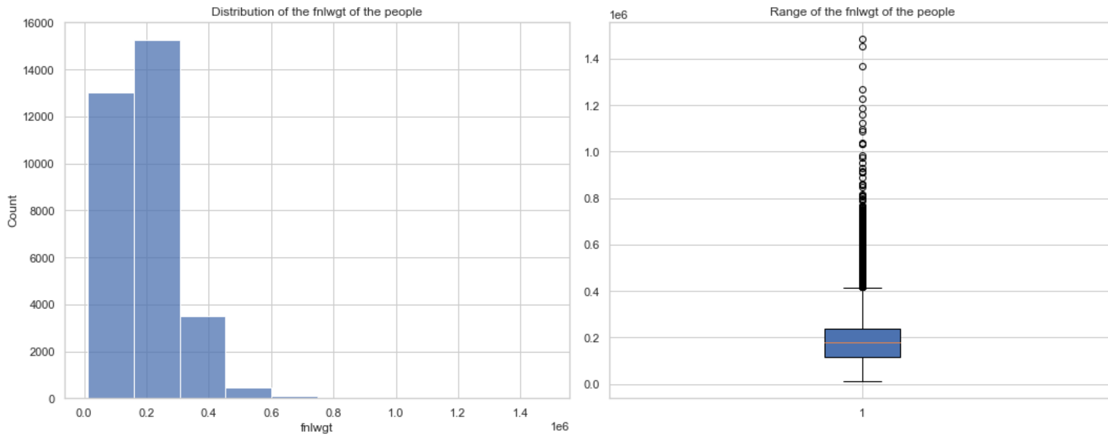
```
income_data.shape
```

```
(32395, 15)
```

```
# Visualizing distribution  of 'age' attribute
sns.boxplot(x = 'age', data = income_data)
plt.tight_layout()
plt.show()
```




```
# Histogram and Boxplot for fnlwgt

fig, axes = plt.subplots(1,2,figsize=(15,6))

sns.histplot(income_data['fnlwgt'],bins=10,ax=axes[0])
plt.boxplot(income_data['fnlwgt'], patch_artist = True)

axes[0].set_title('Distribution of the fnlwgt of the people')
axes[1].set_title('Range of the fnlwgt of the people')
plt.tight_layout()
#plt.savefig("Figure4.png")
plt.show()
```



Detect Outliers based on fnlwgt attribute

```
# Calculating Q1 and Q3 for fnlwgt attribute
Q1 = income_data.fnlwgt.quantile(0.25)
Q3 = income_data.fnlwgt.quantile(0.75)
Q1, Q3
```

```
(117849.0, 237282.5)
```

```
# Calculating Interquartile range for fnlwgt attribute
IQR = Q3-Q1
IQR
```

```
119433.5
```

```
# calculating lower and upper limits
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

```
(-61301.25, 416432.75)
```

```
income_data.fnlwgt.describe()
```

```
count    3.239500e+04
mean     1.899060e+05
std      1.056092e+05
min      1.228500e+04
25%      1.178490e+05
50%      1.785060e+05
75%      2.372825e+05
max      1.484705e+06
Name: fnlwgt, dtype: float64
```

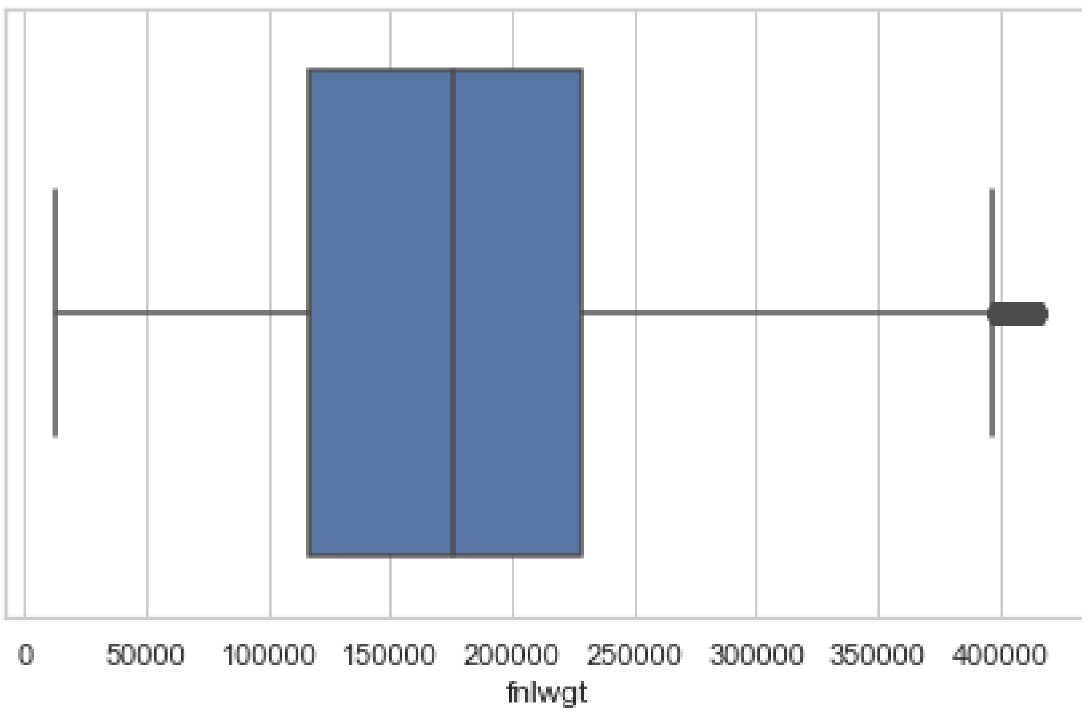
```
# Removing the outliers from fnlwgt attribute
index1 = income_data.loc[~((income_data.fnlwgt<lower_limit)|
(income_data.fnlwgt>upper_limit)),"fnlwgt"].index
index1
```

```
Int64Index([    2,     3,     4,     5,     6,     7,     9,    10,
11,
               12,
            ...
            32551, 32552, 32553, 32554, 32555, 32556, 32557, 32558,
32559,
            32560],
           dtype='int64', length=31414)
```

```
income_data = income_data.loc[index1]
income_data.shape
```

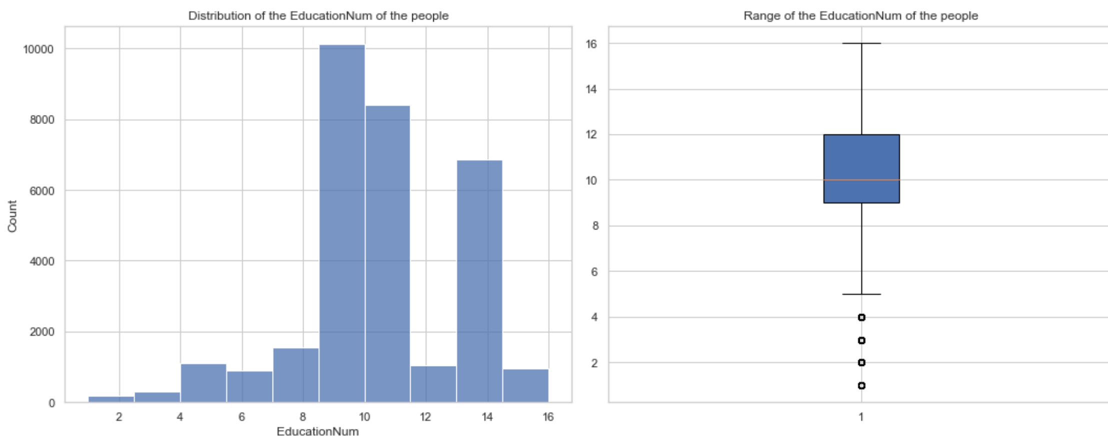
```
(31414, 15)
```

```
sns.boxplot(x = 'fnlwgt', data = income_data)
plt.tight_layout()
plt.show()
```




```
# Histogram and Boxplot for EducationNum

fig, axes = plt.subplots(1,2,figsize=(15,6))

sns.histplot(income_data['EducationNum'],bins=10,ax=axes[0])
plt.boxplot(income_data['EducationNum'], patch_artist = True)

axes[0].set_title('Distribution of the EducationNum of the people')
axes[1].set_title('Range of the EducationNum of the people')
plt.tight_layout()
```

```
#plt.savefig("hist of EducationNum.png")
plt.show()
```



Detect Outliers based on EducationNum attribute

```
# Calculating Q1 and Q3 for EducationNum attribute
Q1 = income_data.EducationNum.quantile(0.25)
Q3 = income_data.EducationNum.quantile(0.75)
Q1, Q3
```

```
(9.0, 12.0)
```

```
# Calculating interquartile range for EducationNum attribute
IQR = Q3-Q1
IQR
```

```
3.0
```

```
# calculating lower and upper limits
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

```
(4.5, 16.5)
```

```
income_data.EducationNum.describe()
```

```
count    31414.000000
mean        10.097600
std          2.557403
min          1.000000
25%          9.000000
50%         10.000000
75%         12.000000
max         16.000000
Name: EducationNum, dtype: float64
```

```
# Removing the outliers from EducationNum attribute
index1= income_data.loc[~((income_data.EducationNum<lower_limit)|
```

```
(income_data.EducationNum>upper_limit)),"EducationNum"].index
index1
```

```
Int64Index([    2,     4,     5,     6,     7,     9,    10,    11,
12,
               13,
            ...
            32551, 32552, 32553, 32554, 32555, 32556, 32557, 32558,
32559,
            32560],
           dtype='int64', length=30303)
```

```
income_data = income_data.loc[index1]
income_data
```

