27. EDA - Income Data

October 31, 2022

1 Exploratory Data Analysis on Income Data

Data-set `income_evaluation.csv` was extracted from the 1994 Census bureau database of USA.

Columns/Features:

1. age: continuous
2. workclass: categorical
3. fnlwgt: continuous
4. education: categorical
5. education-num: continuous
6. marital-status: categorical
7. occupation: categorical
8. relationship: categorical
9. race: categorical
10. sex: categorical
11. capital-gain: continuous
12. capital-loss: continuous
13. hours-per-week: continuous
14. native-country: categorical
15. income: target

2 Import required packages

[1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

3 Load dataset

[2]:
```
df = pd.read_csv('../data/income_evaluation.csv')
```

4 Exploratory Data Analysis

Explore the data to gain insights about the data.

4.1 View dimensions of dataset

[3]:
```
df.shape
```

[3]:
```
(32561, 15)
```

We can see that there are 32561 instances and 15 attributes in the data set.

4.2 Preview the dataset

[4]:
```
df.head()
```

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

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

   capital-gain   capital-loss   hours-per-week  native-country  income
0          2174              0               40   United-States   <=50K
1             0              0               13   United-States   <=50K
2             0              0               40   United-States   <=50K
3             0              0               40   United-States   <=50K
4             0              0               40            Cuba   <=50K
```

4.3 Rename column names

We can see that the dataset does not have proper column names. The column names contain underscore. We should give proper names to the columns. I will do it as follows:-

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

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

[6]:
```python
df.columns = [i.replace('-','_').strip() for i in df.columns]
```

[7]:
```python
df.columns
```

[7]:
```
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')
```

4.4 View summary of dataset

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education_num   32561 non-null  int64
 5   marital_status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital_gain    32561 non-null  int64
 11  capital_loss    32561 non-null  int64
 12  hours_per_week  32561 non-null  int64
 13  native_country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Findings

- We can see that the dataset contains 9 character variables and 6 numerical variables.
- There are no missing values in the dataset.

4.5 Check the data types of columns

- The above `df.info()` command gives us the number of filled values along with the data types of columns.
- If we simply want to check the data type of a particular column, we can use the following command.

[9]:
```
df.dtypes
```

[9]:
```
age                int64
workclass         object
fnlwgt             int64
education         object
education_num      int64
marital_status    object
occupation        object
relationship      object
race              object
sex               object
capital_gain       int64
capital_loss       int64
hours_per_week     int64
native_country    object
income            object
dtype: object
```

4.6 View statistical properties of dataset

[10]:
```
df.describe()
```

[10]:
```
                age        fnlwgt  education_num  capital_gain  capital_loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000  32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844     87.303830
std       13.640433  1.055500e+05       2.572720   7385.292085    402.960219
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.178270e+05       9.000000      0.000000      0.000000
50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       hours_per_week
count    32561.000000
mean        40.437456
std         12.347429
min          1.000000
25%         40.000000
50%         40.000000
```

```
75%          45.000000
max          99.000000
```

- The above `df.describe()` command presents statistical properties in vertical form.
- If we want to view the statistical properties in horizontal form, we should run the following command.

[11]:
```
df.describe().T
```

[11]:
```
                  count           mean            std      min       25%  \
age             32561.0      38.581647      13.640433     17.0      28.0
fnlwgt          32561.0  189778.366512  105549.977697  12285.0  117827.0
education_num   32561.0      10.080679       2.572720      1.0       9.0
capital_gain    32561.0    1077.648844    7385.292085      0.0       0.0
capital_loss    32561.0      87.303830     402.960219      0.0       0.0
hours_per_week  32561.0      40.437456      12.347429      1.0      40.0

                     50%       75%        max
age                 37.0      48.0       90.0
fnlwgt          178356.0  237051.0  1484705.0
education_num       10.0      12.0       16.0
capital_gain         0.0       0.0    99999.0
capital_loss         0.0       0.0     4356.0
hours_per_week      40.0      45.0       99.0
```

[12]:
```
df.describe(include='all')
```

[12]:
```
                 age workclass        fnlwgt education  education_num  \
count   32561.000000     32561  3.256100e+04     32561   32561.000000
unique           NaN         9           NaN        16            NaN
top              NaN   Private           NaN   HS-grad            NaN
freq             NaN     22696           NaN     10501            NaN
mean       38.581647       NaN  1.897784e+05       NaN      10.080679
std        13.640433       NaN  1.055500e+05       NaN       2.572720
min        17.000000       NaN  1.228500e+04       NaN       1.000000
25%        28.000000       NaN  1.178270e+05       NaN       9.000000
50%        37.000000       NaN  1.783560e+05       NaN      10.000000
75%        48.000000       NaN  2.370510e+05       NaN      12.000000
max        90.000000       NaN  1.484705e+06       NaN      16.000000

              marital_status      occupation relationship   race    sex  \
count                  32561           32561        32561  32561  32561
unique                     7              15            6      5      2
top       Married-civ-spouse  Prof-specialty      Husband  White   Male
freq                   14976            4140        13193  27816  21790
mean                     NaN             NaN          NaN    NaN    NaN
std                      NaN             NaN          NaN    NaN    NaN
```

```
min                       NaN              NaN           NaN      NaN    NaN
25%                       NaN              NaN           NaN      NaN    NaN
50%                       NaN              NaN           NaN      NaN    NaN
75%                       NaN              NaN           NaN      NaN    NaN
max                       NaN              NaN           NaN      NaN    NaN

        capital_gain  capital_loss  hours_per_week  native_country  income
count   32561.000000  32561.000000    32561.000000           32561   32561
unique           NaN           NaN             NaN              42       2
top              NaN           NaN             NaN   United-States   <=50K
freq             NaN           NaN             NaN           29170   24720
mean     1077.648844     87.303830       40.437456             NaN     NaN
std      7385.292085    402.960219       12.347429             NaN     NaN
min         0.000000      0.000000        1.000000             NaN     NaN
25%         0.000000      0.000000       40.000000             NaN     NaN
50%         0.000000      0.000000       40.000000             NaN     NaN
75%         0.000000      0.000000       45.000000             NaN     NaN
max     99999.000000   4356.000000       99.000000             NaN     NaN
```

4.7 Check for missing values

- In Python missing data is represented by two values:
  - **None** : None is a Python singleton object that is often used for missing data in Python code.
  - **NaN** : NaN is an acronym for Not a Number. It is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.
- There are different methods in place on how to detect missing values.

Pandas `isnull()` and `notnull()` functions

- Pandas offers two functions to test for missing values - **isnull()** and **notnull()**.
- These are simple functions that return a boolean value indicating whether the passed in argument value is in fact missing data.

Below, I will list some useful commands to deal with missing values.

Useful commands to detect missing values

- **df.isnull()**

The above command checks whether each cell in a dataframe contains missing values or not. If the cell contains missing value, it returns True otherwise it returns False.

- **df.isnull().sum()**

The above command returns total number of missing values in each column in the dataframe.

- **df.isnull().sum().sum()**

It returns total number of missing values in the dataframe.

- **df.isnull().mean()**

It returns percentage of missing values in each column in the dataframe.

- **df.isnull().any()**

It checks which column has null values and which has not. The columns which has null values returns TRUE and FALSE otherwise.

- **df.isnull().any().any()**

It returns a boolean value indicating whether the dataframe has missing values or not. If dataframe contains missing values it returns TRUE and FALSE otherwise.

- **df.isnull().values.any()**

It checks whether a particular column has missing values or not. If the column contains missing values, then it returns TRUE otherwise FALSE.

- **df.isnull().values.sum()**

It returns the total number of missing values in the dataframe.

[13]:
```
# check for missing values
df.isnull().sum()
```

[13]:
```
age               0
workclass         0
fnlwgt            0
education         0
education_num     0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
income            0
dtype: int64
```

Interpretation

We can see that there are no missing values in the dataset.

4.7.1 Check with assert statement

- We must confirm that our dataset has no missing values.
- We can write an **Assert statement** to verify this.

- We can use an assert statement to programmatically check that no missing, unexpected 0 or negative values are present.
- This gives us confidence that our code is running properly.
- **Assert statement** will return nothing if the value being tested is true and will throw an AssertionError if the value is false.
- Asserts
  - assert 1 == 1 (return Nothing if the value is True)
  - assert 1 == 2 (return AssertionError if the value is False)

[14]:
```
#assert that there are no missing values in the dataframe

assert pd.notnull(df).all().all()
```

Interpretation

- The above command does not throw any error. Hence, it is confirmed that there are no missing or negative values in the dataset.
- All the values are greater than or equal to zero excluding character values.

4.8 Explore Categorical Variables

[15]:
```
categorical = ['workclass', 'education', 'marital_status',
               'occupation', 'relationship', 'race', 'sex',
               'native_country']
```

[16]:
```
df[categorical].head()
```

[16]:
```
          workclass   education       marital_status          occupation  \
0         State-gov   Bachelors        Never-married        Adm-clerical
1  Self-emp-not-inc   Bachelors   Married-civ-spouse     Exec-managerial
2           Private     HS-grad             Divorced   Handlers-cleaners
3           Private        11th   Married-civ-spouse   Handlers-cleaners
4           Private   Bachelors   Married-civ-spouse      Prof-specialty

    relationship   race      sex  native_country
0  Not-in-family  White     Male   United-States
1        Husband  White     Male   United-States
2  Not-in-family  White     Male   United-States
3        Husband  Black     Male   United-States
4           Wife  Black   Female            Cuba
```

4.8.1 Frequency distribution of categorical variables

Now, we will check the frequency distribution of categorical variables.

[17]:
```python
for var in categorical:
    print(df[var].value_counts(),'\n')
```

