Classification Models

Introduction:

In this project we will be working with the UCI/Kaggle adult dataset. We aim at predicting if people in the data set belong in a certain class by income, either making <=50k or >50k per year.

We will be performing EDA, Feature Engineering and Visualization while answering relevant questions that we need to ask ourselves in this journey to make the dataset ready for different classification models development and evaluation of the same.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import os
```

In [2]:
```
os.getcwd()
```

Out[2]:
```
'C:\\Users\\ruzdomain\\Desktop\\MLBDA\\NoteBooks_Data\\HTML_NOTEBOOK'
```

In [3]:
```
os.chdir(r'C:\Users\ruzdomain\Desktop\MLBDA\PROJECT')
```

In [4]:
```
os.getcwd()
```

Out[4]:
```
'C:\\Users\\ruzdomain\\Desktop\\MLBDA\\PROJECT'
```

In [5]:
```
df_class= pd.read_csv("adult_sal.csv", na_values  = '?')
df_class.head()
```

Out[5]:

	Unnamed: 0	age	type_employer	fnlwgt	education	education_num	marital	occupation	relationship	race	sex	capital_gain	capital_loss	hr_per_week	country	income
0	1	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	2	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	3	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	4	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	5	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [6]:
```
df_class.describe()
```

Out[6]:

	Unnamed: 0	age	fnlwgt	education_num	capital_gain	capital_loss	hr_per_week
count	32561.000000	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	16281.000000	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	9399.695394	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	1.000000	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	8141.000000	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	16281.000000	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	24421.000000	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	32561.000000	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Detail of the columns of the dataset

This is a table of the findings about all the variables:

Variable	Description
age	Age of the person
type_employer	Class of work
fnlwgt	Final weight of how much of the population it represents
education	Education level
education_num	Numeric education level
marital	Marital status of the person
occupation	Occupation of the person
relationship	Type of relationship
race	Race of the person
sex	Sex of the person
capital_gain	Capital gains obtained
capital_loss	Capital loss suffered
hr_per_week	Average number of hours working per week

country	Country of origin
income	Income level

In [7]:
```
df_class.drop("Unnamed: 0",axis=1, inplace=True)
```

In [8]:
```
df_class.describe(include='all')
```

Out[8]:

	age	type_employer	fnlwgt	education	education_num	marital	occupation	relationship	race	sex	capital_gain	capital_loss	hr_per_week	country	income
count	32561.000000	30725	3.256100e+04	32561	32561.000000	32561	30718	32561	32561	32561	32561.000000	32561.000000	32561.000000	31978	32561
unique	NaN	8	NaN	16	NaN	7	14	6	5	2	NaN	NaN	NaN	41	2
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband	White	Male	NaN	NaN	NaN	United-States	<=50K
freq	NaN	22696	NaN	10501	NaN	14976	4140	13193	27816	21790	NaN	NaN	NaN	29170	24720
mean	38.581647	NaN	1.897784e+05	NaN	10.080679	NaN	NaN	NaN	NaN	NaN	1077.648844	87.303830	40.437456	NaN	NaN
std	13.640433	NaN	1.055500e+05	NaN	2.572720	NaN	NaN	NaN	NaN	NaN	7385.292085	402.960219	12.347429	NaN	NaN
min	17.000000	NaN	1.228500e+04	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	1.000000	NaN	NaN
25%	28.000000	NaN	1.178270e+05	NaN	9.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	40.000000	NaN	NaN
50%	37.000000	NaN	1.783560e+05	NaN	10.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	40.000000	NaN	NaN
75%	48.000000	NaN	2.370510e+05	NaN	12.000000	NaN	NaN	NaN	NaN	NaN	0.000000	0.000000	45.000000	NaN	NaN
max	90.000000	NaN	1.484705e+06	NaN	16.000000	NaN	NaN	NaN	NaN	NaN	99999.000000	4356.000000	99.000000	NaN	NaN

In [9]:
```
df_class.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   type_employer  30725 non-null  object
 2   fnlwgt         32561 non-null  int64 
 3   education      32561 non-null  object
 4   education_num  32561 non-null  int64 
 5   marital        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  capital_gain   32561 non-null  int64 
 11  capital_loss   32561 non-null  int64 
 12  hr_per_week    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
```

In [10]:
```
df_class.shape
```

Out[10]:
```
(32561, 15)
```

In [11]:
```
df_class.dtypes
```

Out[11]:
```
age              int64
type_employer    object
fnlwgt            int64
education        object
education_num     int64
marital          object
occupation       object
relationship     object
race             object
sex              object
capital_gain      int64
capital_loss      int64
hr_per_week       int64
country          object
income           object
dtype: object
```

In [12]:
```
missingrows = df_class.isna().sum()
```

In [13]:
```
missingrows
```

Out[13]:
```
age                 0
type_employer    1836
fnlwgt              0
education           0
education_num       0
marital             0
occupation       1843
relationship        0
race                0
sex                 0
capital_gain        0
capital_loss        0
hr_per_week         0
country           583
income              0
dtype: int64
```

In [14]:
```
def percentage_of_miss(df):
  df1=df[df.columns[df.isnull().sum()>=1]]
  total_miss = df1.isnull().sum().sort_values(ascending=False)
  percent_miss = (df1.isnull().sum()/df1.isnull().count()).sort_values(ascending=False)
  missing_data = pd.concat([total_miss, percent_miss], axis=1, keys=['Number of Missing', 'Percentage'])
  return(missing_data)
```

In [15]:
```
percentage_of_miss(df_class)
```

Out[15]:

	Number of Missing	Percentage
occupation	1843	0.056601
type_employer	1836	0.056386
country	583	0.017905

In [16]:
```
df_class = df_class.drop_duplicates()
```

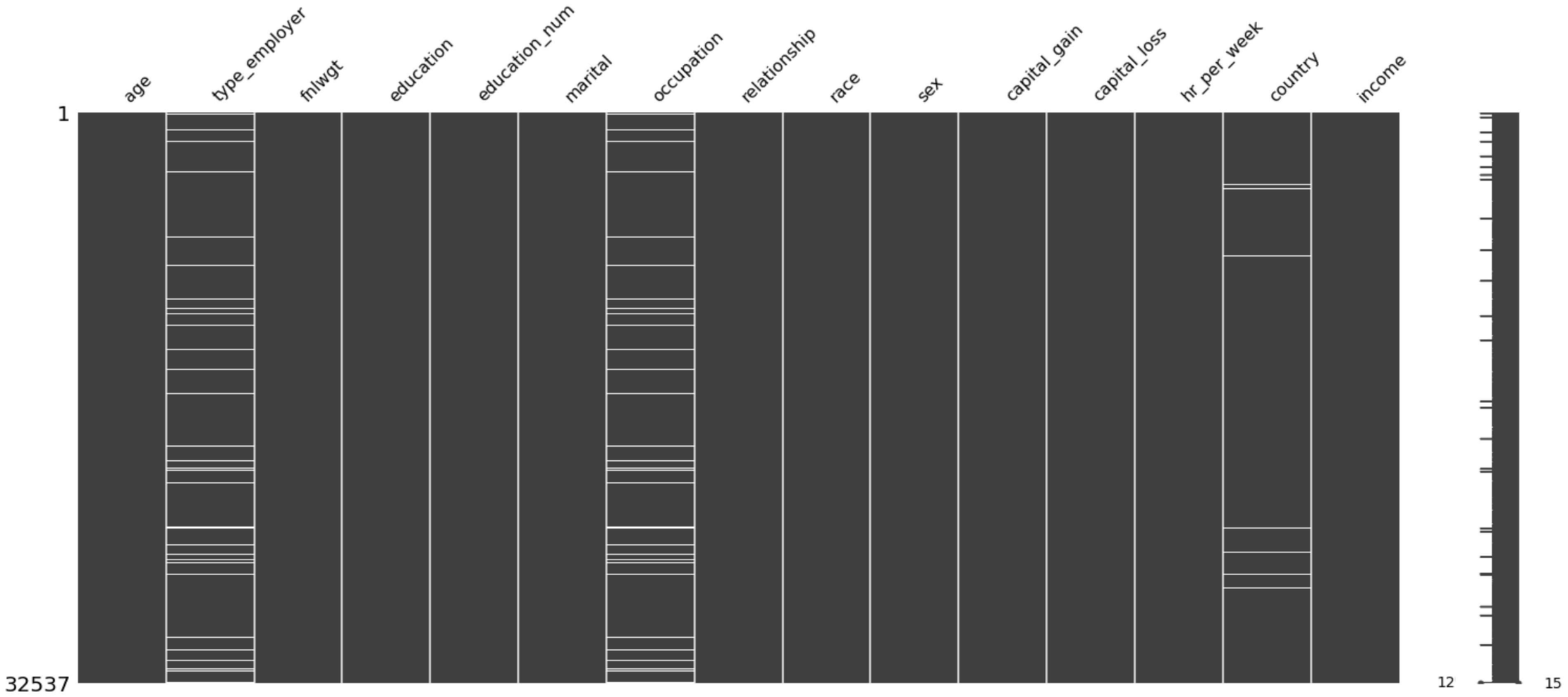
In [17]:
```
dup_df_class = df_class[df_class.duplicated()]
dup_df_class
```

Out[17]:

age	type_employer	fnlwgt	education	education_num	marital	occupation	relationship	race	sex	capital_gain	capital_loss	hr_per_week	country	income

In [18]:
```
import missingno as msno
msno.matrix(df_class)
```

Out[18]: <AxesSubplot:>



In [19]:
```
df_class.columns
```

Out[19]:
```
Index(['age', 'type_employer', 'fnlwgt', 'education', 'education_num',
       'marital', 'occupation', 'relationship', 'race', 'sex', 'capital_gain',
       'capital_loss', 'hr_per_week', 'country', 'income'],
      dtype='object')
```

```
# Exploratory Data Analysis
# Target variable: Segmentation (A,B,C,D) , Potential Predictors: All Others
```

```
# Univariate Analysis
```

In [20]:
```
df_class['country'].value_counts()
```

Out[20]:
```
United-States                 29153
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
Guatemala                        62
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
Greece                           29
France                           29
Ecuador                          28
Ireland                          24
Hong                             20
Cambodia                         19
Trinadad&Tobago                  19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Hungary                          13
Honduras                         13
Scotland                         12
Holand-Netherlands                1
Name: country, dtype: int64
```

In [21]:
```
df_class['occupation'].value_counts()
```

Out[21]:
```
Prof-specialty       4136
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
```

In [22]:
```
df_class['type_employer'].value_counts()
```

Out[22]:
```
Private             22673
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: type_employer, dtype: int64
```

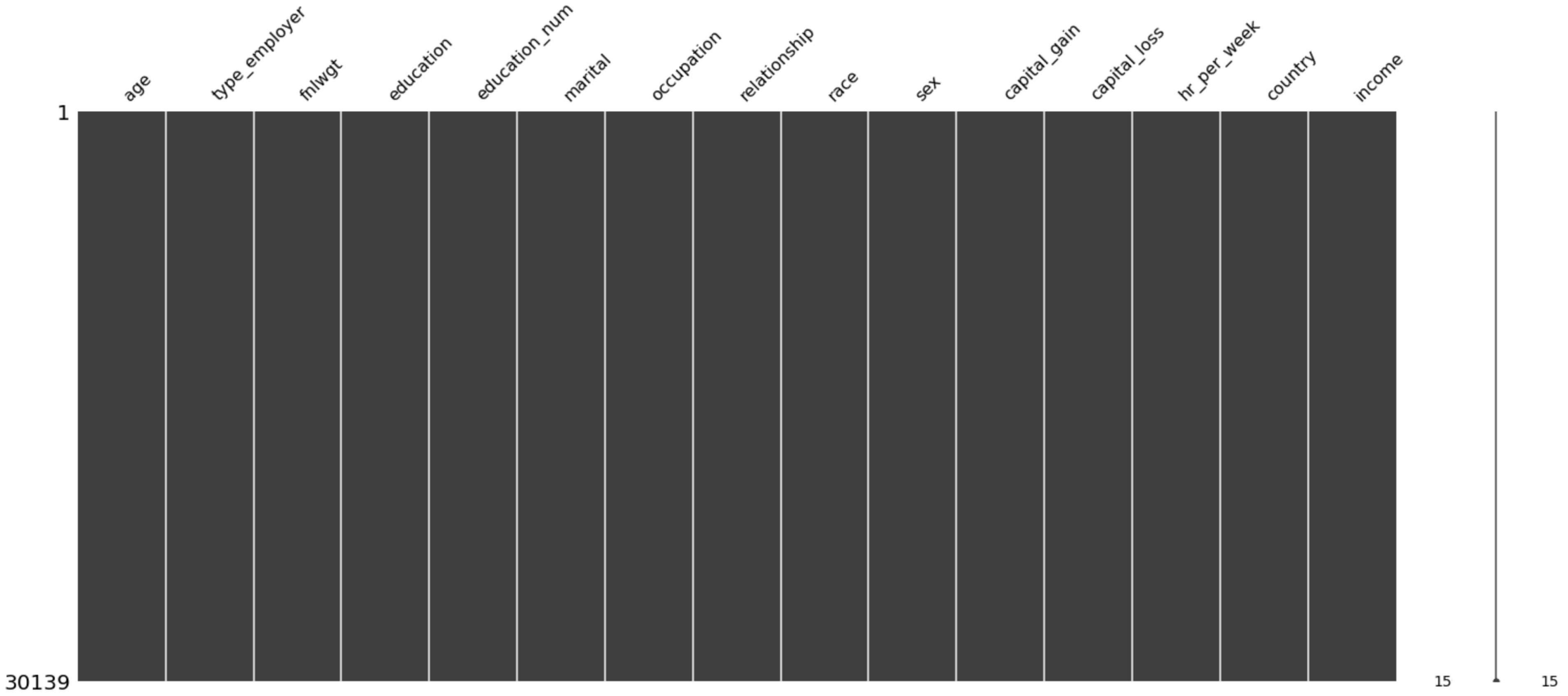
In [23]:
```
df_class['income'].value_counts()
```

Out[23]:
```
<=50K    24698
>50K      7839
Name: income, dtype: int64
```

In [24]:
```
df_class.dropna(axis=0, how='any',inplace=True)
```

In [25]:
```
msno.matrix(df_class)
```

Out[25]: <AxesSubplot:>



In [26]:
```
df_class.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30139 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            30139 non-null  int64 
 1   type_employer  30139 non-null  object
 2   fnlwgt         30139 non-null  int64 
 3   education      30139 non-null  object
 4   education_num  30139 non-null  int64 
 5   marital        30139 non-null  object
 6   occupation     30139 non-null  object
 7   relationship   30139 non-null  object
 8   race           30139 non-null  object
 9   sex            30139 non-null  object
 10  capital_gain   30139 non-null  int64 
 11  capital_loss   30139 non-null  int64 
 12  hr_per_week    30139 non-null  int64 
 13  country        30139 non-null  object
 14  income         30139 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [27]:
```
df_class['type_employer'].value_counts()
```