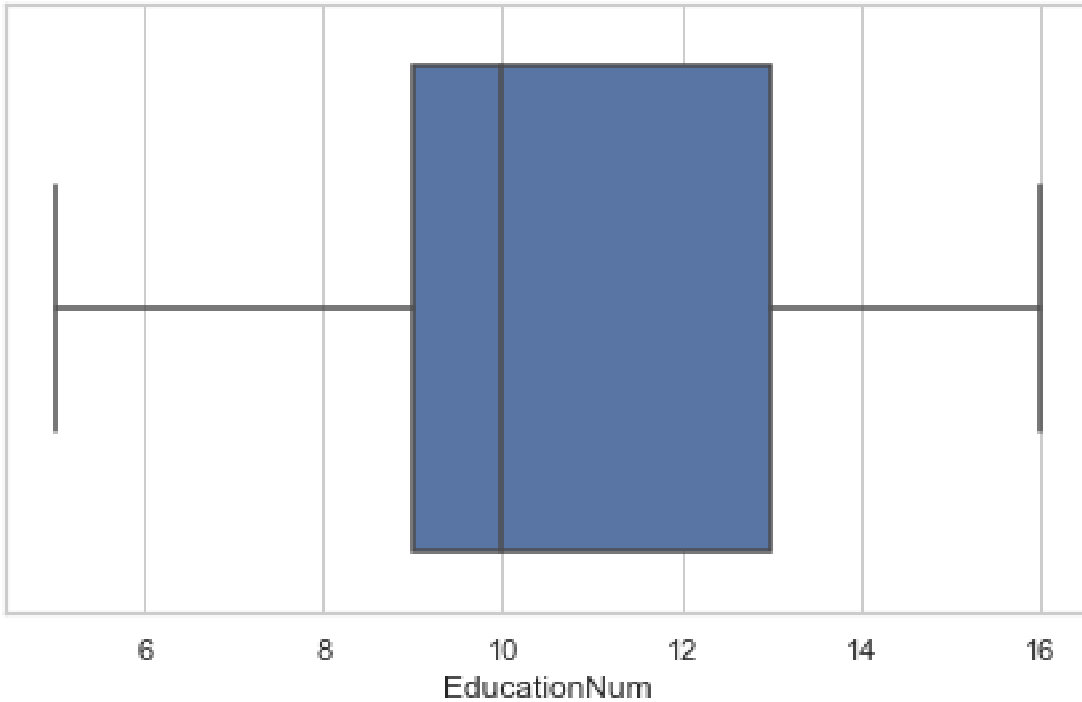
```
       age  workclass  fnlwgt     education  EducationNum
MaritalStatus  \
2       66    Private  186061  Some-college            10
Widowed
4       41    Private  264663  Some-college            10
Separated
5       34    Private  216864       HS-grad             9
Divorced
6       38    Private  150601          10th             6
Separated
7       74  State-gov   88638     Doctorate            16       Never-
married
...    ...        ...     ...           ...           ...
...
32556   22    Private  310152  Some-college            10       Never-
married
32557   27    Private  257302    Assoc-acdm            12  Married-
civ-spouse
32558   40    Private  154374       HS-grad             9  Married-
civ-spouse
32559   58    Private  151910       HS-grad             9
Widowed
32560   22    Private  201490       HS-grad             9       Never-
married

              occupation    relationship   race  gender
CapitalGain  \
2         Prof-specialty       Unmarried  Black  Female            0

4         Prof-specialty       Own-child  White  Female            0

5          Other-service       Unmarried  White  Female            0

6           Adm-clerical       Unmarried  White    Male            0
```

```
7         Prof-specialty  Other-relative  White  Female            0

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

32556    Protective-serv   Not-in-family  White    Male            0

32557       Tech-support            Wife  White  Female            0

32558  Machine-op-inspct         Husband  White    Male            0

32559       Adm-clerical       Unmarried  White  Female            0

32560       Adm-clerical       Own-child  White    Male            0


       CapitalLoss  HoursPerWeek        Country  income
2             4356            40  United-States       0
4             3900            40  United-States       0
5             3770            45  United-States       0
6             3770            40  United-States       0
7             3683            20  United-States       1
...            ...           ...            ...     ...
32556            0            40  United-States       0
32557            0            38  United-States       0
32558            0            40  United-States       1
32559            0            40  United-States       0
32560            0            20  United-States       0

[30303 rows x 15 columns]
```

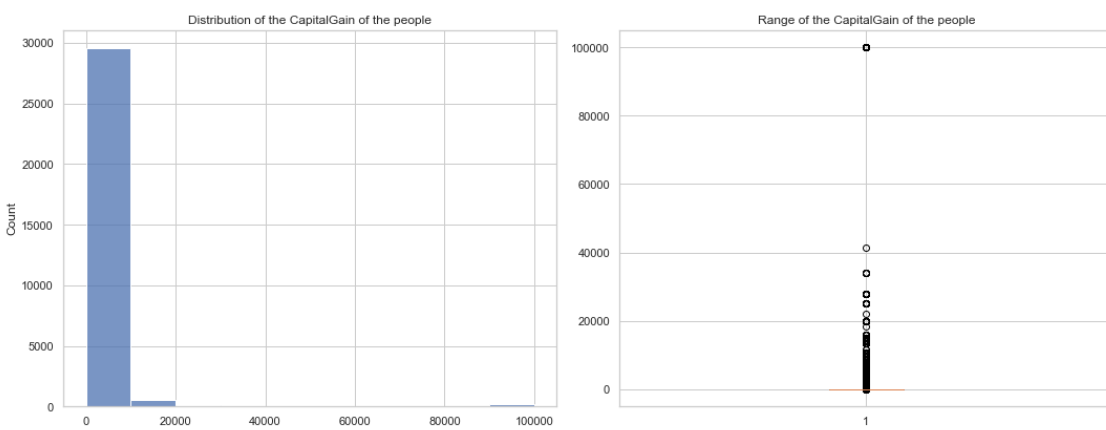
```
income_data.shape
```

```
(30303, 15)
```

```
sns.boxplot(x = 'EducationNum', data = income_data)
plt.tight_layout()
plt.show()
```




```
# Histogram and Boxplot for CapitalGain

fig, axes = plt.subplots(1,2,figsize=(15,6))

sns.histplot(income_data['CapitalGain'],bins=10,ax=axes[0])
plt.boxplot(income_data['CapitalGain'], patch_artist = True)

axes[0].set_title('Distribution of the CapitalGain of the people')
axes[1].set_title('Range of the CapitalGain of the people')
plt.tight_layout()
#plt.savefig("hist of CapitalGain.png")
plt.show()
```



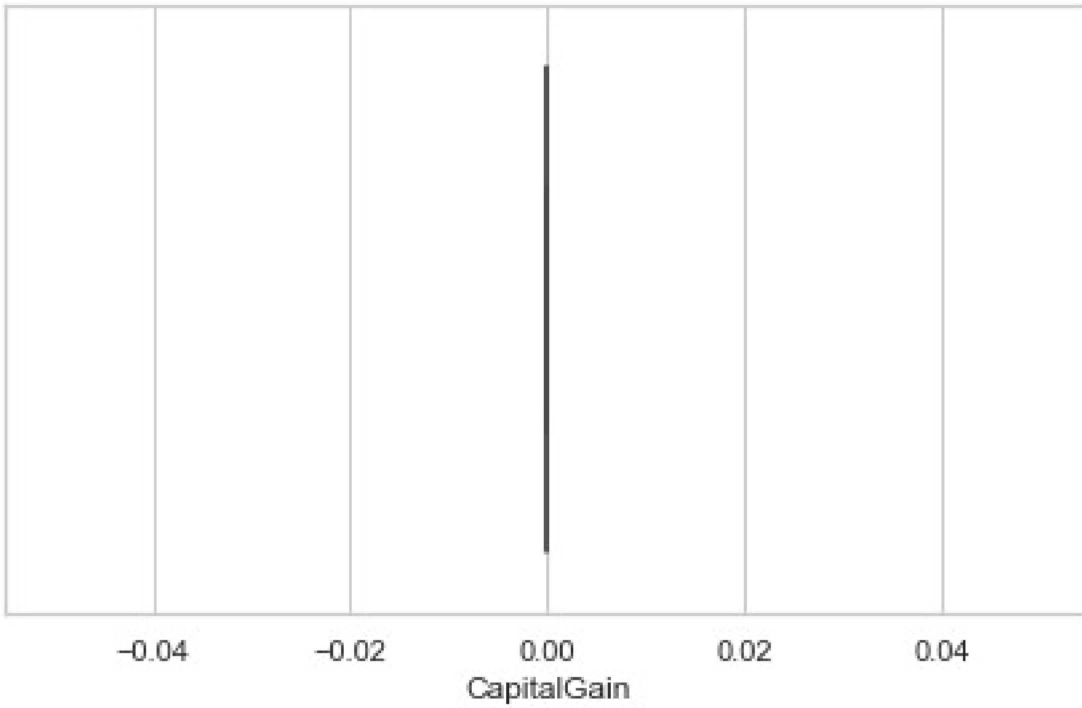
Detect Outliers based on CapitalGain attribute

```python
# Calculating Q1 and Q3 for CapitalGain attribute
Q1 = income_data.CapitalGain.quantile(0.25)
Q3 = income_data.CapitalGain.quantile(0.75)
Q1, Q3

# Calculating IQR for CapitalGain attribute
IQR = Q3-Q1
IQR

# Calculating lower and upper limit for CapitalGain attribute
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit


# Removing the outliers from CapitalGain attribute
income_data = income_data[~((income_data.CapitalGain<lower_limit)|
(income_data.CapitalGain>upper_limit))]

income_data
```