```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64

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

 Married-civ-spouse       14976
 Never-married            10683
 Divorced                  4443
 Separated                 1025
 Widowed                    993
 Married-spouse-absent      418
 Married-AF-spouse           23
Name: marital_status, dtype: int64

 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
```

```
 Machine-op-inspct    2002
 ?                    1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64

 Husband           13193
 Not-in-family      8305
 Own-child          5068
 Unmarried          3446
 Wife               1568
 Other-relative      981
Name: relationship, dtype: int64

 White                 27816
 Black                  3124
 Asian-Pac-Islander     1039
 Amer-Indian-Eskimo      311
 Other                   271
Name: race, dtype: int64

 Male      21790
 Female    10771
Name: sex, dtype: int64

 United-States                 29170
 Mexico                          643
 ?                               583
 Philippines                     198
 Germany                         137
 Canada                          121
 Puerto-Rico                     114
 El-Salvador                     106
 India                           100
 Cuba                             95
 England                          90
 Jamaica                          81
 South                            80
 China                            75
 Italy                            73
 Dominican-Republic               70
 Vietnam                          67
 Guatemala                        64
```

```
Japan                          62
Poland                         60
Columbia                       59
Taiwan                         51
Haiti                          44
Iran                           43
Portugal                       37
Nicaragua                      34
Peru                           31
France                         29
Greece                         29
Ecuador                        28
Ireland                        24
Hong                           20
Cambodia                       19
Trinadad&Tobago                19
Laos                           18
Thailand                       18
Yugoslavia                     16
Outlying-US(Guam-USVI-etc)     14
Honduras                       13
Hungary                        13
Scotland                       12
Holand-Netherlands              1
Name: native_country, dtype: int64
```

4.8.2 Percentage of frequency distribution of values

[18]:
```python
for var in categorical:
     print(df[var].value_counts(normalize=True),'\n')
```

```
Private             0.697030
Self-emp-not-inc    0.078038
Local-gov           0.064279
?                   0.056386
State-gov           0.039864
Self-emp-inc        0.034274
Federal-gov         0.029483
Without-pay         0.000430
Never-worked        0.000215
Name: workclass, dtype: float64

HS-grad         0.322502
Some-college    0.223918
Bachelors       0.164461
Masters         0.052916
Assoc-voc       0.042443
```

```
 11th            0.036086
 Assoc-acdm      0.032769
 10th            0.028654
 7th-8th         0.019840
 Prof-school     0.017690
 9th             0.015786
 12th            0.013298
 Doctorate       0.012684
 5th-6th         0.010227
 1st-4th         0.005160
 Preschool       0.001566
Name: education, dtype: float64

 Married-civ-spouse       0.459937
 Never-married            0.328092
 Divorced                 0.136452
 Separated                0.031479
 Widowed                  0.030497
 Married-spouse-absent    0.012837
 Married-AF-spouse        0.000706
Name: marital_status, dtype: float64

 Prof-specialty       0.127146
 Craft-repair         0.125887
 Exec-managerial      0.124873
 Adm-clerical         0.115783
 Sales                0.112097
 Other-service        0.101195
 Machine-op-inspct    0.061485
 ?                    0.056601
 Transport-moving     0.049046
 Handlers-cleaners    0.042075
 Farming-fishing      0.030527
 Tech-support         0.028500
 Protective-serv      0.019932
 Priv-house-serv      0.004576
 Armed-Forces         0.000276
Name: occupation, dtype: float64

 Husband           0.405178
 Not-in-family     0.255060
 Own-child         0.155646
 Unmarried         0.105832
 Wife              0.048156
 Other-relative    0.030128
Name: relationship, dtype: float64

 White                 0.854274
```

```
 Black                 0.095943
 Asian-Pac-Islander    0.031909
 Amer-Indian-Eskimo    0.009551
 Other                 0.008323
Name: race, dtype: float64

 Male      0.669205
 Female    0.330795
Name: sex, dtype: float64

 United-States                 0.895857
 Mexico                        0.019748
 ?                             0.017905
 Philippines                   0.006081
 Germany                       0.004207
 Canada                        0.003716
 Puerto-Rico                   0.003501
 El-Salvador                   0.003255
 India                         0.003071
 Cuba                          0.002918
 England                       0.002764
 Jamaica                       0.002488
 South                         0.002457
 China                         0.002303
 Italy                         0.002242
 Dominican-Republic            0.002150
 Vietnam                       0.002058
 Guatemala                     0.001966
 Japan                         0.001904
 Poland                        0.001843
 Columbia                      0.001812
 Taiwan                        0.001566
 Haiti                         0.001351
 Iran                          0.001321
 Portugal                      0.001136
 Nicaragua                     0.001044
 Peru                          0.000952
 France                        0.000891
 Greece                        0.000891
 Ecuador                       0.000860
 Ireland                       0.000737
 Hong                          0.000614
 Cambodia                      0.000584
 Trinadad&Tobago               0.000584
 Laos                          0.000553
 Thailand                      0.000553
 Yugoslavia                    0.000491
 Outlying-US(Guam-USVI-etc)    0.000430
```

```
 Honduras                      0.000399
 Hungary                       0.000399
 Scotland                      0.000369
 Holand-Netherlands            0.000031
Name: native_country, dtype: float64
```

Findings

- Now, we can see that there are several variables like `workclass`, `occupation` and `native_country` which contain missing values.
- Generally, the missing values are coded as `NaN` and python will detect them with the usual command of `df.isnull().sum()`.
- But, in this case the missing values are coded as `?`. Pandas fails to detect these as missing values because it does not consider `?` as missing values.
- So, we have to replace `?` with `NaN` so that Python can detect these missing values.
- We will explore these variables and replace `?` with `NaN`.

4.8.3 Explore target variable

[19]:
```python
# check for missing values
df['income'].isnull().sum()
```

[19]:
```
0
```

We can see that there are no missing values in the `income` target variable.

[20]:
```python
# view number of unique values
df['income'].nunique()
```

[20]:
```
2
```

There are 2 unique values in the `income` variable.

[21]:
```python
# view the unique values
df['income'].unique()
```

[21]:
```
array([' <=50K', ' >50K'], dtype=object)
```

The two unique values are `<=50K` and `>50K`.

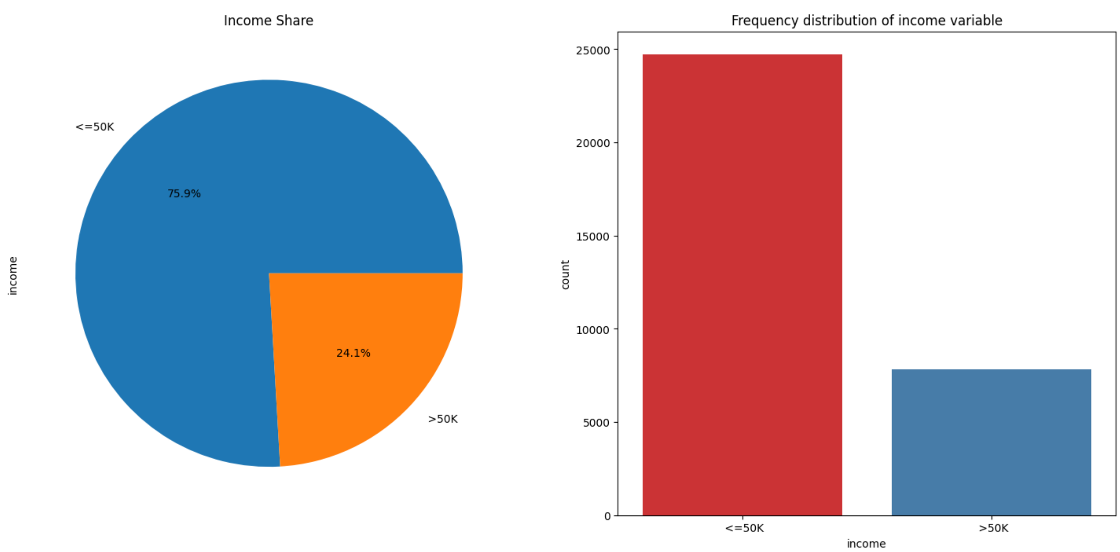
[22]:
```python
# view the frequency distribution of values
df['income'].value_counts()
```

[22]:
```
 <=50K    24720
 >50K      7841
Name: income, dtype: int64
```

[23]:
```
# view percentage of frequency distribution of values
df['income'].value_counts(normalize=True)
```

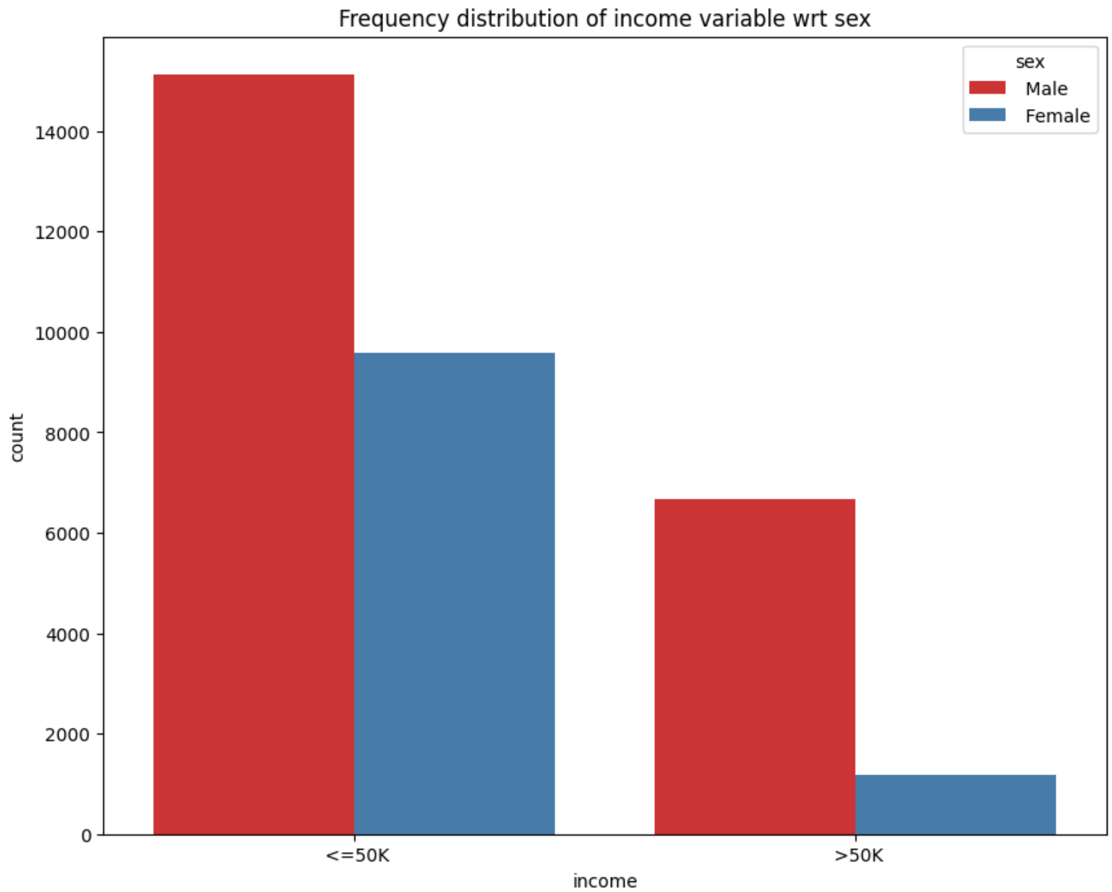
[23]:
```
 <=50K    0.75919
 >50K     0.24081
Name: income, dtype: float64
```

[24]:
```
# visualize frequency distribution of income variable