Out[27]:
```
Private             22264
Self-emp-not-inc     2498
Local-gov            2067
State-gov            1279
Self-emp-inc         1074
Federal-gov           943
Without-pay            14
Name: type_employer, dtype: int64
```

In [28]:
```
df_class.type_employer = df_class.type_employer.map(
                    {
                        'Local-gov': 'SL-gov',
                        'State-gov': 'SL-gov',
                        'Self-emp-inc': 'self-emp',
                        'Self-emp-not-inc': 'self-emp',
                        'Federal-gov': 'Federal-gov',
                        'Private': 'Private',
                        'Without-pay': 'Without-pay'

                    }
                    )
```

In [29]:
```
df_class['type_employer'].value_counts()
```

Out[29]:
```
Private        22264
self-emp        3572
SL-gov          3346
Federal-gov      943
Without-pay       14
Name: type_employer, dtype: int64
```

In [30]:
```
df_class['education'].value_counts()
```

Out[30]:
```
HS-grad         9834
Some-college    6669
Bachelors       5042
Masters         1626
Assoc-voc       1307
11th            1048
Assoc-acdm      1008
10th             820
7th-8th          556
Prof-school      542
9th              455
12th             377
Doctorate        375
5th-6th          287
1st-4th          149
Preschool         44
Name: education, dtype: int64
```

In [31]:
```
df_class['education_num'].value_counts()
```

Out[31]:
```
9     9834
10    6669
13    5042
14    1626
11    1307
7     1048
12    1008
6      820
4      556
15     542
5      455
8      377
16     375
3      287
2      149
1       44
Name: education_num, dtype: int64
```

In [32]:
```
df_class.head(10)
```

Out[32]:

	age	type_employer	fnlwgt	education	education_num	marital	occupation	relationship	race	sex	capital_gain	capital_loss	hr_per_week	country	income
0	39	SL-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	self-emp	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
7	52	self-emp	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K

In [33]:
```
df_class['marital'].value_counts()
```

Out[33]:
```
Married-civ-spouse       14059
Never-married             9711
Divorced                  4212
Separated                  939
Widowed                    827
Married-spouse-absent      370
Married-AF-spouse           21
Name: marital, dtype: int64
```

In [34]:
```
df_class.marital = df_class.marital.map(
                    {
                        'Separated': 'Not-Married',
                        'Divorced': 'Not-Married',
                        'Widowed': 'Not-Married',
                        'Never-married': 'Never-married',
                        'Married-civ-spouse': 'Married',
                        'Married-spouse-absent': 'Married',
                        'Married-AF-spouse': 'Married'

                    }
                    )
```

In [35]:
```
df_class['marital'].value_counts()
```

Out[35]:
```
Married          14450
Never-married     9711
Not-Married       5978
Name: marital, dtype: int64
```

In [36]:
```
df_class['country'].value_counts()
```

Out[36]:
```
United-States                 27487
Mexico                          606
Philippines                     188
Germany                         128
Puerto-Rico                     109
Canada                          107
El-Salvador                     100
India                           100
Cuba                             92
England                          86
Jamaica                          80
South                            71
China                            68
Italy                            68
Dominican-Republic               67
Vietnam                          64
Guatemala                        61
Japan                            59
Poland                           56
Columbia                         56
Taiwan                           42
Iran                             42
Haiti                            42
Portugal                         34
Nicaragua                        33
Peru                             30
Greece                           29
France                           27
Ecuador                          27
Ireland                          24
Hong                             19
Cambodia                         18
Trinadad&Tobago                  18
Laos                             17
Thailand                         17
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Hungary                          13
Honduras                         12
Scotland                         11
Holand-Netherlands                1
Name: country, dtype: int64
```

In [37]:
```
LLL1 = ['China','Hong','India','Iran','Cambodia','Japan', 'Laos' , 'Philippines' ,'Vietnam' ,'Taiwan', 'Thailand']
DD1 = dict.fromkeys(LLL1, 'Asia')
LLL2 = ['Canada','United-States','Puerto-Rico']
DD2 = dict.fromkeys(LLL2, 'North.America')
LLL3 = ['England' ,'France', 'Germany' ,'Greece','Holand-Netherlands','Hungary','Ireland','Italy','Poland','Portugal','Scotland','Yugoslavia']
DD3 = dict.fromkeys(LLL3, 'Europe')
LLL4 = ['Columbia','Cuba','Dominican-Republic','Ecuador','El-Salvador','Guatemala','Haiti','Honduras','Mexico','Nicaragua','Outlying-US(Guam-USVI-etc)','Peru',
DD4 = dict.fromkeys(LLL4, 'Latin.and.South.America')
LLL5 = ['South']
DD5 = dict.fromkeys(LLL5, 'South')
DDD = {**DD1, **DD2, **DD3, **DD4, **DD5}
print(DDD)
```

```
{'China': 'Asia', 'Hong': 'Asia', 'India': 'Asia', 'Iran': 'Asia', 'Cambodia': 'Asia', 'Japan': 'Asia', 'Laos': 'Asia', 'Philippines': 'Asia', 'Vietnam': 'Asi
a', 'Taiwan': 'Asia', 'Thailand': 'Asia', 'Canada': 'North.America', 'United-States': 'North.America', 'Puerto-Rico': 'North.America', 'England': 'Europe', 'F
rance': 'Europe', 'Germany': 'Europe', 'Greece': 'Europe', 'Holand-Netherlands': 'Europe', 'Hungary': 'Europe', 'Ireland': 'Europe', 'Italy': 'Europe', 'Polan
d': 'Europe', 'Portugal': 'Europe', 'Scotland': 'Europe', 'Yugoslavia': 'Europe', 'Columbia': 'Latin.and.South.America', 'Cuba': 'Latin.and.South.America', 'D
ominican-Republic': 'Latin.and.South.America', 'Ecuador': 'Latin.and.South.America', 'El-Salvador': 'Latin.and.South.America', 'Guatemala': 'Latin.and.South.A
merica', 'Haiti': 'Latin.and.South.America', 'Honduras': 'Latin.and.South.America', 'Mexico': 'Latin.and.South.America', 'Nicaragua': 'Latin.and.South.Americ
a', 'Outlying-US(Guam-USVI-etc)': 'Latin.and.South.America', 'Peru': 'Latin.and.South.America', 'Jamaica': 'Latin.and.South.America', 'Trinadad&Tobago': 'Lati
n.and.South.America', 'South': 'South'}
```

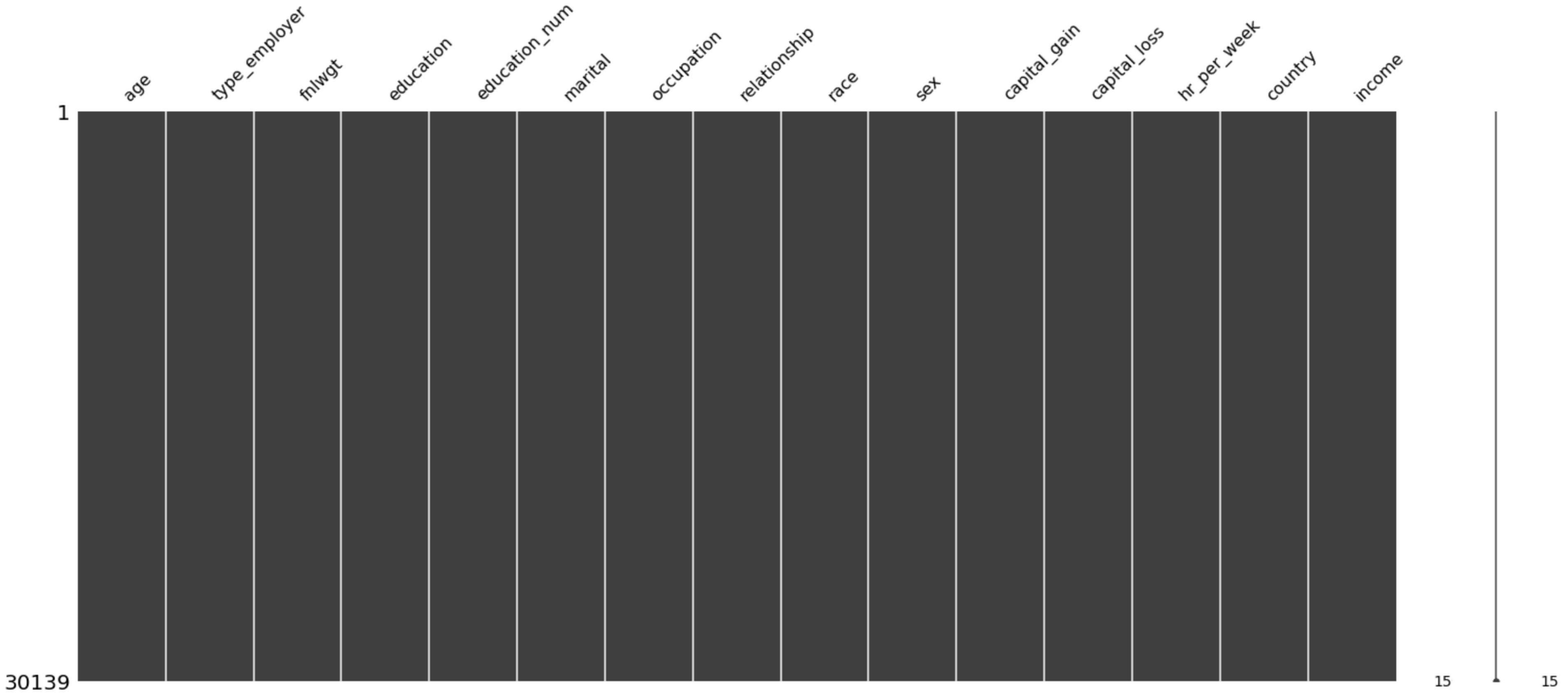
In [38]:
```
df_class['country'] = df_class['country'].map(DDD)
```

In [39]:
```
df_class['country'].value_counts()
```

Out[39]:
```
North.America             27703
Latin.and.South.America    1238
Asia                        634
Europe                      493
South                        71
Name: country, dtype: int64
```

In [40]:
```
msno.matrix(df_class)
```

Out[40]: <AxesSubplot:>

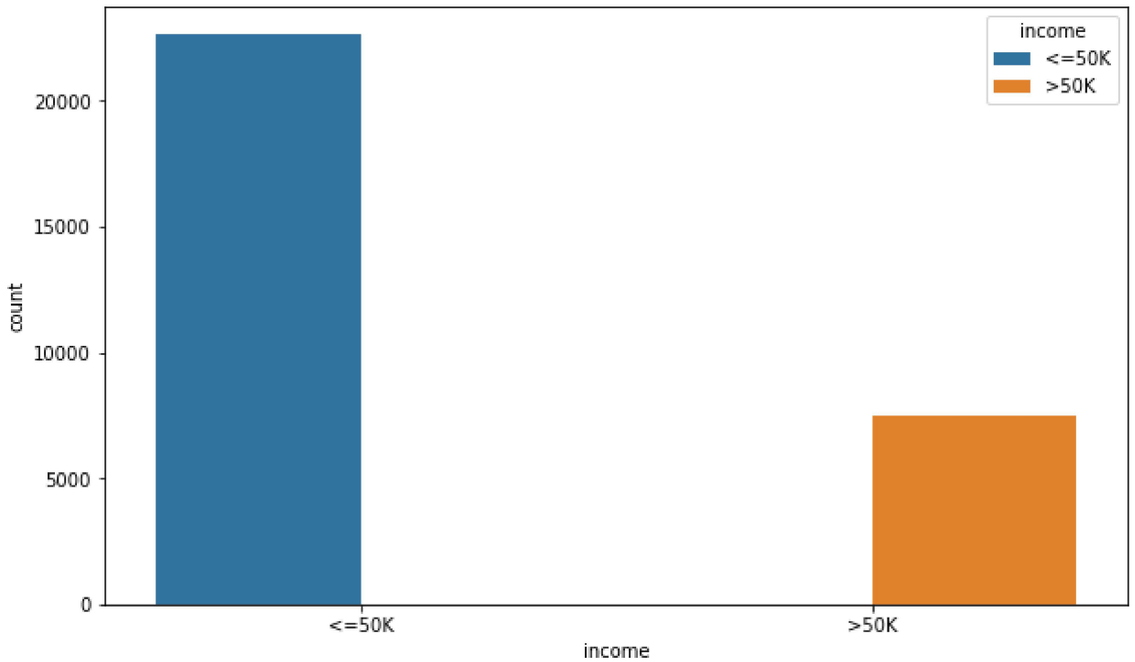


What is the distribution of target?