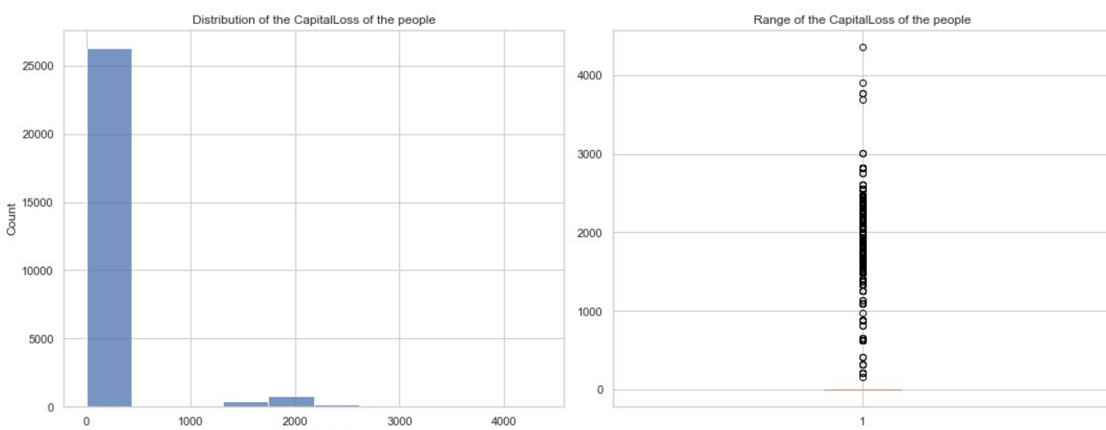
```
       age  workclass  fnlwgt     education  EducationNum
MaritalStatus  \
2       66    Private  186061  Some-college            10
Widowed
4       41    Private  264663  Some-college            10
Separated
5       34    Private  216864       HS-grad             9
Divorced
6       38    Private  150601          10th             6
Separated
7       74  State-gov   88638     Doctorate            16       Never-
married
...    ...        ...     ...           ...           ...
...
32556   22    Private  310152  Some-college            10       Never-
married
32557   27    Private  257302    Assoc-acdm            12  Married-
civ-spouse
32558   40    Private  154374       HS-grad             9  Married-
civ-spouse
32559   58    Private  151910       HS-grad             9
Widowed
32560   22    Private  201490       HS-grad             9       Never-
married

              occupation    relationship   race  gender
CapitalGain  \
2         Prof-specialty       Unmarried  Black  Female           0
```

```
4         Prof-specialty       Own-child  White  Female             0
5          Other-service       Unmarried  White  Female             0
6           Adm-clerical       Unmarried  White    Male             0
7         Prof-specialty  Other-relative  White  Female             0
...                  ...             ...    ...     ...           ...
32556    Protective-serv   Not-in-family  White    Male             0
32557       Tech-support            Wife  White  Female             0
32558  Machine-op-inspct         Husband  White    Male             0
32559       Adm-clerical       Unmarried  White  Female             0
32560       Adm-clerical       Own-child  White    Male             0

       CapitalLoss  HoursPerWeek        Country  income
2             4356            40  United-States       0
4             3900            40  United-States       0
5             3770            45  United-States       0
6             3770            40  United-States       0
7             3683            20  United-States       1
...            ...           ...            ...     ...
32556            0            40  United-States       0
32557            0            38  United-States       0
32558            0            40  United-States       1
32559            0            40  United-States       0
32560            0            20  United-States       0

[27743 rows x 15 columns]
```

```
sns.boxplot(x = 'CapitalGain', data = income_data)
plt.tight_layout()
plt.show()
```




```
# Histogram and Boxplot for CapitalLoss

fig, axes = plt.subplots(1,2,figsize=(15,6))

sns.histplot(income_data['CapitalLoss'],bins=10,ax=axes[0])
plt.boxplot(income_data['CapitalLoss'], patch_artist = True)

axes[0].set_title('Distribution of the CapitalLoss of the people')
axes[1].set_title('Range of the CapitalLoss of the people')
plt.tight_layout()
#plt.savefig("hist of CapitalLoss.png")
plt.show()
```



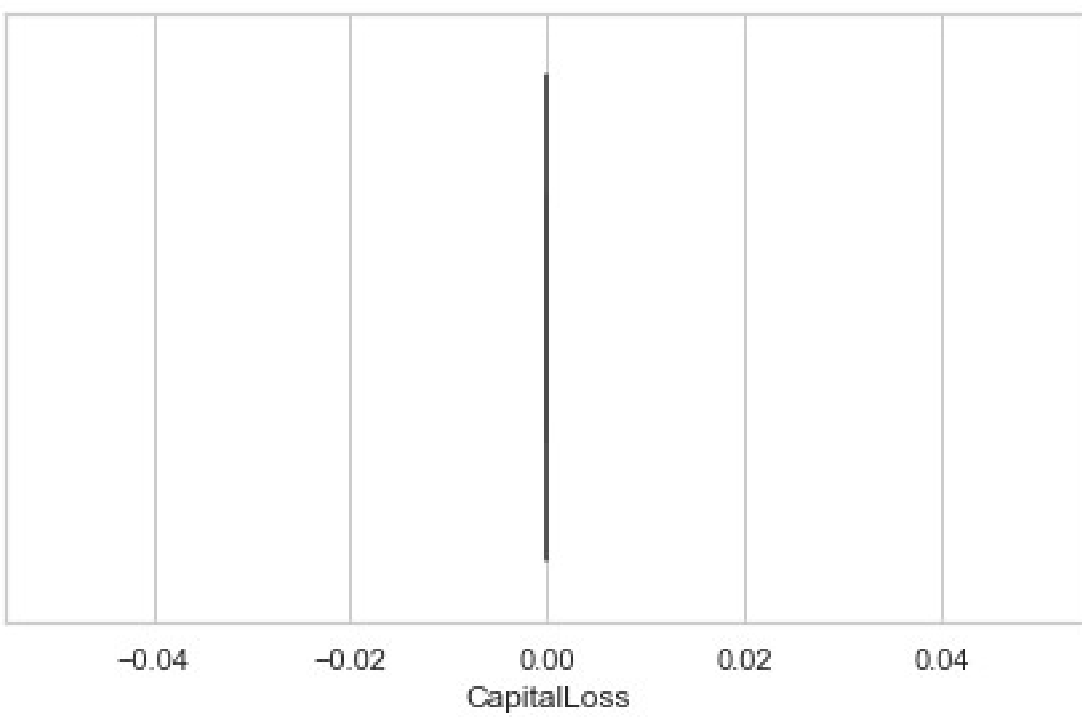
Detect Outliers based on CapitalLoss attribute

```
# Calculating Q1 and Q3 for CapitalLoss attribute
Q1 = income_data.CapitalLoss.quantile(0.25)
Q3 = income_data.CapitalLoss.quantile(0.75)
Q1, Q3

# Calculating IQR for CapitalLoss attribute
IQR = Q3-Q1
IQR

# Calculating lower and upper limit for CapitalLoss attribute
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit


# Removing the outliers from CapitalLoss attribute
income_data = income_data[~((income_data.CapitalLoss<lower_limit)|
(income_data.CapitalLoss>upper_limit))]
income_data
```

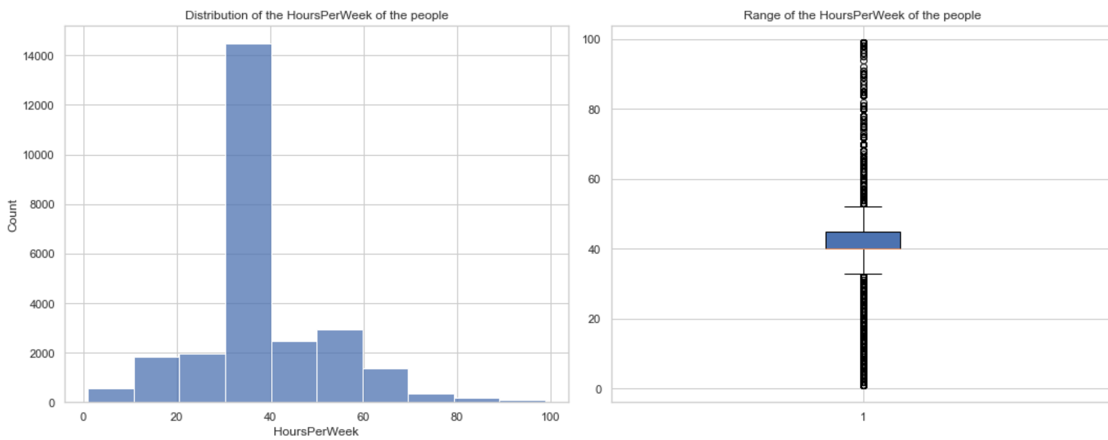
```
       age         workclass  fnlwgt     education  EducationNum  \
4231    50  Self-emp-not-inc   83311     Bachelors            13
4232    38           Private  215646       HS-grad             9
4233    53           Private  234721          11th             7
4234    28           Private  338409     Bachelors            13
4235    37           Private  284582       Masters            14
...    ...               ...     ...           ...           ...
32556   22           Private  310152  Some-college            10
32557   27           Private  257302    Assoc-acdm            12
32558   40           Private  154374       HS-grad             9
32559   58           Private  151910       HS-grad             9
32560   22           Private  201490       HS-grad             9

            MaritalStatus         occupation   relationship   race
gender  \
4231   Married-civ-spouse    Exec-managerial        Husband  White
Male
4232             Divorced  Handlers-cleaners  Not-in-family  White
Male
4233   Married-civ-spouse  Handlers-cleaners        Husband  Black
Male
4234   Married-civ-spouse     Prof-specialty           Wife  Black
Female
4235   Married-civ-spouse    Exec-managerial           Wife  White
Female
...                   ...                ...            ...    ...
...
32556       Never-married    Protective-serv  Not-in-family  White
Male
32557  Married-civ-spouse       Tech-support           Wife  White
```

```
Female
32558  Married-civ-spouse  Machine-op-inspct        Husband  White
Male
32559             Widowed       Adm-clerical      Unmarried  White
Female
32560       Never-married       Adm-clerical      Own-child  White
Male

       CapitalGain  CapitalLoss  HoursPerWeek        Country  income
4231             0            0            13  United-States       0
4232             0            0            40  United-States       0
4233             0            0            40  United-States       0
4234             0            0            40           Cuba       0
4235             0            0            40  United-States       0
...            ...          ...           ...            ...     ...
32556            0            0            40  United-States       0
32557            0            0            38  United-States       0
32558            0            0            40  United-States       1
32559            0            0            40  United-States       0
32560            0            0            20  United-States       0

[26300 rows x 15 columns]
```

```
sns.boxplot(x = 'CapitalLoss', data = income_data)
plt.tight_layout()
plt.show()
```



```
# Histogram and Boxplot for HoursPerWeek

fig, axes = plt.subplots(1,2,figsize=(15,6))

sns.histplot(income_data['HoursPerWeek'],bins=10,ax=axes[0])
plt.boxplot(income_data['HoursPerWeek'], patch_artist = True)

axes[0].set_title('Distribution of the HoursPerWeek of the people')
axes[1].set_title('Range of the HoursPerWeek of the people')
plt.tight_layout()
#plt.savefig("hist of HoursPerWeek.png")
plt.show()
```