f,ax=plt.subplots(1,2,figsize=(18,8))

ax[0] = df['income'].value_counts().plot.pie(explode=[0,0],autopct='%1.
 ↪1f%%',ax=ax[0])
ax[0].set_title('Income Share')


#f, ax = plt.subplots(figsize=(6, 8))
ax[1] = sns.countplot(x="income", data=df, palette="Set1")
ax[1].set_title("Frequency distribution of income variable")

plt.show()
```



4.8.4 Visualize income wrt sex variable

[25]:
```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.countplot(x="income", hue="sex", data=df, palette="Set1")
ax.set_title("Frequency distribution of income variable wrt sex")
plt.show()
```

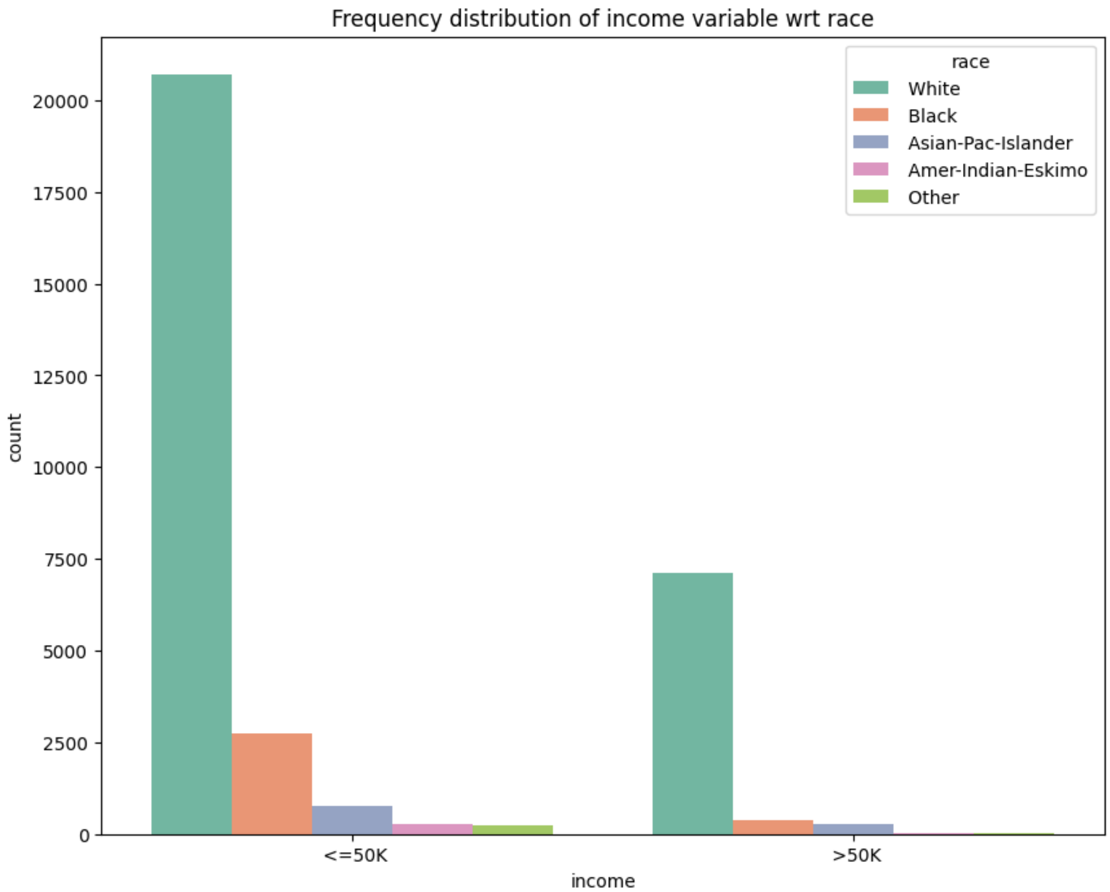


Interpretation

- We can see that males make more money than females in both the income categories.

4.8.5 Visualize income wrt race

[26]:
```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.countplot(x="income", hue="race", data=df, palette="Set2")
ax.set_title("Frequency distribution of income variable wrt race")
plt.show()
```



Interpretation

- We can see that whites make more money than non-whites in both the income categories.

4.8.6 Explore `workclass` variable

[27]:
```python
# check number of unique labels
df.workclass.nunique()
```

[27]: 9

[28]:
```python
# view the unique labels
df.workclass.unique()
```

[28]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

[29]:
```python
# view frequency distribution of values
df.workclass.value_counts()
```

[29]:
```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

We can see that there are 1836 values encoded as `?` in workclass variable. I will replace these `?` with `NaN`.

[30]:
```
# replace '?' values in workclass variable with `NaN`
df['workclass'].replace(' ?', np.NaN, inplace=True)
```

