adult-census-income-prediction

June 7, 2023

1 Adult census income prediction

1.1 Problem statement :

- Problem Statement: The Goal is to predict whether a person has an income of more than 50K a year or not. This is basically a binary classification problem where a person is classified into the >50K group or <=50K group.

1.1.1 import required libraries of python

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import model_selection,preprocessing,tree
from sklearn.metrics import␣
↪confusion_matrix,classification_report,accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import seaborn as sns
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeClassifier
```

1.1.2 import dataset :

kaggel link :https://www.kaggle.com/datasets/overload10/adult-census-dataset

[2]:
```
df = pd.read_csv('adult.csv')
```

[3]:
```
df
```

[3]:
```
       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
…     …                 …       …           …              …
```

```
32556   27           Private  257302   Assoc-acdm            12
32557   40           Private  154374      HS-grad             9
32558   58           Private  151910      HS-grad             9
32559   22           Private  201490      HS-grad             9
32560   52      Self-emp-inc  287927      HS-grad             9

            marital-status          occupation    relationship    race  \
0            Never-married        Adm-clerical   Not-in-family   White
1       Married-civ-spouse     Exec-managerial         Husband   White
2                 Divorced   Handlers-cleaners   Not-in-family   White
3       Married-civ-spouse   Handlers-cleaners         Husband   Black
4       Married-civ-spouse      Prof-specialty            Wife   Black
…                        …                   …               …      …
32556   Married-civ-spouse        Tech-support            Wife   White
32557   Married-civ-spouse   Machine-op-inspct         Husband   White
32558              Widowed        Adm-clerical       Unmarried   White
32559        Never-married        Adm-clerical       Own-child   White
32560   Married-civ-spouse     Exec-managerial            Wife   White

           sex  capital-gain  capital-loss  hours-per-week         country  \
0         Male          2174             0              40   United-States
1         Male             0             0              13   United-States
2         Male             0             0              40   United-States
3         Male             0             0              40   United-States
4       Female             0             0              40            Cuba
…           …             …             …               …               …
32556   Female             0             0              38   United-States
32557     Male             0             0              40   United-States
32558   Female             0             0              40   United-States
32559     Male             0             0              20   United-States
32560   Female         15024             0              40   United-States

       salary
0       <=50K
1       <=50K
2       <=50K
3       <=50K
4       <=50K
…         …
32556   <=50K
32557    >50K
32558   <=50K
32559   <=50K
32560    >50K

[32561 rows x 15 columns]
```

1.2 EDA

1.2.1 The info() method prints information about the DataFrame.

The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

[4]:
```
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       32560 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  country         32561 non-null  object
 14  salary          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[5]:
```
df.describe()
```

[5]:

	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
```

[6]:
```
df
```

[6]:
```
         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
…     …                  …      …           …           …
32556     27           Private  257302   Assoc-acdm             12
32557     40           Private  154374      HS-grad              9
32558     58           Private  151910      HS-grad              9
32559     22           Private  201490      HS-grad              9
32560     52      Self-emp-inc  287927      HS-grad              9

            marital-status          occupation   relationship    race  \
0            Never-married        Adm-clerical  Not-in-family   White
1       Married-civ-spouse     Exec-managerial        Husband   White
2                 Divorced   Handlers-cleaners  Not-in-family   White
3       Married-civ-spouse   Handlers-cleaners        Husband   Black
4       Married-civ-spouse      Prof-specialty           Wife   Black
…                       …                   …              …     …
32556   Married-civ-spouse        Tech-support           Wife   White
32557   Married-civ-spouse   Machine-op-inspct        Husband   White
32558              Widowed        Adm-clerical      Unmarried   White
32559        Never-married        Adm-clerical      Own-child   White
32560   Married-civ-spouse     Exec-managerial           Wife   White

           sex  capital-gain  capital-loss  hours-per-week        country  \
0         Male          2174             0              40  United-States
1         Male             0             0              13  United-States
2         Male             0             0              40  United-States
3         Male             0             0              40  United-States
4       Female             0             0              40           Cuba
…          …             …             …               …              …
32556   Female             0             0              38  United-States
32557     Male             0             0              40  United-States
32558   Female             0             0              40  United-States
32559     Male             0             0              20  United-States
32560   Female         15024             0              40  United-States

       salary
```

```
0        <=50K
1        <=50K
2        <=50K
3        <=50K
4        <=50K
…        …
32556    <=50K
32557     >50K
32558    <=50K
32559    <=50K
32560     >50K

[32561 rows x 15 columns]
```

[7]:
```
print(df['salary'].value_counts())
```

```
 <=50K    24720
 >50K      7841
Name: salary, dtype: int64
```

[8]:
```
print(df['sex'].value_counts())
```

```
 Male      21790
 Female    10771
Name: sex, dtype: int64
```

[9]:
```
a = [df['country'].value_counts()]
a
```

[9]:
```
[ 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: country, dtype: int64]
```

[10]:
```
df['workclass'].value_counts()
```

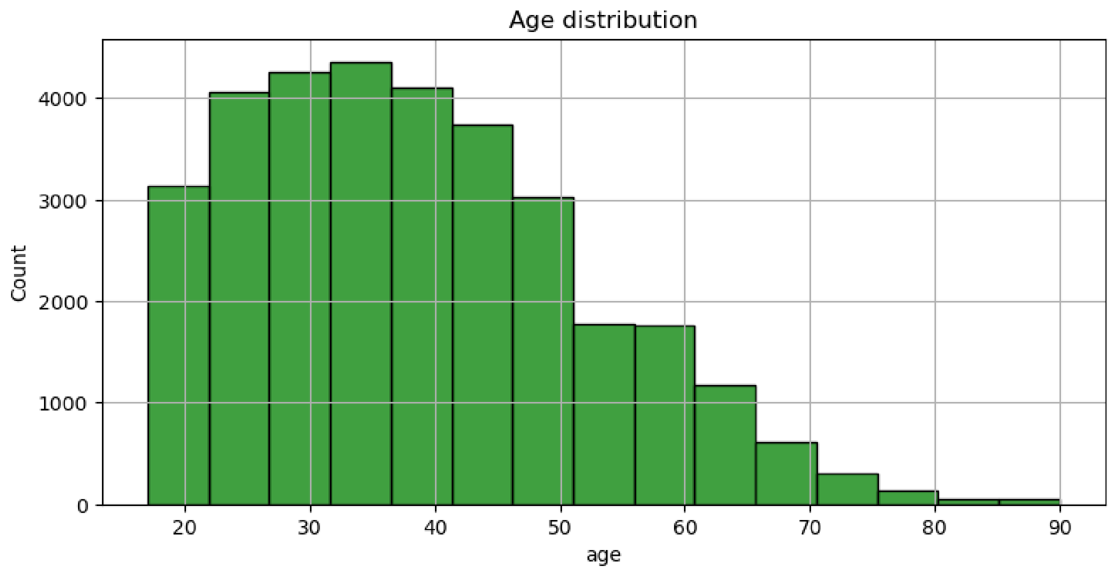
[10]:
```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1835
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