Our target here is income and since it is a categorical variable, in univaraite analysis for summarization we will find frequency and for visualization I would like to plot: pie chart or barchart.

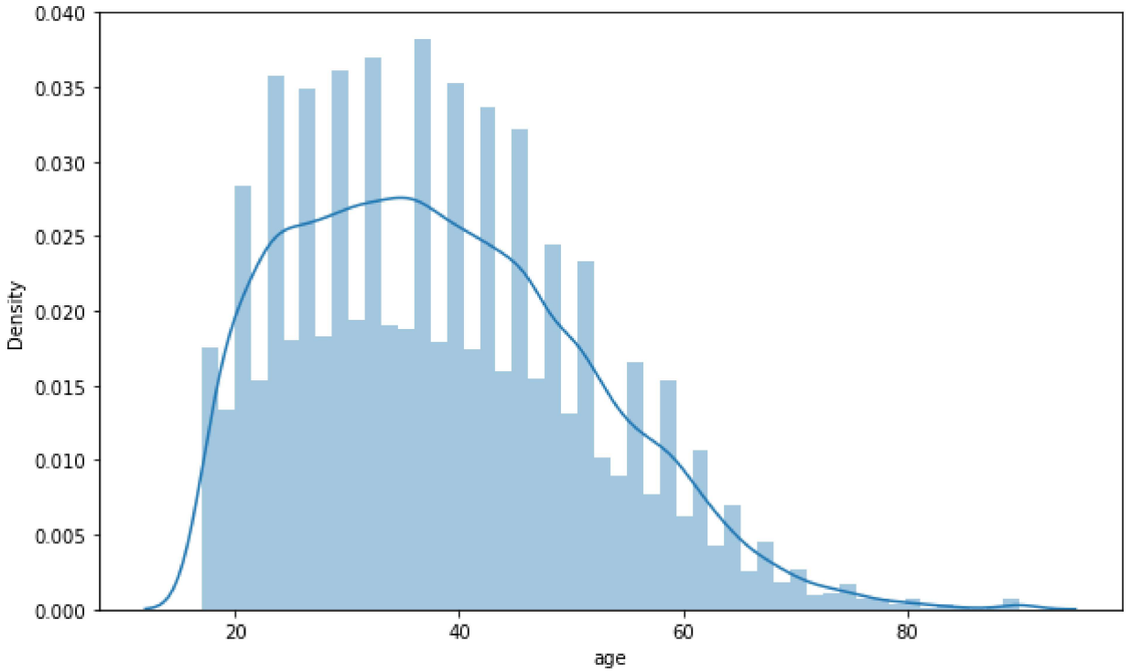
In [41]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['income'],hue=df_class['income'])
```

Out[41]: <AxesSubplot:xlabel='income', ylabel='count'>



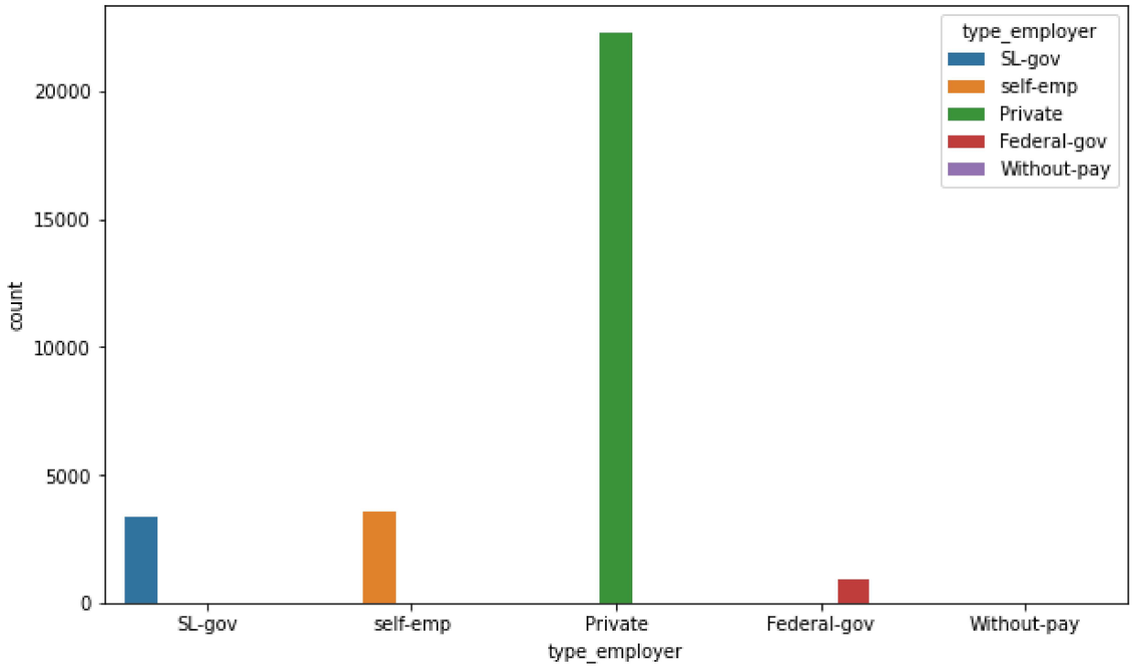
In [42]:
```
#Age
plt.rcParams['figure.figsize'] = (10, 6)
sns.distplot(df_class['age'])
```

Out[42]: <AxesSubplot:xlabel='age', ylabel='Density'>



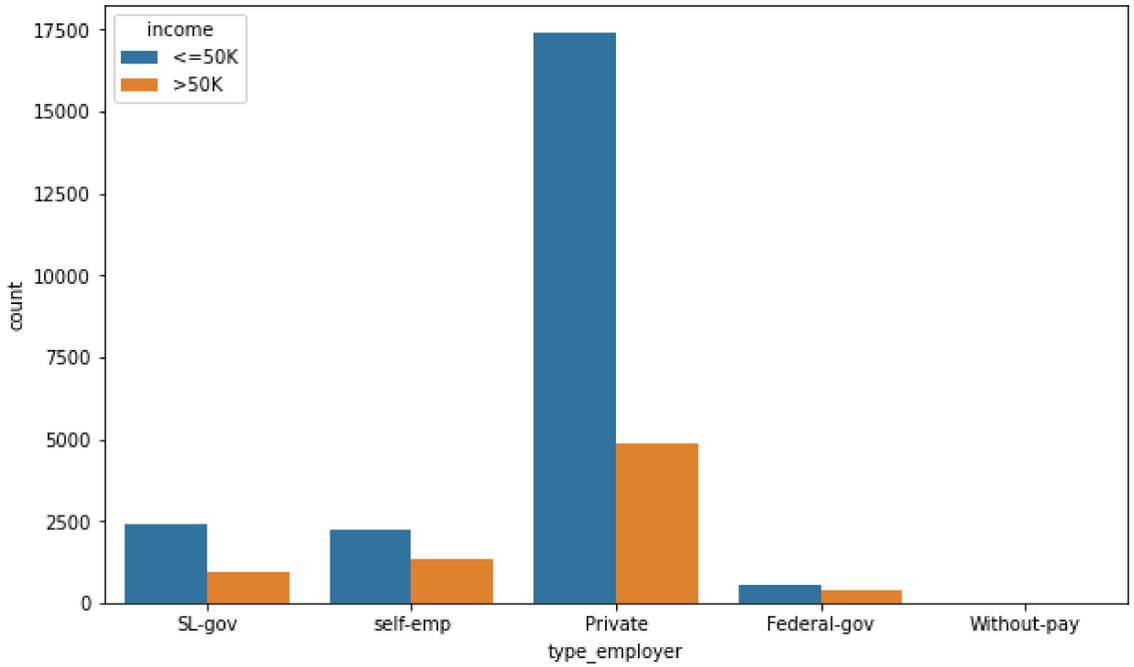
In [43]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['type_employer'],hue=df_class['type_employer'])
```

Out[43]: <AxesSubplot:xlabel='type_employer', ylabel='count'>



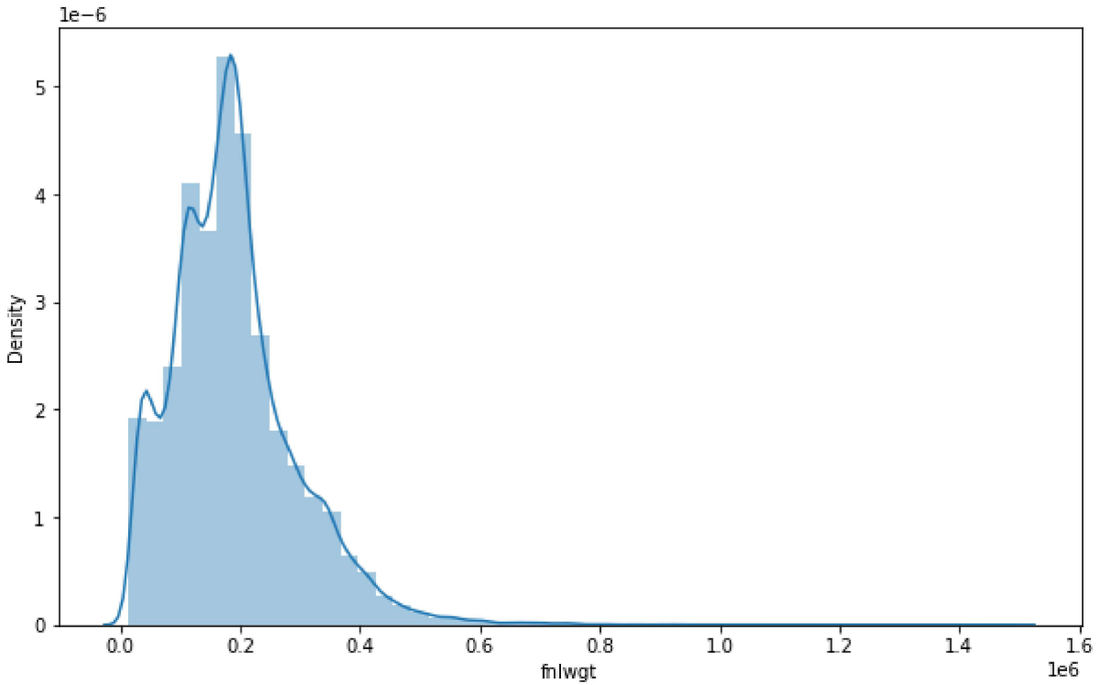
In [44]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['type_employer'],hue=df_class['income'])
```

Out[44]:
```
<AxesSubplot:xlabel='type_employer', ylabel='count'>
```



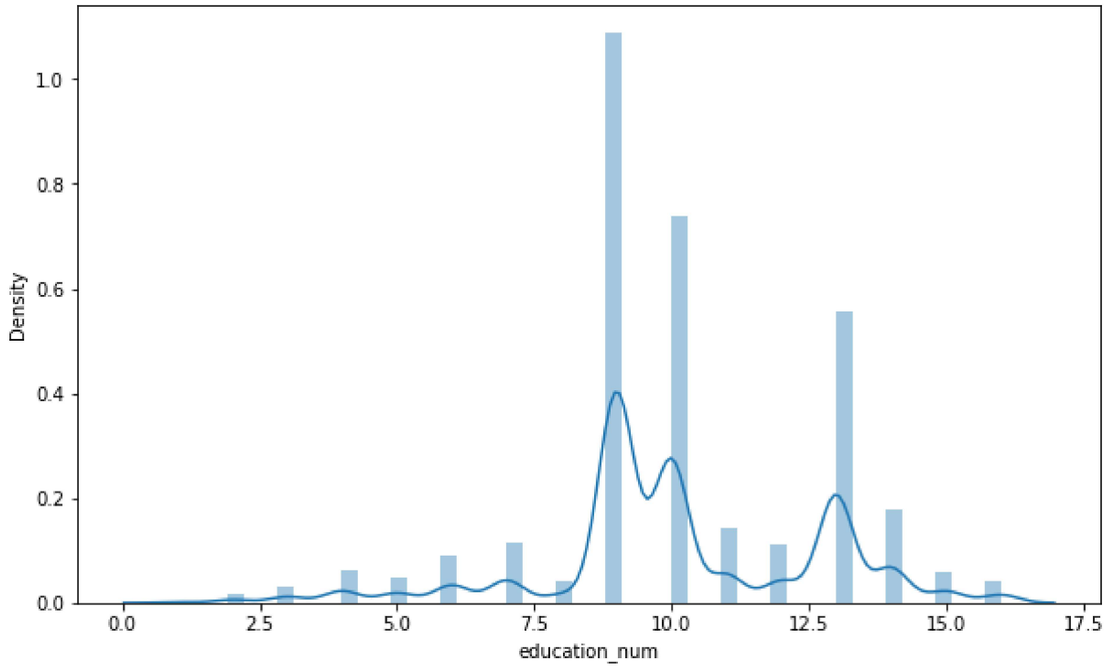
In [45]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.distplot(df_class['fnlwgt'])
```