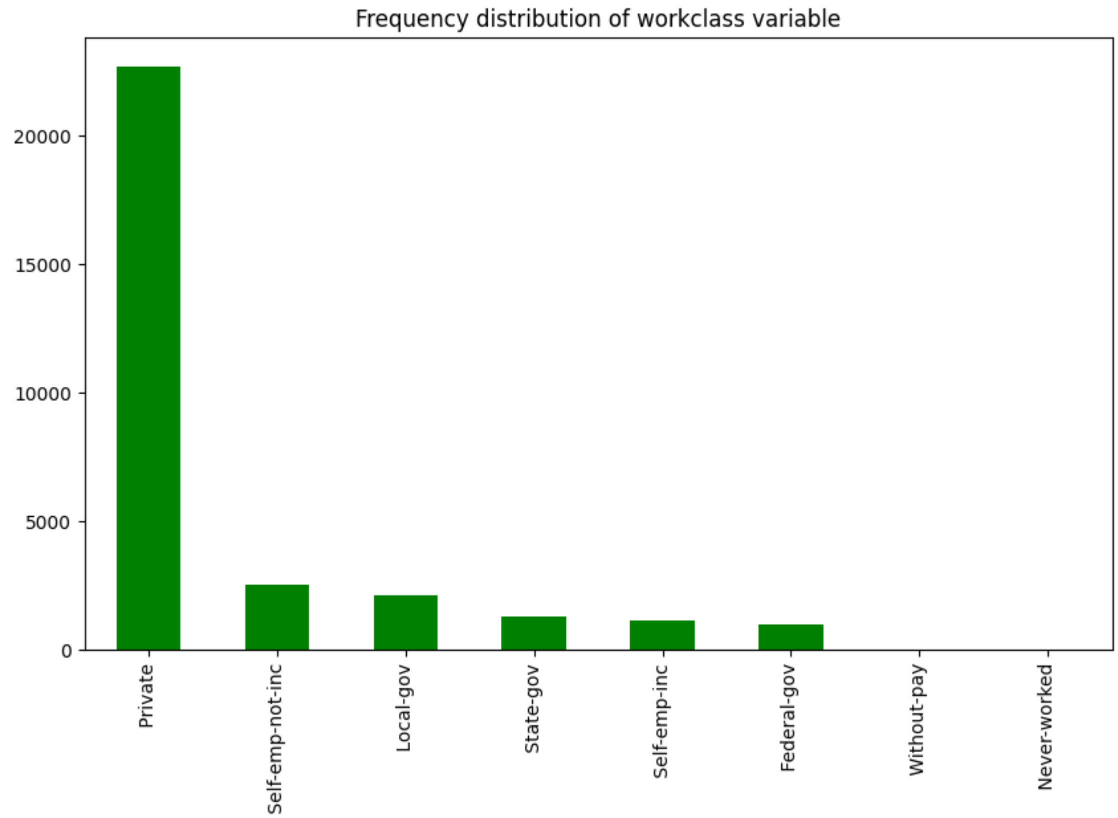
[31]:
```
# again check the frequency distribution of values in workclass variable
df.workclass.value_counts()
```

[31]:
```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

- Now, we can see that there are no values encoded as `?` in the workclass variable.
- We will adopt similar approach with `occupation` and `native_country` column.

4.8.7 Visualize `workclass` variable

[32]:
```
f, ax = plt.subplots(figsize=(10, 6))
ax = df.workclass.value_counts().plot(kind="bar", color="green")
ax.set_title("Frequency distribution of workclass variable")
ax.set_xticklabels(df.workclass.value_counts().index, rotation=90)
plt.show()
```

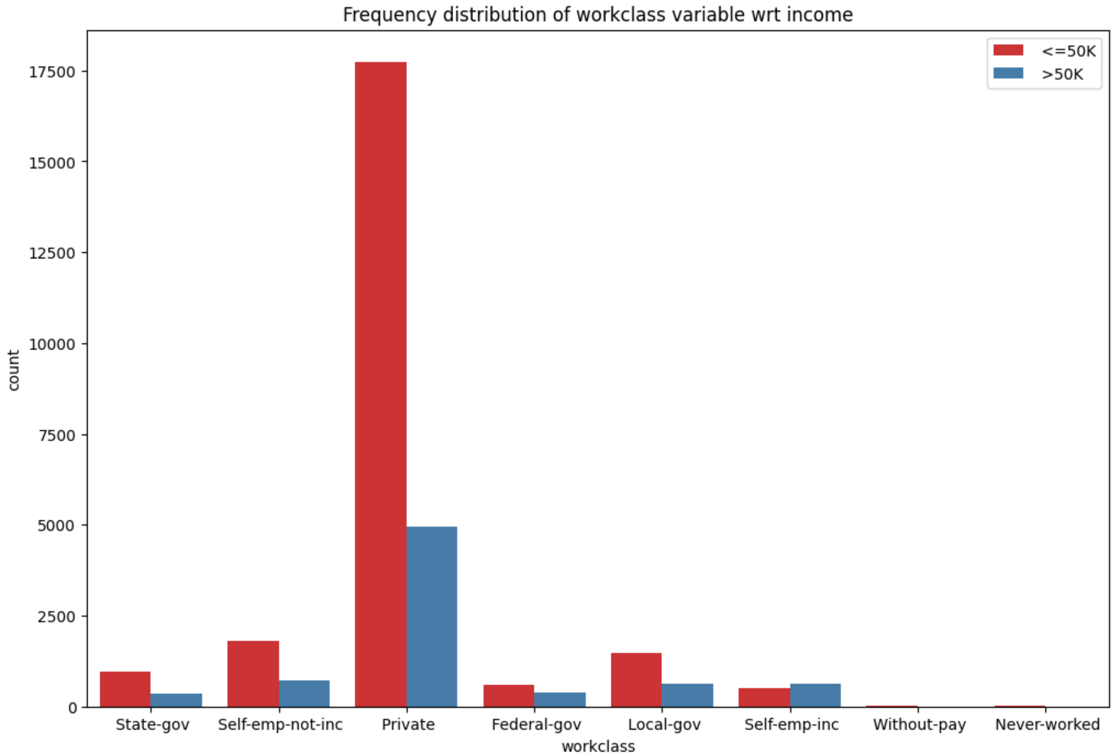


Interpretation

- We can see that there are lot more private workers than other category of workers.

4.8.8 Visualize `workclass` variable wrt `income` variable

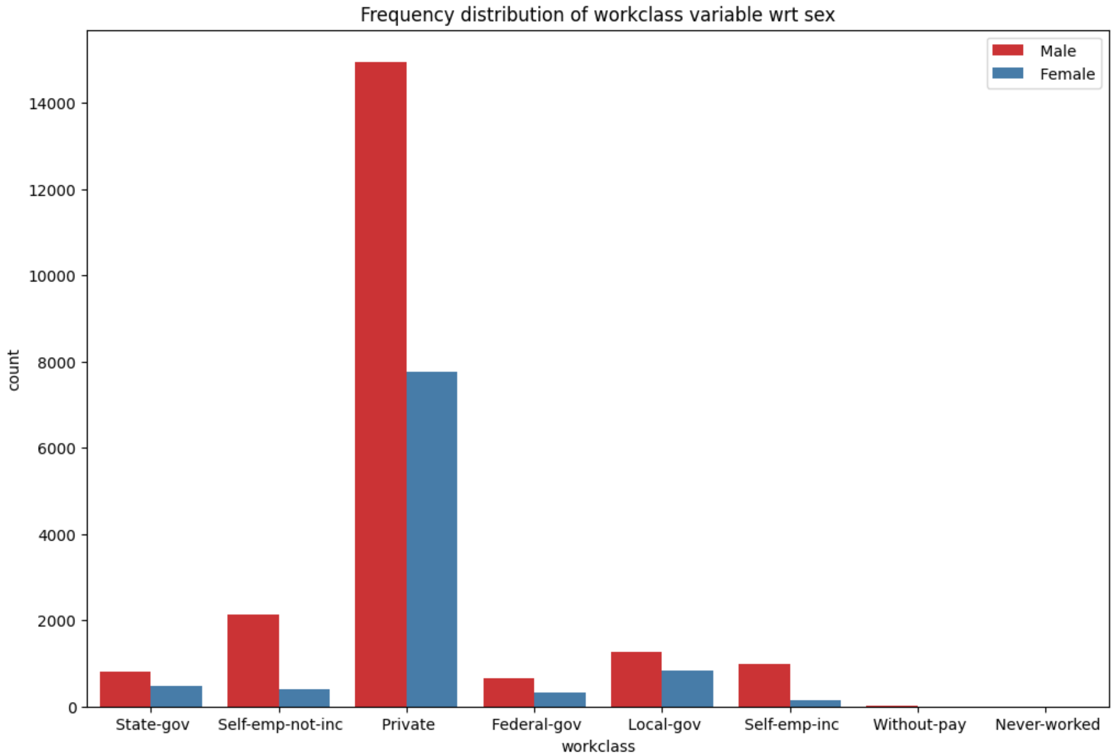
[33]:
```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.countplot(x="workclass", hue="income", data=df, palette="Set1")
ax.set_title("Frequency distribution of workclass variable wrt income")
ax.legend(loc='upper right')
plt.show()
```



Interpretation - We can see that workers make less than equal to 50k in most of the working categories. - But this trend is more appealing in Private `workclass` category.

4.8.9 Visualize `workclass` variable wrt `sex` variable

[34]:
```
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.countplot(x="workclass", hue="sex", data=df, palette="Set1")
ax.set_title("Frequency distribution of workclass variable wrt sex")
ax.legend(loc='upper right')
plt.show()
```



Interpretation - We can see that there are more male workers than female workers in all the working category. - The trend is more appealing in Private sector.

4.8.10 Explore occupation variable

[35]:
```python
# check number of unique labels
df.occupation.nunique()
```

[35]: 15

[36]:
```python
# view unique labels
df.occupation.unique()
```

[36]:
```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Craft-repair',
       ' Transport-moving', ' Farming-fishing', ' Machine-op-inspct',
       ' Tech-support', ' ?', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

[37]:
```python
# view frequency distribution of values
df.occupation.value_counts()
```

[37]:

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

We can see that there are 1843 values encoded as `?` in occupation variable. I will replace these `?` with `NaN`.