1.3 Data visualization

1.4 Age

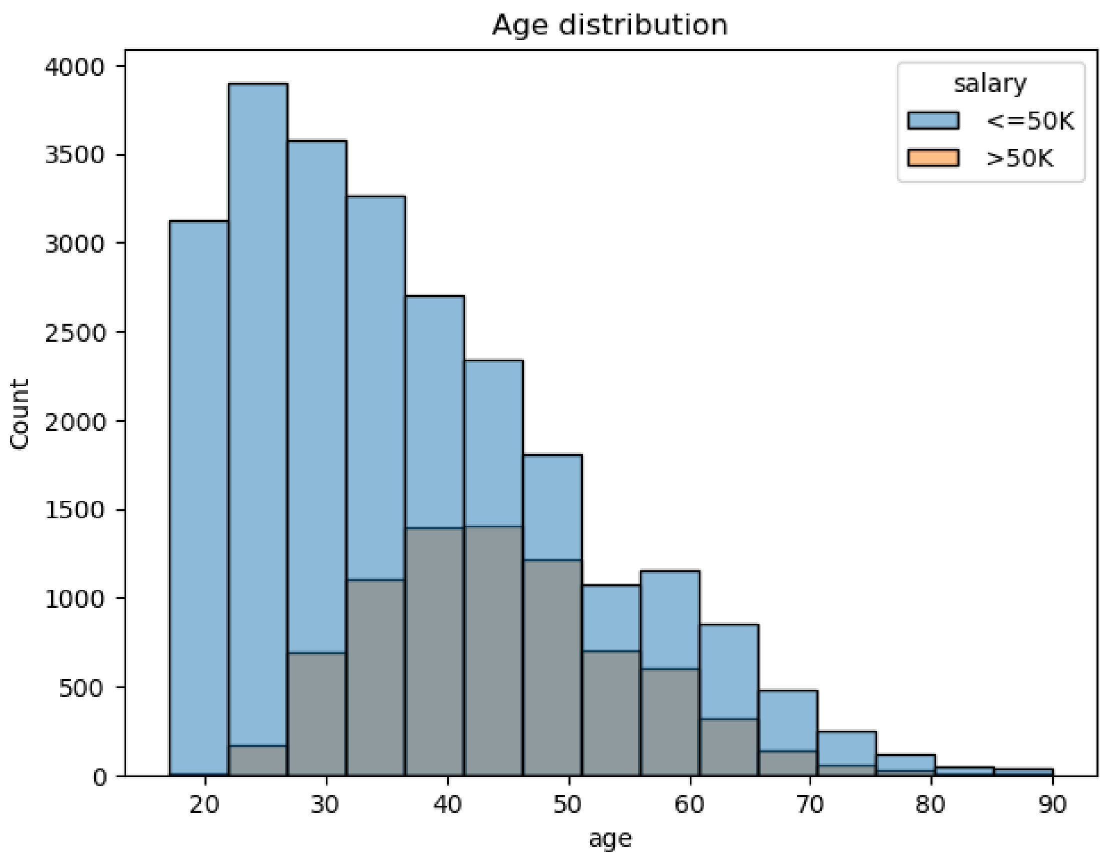
[11]:
```
plt.figure(figsize=(8,4))
sns.histplot(df['age'],color='green',bins=15)
plt.tight_layout()
plt.grid(True)
```

```
plt.title('Age distribution')
plt.show()
```



[12]:

```
sns.histplot(x=df['age'],hue=df['salary'],color='green',bins=15)
plt.tight_layout()
plt.title('Age distribution')
plt.show()
```



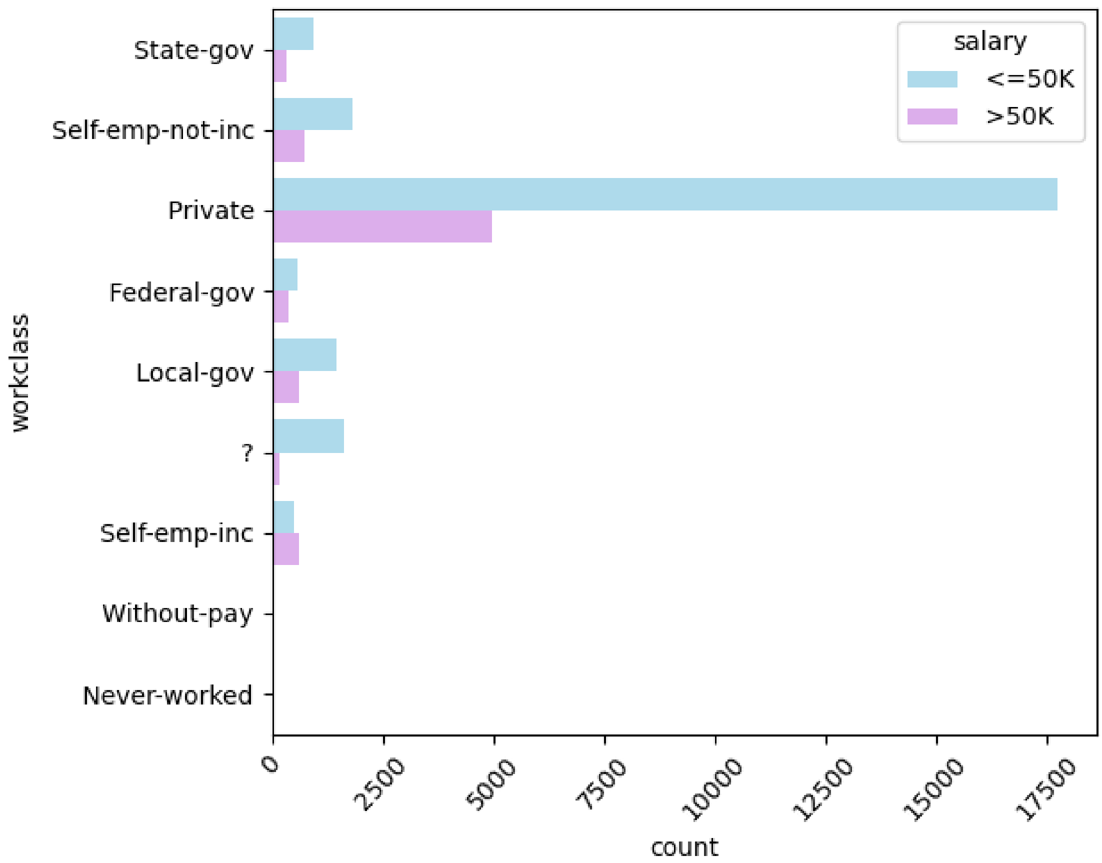
From the graph we can see that in the age group 0-20 there isn't any entry of salary greater than 50k, same goes with the group greater than 75 years.