Out[45]:
```
<AxesSubplot:xlabel='fnlwgt', ylabel='Density'>
```



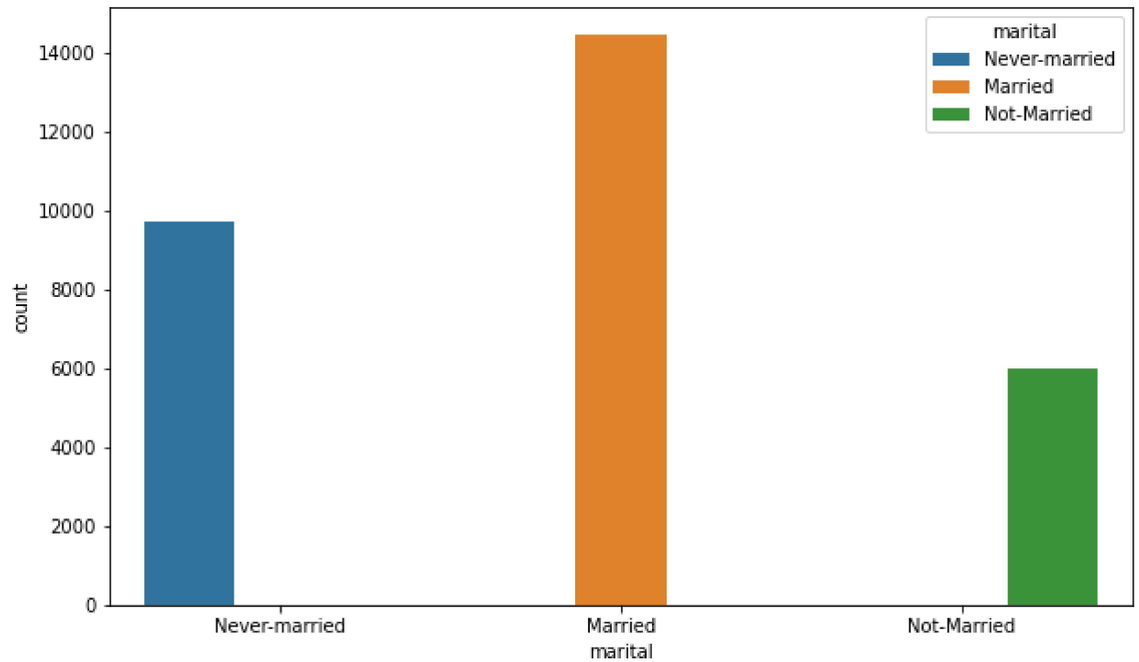
In [46]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.distplot(df_class['education_num'])
```

Out[46]:
```
<AxesSubplot:xlabel='education_num', ylabel='Density'>
```



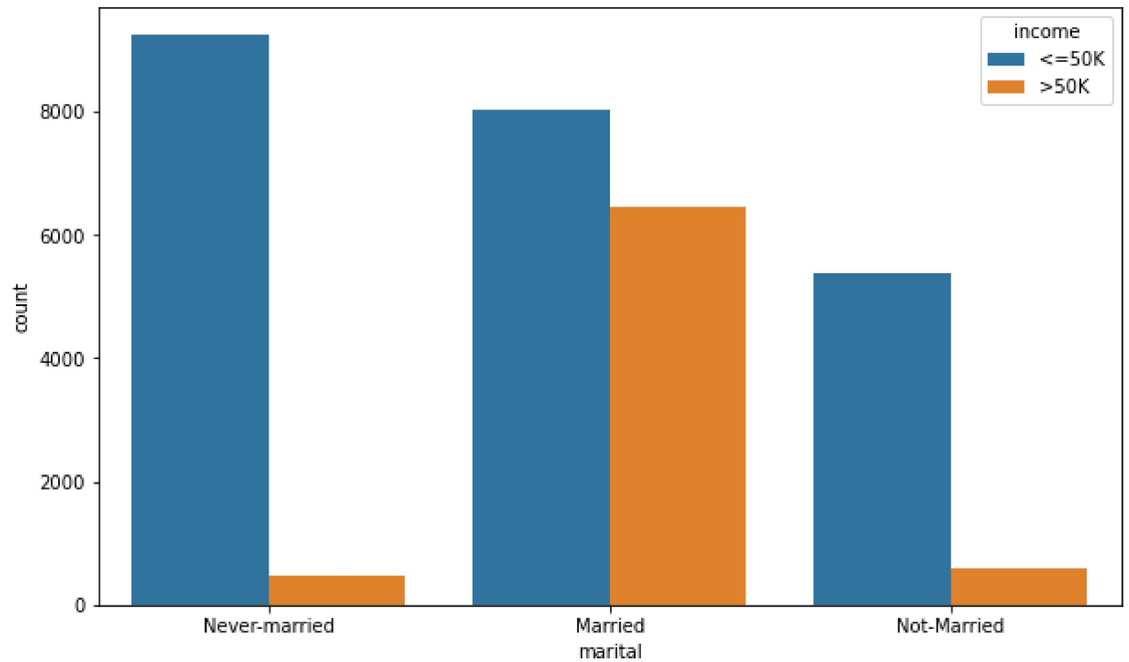
In [47]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['marital'],hue=df_class['marital'])
```

Out[47]: <AxesSubplot:xlabel='marital', ylabel='count'>



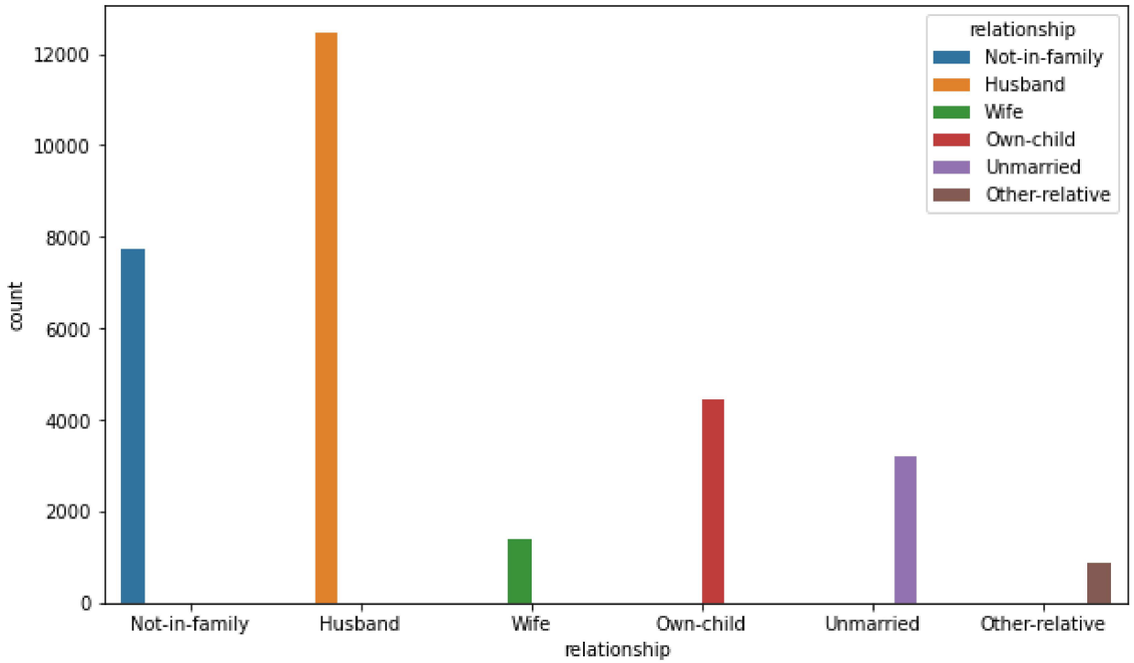
In [48]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['marital'],hue=df_class['income'])
```

Out[48]:
```
<AxesSubplot:xlabel='marital', ylabel='count'>
```



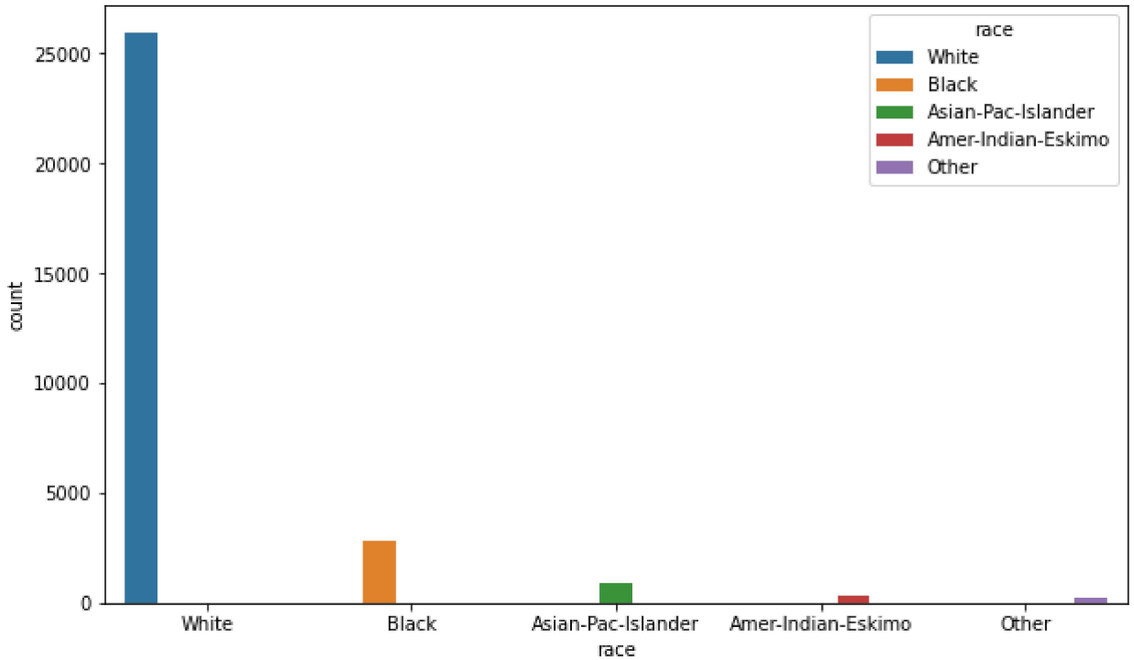
In [49]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['relationship'],hue=df_class['relationship'])
```

Out[49]: <AxesSubplot:xlabel='relationship', ylabel='count'>



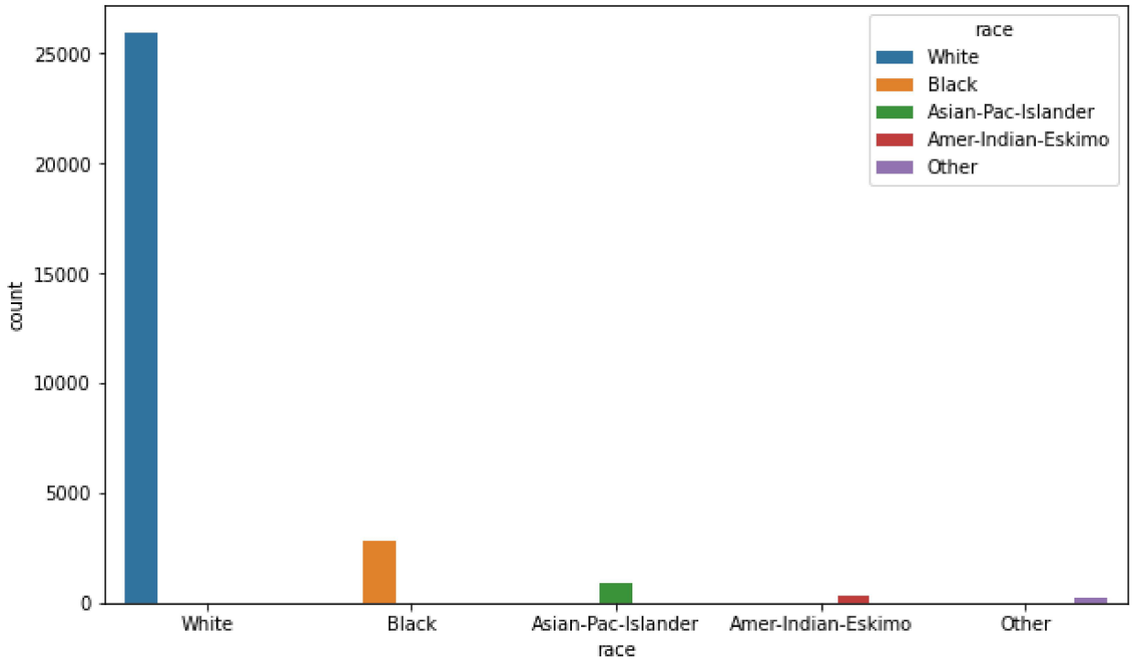
In [50]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['race'],hue=df_class['race'])
```

Out[50]:
```
<AxesSubplot:xlabel='race', ylabel='count'>
```



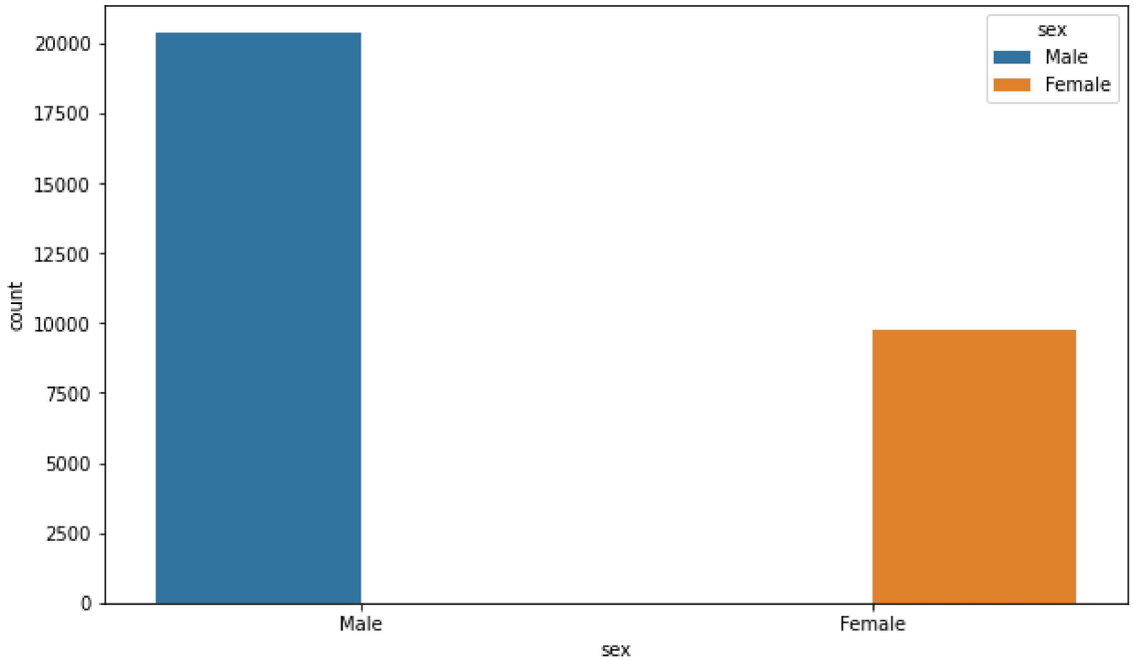
In [51]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['race'],hue=df_class['race'])
```