[38]:

```
# replace '?' values in occupation variable with `NaN`
df['occupation'].replace(' ?', np.NaN, inplace=True)
```

[39]:

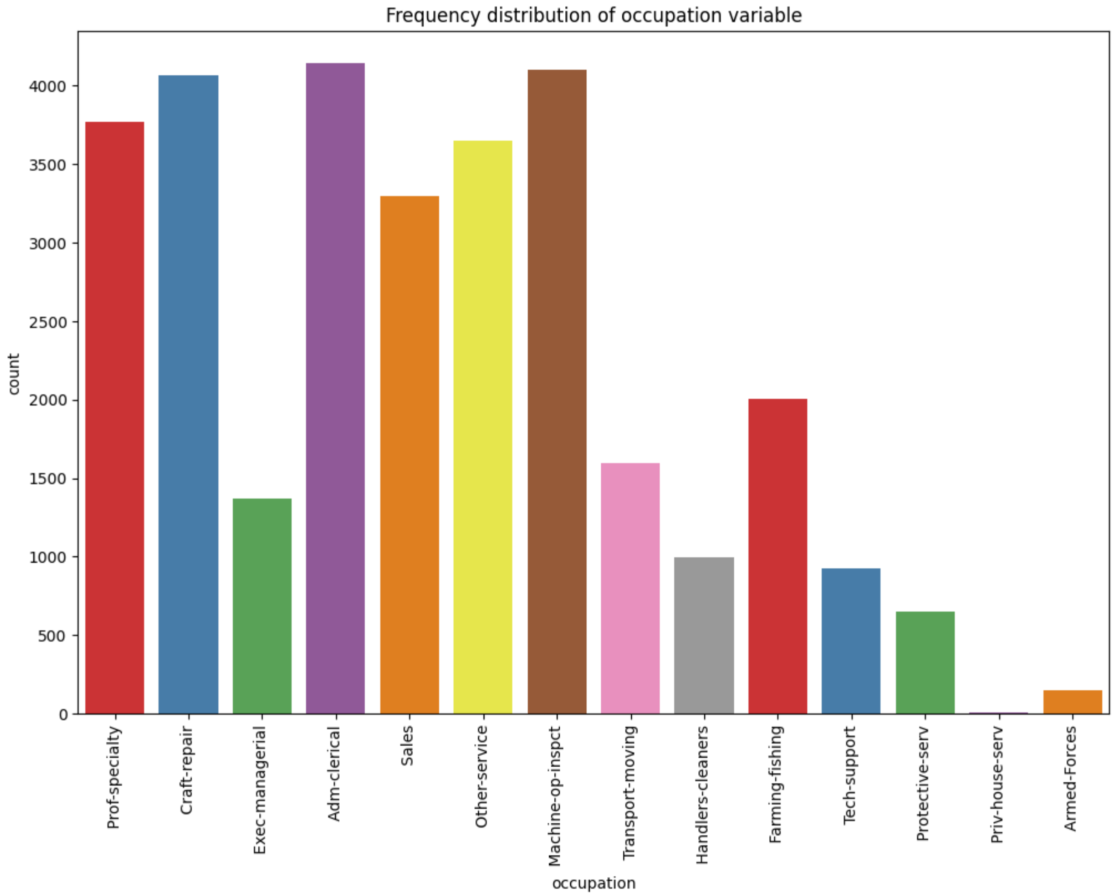
```
# again check the frequency distribution of values
df.occupation.value_counts()
```

[39]:

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

[40]:

```
# visualize frequency distribution of `occupation` variable
f, ax = plt.subplots(figsize=(12, 8))
ax = sns.countplot(x="occupation", data=df, palette="Set1")
ax.set_title("Frequency distribution of occupation variable")
ax.set_xticklabels(df.occupation.value_counts().index, rotation=90)
plt.show()
```



4.8.11 Explore native_country variable

[41]:
```
# check number of unique labels
df.native_country.nunique()
```

[41]: 42

[42]:
```
# view unique labels
df.native_country.unique()
```

[42]:
```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' ?', ' Mexico',
       ' South', ' Puerto-Rico', ' Honduras', ' England', ' Canada',
       ' Germany', ' Iran', ' Philippines', ' Italy', ' Poland',
       ' Columbia', ' Cambodia', ' Thailand', ' Ecuador', ' Laos',
       ' Taiwan', ' Haiti', ' Portugal', ' Dominican-Republic',
       ' El-Salvador', ' France', ' Guatemala', ' China', ' Japan',
       ' Yugoslavia', ' Peru', ' Outlying-US(Guam-USVI-etc)', ' Scotland',
       ' Trinadad&Tobago', ' Greece', ' Nicaragua', ' Vietnam', ' Hong',
```

```
       ' Ireland', ' Hungary', ' Holand-Netherlands'], dtype=object)
```

[43]:
```python
# check frequency distribution of values
df.native_country.value_counts()
```

[43]:
```
 United-States                  29170
 Mexico                           643
 ?                                583
 Philippines                      198
 Germany                          137
 Canada                           121
 Puerto-Rico                      114
 El-Salvador                      106
 India                            100
 Cuba                              95
 England                           90
 Jamaica                           81
 South                             80
 China                             75
 Italy                             73
 Dominican-Republic                70
 Vietnam                           67
 Guatemala                         64
 Japan                             62
 Poland                            60
 Columbia                          59
 Taiwan                            51
 Haiti                             44
 Iran                              43
 Portugal                          37
 Nicaragua                         34
 Peru                              31
 France                            29
 Greece                            29
 Ecuador                           28
 Ireland                           24
 Hong                              20
 Cambodia                          19
 Trinadad&Tobago                   19
 Laos                              18
 Thailand                          18
 Yugoslavia                        16
 Outlying-US(Guam-USVI-etc)        14
 Honduras                          13
 Hungary                           13
 Scotland                          12
 Holand-Netherlands                 1
```

```
Name: native_country, dtype: int64
```

We can see that there are 583 values encoded as `?` in native_country variable. I will replace these `?` with `NaN`.

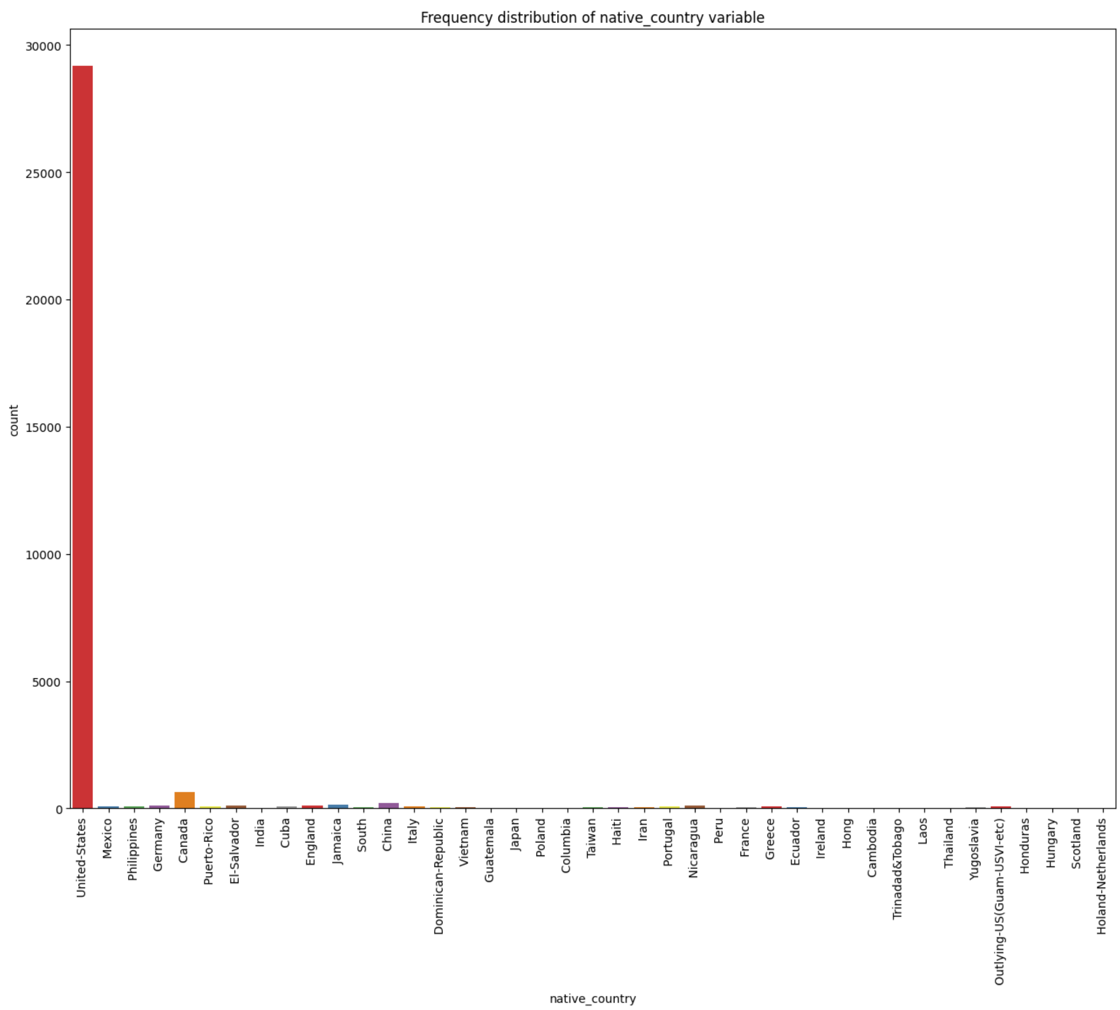
[44]:
```python
# replace '?' values in native_country variable with `NaN`
df['native_country'].replace(' ?', np.NaN, inplace=True)
```