1.4.1 work class

[13]:
```
df['workclass'].unique()
```

[13]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', nan, ' Without-pay',
       ' Never-worked'], dtype=object)
```

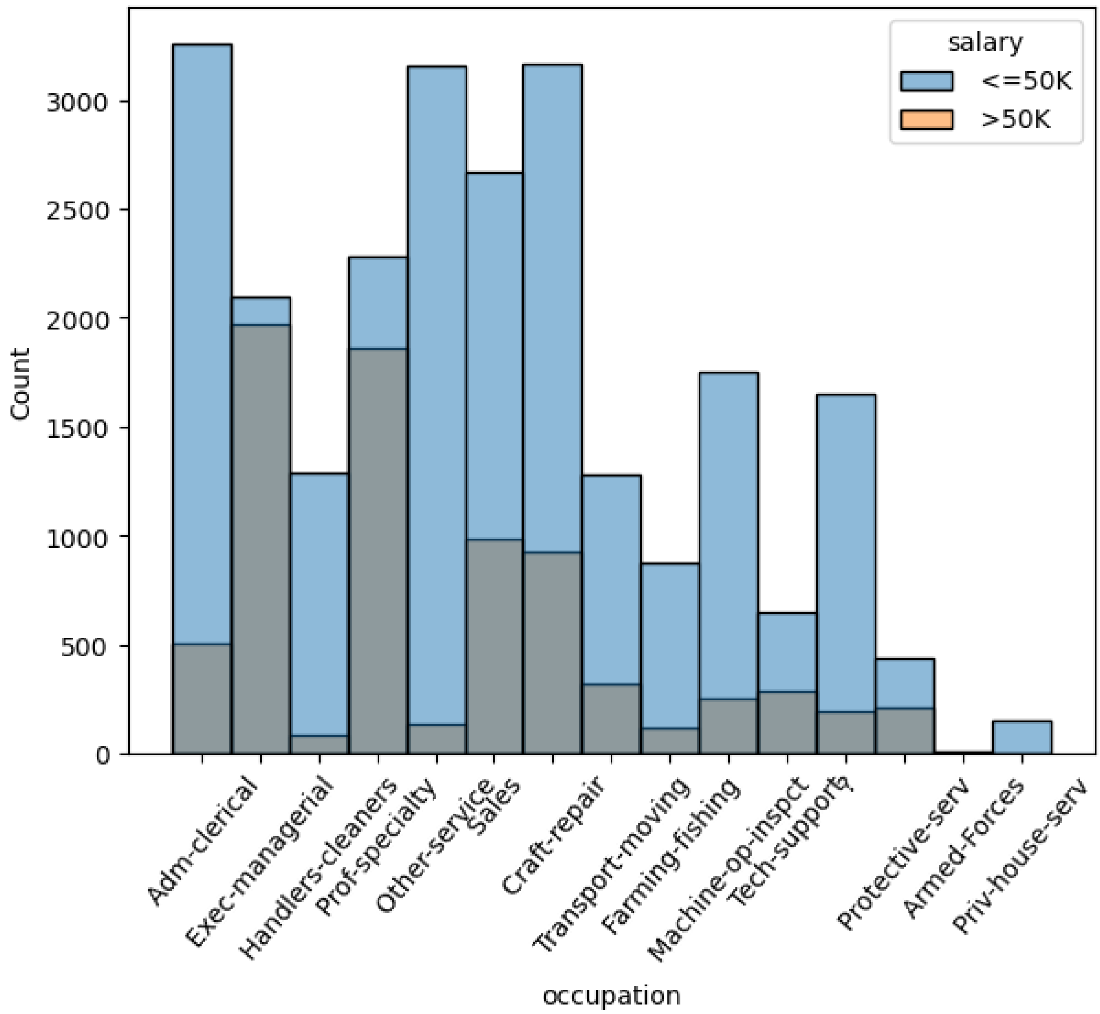
[17]:
```
sns.countplot(y=df['workclass'], hue = df['salary'],␣
 ↪palette=['#a4def5','#e1a4f5'])
plt.tight_layout()
plt.xticks(rotation = 47)
plt.show()
```



- The majority of the individuals work in the private sector. The probabilities of making above 50,000 are similar among the work classes except for self-emp-inc and federal government. Federal government is seen as the most elite in the public sector, which most likely explains the higher chance of earning more than 50,000.

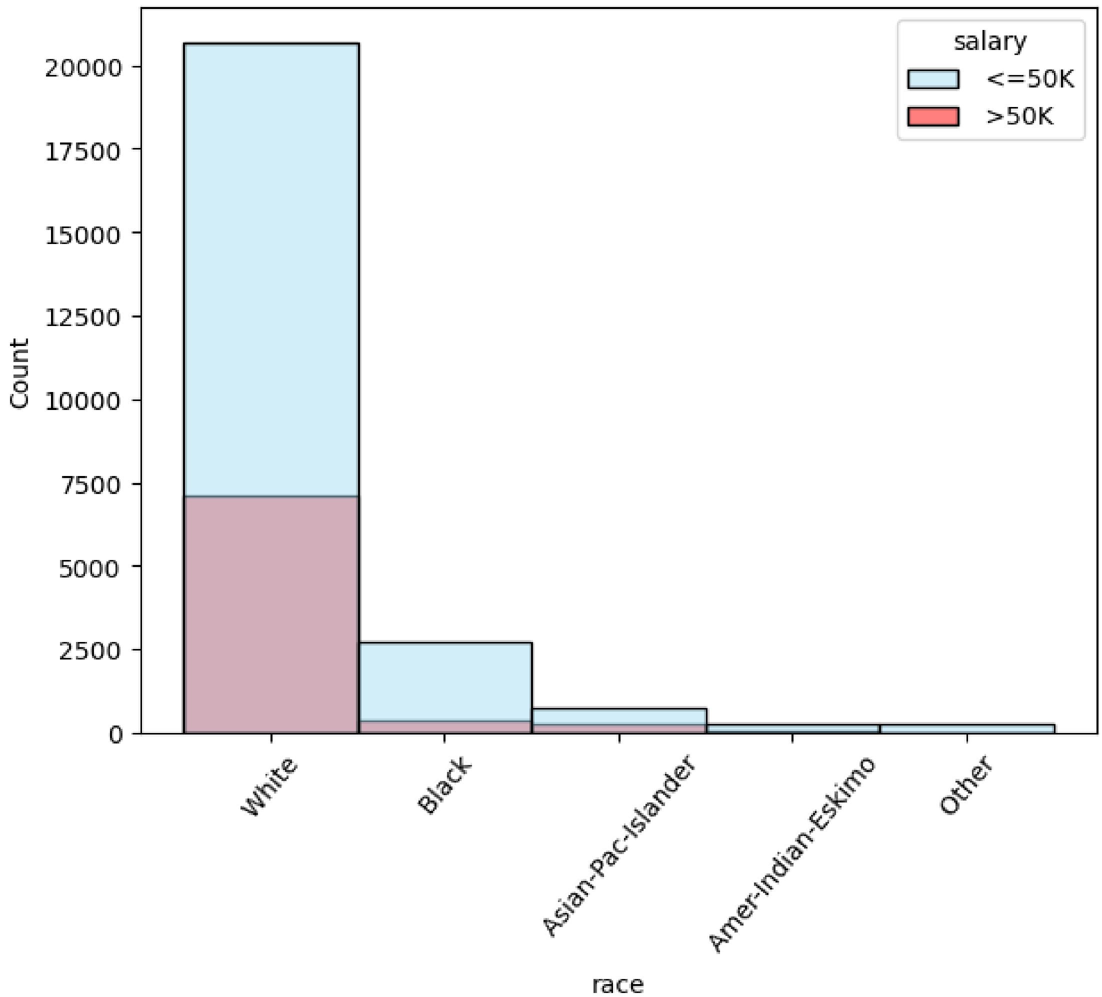
1.4.2 occupation

[19]:
```
sns.histplot(x=df['occupation'], hue=df['salary'], color='green',bins=15)
plt.tight_layout()
plt.xticks(rotation = 50)
plt.show()
```



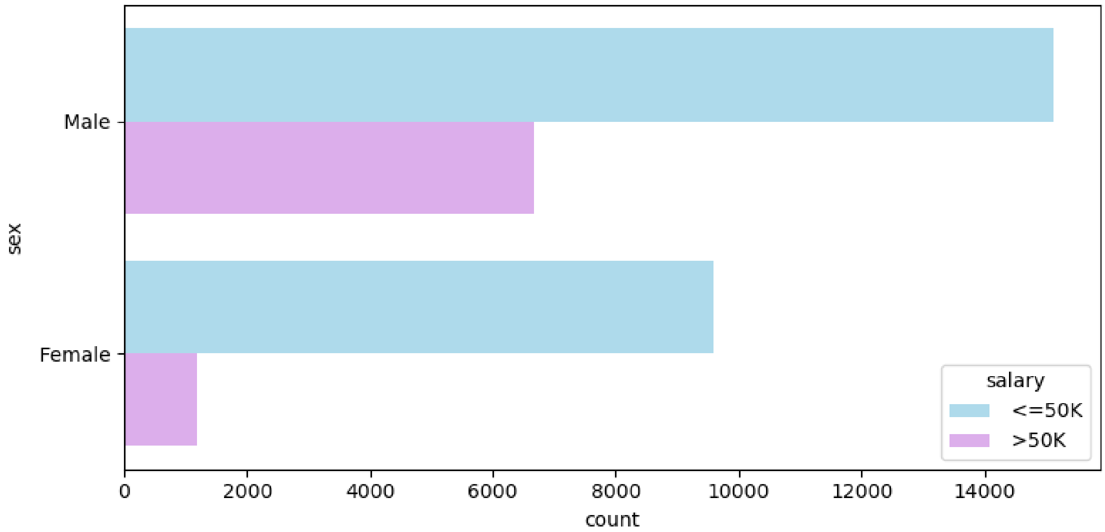
1.4.3 race

[26]:
```
sns.histplot(x = df['race'], hue = df['salary'],color = 'green',bins = 15␣