Detect Outliers based on HoursPerWeek attribute

```
# Calculating Q1 and Q3 for HoursPerWeek attribute
Q1 = income_data.HoursPerWeek.quantile(0.25)
Q3 = income_data.HoursPerWeek.quantile(0.75)
Q1, Q3

# Calculating IQR for HoursPerWeek attribute
IQR = Q3-Q1
IQR

# Calculating lower and upper limit for HoursPerWeek attribute
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

```
(32.5, 52.5)
```

```
income_data.HoursPerWeek.describe()
```

```
count    26300.00000
mean        40.12384
std         12.26543
min          1.00000
```

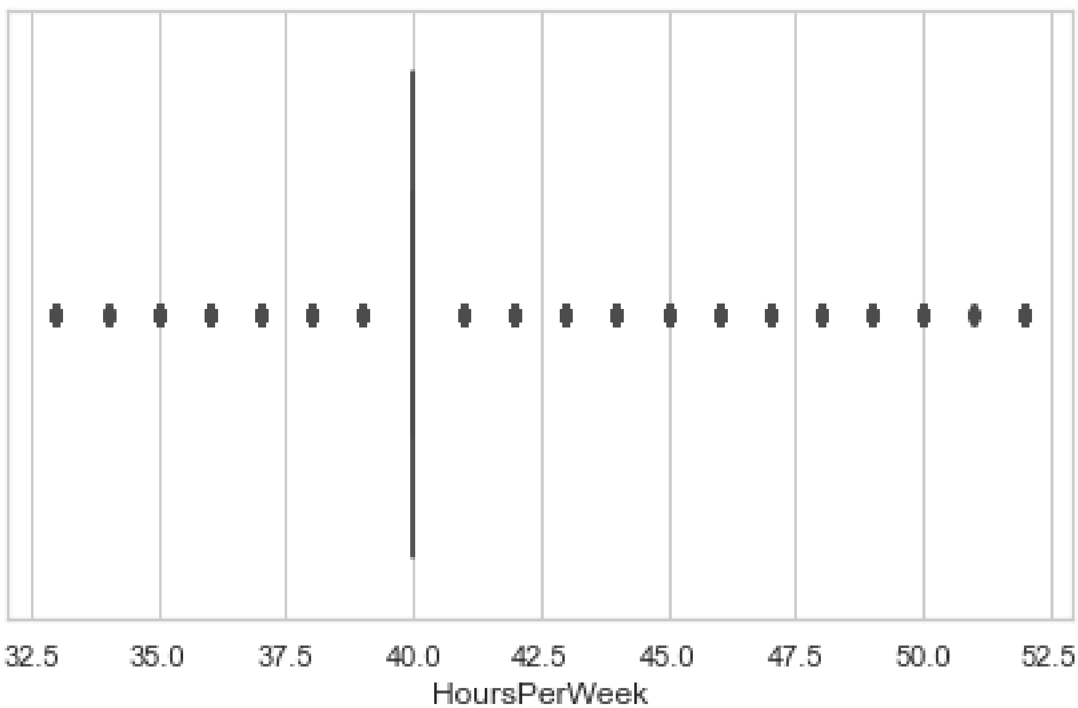
```
25%          40.00000
50%          40.00000
75%          45.00000
max          99.00000
Name: HoursPerWeek, dtype: float64
```

```
# Removing the outliers from HoursPerWeek attribute
income_data = income_data[~((income_data.HoursPerWeek<lower_limit)|
(income_data.HoursPerWeek>upper_limit))]
income_data
```

```
       age         workclass  fnlwgt     education  EducationNum  \
4232    38           Private  215646       HS-grad             9   
4233    53           Private  234721          11th             7   
4234    28           Private  338409     Bachelors            13   
4235    37           Private  284582       Masters            14   
4237    52  Self-emp-not-inc  209642       HS-grad             9   
...    ...               ...     ...           ...           ...   
32555   53           Private  321865       Masters            14   
32556   22           Private  310152  Some-college            10   
32557   27           Private  257302    Assoc-acdm            12   
32558   40           Private  154374       HS-grad             9   
32559   58           Private  151910       HS-grad             9   

            MaritalStatus         occupation   relationship   race 
gender  \
4232             Divorced  Handlers-cleaners  Not-in-family  White 
Male   
4233   Married-civ-spouse  Handlers-cleaners        Husband  Black 
Male   
4234   Married-civ-spouse     Prof-specialty           Wife  Black 
Female   
4235   Married-civ-spouse    Exec-managerial           Wife  White 
Female   
4237   Married-civ-spouse    Exec-managerial        Husband  White 
Male   
...                   ...                ...            ...    ... 
...   
32555  Married-civ-spouse    Exec-managerial        Husband  White 
Male   
32556       Never-married    Protective-serv  Not-in-family  White 
Male   
32557  Married-civ-spouse       Tech-support           Wife  White 
Female   
32558  Married-civ-spouse  Machine-op-inspct        Husband  White 
Male   
32559             Widowed       Adm-clerical      Unmarried  White 
Female   

       CapitalGain  CapitalLoss  HoursPerWeek        Country  income
```

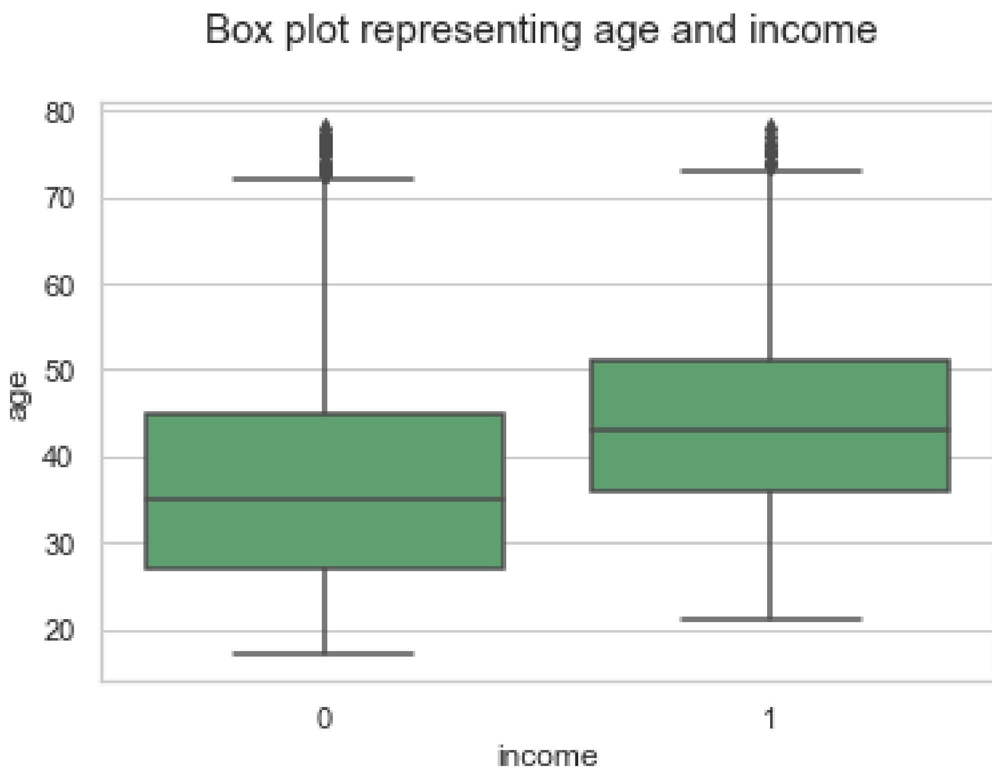
```
4232              0            0          40  United-States       0
4233              0            0          40  United-States       0
4234              0            0          40           Cuba       0
4235              0            0          40  United-States       0
4237              0            0          45  United-States       1
...             ...          ...         ...            ...     ...
32555             0            0          40  United-States       1
32556             0            0          40  United-States       0
32557             0            0          38  United-States       0
32558             0            0          40  United-States       1
32559             0            0          40  United-States       0

[18991 rows x 15 columns]
```

```
sns.boxplot(x = 'HoursPerWeek', data = income_data)
plt.tight_layout()
plt.show()
```




```
#Variation between age and income level
sns.boxplot(x=income_data['income'],y = income_data['age'],color='g')
plt.title("Box plot representing age and income \n", size = 15)
#plt.savefig("Box plot representing age and income.png")
```

```
Text(0.5, 1.0, 'Box plot representing age and income \n')
```




```
# checking the variance of the variables

variance=round(income_data.var(),0)
#variance.to_csv("variance.csv")
```

I delete attributes with nearly zero variance from the data set because they don't provide any information about the dataset..