[45]:
```python
# again check the frequency distribution of values
df.native_country.value_counts()
```

[45]:
```
United-States                 29170
Mexico                          643
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        64
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
France                           29
Greece                           29
Ecuador                          28
Ireland                          24
Hong                             20
Cambodia                         19
Trinadad&Tobago                  19
Laos                             18
Thailand                         18
Yugoslavia                       16
```

```
 Outlying-US(Guam-USVI-etc)       14
 Honduras                         13
 Hungary                          13
 Scotland                         12
 Holand-Netherlands                1
Name: native_country, dtype: int64
```

[46]:
```python
# visualize frequency distribution of `native_country` variable
f, ax = plt.subplots(figsize=(16, 12))
ax = sns.countplot(x="native_country", data=df, palette="Set1")
ax.set_title("Frequency distribution of native_country variable")
ax.set_xticklabels(df.native_country.value_counts().index, rotation=90)
plt.show()
```



We can see that `United-States` dominate amongst the `native_country` variables.

4.8.12 Check missing values in categorical variables

[47]:
```
df[categorical].isnull().sum()
```

[47]:
```
workclass         1836
education            0
marital_status       0
occupation        1843
relationship         0
race                 0
sex                  0
native_country     583
dtype: int64
```

Now, we can see that `workclass`, `occupation` and `native_country` variable contains missing values.

4.8.13 Number of labels: Cardinality

- The number of labels within a categorical variable is known as **cardinality**.
- A high number of labels within a variable is known as **high cardinality**.
- High cardinality may pose some serious problems in the machine learning model. So, we will check for high cardinality.

[48]:
```
# check for cardinality in categorical variables
for var in categorical:
    print(var, ' contains ', len(df[var].unique()), ' labels')
```

```
workclass  contains  9  labels
education  contains  16  labels
marital_status  contains  7  labels
occupation  contains  15  labels
relationship  contains  6  labels
race  contains  5  labels
sex  contains  2  labels
native_country  contains  42  labels
```

We can see that `native_country` column contains relatively large number of labels as compared to other columns.

4.9 Explore Numerical Variables

[49]:
```
numerical = ['age', 'fnlwgt', 'education_num', 'capital_gain',
             'capital_loss', 'hours_per_week']
```

4.9.1 Preview the numerical variables

[50]:
```
df[numerical].head()
```

[50]:

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40

4.9.2 Check missing values in numerical variables

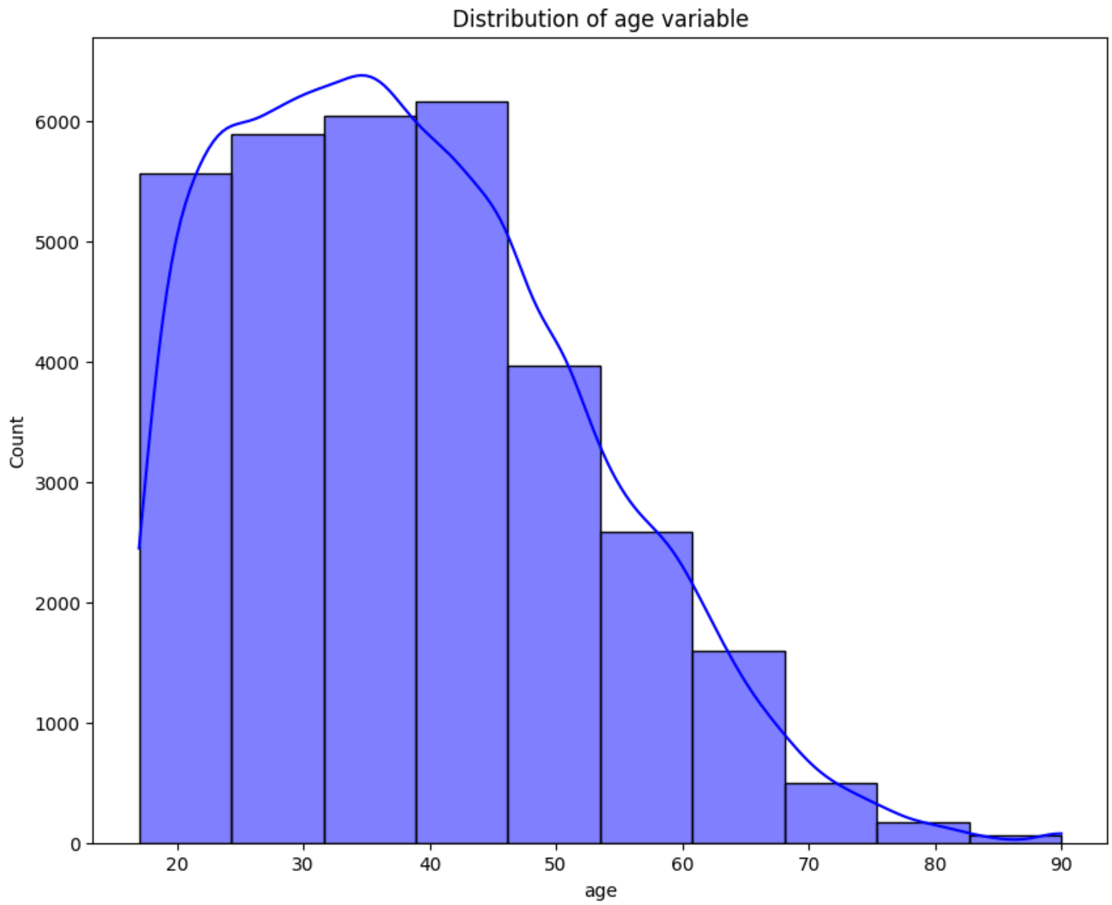
[51]:
```
df[numerical].isnull().sum()
```

[51]:
```
age               0
fnlwgt            0
education_num     0
capital_gain      0
capital_loss      0
hours_per_week    0
dtype: int64
```

We can see that there are no missing values in the numerical variables.

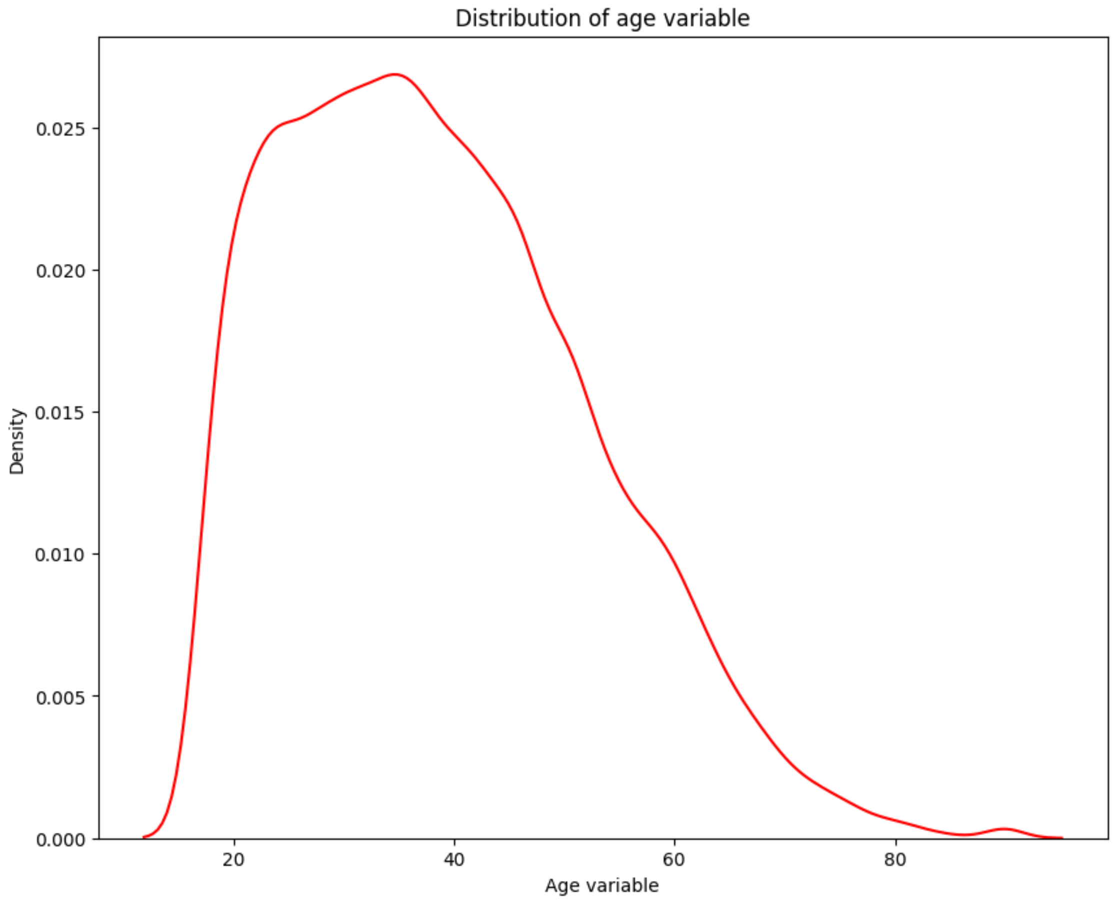
4.9.3 Explore age variable

[52]:
```
# View the distribution of `age` variable
f, ax = plt.subplots(figsize=(10,8))
x = df['age']
ax = sns.histplot(x, bins=10, color='blue', kde=True)
ax.set_title("Distribution of age variable")
plt.show()
```