  ↪,palette = ['#a4def5','red'])
plt.tight_layout()
plt.xticks(rotation = 50)
plt.show()
```



1.4.4 sex

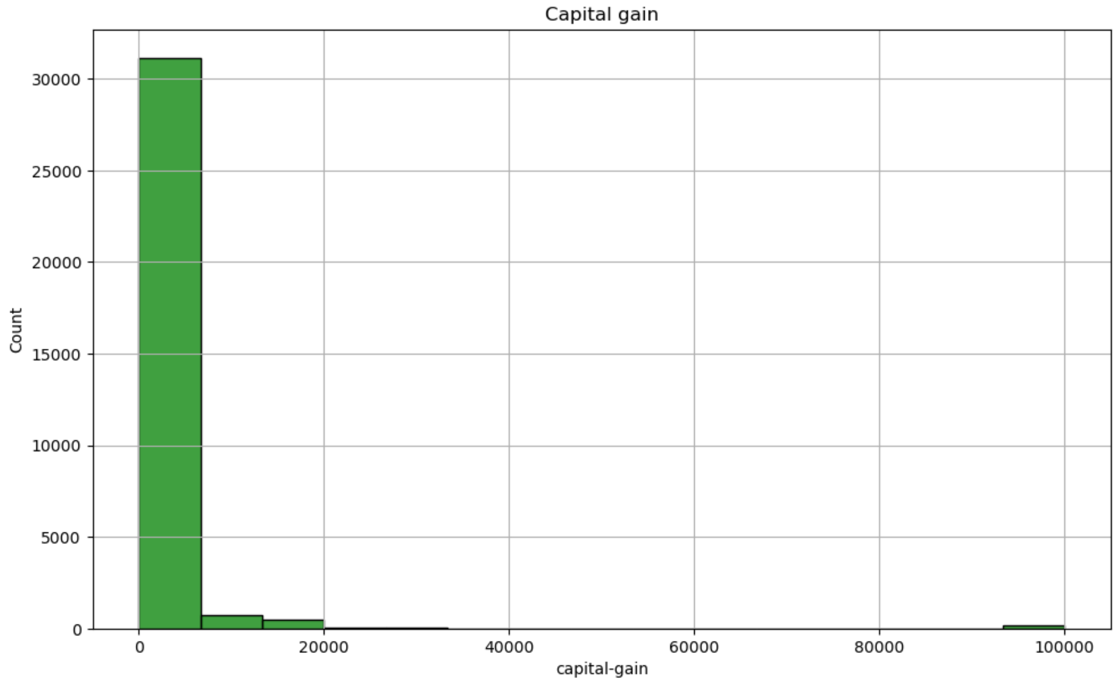
[27]:
```
plt.figure(figsize = [8,4])
sns.countplot(y=df['sex'],hue = df['salary'],palette=['#a4def5','#e1a4f5'])
plt.tight_layout()
plt.show()
```



- The percentage of males who make greater than 50,000 is much greater than the percentage of females that make the same amount. This will certainly be a significant factor, and should be a feature considered in our prediction model.

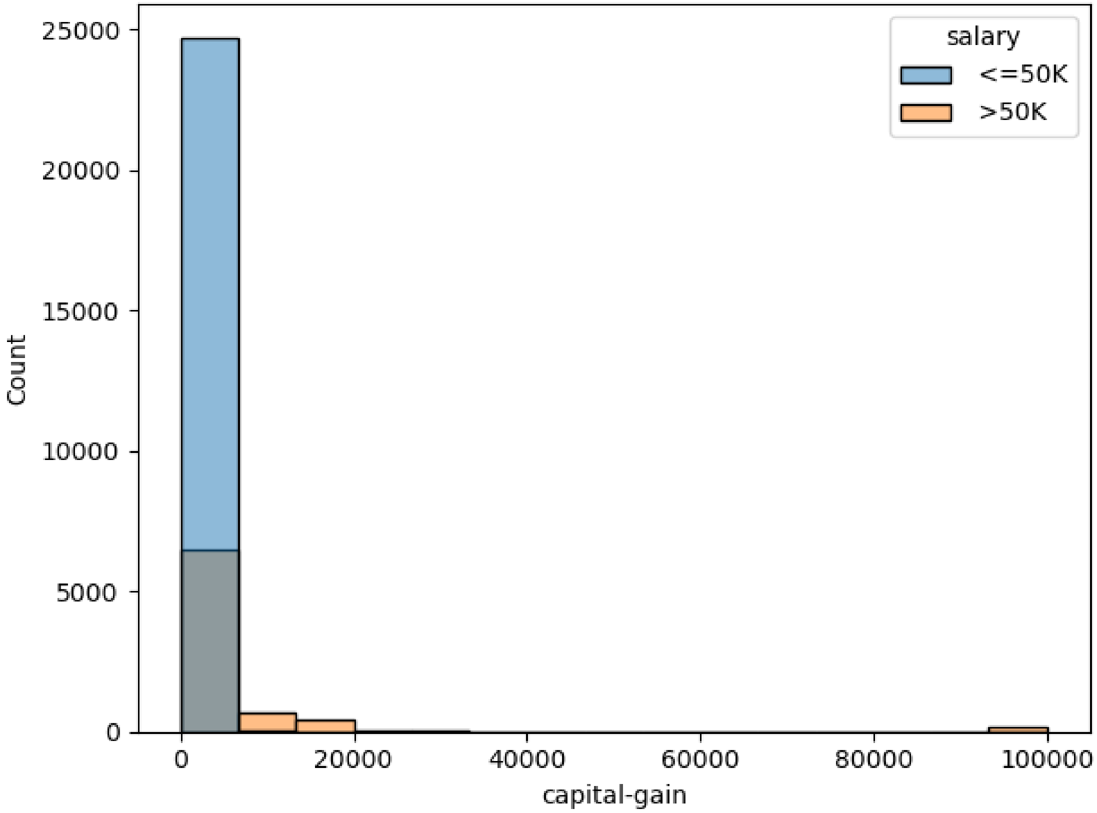
1.4.5 capital gain

[28]:
```
plt.figure(figsize=(10,6))
sns.histplot(df['capital-gain'], color = 'green', bins = 15)
plt.tight_layout()
plt.grid(True)
plt.title('Capital gain')
plt.show()
```



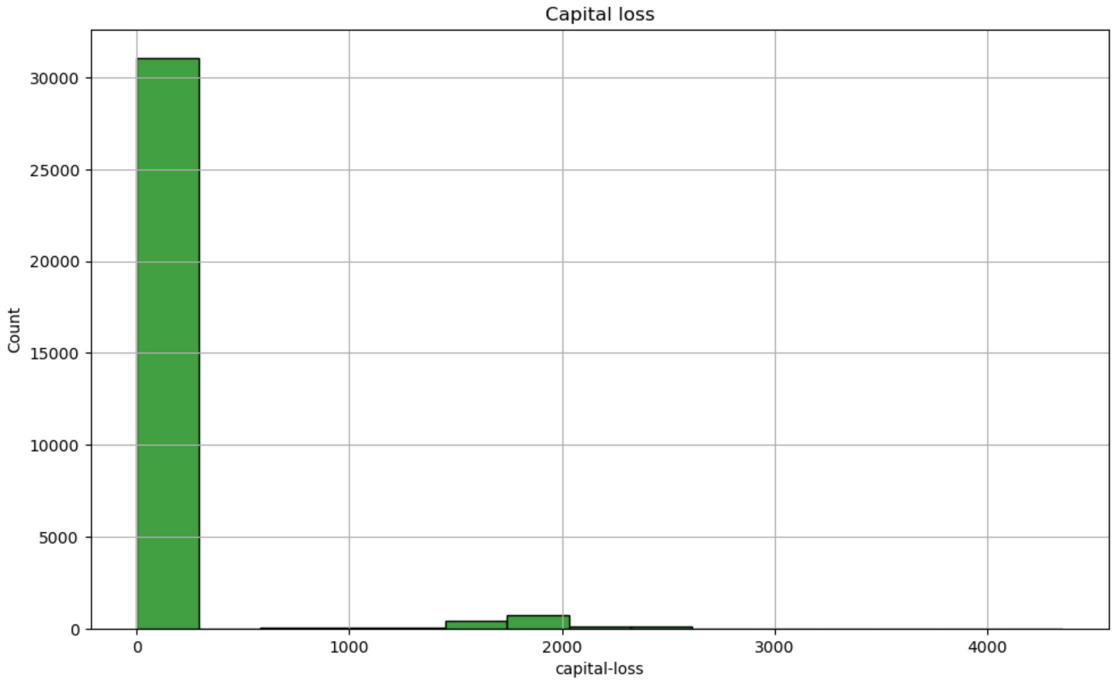
- From the graph we can see that the distribution of capital gain is very skewed.
- And there are outliers at data poitn 100000.

[29]:
```
sns.histplot(x=df['capital-gain'],hue=df['salary'],color='green',bins=15)