Out[51]: <AxesSubplot:xlabel='race', ylabel='count'>



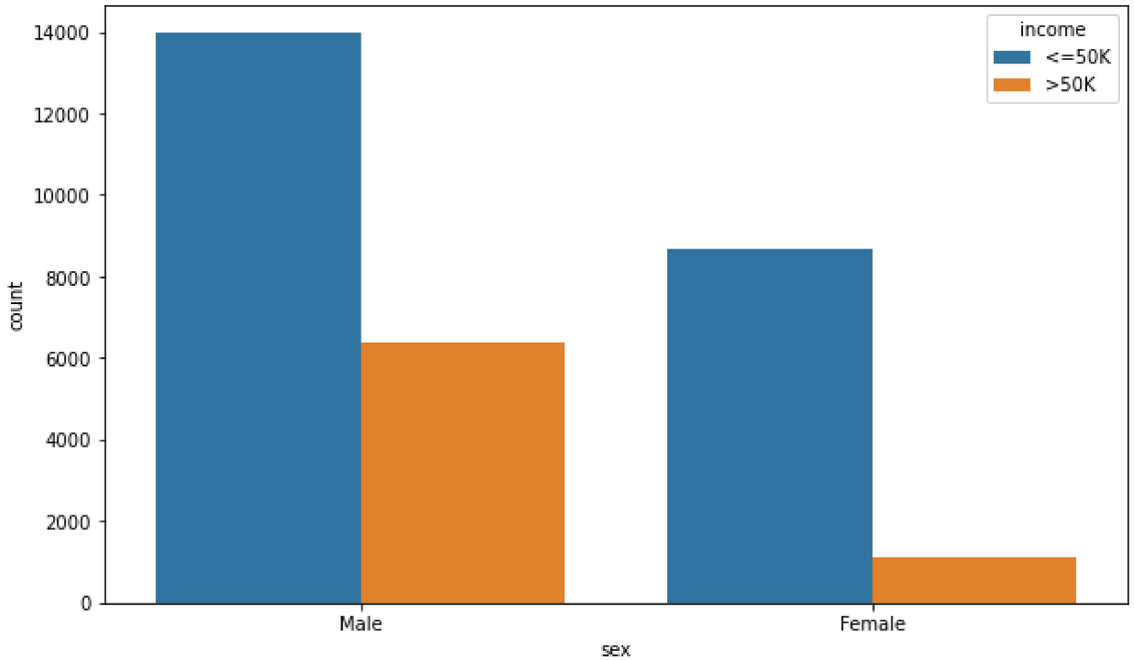
In [52]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['sex'],hue=df_class['sex'])
```

Out[52]: <AxesSubplot:xlabel='sex', ylabel='count'>



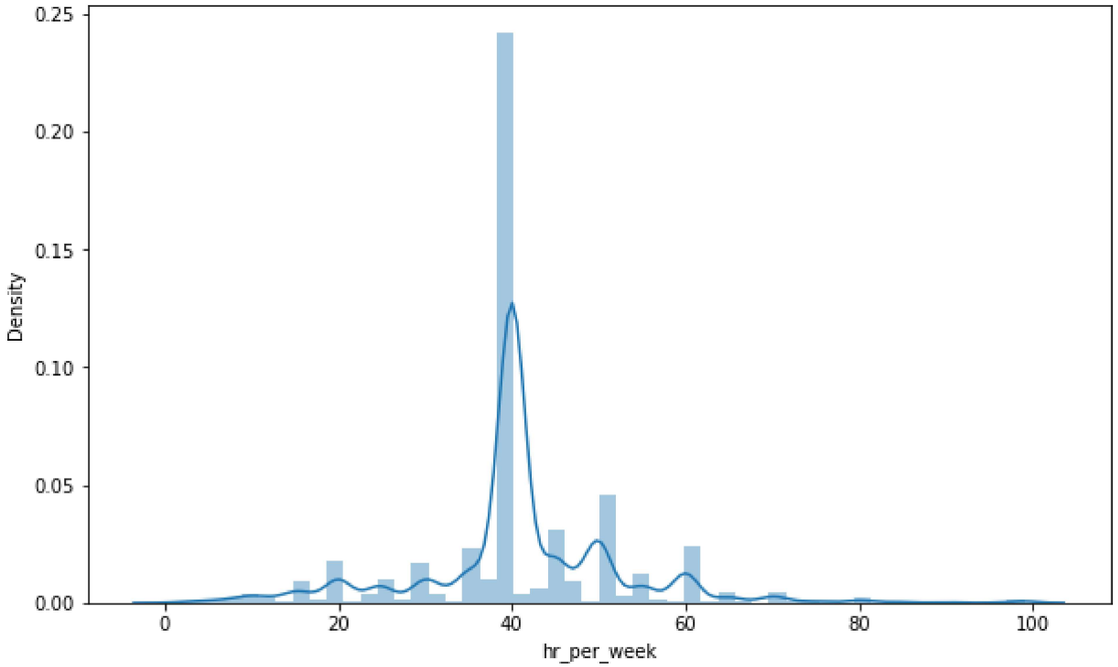
In [53]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['sex'],hue=df_class['income'])
```

Out[53]:
```
<AxesSubplot:xlabel='sex', ylabel='count'>
```



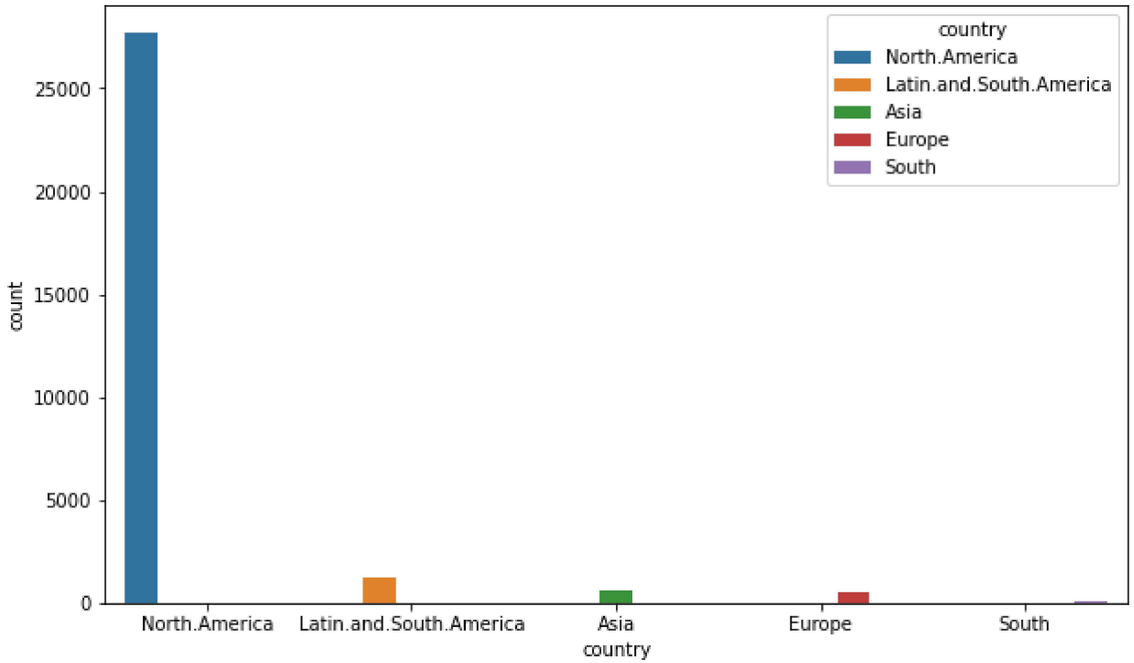
In [54]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.distplot(df_class['hr_per_week'])
```

Out[54]:
```
<AxesSubplot:xlabel='hr_per_week', ylabel='Density'>
```



In [55]:
```
plt.rcParams['figure.figsize'] = (10, 6)
sns.countplot(df_class['country'],hue=df_class['country'])
```

Out[55]:
```
<AxesSubplot:xlabel='country', ylabel='count'>
```

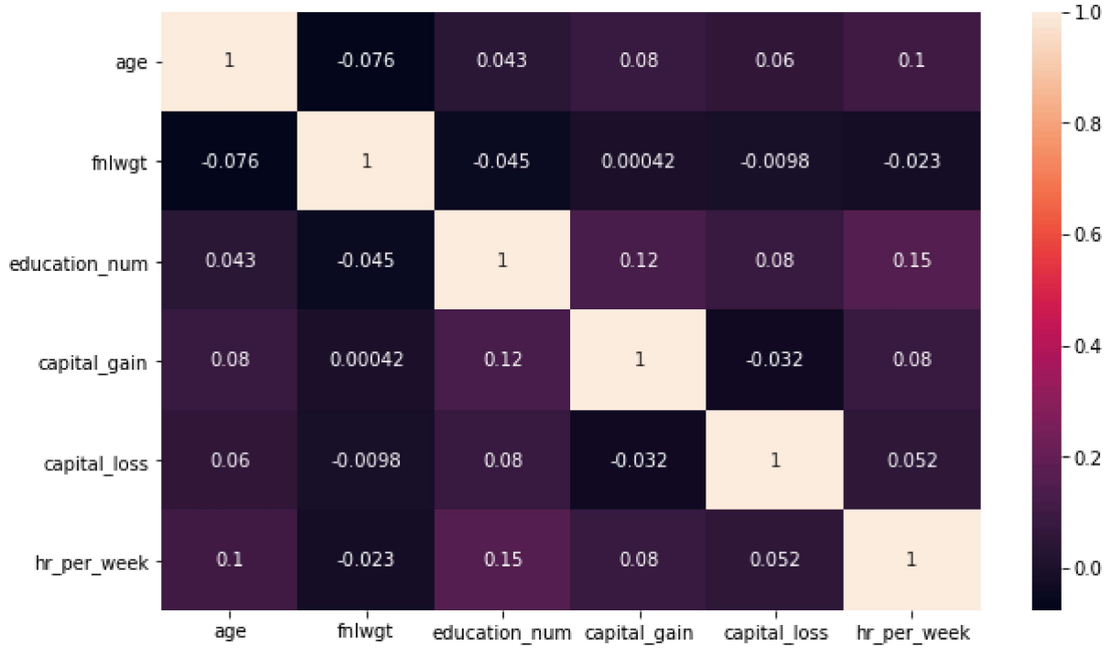