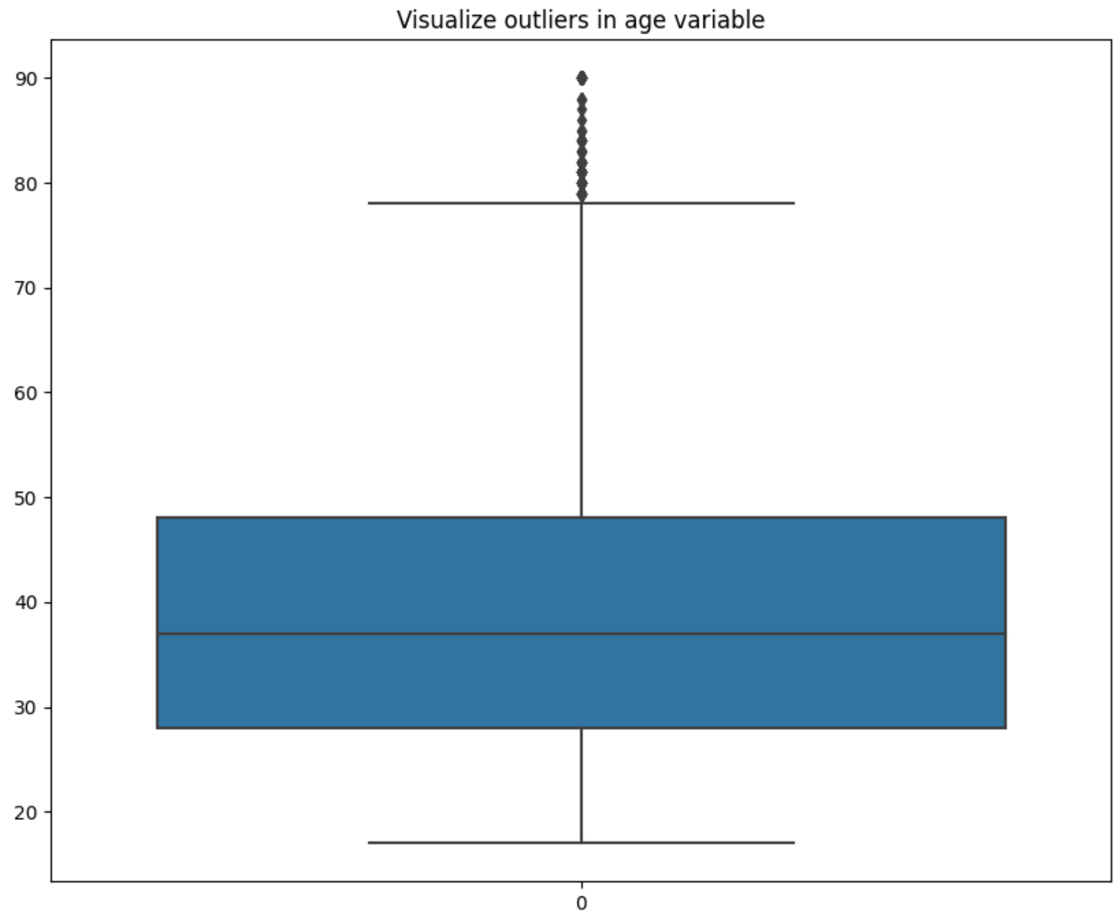
We can see that `age` is slightly positively skewed.

[53]:
```
f, ax = plt.subplots(figsize=(10,8))
x = df['age']
x = pd.Series(x, name="Age variable")
ax = sns.kdeplot(x, color='red')
ax.set_title("Distribution of age variable")
plt.show()
```



4.9.4 Detect outliers in age variable with boxplot

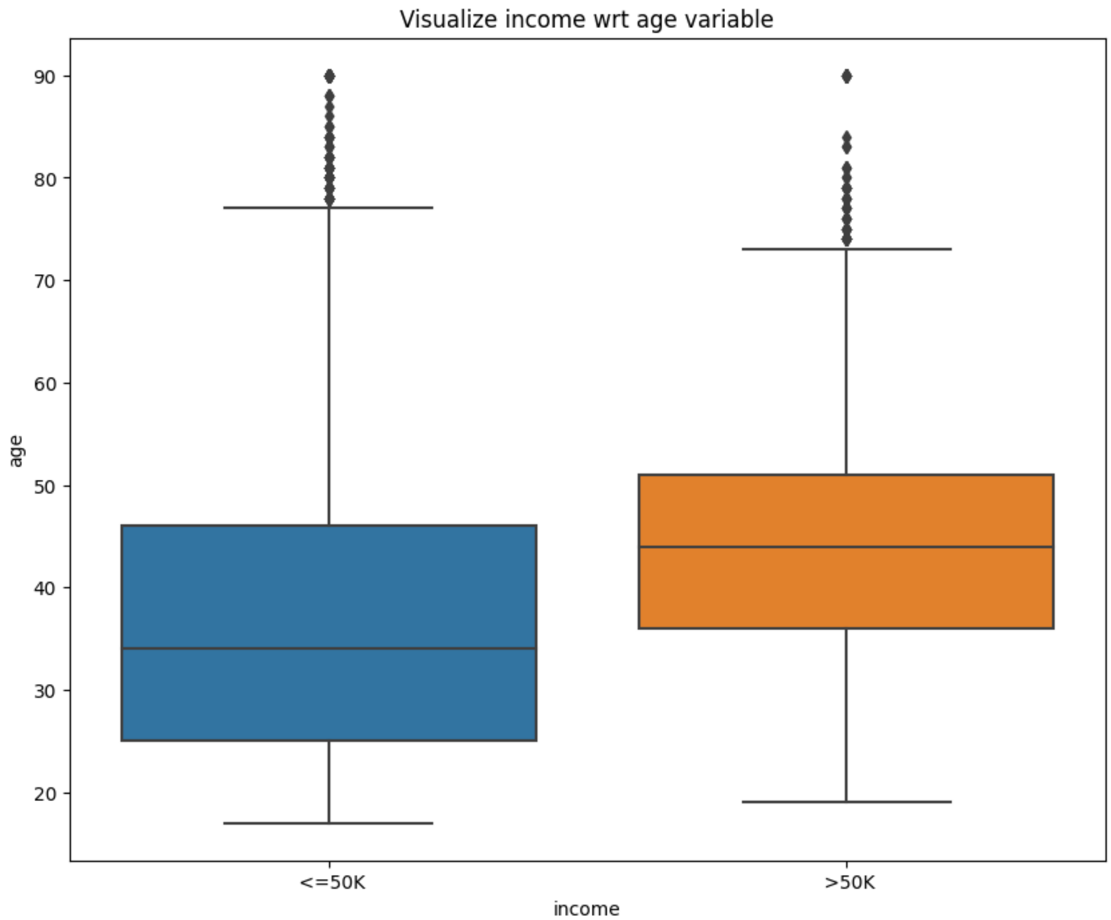
[54]:
```
f, ax = plt.subplots(figsize=(10,8))
x = df['age']
ax = sns.boxplot(x)
ax.set_title("Visualize outliers in age variable")
plt.show()
```



We can see that there are lots of outliers in `age` variable.