plt.tight_layout()
plt.show()
```



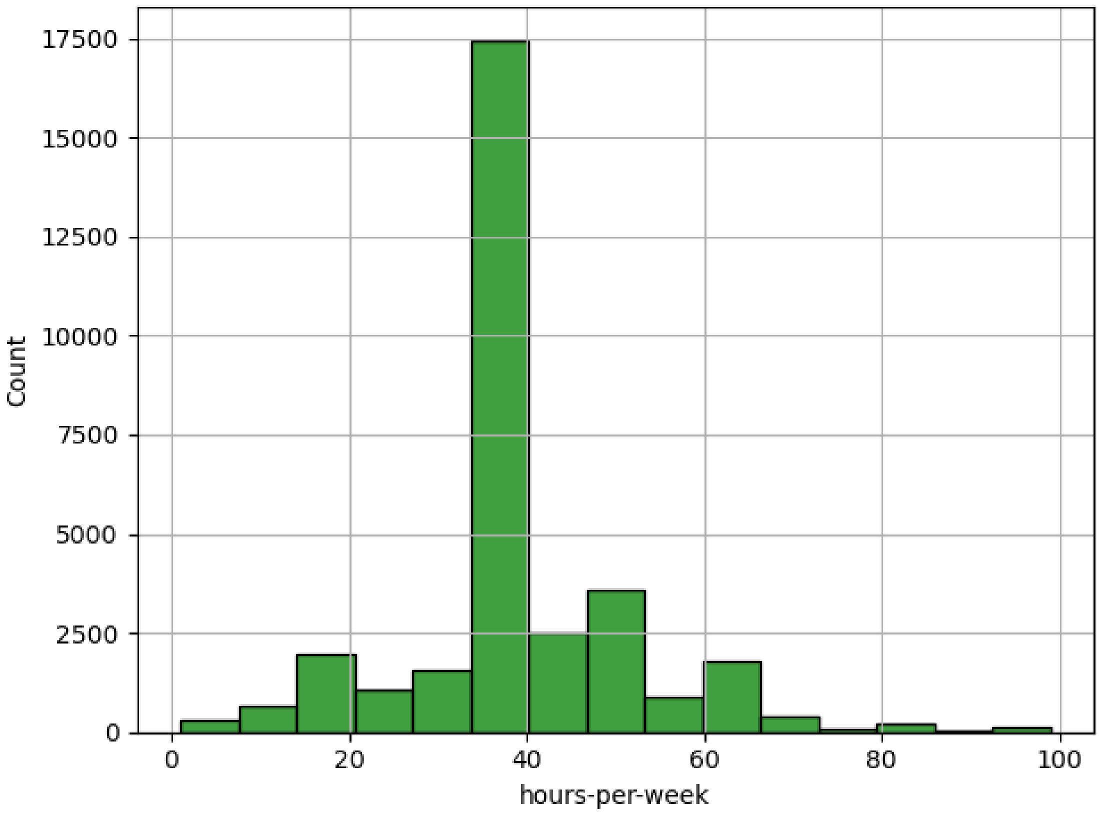
1.4.6 capital loss

[30]:
```
plt.figure(figsize=(10,6))
sns.histplot(df['capital-loss'],color='green',bins=15)
plt.tight_layout()
plt.grid(True)
plt.title('Capital loss')
plt.show()
```



1.4.7 hours per week

[31]:
```
sns.histplot(df['hours-per-week'],color='green',bins=15)
plt.tight_layout()
plt.grid(True)
plt.show()
```

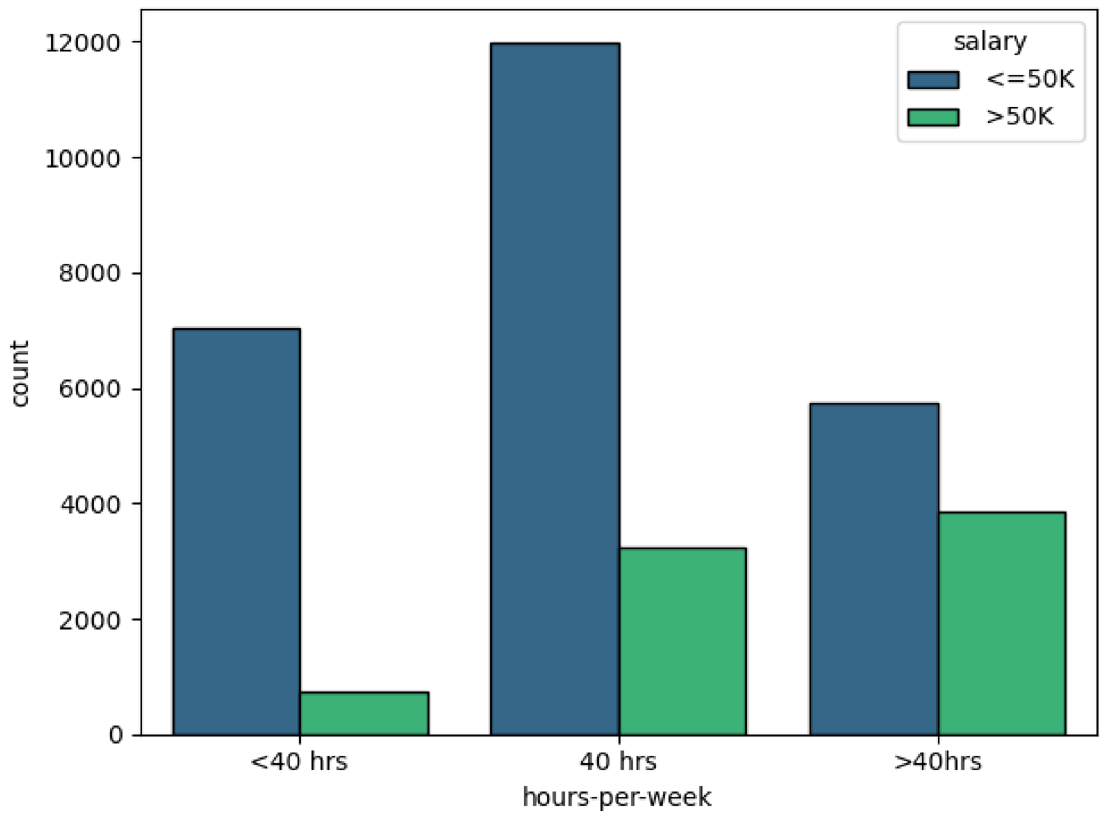


- We can see that vast majority of values are of 40 hours. what we can do is make 3 classes, i.e <40hrs, 40hrs and >40hrs, and check whther it is significant or not.

[32]:
```
def hrs_edit(val):
    if (val<40):
        return ('<40 hrs')
    elif (val==40):
        return ('40 hrs')
    else:
        return ('>40hrs')
```

[33]:
```
df['hours-per-week']=df['hours-per-week'].apply(hrs_edit)
```

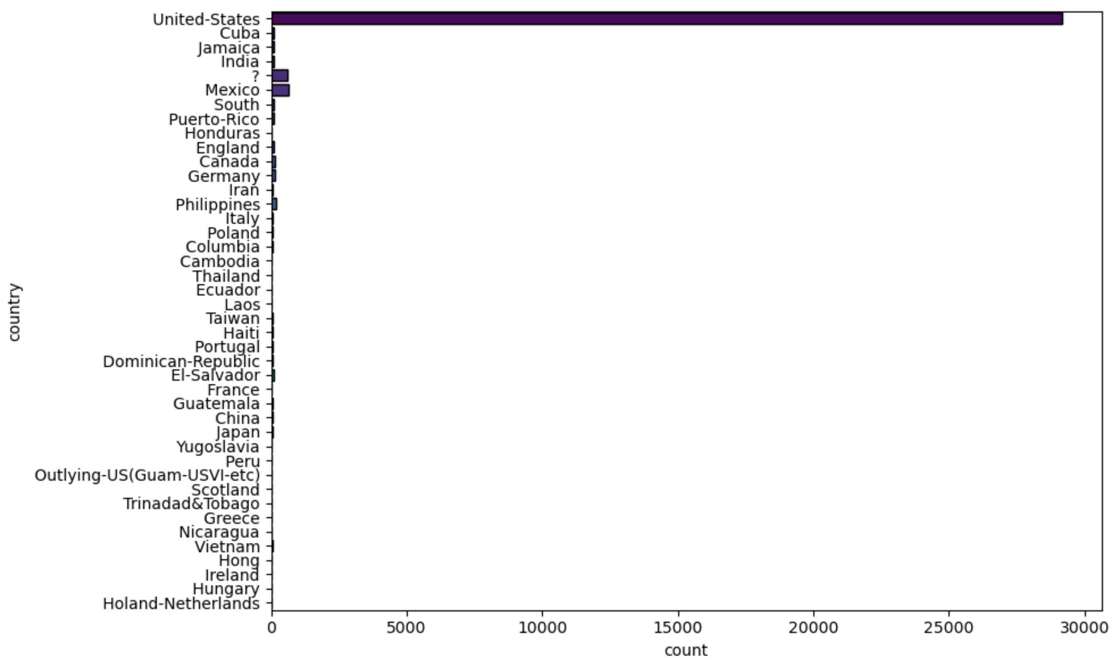
[34]:
```
sns.countplot(x=df['hours-per-week'],hue=df['salary'],palette='viridis',
              saturation=0.9,edgecolor="black",order=['<40 hrs','40␣
 ↪hrs','>40hrs'])
plt.tight_layout()
plt.show()
```



- The percentage of individuals making over 50,000 drastically decreases when less than 40 hours per week, and increases significantly when greater than 40 hours per week.

1.4.8 country

[35]:
```
plt.figure(figsize=(10,6))
sns.countplot(y=df['country'],palette='viridis',saturation=0.
  ↪9,edgecolor="black")
plt.tight_layout()
plt.show()
```



1.4.9 Feature engineering

1. We will drop the features:

- fnlwgt
- education
- relationship
- race

2. Impute Nan values with mode.

3. train - test split.

4. Lable encoding.

1.4.10 Dropping Education - Education Num is enough.

1.4.11 Dropping Final Weight - highly discrete data so not useful

[36]:
```
df = df.drop(['education','fnlwgt','race','relationship'],axis = 1)
df
```

[36]:

	age	workclass	education-num	marital-status \
0	39	State-gov	13	Never-married
1	50	Self-emp-not-inc	13	Married-civ-spouse
2	38	Private	9	Divorced
3	53	Private	7	Married-civ-spouse
4	28	Private	13	Married-civ-spouse