```
# A,B,C have higher Married people vs D has more Single people
```

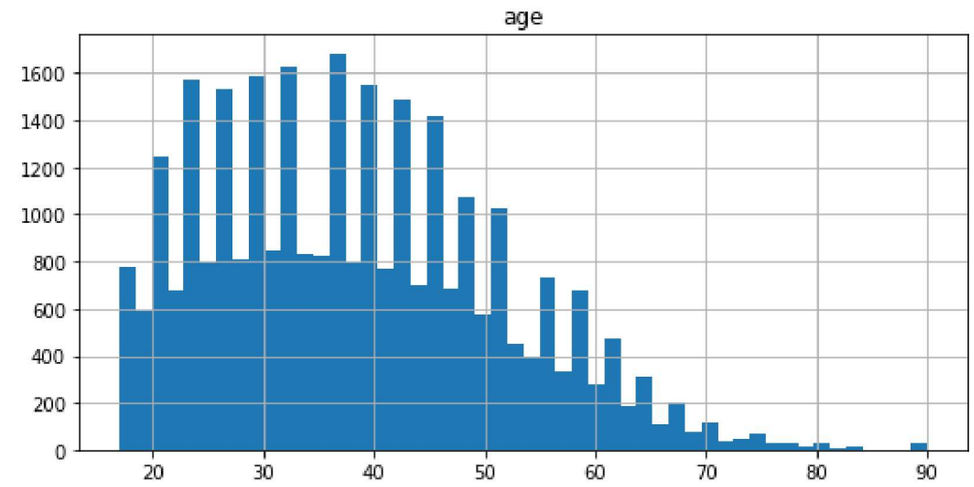
```
# Bivariate Analysis
```

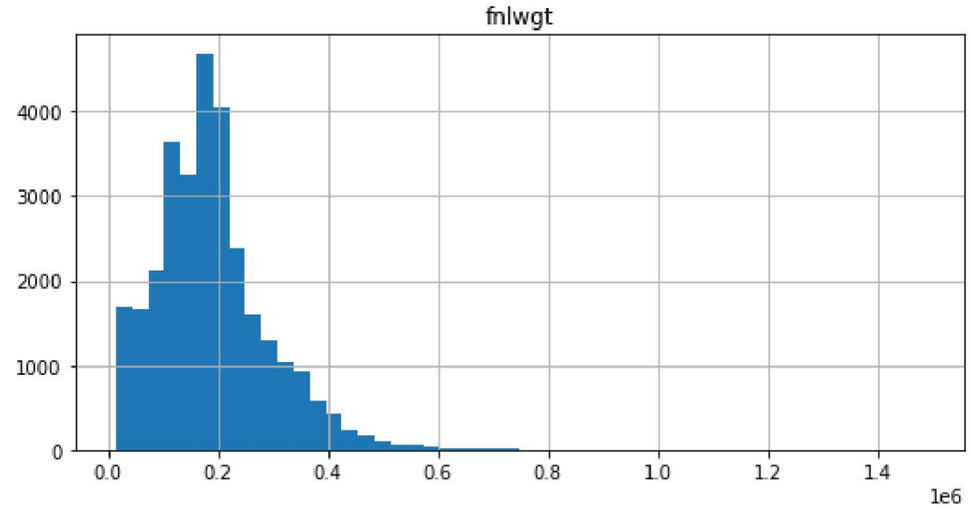
In [56]:
```
sns.heatmap(df_class.corr(),annot=True)
```

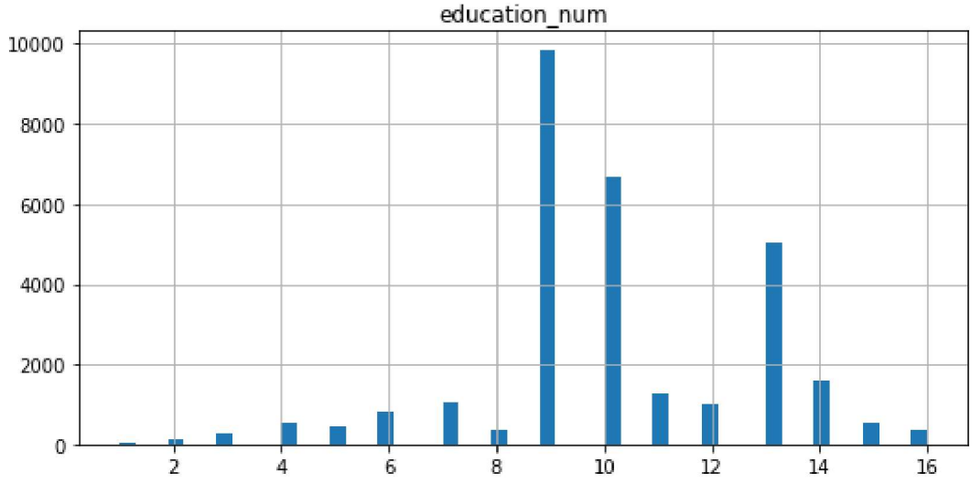
Out[56]: <AxesSubplot:>

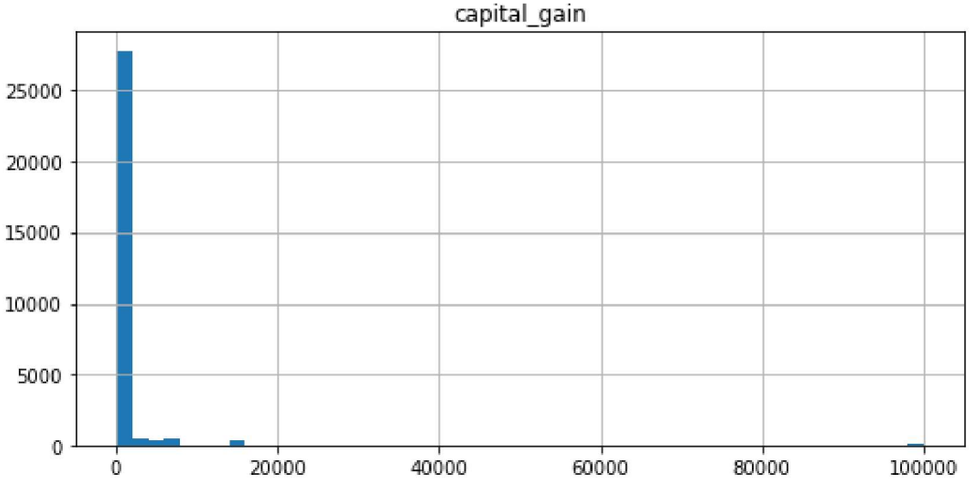


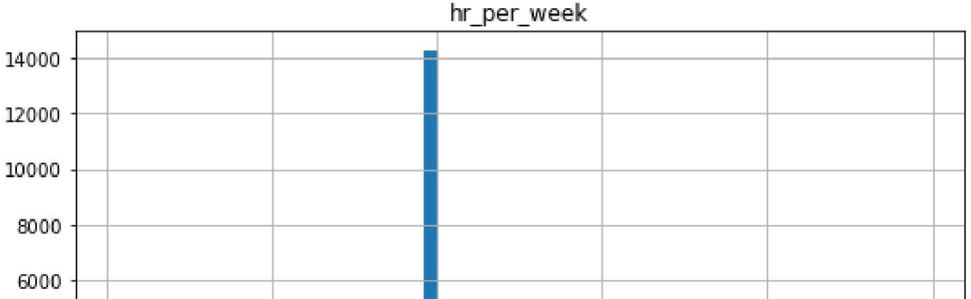
In [57]:
```
df_class.hist(bins=50, figsize=(20,15))
plt.show()
```

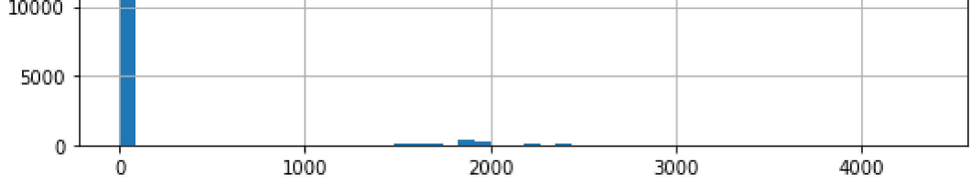


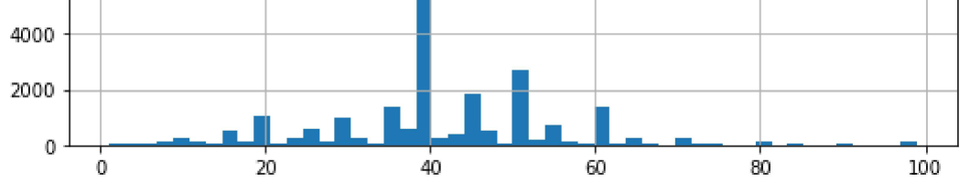












In [58]:
```
df_class.head(10)
```

Out[58]:

	age	type_employer	fnlwgt	education	education_num	marital	occupation	relationship	race	sex	capital_gain	capital_loss	hr_per_week	country	income
0	39	SL-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	North.America	<=50K
1	50	self-emp	83311	Bachelors	13	Married	Exec-managerial	Husband	White	Male	0	0	13	North.America	<=50K
2	38	Private	215646	HS-grad	9	Not-Married	Handlers-cleaners	Not-in-family	White	Male	0	0	40	North.America	<=50K
3	53	Private	234721	11th	7	Married	Handlers-cleaners	Husband	Black	Male	0	0	40	North.America	<=50K
4	28	Private	338409	Bachelors	13	Married	Prof-specialty	Wife	Black	Female	0	0	40	Latin.and.South.America	<=50K
5	37	Private	284582	Masters	14	Married	Exec-managerial	Wife	White	Female	0	0	40	North.America	<=50K
6	49	Private	160187	9th	5	Married	Other-service	Not-in-family	Black	Female	0	0	16	Latin.and.South.America	<=50K
7	52	self-emp	209642	HS-grad	9	Married	Exec-managerial	Husband	White	Male	0	0	45	North.America	>50K
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	North.America	>50K
9	42	Private	159449	Bachelors	13	Married	Exec-managerial	Husband	White	Male	5178	0	40	North.America	>50K

```
Let us drop education column since education and education_num seems to convey same information
```

In [59]:
```
df_class = df_class.drop(['education'],axis=1)
```

In [60]:
```
df_class.head(5)
```

Out[60]:

	age	type_employer	fnlwgt	education_num	marital	occupation	relationship	race	sex	capital_gain	capital_loss	hr_per_week	country	income
0	39	SL-gov	77516	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	North.America	<=50K
1	50	self-emp	83311	13	Married	Exec-managerial	Husband	White	Male	0	0	13	North.America	<=50K
2	38	Private	215646	9	Not-Married	Handlers-cleaners	Not-in-family	White	Male	0	0	40	North.America	<=50K
3	53	Private	234721	7	Married	Handlers-cleaners	Husband	Black	Male	0	0	40	North.America	<=50K
4	28	Private	338409	13	Married	Prof-specialty	Wife	Black	Female	0	0	40	Latin.and.South.America	<=50K