```
income_data = income_data.drop(["CapitalGain","CapitalLoss"],axis = 1)
```

```
income_data
```

```
       age         workclass  fnlwgt     education  EducationNum  \
4232    38           Private  215646       HS-grad             9   
4233    53           Private  234721          11th             7   
4234    28           Private  338409     Bachelors            13   
4235    37           Private  284582       Masters            14   
4237    52  Self-emp-not-inc  209642       HS-grad             9   
...    ...               ...     ...           ...           ...   
32555   53           Private  321865       Masters            14   
32556   22           Private  310152  Some-college            10   
32557   27           Private  257302    Assoc-acdm            12   
32558   40           Private  154374       HS-grad             9   
32559   58           Private  151910       HS-grad             9   

            MaritalStatus         occupation   relationship   race  
gender  \
```

```
4232             Divorced  Handlers-cleaners  Not-in-family  White
Male
4233   Married-civ-spouse  Handlers-cleaners        Husband  Black
Male
4234   Married-civ-spouse     Prof-specialty           Wife  Black
Female
4235   Married-civ-spouse    Exec-managerial           Wife  White
Female
4237   Married-civ-spouse    Exec-managerial        Husband  White
Male
...                   ...                ...            ...    ...
...
32555  Married-civ-spouse    Exec-managerial        Husband  White
Male
32556       Never-married    Protective-serv  Not-in-family  White
Male
32557  Married-civ-spouse       Tech-support           Wife  White
Female
32558  Married-civ-spouse  Machine-op-inspct        Husband  White
Male
32559             Widowed       Adm-clerical      Unmarried  White
Female

       HoursPerWeek        Country  income
4232             40  United-States       0
4233             40  United-States       0
4234             40           Cuba       0
4235             40  United-States       0
4237             45  United-States       1
...             ...            ...     ...
32555            40  United-States       1
32556            40  United-States       0
32557            38  United-States       0
32558            40  United-States       1
32559            40  United-States       0

[18991 rows x 13 columns]
```

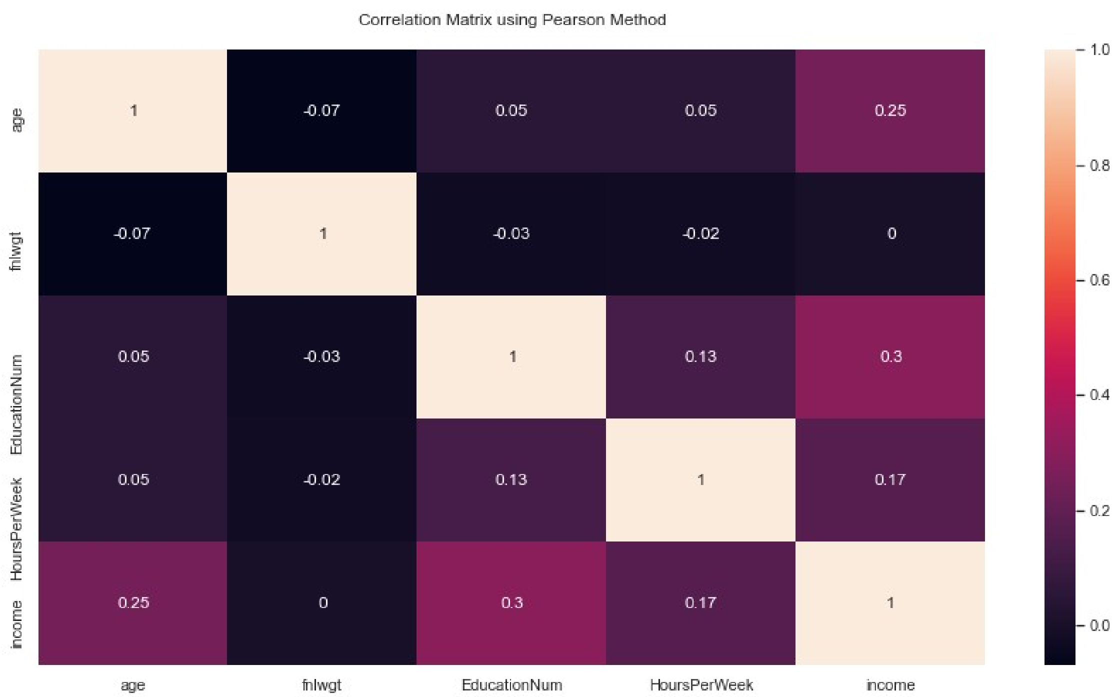
```
# printing the numerical and categorical variables individually

numeric = []
category = []
for col in income_data:
    if pd.api.types.is_numeric_dtype(income_data[col]):
        numeric.append(col)
    else:
        category.append(col)
print("category:", category)
print("numeric:", numeric)
```

```
category: ['workclass', 'education', 'MaritalStatus', 'occupation',
'relationship', 'race', 'gender', 'Country']
numeric: ['age', 'fnlwgt', 'EducationNum', 'HoursPerWeek', 'income']
```

```python
corr= income_data.corr().round(2)
corr
```

	age	fnlwgt	EducationNum	HoursPerWeek	income
age	1.00	-0.07	0.05	0.05	0.25
fnlwgt	-0.07	1.00	-0.03	-0.02	0.00
EducationNum	0.05	-0.03	1.00	0.13	0.30
HoursPerWeek	0.05	-0.02	0.13	1.00	0.17
income	0.25	0.00	0.30	0.17	1.00

```python
# plotting the correlation matrix
plt.figure(figsize=(15,8))
correlation_mat = income_data.corr().round(2)

sns.heatmap(correlation_mat, annot = True)
plt.title("\nCorrelation Matrix using Pearson Method\n")
#plt.savefig("corr.png",bbox_inches='tight')
plt.show()
```

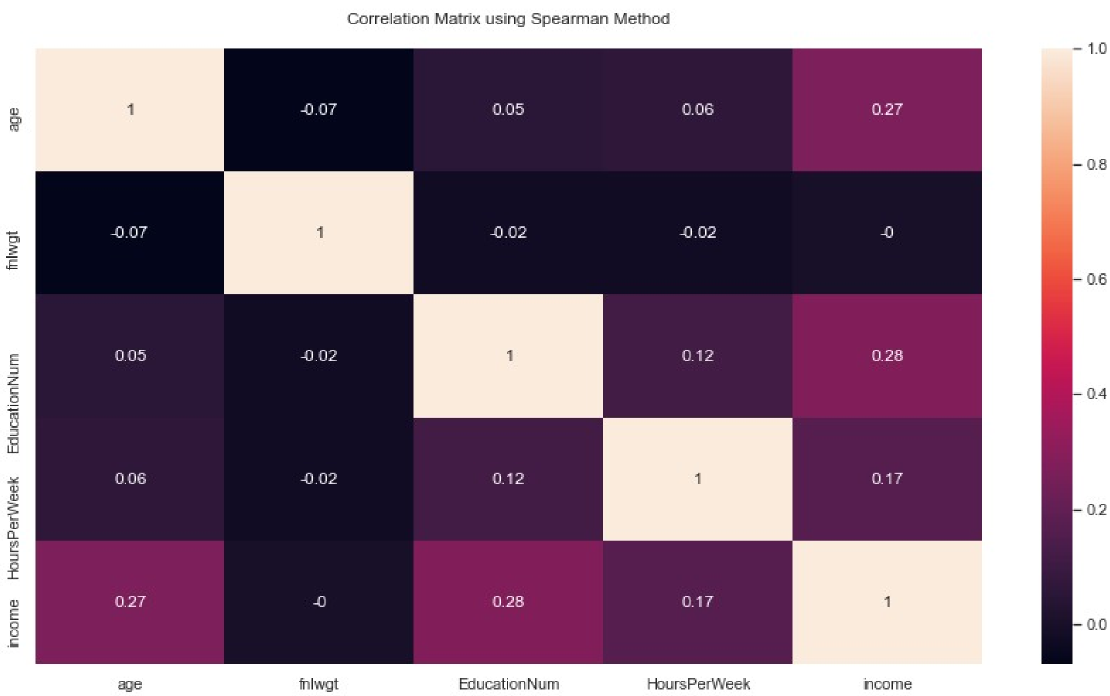



```python
corr1 = income_data.corr(method='spearman').round(2)
corr1
```

	age	fnlwgt	EducationNum	HoursPerWeek	income
age	1.00	-0.07	0.05	0.06	0.27
fnlwgt	-0.07	1.00	-0.02	-0.02	-0.00

```
EducationNum  0.05   -0.02          1.00          0.12     0.28
HoursPerWeek  0.06   -0.02          0.12          1.00     0.17
income        0.27   -0.00          0.28          0.17     1.00
```

```python
# plotting the correlation matrix
plt.figure(figsize=(15,8))
correlation_mat = income_data.corr(method='spearman').round(2)

sns.heatmap(correlation_mat, annot = True)
plt.title("\nCorrelation Matrix using Spearman Method\n")
#plt.savefig("corr2.png",bbox_inches='tight')
plt.show()
```



The above graphs show the relation between two variables using two different methods and shows how the change in one variable impact other. The value varies between -1 to 1. No strong correlation has been observed between any of the variables.