```
…     …                …             …                  …
32556   27          Private            12   Married-civ-spouse
32557   40          Private             9   Married-civ-spouse
32558   58          Private             9              Widowed
32559   22          Private             9        Never-married
32560   52     Self-emp-inc             9   Married-civ-spouse

               occupation      sex  capital-gain  capital-loss hours-per-week  \
0            Adm-clerical     Male          2174             0         40 hrs
1         Exec-managerial     Male             0             0        <40 hrs
2       Handlers-cleaners     Male             0             0         40 hrs
3       Handlers-cleaners     Male             0             0         40 hrs
4          Prof-specialty   Female             0             0         40 hrs
…                     …      …            …            …             …
32556        Tech-support   Female             0             0        <40 hrs
32557   Machine-op-inspct     Male             0             0         40 hrs
32558        Adm-clerical   Female             0             0         40 hrs
32559        Adm-clerical     Male             0             0        <40 hrs
32560     Exec-managerial   Female         15024             0         40 hrs

              country  salary
0       United-States   <=50K
1       United-States   <=50K
2       United-States   <=50K
3       United-States   <=50K
4                Cuba   <=50K
…                 …      …
32556   United-States   <=50K
32557   United-States    >50K
32558   United-States   <=50K
32559   United-States   <=50K
32560   United-States    >50K