```
# Feature Engineering & Missing Value Treatment
```

In [61]:
```
df_class['occupation'].value_counts()
```

Out[61]:
```
Prof-specialty       4034
Craft-repair         4025
Exec-managerial      3991
Adm-clerical         3719
Sales                3584
Other-service        3209
Machine-op-inspct    1964
Transport-moving     1572
Handlers-cleaners    1349
Farming-fishing       987
Tech-support          911
Protective-serv       644
Priv-house-serv       141
Armed-Forces            9
Name: occupation, dtype: int64
```

In [62]:
```
df_class.type_employer = df_class.type_employer.map(
                    {
                        'Private': 0,
                        'self-emp': 1,
                        'SL-gov': 2,
                        'Federal-gov': 3,
                        'Without-pay': 4
                    }
                    )
```

In [63]:
```
df_class.marital = df_class.marital.map(
                    {
                        'Married': 0,
                        'Never-married': 1,
                        'Not-Married': 2
                    }
                    )
```

In [64]:
```
df_class = pd.get_dummies(df_class, columns=["occupation","relationship","country", "race"],drop_first=True)
```

In [77]:
```
df_class.head()
```

Out[77]:

	age	type_employer	fnlwgt	education_num	marital	race	sex	capital_gain	capital_loss	hr_per_week	...	occupation_Transport-moving	relationship_Not-in-family	relationship_Other-relative	relationship_Own-child
0	39	2	77516	13	1	White	Male	2174	0	40	...	0	1	0	0
1	50	1	83311	13	0	White	Male	0	0	13	...	0	0	0	0
2	38	0	215646	9	2	White	Male	0	0	40	...	0	1	0	0
3	53	0	234721	7	0	Black	Male	0	0	40	...	0	0	0	0
4	28	0	338409	13	0	Black	Female	0	0	40	...	0	0	0	0

5 rows × 33 columns

In [65]:
```
df_class['sex'].value_counts()
```

Out[65]:
```
Male      20366
Female     9773
Name: sex, dtype: int64
```

In [66]:
```
df_class.sex = df_class.sex.map(
                    {
                        'Female': 0,
                        'Male': 1
                    }
                    )
```

In [67]:
```
df_class.head()
```

Out[67]:

	age	type_employer	fnlwgt	education_num	marital	sex	capital_gain	capital_loss	hr_per_week	income	...	relationship_Unmarried	relationship_Wife	country_Europe	country_Latin.and.South
0	39	2	77516	13	1	1	2174	0	40	<=50K	...	0	0	0	
1	50	1	83311	13	0	1	0	0	13	<=50K	...	0	0	0	
2	38	0	215646	9	2	1	0	0	40	<=50K	...	0	0	0	
3	53	0	234721	7	0	1	0	0	40	<=50K	...	0	0	0	
4	28	0	338409	13	0	0	0	0	40	<=50K	...	0	1	0	

5 rows × 36 columns

In [68]:
```
df_class['income'].value_counts()
```

Out[68]:
```
<=50K    22633
>50K      7506
Name: income, dtype: int64
```

In [69]:
```
df_class.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30139 entries, 0 to 32560
Data columns (total 36 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   age                              30139 non-null  int64
 1   type_employer                    30139 non-null  int64
 2   fnlwgt                           30139 non-null  int64
 3   education_num                    30139 non-null  int64
 4   marital                          30139 non-null  int64
 5   sex                              30139 non-null  int64
 6   capital_gain                     30139 non-null  int64
 7   capital_loss                     30139 non-null  int64
 8   hr_per_week                      30139 non-null  int64
 9   income                           30139 non-null  object
 10  occupation_Armed-Forces          30139 non-null  uint8
 11  occupation_Craft-repair          30139 non-null  uint8
 12  occupation_Exec-managerial       30139 non-null  uint8
 13  occupation_Farming-fishing       30139 non-null  uint8
 14  occupation_Handlers-cleaners     30139 non-null  uint8
 15  occupation_Machine-op-inspct     30139 non-null  uint8
 16  occupation_Other-service         30139 non-null  uint8
 17  occupation_Priv-house-serv       30139 non-null  uint8
 18  occupation_Prof-specialty        30139 non-null  uint8
 19  occupation_Protective-serv       30139 non-null  uint8
 20  occupation_Sales                 30139 non-null  uint8
 21  occupation_Tech-support          30139 non-null  uint8
 22  occupation_Transport-moving      30139 non-null  uint8
 23  relationship_Not-in-family       30139 non-null  uint8
 24  relationship_Other-relative      30139 non-null  uint8
 25  relationship_Own-child           30139 non-null  uint8
 26  relationship_Unmarried           30139 non-null  uint8
 27  relationship_Wife                30139 non-null  uint8
 28  country_Europe                   30139 non-null  uint8
 29  country_Latin.and.South.America  30139 non-null  uint8
 30  country_North.America            30139 non-null  uint8
 31  country_South                    30139 non-null  uint8
 32  race_Asian-Pac-Islander          30139 non-null  uint8
 33  race_Black                       30139 non-null  uint8
 34  race_Other                       30139 non-null  uint8
 35  race_White                       30139 non-null  uint8
dtypes: int64(9), object(1), uint8(26)
memory usage: 4.5+ MB
```

In [70]:
```
df_class.income = df_class.income.map(
                    {
                        '<=50K': 0,
                        '>50K': 1
                    }
                    )
```

In [71]:
```
df_class.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30139 entries, 0 to 32560
Data columns (total 36 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   age                              30139 non-null  int64
 1   type_employer                    30139 non-null  int64
 2   fnlwgt                           30139 non-null  int64
 3   education_num                    30139 non-null  int64
 4   marital                          30139 non-null  int64
 5   sex                              30139 non-null  int64
 6   capital_gain                     30139 non-null  int64
 7   capital_loss                     30139 non-null  int64
 8   hr_per_week                      30139 non-null  int64
 9   income                           30139 non-null  int64
 10  occupation_Armed-Forces          30139 non-null  uint8
 11  occupation_Craft-repair          30139 non-null  uint8
 12  occupation_Exec-managerial       30139 non-null  uint8
 13  occupation_Farming-fishing       30139 non-null  uint8
 14  occupation_Handlers-cleaners     30139 non-null  uint8
 15  occupation_Machine-op-inspct     30139 non-null  uint8
 16  occupation_Other-service         30139 non-null  uint8
 17  occupation_Priv-house-serv       30139 non-null  uint8
 18  occupation_Prof-specialty        30139 non-null  uint8
 19  occupation_Protective-serv       30139 non-null  uint8
 20  occupation_Sales                 30139 non-null  uint8
 21  occupation_Tech-support          30139 non-null  uint8
 22  occupation_Transport-moving      30139 non-null  uint8
 23  relationship_Not-in-family       30139 non-null  uint8
 24  relationship_Other-relative      30139 non-null  uint8
 25  relationship_Own-child           30139 non-null  uint8
 26  relationship_Unmarried           30139 non-null  uint8
 27  relationship_Wife                30139 non-null  uint8
 28  country_Europe                   30139 non-null  uint8
 29  country_Latin.and.South.America  30139 non-null  uint8
 30  country_North.America            30139 non-null  uint8
 31  country_South                    30139 non-null  uint8
 32  race_Asian-Pac-Islander          30139 non-null  uint8
 33  race_Black                       30139 non-null  uint8
 34  race_Other                       30139 non-null  uint8
 35  race_White                       30139 non-null  uint8
dtypes: int64(10), uint8(26)
memory usage: 4.5 MB
```

In [74]:
```
df_class.dtypes
```

Out[74]:
```
age                                int64
type_employer                      int64
fnlwgt                             int64
education_num                      int64
marital                            int64
sex                                int64
capital_gain                       int64
capital_loss                       int64
hr_per_week                        int64
income                             int64
occupation_Armed-Forces            uint8
occupation_Craft-repair            uint8
occupation_Exec-managerial         uint8
occupation_Farming-fishing         uint8
occupation_Handlers-cleaners       uint8
occupation_Machine-op-inspct       uint8
occupation_Other-service           uint8
occupation_Priv-house-serv         uint8
occupation_Prof-specialty          uint8
occupation_Protective-serv         uint8
occupation_Sales                   uint8
occupation_Tech-support            uint8
occupation_Transport-moving        uint8
relationship_Not-in-family         uint8
relationship_Other-relative        uint8
relationship_Own-child             uint8
relationship_Unmarried             uint8
relationship_Wife                  uint8
country_Europe                     uint8
country_Latin.and.South.America    uint8
country_North.America              uint8
country_South                      uint8
race_Asian-Pac-Islander            uint8
race_Black                         uint8
race_Other                         uint8
race_White                         uint8
dtype: object
```

In [75]:
```
df_class.shape
```

Out[75]:
```
(30139, 36)
```

In [76]:
```
X_fin = df_class.drop(['income'],axis=1)
y_fin = df_class.income
```

In [77]:
```
X_fin.head()
```

Out[77]:

	age	type_employer	fnlwgt	education_num	marital	sex	capital_gain	capital_loss	hr_per_week	occupation_Armed-Forces	...	relationship_Unmarried	relationship_Wife	country_Europe	country_Lat
0	39	2	77516	13	1	1	2174	0	40	0	...	0	0	0	
1	50	1	83311	13	0	1	0	0	13	0	...	0	0	0	
2	38	0	215646	9	2	1	0	0	40	0	...	0	0	0	
3	53	0	234721	7	0	1	0	0	40	0	...	0	0	0	
4	28	0	338409	13	0	0	0	0	40	0	...	0	1	0	

5 rows × 35 columns

In [78]:
```
y_fin.head()
```

Out[78]:
```
0    0
1    0
2    0
3    0
4    0
Name: income, dtype: int64
```

In [79]:
```
from sklearn.preprocessing import MinMaxScaler
sc= MinMaxScaler()
X_fins = pd.DataFrame(sc.fit_transform(X_fin),columns = X_fin.columns)
X_fins.head()
```

Out[79]:

	age	type_employer	fnlwgt	education_num	marital	sex	capital_gain	capital_loss	hr_per_week	occupation_Armed-Forces	...	relationship_Unmarried	relationship_Wife	country_Europe	coun
0	0.301370	0.50	0.043338	0.800000	0.5	1.0	0.02174	0.0	0.397959	0.0	...	0.0	0.0	0.0	
1	0.452055	0.25	0.047277	0.800000	0.0	1.0	0.00000	0.0	0.122449	0.0	...	0.0	0.0	0.0	
2	0.287671	0.00	0.137244	0.533333	1.0	1.0	0.00000	0.0	0.397959	0.0	...	0.0	0.0	0.0	
3	0.493151	0.00	0.150212	0.400000	0.0	1.0	0.00000	0.0	0.397959	0.0	...	0.0	0.0	0.0	
4	0.150685	0.00	0.220703	0.800000	0.0	0.0	0.00000	0.0	0.397959	0.0	...	0.0	1.0	0.0	

5 rows × 35 columns

In [166]:
```
X_fins.shape
```

Out[166]:
```
(30139, 35)
```

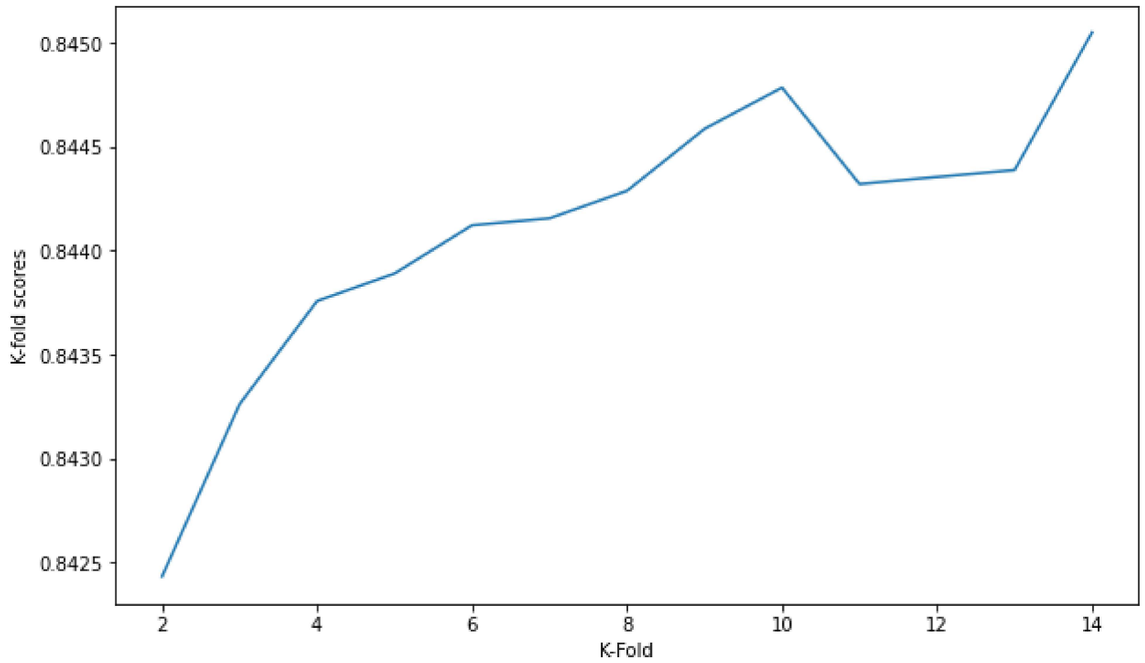
In [167]:
```
y_fin.shape
```

Out[167]:
```
(30139,)
```

In [89]:
```
#k-fold cross-validation score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
cross_val_score(LogisticRegression(max_iter=1000,tol=0.001),
                X_fins,y_fin,cv=10).mean()
```

Out[89]:
```
0.8447524782566852
```

In [86]:
```
#Graph k-fold score for Logistic Regression Classification

mlr_scores = []
for i in range(2,15):
    mlr_scores.append(cross_val_score(LogisticRegression(),X_fins,y_fin,cv=i).mean())



plt.plot(range(2,15),mlr_scores)
plt.xlabel('K-Fold')
plt.ylabel('K-fold scores')
plt.show()
```



In [87]:
```
from sklearn.model_selection import train_test_split
X_train_mlr, X_test_mlr, y_train_mlr, y_test_mlr = train_test_split(X_fins,
                                      y_fin,random_state=0,test_size=0.10)
```

In [88]:
```
from sklearn.metrics import classification_report, confusion_matrix
model_mlr = LogisticRegression()
model_mlr.fit(X_train_mlr,y_train_mlr)
y_pred_mlr = model_mlr.predict(X_test_mlr)
print(classification_report(y_test_mlr,y_pred_mlr))
```

```
              precision    recall  f1-score   support

           0       0.87      0.92      0.90      2248
           1       0.73      0.61      0.66       766

    accuracy                           0.84      3014
   macro avg       0.80      0.76      0.78      3014
weighted avg       0.84      0.84      0.84      3014
```

In [90]:
```
print(confusion_matrix(y_test_mlr,y_pred_mlr))
```

```
[[2077  171]
 [ 302  464]]
```

In [95]:
```
# Grid search cross validation
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
grid={"C":np.logspace(-3,3,7), "penalty":["l1","l2"]}# l1 lasso l2 ridge
logreg=LogisticRegression()
logreg_cv=GridSearchCV(logreg,grid,cv=10)
logreg_cv.fit(X_train_mlr,y_train_mlr)

print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
best_model_logreg = logreg_cv.best_estimator_
```

```
tuned hpyerparameters :(best parameters)  {'C': 100.0, 'penalty': 'l2'}
accuracy : 0.8471149915136016
```

In [96]:
```
best_model_logreg.fit(X_train_mlr,y_train_mlr)
y_pred_mlr_best = model_mlr.predict(X_test_mlr)
print(classification_report(y_test_mlr,y_pred_mlr_best))
```

```
              precision    recall  f1-score   support

           0       0.87      0.92      0.90      2248
           1       0.73      0.61      0.66       766

    accuracy                           0.84      3014
   macro avg       0.80      0.76      0.78      3014
weighted avg       0.84      0.84      0.84      3014
```

In [225]:
```
confusion_matrix(y_test_mlr,y_pred_mlr_best)
```

Out[225]:
```
array([[2077,  171],
       [ 302,  464]], dtype=int64)
```