```python
# printing the numerical and categorical variables individually

numeric = []
category = []
for col in income_data:
    if pd.api.types.is_numeric_dtype(income_data[col]):
        numeric.append(col)
    else:
        category.append(col)
print("category:", category)
print("numeric:", numeric)
```

```
category: ['workclass', 'education', 'MaritalStatus', 'occupation',
'relationship', 'race', 'gender', 'Country']
numeric: ['age', 'fnlwgt', 'EducationNum', 'HoursPerWeek', 'income']
```

```
# choosing only numerical attributes from whole dataset
income_data.columns[~income_data.columns.isin(category)]
```

```
Index(['age', 'fnlwgt', 'EducationNum', 'HoursPerWeek', 'income'],
dtype='object')
```

```
# Choosing features for our model
income_new_data=
income_data[income_data.columns[~income_data.columns.isin(category)]]
income_new_data
```

```
       age  fnlwgt  EducationNum  HoursPerWeek  income
4232    38  215646             9            40       0
4233    53  234721             7            40       0
4234    28  338409            13            40       0
4235    37  284582            14            40       0
4237    52  209642             9            45       1
...    ...     ...           ...           ...     ...
32555   53  321865            14            40       1
32556   22  310152            10            40       0
32557   27  257302            12            38       0
32558   40  154374             9            40       1
32559   58  151910             9            40       0

[18991 rows x 5 columns]
```

```
income_new_data.dtypes
```

```
age             int64
fnlwgt          int64
EducationNum    int64
HoursPerWeek    int64
income          int32
dtype: object
```

```
income1= income_new_data.corr()
['income'].abs().sort_values(ascending=False)
income1
#income1.to_csv("corr_target.csv",index=True)
```

```
income          1.000000
EducationNum    0.295074
age             0.245649
HoursPerWeek    0.172764
fnlwgt          0.000373
Name: income, dtype: float64
```

```
#Correlation with output variable
cor_target = abs(corr1["income"])

#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.5]
relevant_features
```

```
income    1.0
Name: income, dtype: float64
```

```
income_data.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'EducationNum',
       'MaritalStatus', 'occupation', 'relationship', 'race',
'gender',
       'HoursPerWeek', 'Country', 'income'],
      dtype='object')
```

```
income_data
```

```
       age         workclass  fnlwgt     education  EducationNum  \
4232    38           Private  215646       HS-grad             9
4233    53           Private  234721          11th             7
4234    28           Private  338409     Bachelors            13
4235    37           Private  284582       Masters            14
4237    52  Self-emp-not-inc  209642       HS-grad             9
...    ...               ...     ...           ...           ...
32555   53           Private  321865       Masters            14
32556   22           Private  310152  Some-college            10
32557   27           Private  257302    Assoc-acdm            12
32558   40           Private  154374       HS-grad             9
32559   58           Private  151910       HS-grad             9

            MaritalStatus         occupation   relationship   race
gender  \
4232             Divorced  Handlers-cleaners  Not-in-family  White
Male
4233   Married-civ-spouse  Handlers-cleaners        Husband  Black
Male
4234   Married-civ-spouse     Prof-specialty           Wife  Black
Female
4235   Married-civ-spouse    Exec-managerial           Wife  White
Female
4237   Married-civ-spouse    Exec-managerial        Husband  White
Male
...                   ...                ...            ...    ...
...
32555  Married-civ-spouse    Exec-managerial        Husband  White
Male
32556       Never-married    Protective-serv  Not-in-family  White
Male
```

```
32557  Married-civ-spouse       Tech-support          Wife  White
Female
32558  Married-civ-spouse  Machine-op-inspct       Husband  White
Male
32559             Widowed       Adm-clerical      Unmarried  White
Female

       HoursPerWeek        Country  income
4232             40  United-States       0
4233             40  United-States       0
4234             40           Cuba       0
4235             40  United-States       0
4237             45  United-States       1
...             ...            ...     ...
32555            40  United-States       1
32556            40  United-States       0
32557            38  United-States       0
32558            40  United-States       1
32559            40  United-States       0

[18991 rows x 13 columns]
```

```
# Normalizing our attributes using min max scaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
income_data[numeric] = scaler.fit_transform(income_data[numeric])
# scaled_data = pd.DataFrame(scaler.fit_transform(X1),
columns=X1.columns)
# scaled_data

income_data
```

```
            age         workclass    fnlwgt     education
EducationNum  \
4232   0.344262           Private  0.499999       HS-grad
0.363636
4233   0.590164           Private  0.547504          11th
0.181818
4234   0.180328           Private  0.805731     Bachelors
0.727273
4235   0.327869           Private  0.671679       Masters
0.818182
4237   0.573770  Self-emp-not-inc  0.485046       HS-grad
0.363636
...         ...               ...       ...           ...           ..
.
32555  0.590164           Private  0.764530       Masters
0.818182
32556  0.081967           Private  0.735359  Some-college
0.454545
32557  0.163934           Private  0.603740    Assoc-acdm
```

```
0.636364
32558  0.377049           Private  0.347405       HS-grad
0.363636
32559  0.672131           Private  0.341269       HS-grad
0.363636

            MaritalStatus         occupation   relationship   race
gender  \
4232             Divorced  Handlers-cleaners  Not-in-family  White
Male
4233   Married-civ-spouse  Handlers-cleaners        Husband  Black
Male
4234   Married-civ-spouse     Prof-specialty           Wife  Black
Female
4235   Married-civ-spouse    Exec-managerial           Wife  White
Female
4237   Married-civ-spouse    Exec-managerial        Husband  White
Male
...                   ...                ...            ...    ...
...
32555  Married-civ-spouse    Exec-managerial        Husband  White
Male
32556       Never-married    Protective-serv  Not-in-family  White
Male
32557  Married-civ-spouse       Tech-support           Wife  White
Female
32558  Married-civ-spouse  Machine-op-inspct        Husband  White
Male
32559             Widowed       Adm-clerical      Unmarried  White
Female

       HoursPerWeek        Country  income
4232       0.368421  United-States     0.0
4233       0.368421  United-States     0.0
4234       0.368421           Cuba     0.0
4235       0.368421  United-States     0.0
4237       0.631579  United-States     1.0
...             ...            ...     ...
32555      0.368421  United-States     1.0
32556      0.368421  United-States     0.0
32557      0.263158  United-States     0.0
32558      0.368421  United-States     1.0
32559      0.368421  United-States     0.0

[18991 rows x 13 columns]
```

```
from sklearn.preprocessing import OneHotEncoder #used for one hot
encoding
```

```
# Applying one hot encoding for our categorical attributes
income_data = pd.get_dummies(income_data)

income_data
```

```
            age    fnlwgt  EducationNum  HoursPerWeek  income  \
4232   0.344262  0.499999      0.363636      0.368421     0.0
4233   0.590164  0.547504      0.181818      0.368421     0.0
4234   0.180328  0.805731      0.727273      0.368421     0.0
4235   0.327869  0.671679      0.818182      0.368421     0.0
4237   0.573770  0.485046      0.363636      0.631579     1.0
...         ...       ...           ...           ...     ...
32555  0.590164  0.764530      0.818182      0.368421     1.0
32556  0.081967  0.735359      0.454545      0.368421     0.0
32557  0.163934  0.603740      0.636364      0.263158     0.0
32558  0.377049  0.347405      0.363636      0.368421     1.0
32559  0.672131  0.341269      0.363636      0.368421     0.0

       workclass_Federal-gov  workclass_Local-gov  workclass_Never-
worked  \
4232                       0                    0
0
4233                       0                    0
0
4234                       0                    0
0
4235                       0                    0
0
4237                       0                    0
0
...                      ...                  ...
...
32555                      0                    0
0
32556                      0                    0
0
32557                      0                    0
0
32558                      0                    0
0
32559                      0                    0
0

       workclass_Private  workclass_Self-emp-inc  ...
Country_Portugal  \
4232                   1                       0  ...
0
4233                   1                       0  ...
0
4234                   1                       0  ...
```

```
0
4235                1                       0  ...
0
4237                0                       0  ...
0
...               ...                     ...  ...                 ..
.
32555               1                       0  ...
0
32556               1                       0  ...
0
32557               1                       0  ...
0
32558               1                       0  ...
0
32559               1                       0  ...
0

       Country_Puerto-Rico  Country_Scotland  Country_South
Country_Taiwan  \
4232                     0                 0              0
0
4233                     0                 0              0
0
4234                     0                 0              0
0
4235                     0                 0              0
0
4237                     0                 0              0
0
...                    ...               ...            ...
...
32555                    0                 0              0
0
32556                    0                 0              0
0
32557                    0                 0              0
0
32558                    0                 0              0
0
32559                    0                 0              0
0

       Country_Thailand  Country_Trinadad&Tobago  Country_United-
States  \
4232                  0                        0
1
4233                  0                        0
1
4234                  0                        0
```

```
0
4235                  0                        0
1
4237                  0                        0
1
...                 ...                      ...                     ..
.
32555                 0                        0
1
32556                 0                        0
1
32557                 0                        0
1
32558                 0                        0
1
32559                 0                        0
1

       Country_Vietnam  Country_Yugoslavia
4232                 0                   0
4233                 0                   0
4234                 0                   0
4235                 0                   0
4237                 0                   0
...                ...                 ...
32555                0                   0
32556                0                   0
32557                0                   0
32558                0                   0
32559                0                   0

[18991 rows x 104 columns]
```

```
# Here we pop our target variable i.e., income and append at the last
position

cols = list(income_data.columns.values)
cols.pop(cols.index('income'))
income_data = income_data[cols+['income']]
income_data.head()
```

```
           age    fnlwgt  EducationNum  HoursPerWeek
workclass_Federal-gov  \
4232  0.344262  0.499999      0.363636      0.368421
0
4233  0.590164  0.547504      0.181818      0.368421
0
4234  0.180328  0.805731      0.727273      0.368421
0
4235  0.327869  0.671679      0.818182      0.368421
```

```
0
4237  0.573770  0.485046      0.363636      0.631579
0

      workclass_Local-gov  workclass_Never-worked
workclass_Private  \
4232                    0                       0                  1

4233                    0                       0                  1

4234                    0                       0                  1

4235                    0                       0                  1

4237                    0                       0                  0


      workclass_Self-emp-inc  workclass_Self-emp-not-inc  ...  \
4232                       0                           0  ...
4233                       0                           0  ...
4234                       0                           0  ...
4235                       0                           0  ...
4237                       0                           1  ...