[32561 rows x 11 columns]
```

1.4.12 Replacing ? with NaN

[37]:
```
df.isin([' ?']).sum()
```

[37]:
```
age                  0
workclass         1835
education-num        0
marital-status       0
occupation        1843
sex                  0
capital-gain         0
capital-loss         0
```

```
hours-per-week        0
country             583
salary                0
dtype: int64
```

[38]:
```
df['workclass'].replace(' ?',0,inplace = True)
df['occupation'].replace(' ?',0,inplace = True)
df['country'].replace(' ?',0,inplace = True)
```

[39]:
```
df['workclass'].replace(0,np.nan,inplace = True)
df['occupation'].replace(0,np.nan,inplace = True)
df['country'].replace(0,np.nan,inplace = True)
```

[40]:
```
df["workclass"] = df["workclass"].fillna(df["workclass"].mode()[0])
df["occupation"] = df["occupation"].fillna(df["occupation"].mode()[0])
df["country"] = df["country"].fillna(df["country"].mode()[0])
```

[41]:
```
df["workclass"].value_counts()
```

[41]:
```
 Private             24532
 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
```

[42]:
```
df['marital-status'].unique()
```

[42]:
```
array([' Never-married', ' Married-civ-spouse', ' Divorced',
       ' Married-spouse-absent', ' Separated', ' Married-AF-spouse',
       ' Widowed'], dtype=object)
```

[43]:
```
def married(val):
    if val==' Never-married':
        return 'not-married'
    elif val==' Divorced':
        return 'not-married'
    elif val==' Separated':
        return 'not-married'
    elif val==' Widowed':
        return 'not-married'
    else:
        return 'married'
```

[44]:
```
df['marital-status']=df['marital-status'].apply(married)
```

[45]:
```
df['marital-status'].unique()
```

[45]:
```
array(['not-married', 'married'], dtype=object)
```

1.5 Label Encoding

[46]:
```
encoder = preprocessing.LabelEncoder()
```

[47]:
```
df['workclass'] = encoder.fit_transform(df['workclass'])
df['marital-status'] = encoder.fit_transform(df['marital-status'])
df['occupation'] = encoder.fit_transform(df['occupation'])
df['sex'] = encoder.fit_transform(df['sex'])
df['country'] = encoder.fit_transform(df['country'])
df['salary'] = encoder.fit_transform(df['salary'])
df['hours-per-week'] = encoder.fit_transform(df['hours-per-week'])
```

[48]:
```
df
```

[48]:
```
       age  workclass  education-num  marital-status  occupation  sex  \
0       39          6             13               1           0    1
1       50          5             13               0           3    1
2       38          3              9               1           5    1
3       53          3              7               0           5    1
4       28          3             13               0           9    0
…      …          …              …               …           …   …
32556   27          3             12               0          12    0
32557   40          3              9               0           6    1
32558   58          3              9               1           0    0
32559   22          3              9               1           0    1
32560   52          4              9               0           3    0

       capital-gain  capital-loss  hours-per-week  country  salary
0              2174             0               0       38       0
1                 0             0               1       38       0
2                 0             0               0       38       0
3                 0             0               0       38       0
4                 0             0               0        4       0
…               …             …               …        …      …
32556             0             0               1       38       0
32557             0             0               0       38       1
32558             0             0               0       38       0
32559             0             0               1       38       0
32560         15024             0               0       38       1

[32561 rows x 11 columns]
```

[49]:
```
sns.pairplot(df,hue=None)
```

[49]: <seaborn.axisgrid.PairGrid at 0x2054bbfc580>

1.5.1 Model train and test

1.6 Logistic Regression

[50]:
```
logistic = LogisticRegression()
```

[51]:
```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

[52]:
```python
X_train,X_test,y_train,y_test = model_selection.train_test_split(X,y)
logistic.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[52]:
```
LogisticRegression()
```

[65]:
```python
y_pred = logistic.predict(X_test)
```

[66]:
```python
print(confusion_matrix(y_test,y_pred))
```

```
[[5726  430]
 [1108  877]]
```

[67]:
```python
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.93      0.88      6156
           1       0.67      0.44      0.53      1985

    accuracy                           0.81      8141
   macro avg       0.75      0.69      0.71      8141
weighted avg       0.80      0.81      0.80      8141
```

[68]:
```python
print("Accurracy",round(accuracy_score(y_test, y_pred)*100),"%")
```

```
Accurracy 81 %
```

1.7 Random Forest

[57]:
```python
random_forest = RandomForestClassifier(n_estimators=10,
                            random_state=0)
random_forest.fit(X_train, y_train)
Y_prediction = random_forest.predict(X_test)
random_forest.score(X_train, y_train)
acc_random_forest = round(random_forest.score(X_train, y_train) * 100, 2)
```

[58]:
```
print(confusion_matrix(y_test,Y_prediction))
```

```
[[5627  529]
 [ 779 1206]]
```

[59]:
```
print(classification_report(y_test,Y_prediction))
```

```
              precision    recall  f1-score   support

           0       0.88      0.91      0.90      6156
           1       0.70      0.61      0.65      1985

    accuracy                           0.84      8141
   macro avg       0.79      0.76      0.77      8141
weighted avg       0.83      0.84      0.84      8141
```

[60]:
```
print("Accurracy",round(accuracy_score(y_test, Y_prediction)*100),"%")
```

```
Accurracy 84 %
```

1.8 Decision tree

[61]:
```
classifier= DecisionTreeClassifier(criterion='entropy', random_state=10)
classifier.fit(X_train, y_train)
```

[61]:
```
DecisionTreeClassifier(criterion='entropy', random_state=10)
```

[62]:
```
y_pred = classifier.predict(X_test)
```

[64]:
```
print("Accurracy",  round(accuracy_score(y_test,  y_pred)*100),  "%")
```

```
Accurracy 82 %
```

[ ]:
```

```