4.9.5 Explore relationship between `age` and `income` variables

[55]:
```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(x="income", y="age", data=df)
ax.set_title("Visualize income wrt age variable")
plt.show()
```

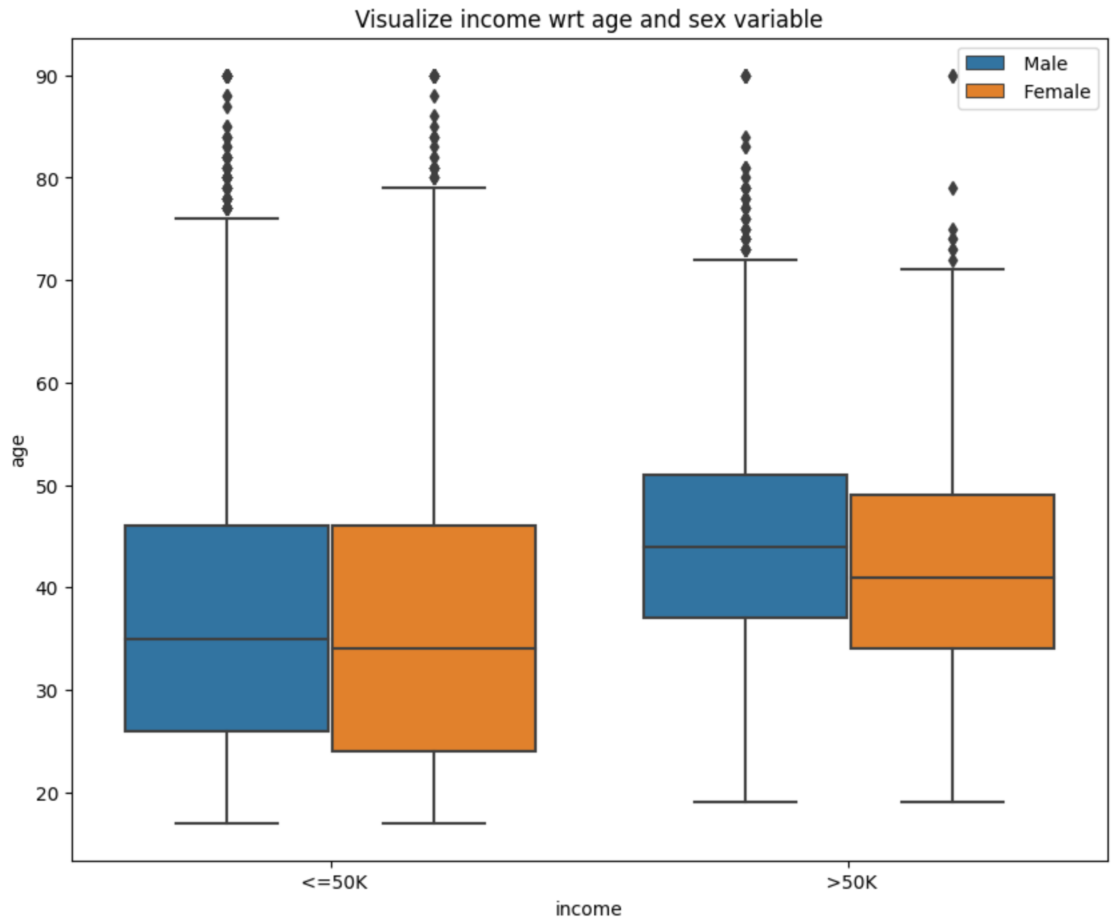


Interpretation

- As expected, younger people make less money as compared to senior people.

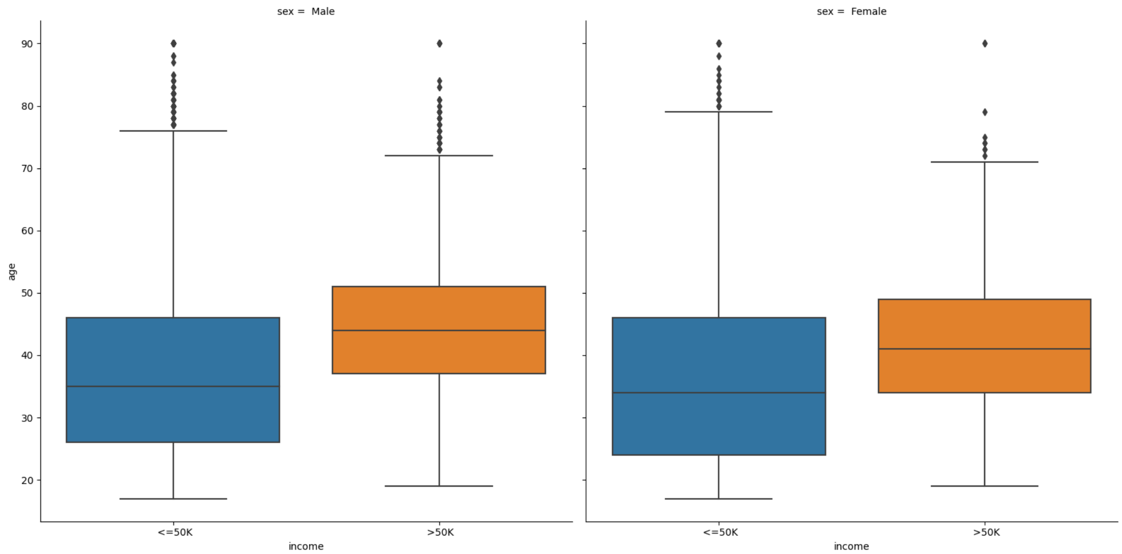
4.9.6 Visualize income wrt age and sex variable

[56]:
```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(x="income", y="age", hue="sex", data=df)
ax.set_title("Visualize income wrt age and sex variable")
ax.legend(loc='upper right')
plt.show()
```



[57]:
```
plt.figure(figsize=(8,6))
ax = sns.catplot(x="income", y="age", col="sex", data=df, kind="box", height=8,
  aspect=1)
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```

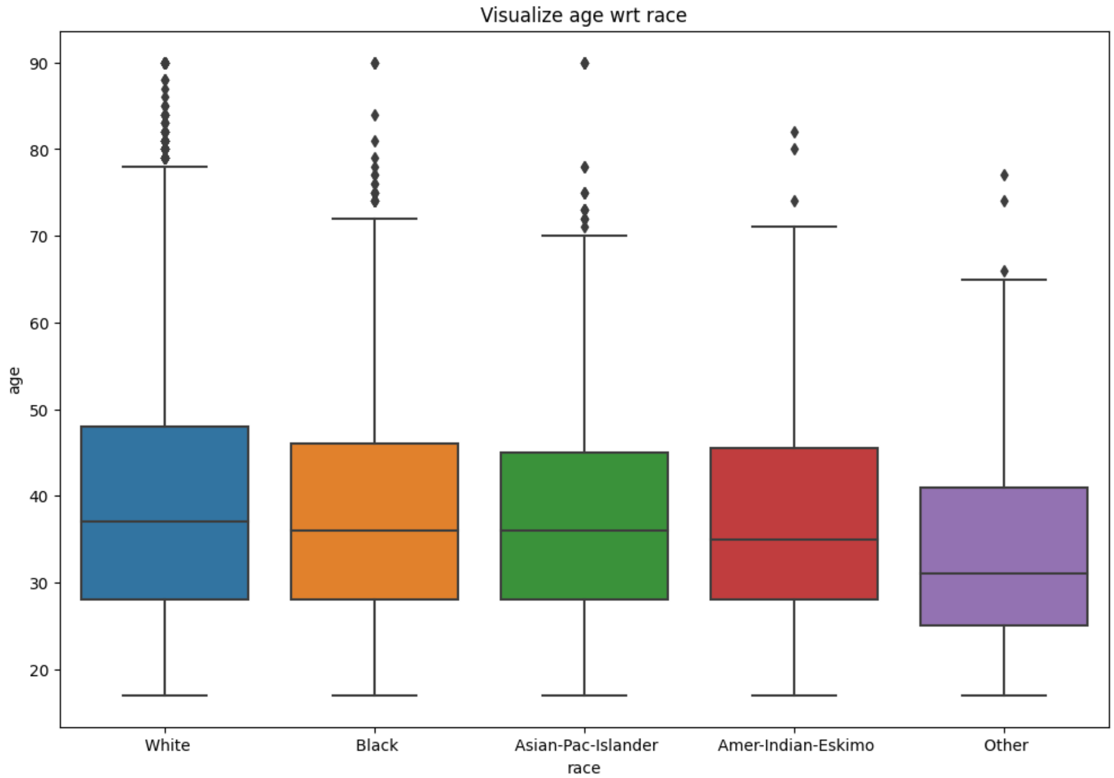


Interpretation

- Senior people make more money than younger people.

4.9.7 Visualize relationship between race and age

[58]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x ='race', y="age", data = df)
plt.title("Visualize age wrt race")
plt.show()
```



Interpretation

- Whites are more older than other groups of people.

4.9.8 Find out the correlations

[59]:
```
df.corr() # Compute pairwise correlation of columns, excluding NA/null values.
```

```
/tmp/ipykernel_107355/605702514.py:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df.corr() # Compute pairwise correlation of columns, excluding NA/null values.
```

[59]:
```
                     age    fnlwgt  education_num  capital_gain  capital_loss  \
age             1.000000 -0.076646       0.036527      0.077674      0.057775
fnlwgt         -0.076646  1.000000      -0.043195      0.000432     -0.010252
education_num   0.036527 -0.043195       1.000000      0.122630      0.079923
capital_gain    0.077674  0.000432       0.122630      1.000000     -0.031615
capital_loss    0.057775 -0.010252       0.079923     -0.031615      1.000000
hours_per_week  0.068756 -0.018768       0.148123      0.078409      0.054256

                hours_per_week
```

```
age               0.068756
fnlwgt           -0.018768
education_num     0.148123
capital_gain      0.078409
capital_loss      0.054256
hours_per_week    1.000000
```

Interpretation - We can see that there is no strong correlation between variables.

[60]:

```
# plot correlation heatmap to find out correlations
fig = plt.figure(dpi=300)
sns.heatmap(df.corr(),annot=True,linewidths=.5,cmap="viridis")
plt.show()
```

```
/tmp/ipykernel_107355/1293130057.py:3: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(),annot=True,linewidths=.5,cmap="viridis")
```