      Country_Puerto-Rico  Country_Scotland  Country_South
Country_Taiwan  \
4232                    0                 0              0
0
4233                    0                 0              0
0
4234                    0                 0              0
0
4235                    0                 0              0
0
4237                    0                 0              0
0

      Country_Thailand  Country_Trinadad&Tobago  Country_United-States
\
4232                 0                        0                      1

4233                 0                        0                      1

4234                 0                        0                      0

4235                 0                        0                      1

4237                 0                        0                      1
```

```
      Country_Vietnam  Country_Yugoslavia  income
4232                0                   0     0.0
4233                0                   0     0.0
4234                0                   0     0.0
4235                0                   0     0.0
4237                0                   0     1.0

[5 rows x 104 columns]
```

```
x= income_data.iloc[:,:-1]# independent
y= income_data.iloc[:,-1]# target
```

```
x.head(3)
```

```
           age    fnlwgt  EducationNum  HoursPerWeek
workclass_Federal-gov  \
4232  0.344262  0.499999      0.363636      0.368421
0
4233  0.590164  0.547504      0.181818      0.368421
0
4234  0.180328  0.805731      0.727273      0.368421
0

      workclass_Local-gov  workclass_Never-worked
workclass_Private  \
4232                    0                       0                  1

4233                    0                       0                  1

4234                    0                       0                  1


      workclass_Self-emp-inc  workclass_Self-emp-not-inc  ...  \
4232                       0                           0  ...
4233                       0                           0  ...
4234                       0                           0  ...

      Country_Portugal  Country_Puerto-Rico  Country_Scotland
Country_South  \
4232                 0                    0                 0
0
4233                 0                    0                 0
0
4234                 0                    0                 0
0

      Country_Taiwan  Country_Thailand  Country_Trinadad&Tobago  \
4232               0                 0                        0
4233               0                 0                        0
```

```
4234               0                 0                        0

      Country_United-States  Country_Vietnam  Country_Yugoslavia
4232                      1                0                   0
4233                      1                0                   0
4234                      0                0                   0

[3 rows x 103 columns]
```

```
y.head(3)
```

```
4232    0.0
4233    0.0
4234    0.0
Name: income, dtype: float64
```

```
from sklearn.model_selection import train_test_split
```

```
train_input,test_input, train_output,test_output =
train_test_split(x,y, test_size = 0.3, random_state=6)
```

```
print("Before SMOTE")
train_output.value_counts()
```

```
Before SMOTE

0.0    10469
1.0     2824
Name: income, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
SMOTE = SMOTE(sampling_strategy='minority', random_state=10)
```

```
train_input_SMOTE, train_output_SMOTE= SMOTE.fit_resample(train_input,
train_output)
train_input_SMOTE.head(3)
```

```
        age    fnlwgt  EducationNum  HoursPerWeek  workclass_Federal-
gov  \
0  0.311475  0.035160      0.363636      0.368421
0
1  0.114754  0.044935      0.363636      0.368421
0
2  0.196721  0.921992      0.454545      0.157895
0

   workclass_Local-gov  workclass_Never-worked  workclass_Private  \
0                    1                       0                  0
1                    0                       0                  0
2                    0                       0                  1

   workclass_Self-emp-inc  workclass_Self-emp-not-inc  ...
```

```
Country_Portugal  \
0                       0                            0  ...
0
1                       0                            1  ...
0
2                       0                            0  ...
0

   Country_Puerto-Rico  Country_Scotland  Country_South
Country_Taiwan  \
0                    0                 0              0
0
1                    0                 0              0
0
2                    0                 0              0
0

   Country_Thailand  Country_Trinadad&Tobago  Country_United-States  \
0                 0                        0                      1
1                 0                        0                      1
2                 0                        0                      1

   Country_Vietnam  Country_Yugoslavia
0                0                   0
1                0                   0
2                0                   0

[3 rows x 103 columns]
```

```
print("after SMOTE:")
train_output_SMOTE.value_counts()
```

```
after SMOTE:

0.0    10469
1.0    10469
Name: income, dtype: int64
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from sklearn import metrics
```

```
#from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(train_input_SMOTE, train_output_SMOTE)
predict_logistic=model.predict(test_input)
cf_matrix_logistic = confusion_matrix(test_output,predict_logistic)
```

```
logistic_acc = accuracy_score(test_output,predict_logistic)*100
print("accuracy of Logistic Regression :", round(logistic_acc,2),"%")
```

```
accuracy of Logistic Regression : 80.4 %
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(train_input_SMOTE, train_output_SMOTE)
pred_k=knn.predict(test_input)
knn_acc = accuracy_score(test_output,pred_k)*100
print("accuracy of KNN :", round(knn_acc,2),"%")
```

```
accuracy of KNN : 75.32 %
```

```
model=RandomForestClassifier(n_estimators=40)
RFC=model.fit(train_input_SMOTE,train_output_SMOTE)
pred_RFC=RFC.predict(test_input)
RFC_acc = accuracy_score(test_output,pred_RFC)*100
print("accuracy of RFC :", round(RFC_acc,2),"%")
```

```
accuracy of RFC : 81.33 %
```

```
GNB=GaussianNB()
GNB.fit(train_input_SMOTE,train_output_SMOTE)
pred_GNB=GNB.predict(test_input)
GNB_acc = accuracy_score(test_output,pred_GNB)*100
print("accuracy of GNB :", round(GNB_acc,2),"%")
```

```
accuracy of GNB : 48.44 %
```

```
# Comparing Accuracies

labels =["Logistic Regression","KNN", "Naive Bayes", "Random Forest"]
x = [ logistic_acc,knn_acc,GNB_acc,RFC_acc]
eval_frame=pd.DataFrame()
eval_frame['Model']=labels
eval_frame['Train_test_split'] = x
eval_frame
```

	Model	Train_test_split
0	Logistic Regression	80.396630
1	KNN	75.324675
2	Naive Bayes	48.438048
3	Random Forest	81.326781

K-Folds Cross Validation : To reduce the model's bias, the K-Folds method is applied. Because each data record has a chance to appear in both the training and test data sets. The K-Folds approach partitions the dataset into k-folds. At random, I divided the data into five folds. The four folds are then applied to the model, and the fifth fold is used to test it. Repeat until each fold has been utilised as a test set. After that, the average is calculated by averaging all of the findings.

```
from sklearn.model_selection import KFold
kfold = KFold(n_splits=5)

# Modeling step Test differents algorithm
classifiers1 = []

classifiers1.append(KNeighborsClassifier())
classifiers1.append(LogisticRegression())
classifiers1.append(GaussianNB())
classifiers1.append(RandomForestClassifier())

from sklearn.model_selection import cross_val_score

accuracy_results1 = []
for a in classifiers1:
    accuracy_results1.append(cross_val_score(a, train_input_SMOTE,
train_output_SMOTE, scoring= "accuracy", cv=kfold))

accuracy_results1
```

```
[array([0.73997135, 0.75716332, 0.76408787, 0.89586816, 0.91712443]),
 array([0.77889207, 0.78127985, 0.78939828, 0.86696919, 0.86195367]),
 array([0.49212034, 0.49140401, 0.49307545, 0.89491283, 0.99546214]),
 array([0.82808023, 0.83643744, 0.834766  , 0.89921185, 0.91473609])]
```

```
accuracy_means1 = []
for e in accuracy_results1:
    accuracy_means1.append(e.mean()*100)

accuracy_means1
```

```
[81.48430273446098, 81.56986114067078, 67.33949558247443,
86.26463203406915]
```

```
eval_frame['kfold_5']=accuracy_means1
eval_frame
```

	Model	Train_test_split	kfold_5
0	Logistic Regression	80.396630	81.484303
1	KNN	75.324675	81.569861
2	Naive Bayes	48.438048	67.339496
3	Random Forest	81.326781	86.264632

Stratified K Fold: This cross-validation object returns stratified folds and is a version of K-Fold. Folds are made by keeping track of the number of samples in each class. Five stratified folds were created using the data. The four folds are then applied to the model, and the fifth fold is used to test it. Repeat until each fold has been utilised as a test set. After that, the average is calculated by averaging all of the findings.

```
from sklearn.model_selection import StratifiedKFold
Stratifiedkfold = StratifiedKFold(n_splits=5)
# Modeling step Test differents algorithm
classifiers_4 = []
```

```
classifiers_4.append(KNeighborsClassifier())
classifiers_4.append(LogisticRegression())
classifiers_4.append(GaussianNB())
classifiers_4.append(RandomForestClassifier())

accuracy_results_4 = []
for a in classifiers_4:
    accuracy_results_4.append(cross_val_score(a, train_input_SMOTE,
train_output_SMOTE, scoring= "accuracy", cv=Stratifiedkfold))
accuracy_means_4 = []
for e in accuracy_results_4:
    accuracy_means_4.append(e.mean()*100)
accuracy_means_4
eval_frame['Stratifiedkfold_5']=accuracy_means_4

eval_frame
```

	Model	Train_test_split	kfold_5	Stratifiedkfold_5
0	Logistic Regression	80.396630	81.484303	84.330058
1	KNN	75.324675	81.569861	84.373121
2	Naive Bayes	48.438048	67.339496	67.131627
3	Random Forest	81.326781	86.264632	88.198620

Repeated Random Test-Train Splits: In this strategy, k-fold cross-validation is paired with typical train-test splits. I utilise a cross-validation approach to generate random divisions of the data in the training test set, and then I split and test the algorithms multiple times. The data was used to produce five Repeated Random Test-Train Splits.

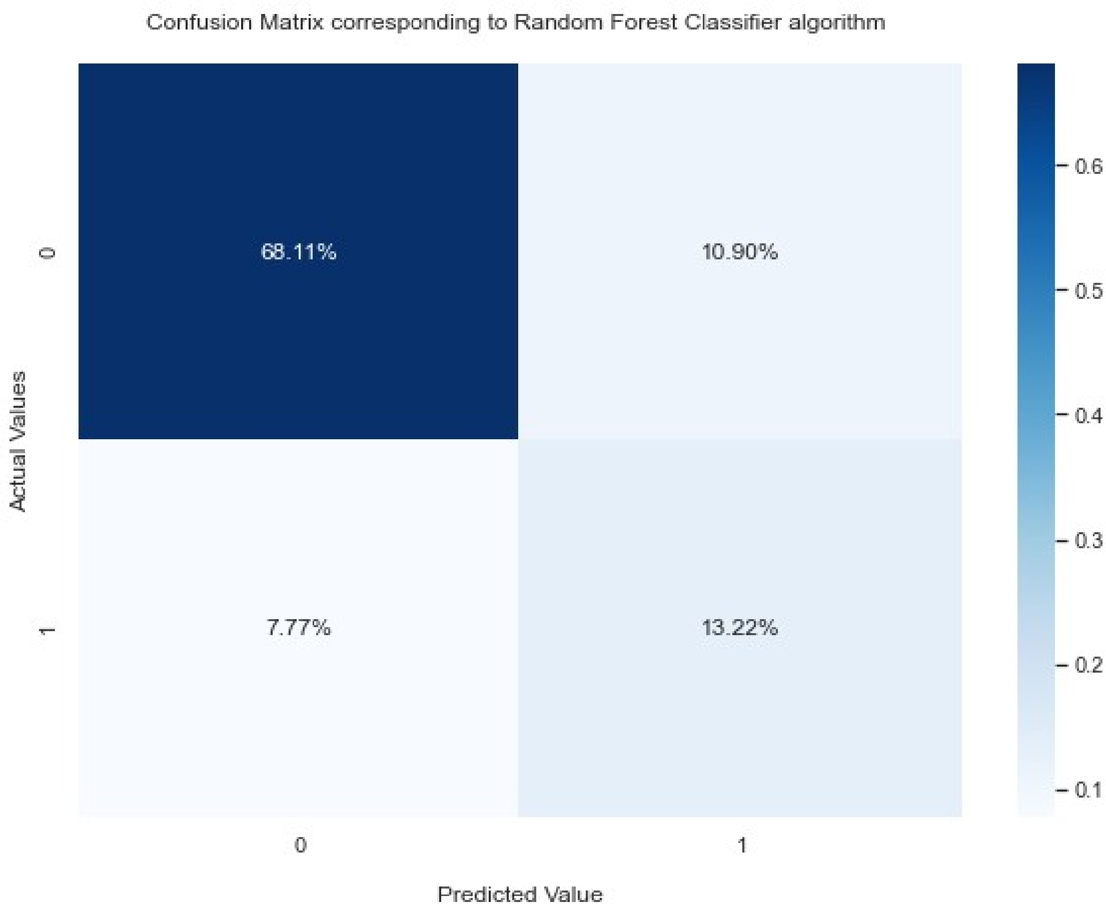
```
from sklearn.model_selection import ShuffleSplit
kfold = ShuffleSplit(n_splits=5,test_size=0.3)
# Modeling step Test differents algorithm
classifiers_2 = []
classifiers_2.append(KNeighborsClassifier())
classifiers_2.append(LogisticRegression())
classifiers_2.append(GaussianNB())
classifiers_2.append(RandomForestClassifier())

accuracy_results_2 = []
for a in classifiers_2:
    accuracy_results_2.append(cross_val_score(a, train_input_SMOTE,
train_output_SMOTE, scoring= "accuracy", cv=kfold))
accuracy_means_2 = []
for e in accuracy_results_2:
    accuracy_means_2.append(e.mean()*100)
accuracy_means_2
eval_frame['RRTestTrainSplits_5']=accuracy_means_2
eval_frame.round(2).to_csv("table.csv")
```

In every case of train-test split cross validation method, Random Forest Model gives the highest accuracy. So, I choose Random Forest model as a best-fit model for my project.

```
# Confusion Matrix corresponding to Random Forest Classifier Model
cf_matrix_RFC = confusion_matrix(test_output, pred_RFC)
cf_matrix_RFC
```

```
array([[3881,  621],
       [ 443,  753]], dtype=int64)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(test_output, pred_RFC)
plt.figure(figsize=(10,7))
ax=sns.heatmap(cm/np.sum(cm),annot=True,fmt='.2%',cmap="Blues")
ax.set_title('Confusion Matrix corresponding to Random Forest
Classifier algorithm \n')
ax.set_xlabel("\nPredicted Value")
ax.set_ylabel("Actual Values")


## Ticket labels - list must be in alphabetical order
ax.xaxis.set_ticklabels(['0','1'])
ax.yaxis.set_ticklabels(['0','1'])

## Display the visualization of the Confusion Matrix
#plt.savefig("cf_matrix_KNN.png",bbox_inches = 'tight')
plt.show()
print("accuracy of Random Forest :", RFC_acc)
```




```
accuracy of Random Forest : 81.32678132678133
```

```
print(classification_report(test_output, pred_RFC))
```

```
              precision    recall  f1-score   support

         0.0       0.90      0.86      0.88      4502
         1.0       0.55      0.63      0.59      1196

    accuracy                           0.81      5698
   macro avg       0.72      0.75      0.73      5698
weighted avg       0.82      0.81      0.82      5698
```

f1 score for <=50k means 0 class # f1 gives the overall performance of our model

```
f1_score = 2*(0.90*0.86)/(0.90+0.86)
round(f1_score,2)
```

```
0.88
```

f1 score for >50k means 1 class

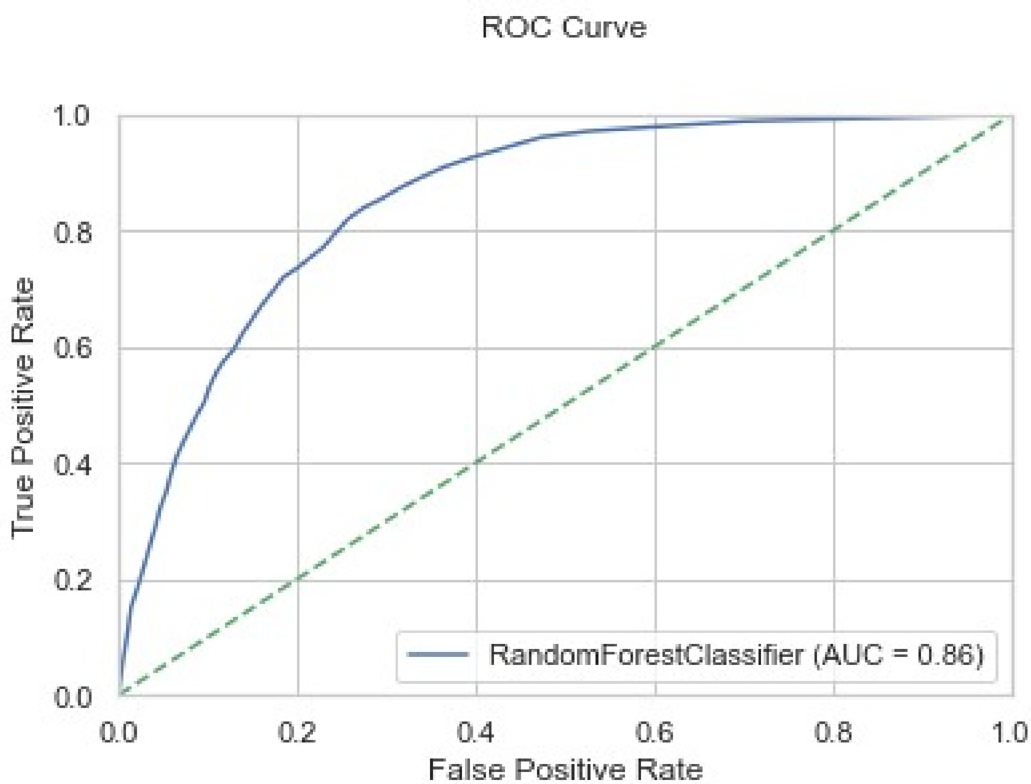
```
f1_score = 2*(0.55*0.63)/(0.55+0.63)
round(f1_score,2)
```

```
0.59
```

```
# normalization comes before train test split
#balancing is after train and split
```

ROC Curve Receiver Operating Characteristic curve, or ROC curve is a tool when predicting the probability of binary output. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis).

```
RFC = RFC.fit(train_input_SMOTE, train_output_SMOTE)
metrics.plot_roc_curve(RFC, test_input,test_output)
plt.plot([0,1],[0,1],'g--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve\n\n')
plt.savefig("ROC Curve.png", bbox_inches='tight')
plt.show()
```




```
#important feature to get the high accuracy in optimal model
feature_imp =
pd.DataFrame(RFC.feature_importances_,index=train_input_SMOTE.columns,
columns=['feature importance']).sort_values('feature
```

```
importance',ascending=False)
```

```
feature_imp
```

```
                                    feature importance
age                                           0.194611
fnlwgt                                        0.149432
MaritalStatus_Married-civ-spouse              0.084177
HoursPerWeek                                  0.065262
EducationNum                                  0.064032
...                                                ...
education_1st-4th                             0.000000
education_7th-8th                             0.000000
Country_Holand-Netherlands                    0.000000
education_Preschool                           0.000000
workclass_Without-pay                         0.000000

[103 rows x 1 columns]
```

```
feature_imp.sort_values(by='feature importance', ascending=False)
```

```
                                    feature importance
age                                           0.194611
fnlwgt                                        0.149432
MaritalStatus_Married-civ-spouse              0.084177
HoursPerWeek                                  0.065262
EducationNum                                  0.064032
...                                                ...
education_1st-4th                             0.000000
education_7th-8th                             0.000000
Country_Holand-Netherlands                    0.000000
education_Preschool                           0.000000
workclass_Without-pay                         0.000000

[103 rows x 1 columns]
```

Most important features are age, fnlwgt, MaritalStatus_Married-civ-spouse, HoursPerWeek, EducationNum.