```
################################### KNN ##############################################################
```

In [97]:
```
X_knnc = X_fins
X_knnc.head()
```

Out[97]:

	age	type_employer	fnlwgt	education_num	marital	sex	capital_gain	capital_loss	hr_per_week	occupation_Armed-Forces	...	relationship_Unmarried	relationship_Wife	country_Europe	coun
0	0.301370	0.50	0.043338	0.800000	0.5	1.0	0.02174	0.0	0.397959	0.0	...	0.0	0.0	0.0	
1	0.452055	0.25	0.047277	0.800000	0.0	1.0	0.00000	0.0	0.122449	0.0	...	0.0	0.0	0.0	
2	0.287671	0.00	0.137244	0.533333	1.0	1.0	0.00000	0.0	0.397959	0.0	...	0.0	0.0	0.0	
3	0.493151	0.00	0.150212	0.400000	0.0	1.0	0.00000	0.0	0.397959	0.0	...	0.0	0.0	0.0	
4	0.150685	0.00	0.220703	0.800000	0.0	0.0	0.00000	0.0	0.397959	0.0	...	0.0	1.0	0.0	

5 rows × 35 columns

In [98]:
```
y_knnc = y_fin
y_knnc.head()
```

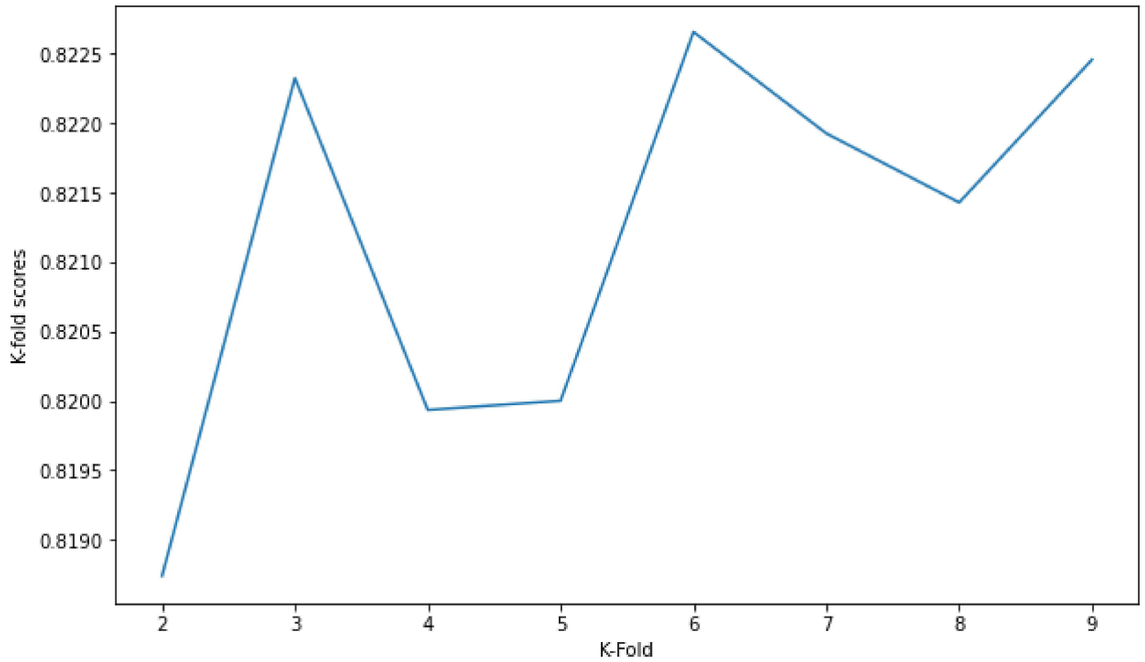
Out[98]:
```
0    0
1    0
2    0
3    0
4    0
Name: income, dtype: int64
```

In [99]:
```
#import the knn model
from sklearn.neighbors import KNeighborsClassifier
knnc = KNeighborsClassifier()
```

In [102]:
```
#see the cross_validated score for cv=4
from sklearn.model_selection import cross_val_score
cross_val_score(knnc,X_knnc,y_knnc,cv=6).mean()
```

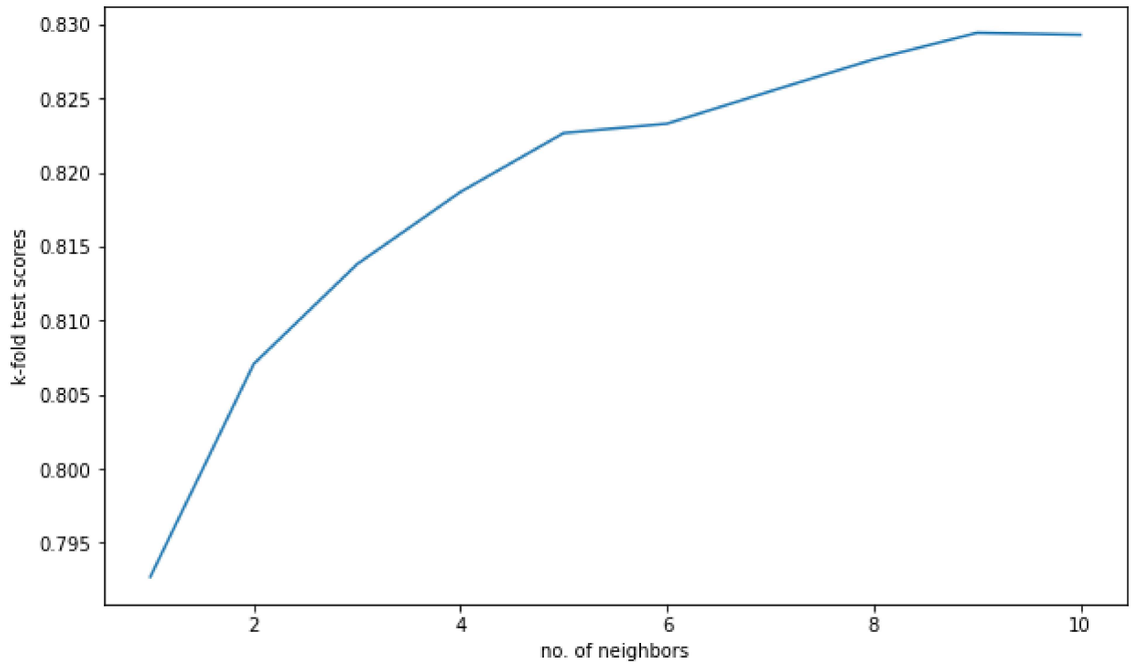
Out[102]:
```
0.8226553422198438
```

In [101]:
```
#Graph k-fold score for KNN

k_scores_mlr = []
for i in range(2,10):
    k_scores_mlr.append(cross_val_score(KNeighborsClassifier(),X_knnc,y_knnc,cv=i).mean())



plt.plot(range(2,10),k_scores_mlr)
plt.xlabel('K-Fold')
plt.ylabel('K-fold scores')
plt.show()
```



In [104]:
```
#for no.of neighbors from 1 - 10, graph the k-fold scores
scores_knnc = []
for i in range(1,11,1):
    knnc_p = KNeighborsClassifier(n_neighbors=i, weights='uniform')
    scores_knnc.append(cross_val_score(knnc_p,X_knnc,y_knnc,cv=6).mean())
```

In [105]:
```
import matplotlib.pyplot as plt
plt.plot(range(1,11,1),scores_knnc)
plt.xlabel('no. of neighbors')
plt.ylabel('k-fold test scores')
plt.show()
```



In [107]:
```
# Grid Search
from sklearn.model_selection import GridSearchCV
k_range = list(range(1, 16))
param_grid = dict(n_neighbors=k_range)
grid = GridSearchCV(knnc, param_grid, cv=6, scoring='accuracy')
grid.fit(X_knnc, y_knnc)
```

Out[107]:
```
GridSearchCV(cv=6, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                         13, 14, 15]},
             scoring='accuracy')
```

In [108]:
```
# view the complete results (list of named tuples)
grid.best_score_
```

Out[108]:
```
0.8311160910343206
```

In [109]:
```
from sklearn.model_selection import train_test_split
X_train_knnc, X_test_knnc, y_train_knnc, y_test_knnc = train_test_split(X_knnc,
                                        y_knnc,random_state=0,test_size=0.25)
```

In [110]:
```
from sklearn.metrics import classification_report, confusion_matrix
model_knnc = grid.best_estimator_
model_knnc.fit(X_train_knnc,y_train_knnc)
y_pred_knnc = model_knnc.predict(X_test_knnc)
print(classification_report(y_test_knnc,y_pred_knnc))
```

```
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      5655
           1       0.68      0.60      0.64      1880

    accuracy                           0.83      7535
   macro avg       0.78      0.75      0.76      7535
weighted avg       0.82      0.83      0.83      7535
```

In [111]:
```
print(confusion_matrix(y_test_knnc,y_pred_knnc))
```

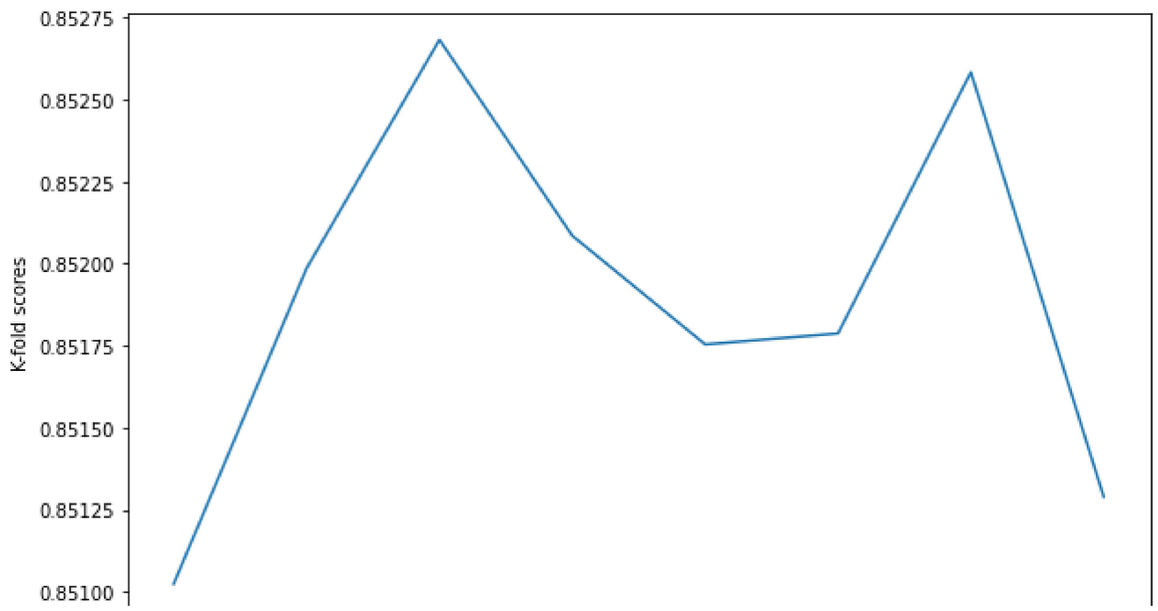
```
[[5140  515]
 [ 761 1119]]
```

```
################################### Random Forest Classifier ######################################
```

In [231]:
```
#import libraries
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
```

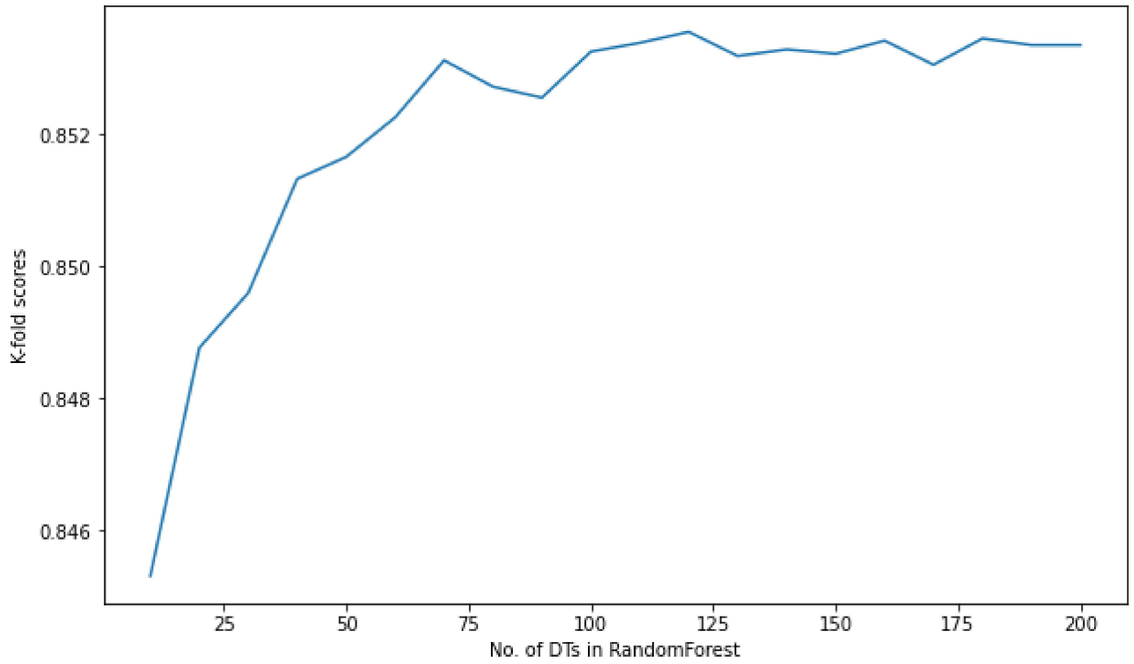
In [229]:
```
X_rfc = X_fins
y_rfc = y_fin
```

In [232]:
```
#Graph k-fold score for RF

k_scores_rfc = []
for i in range(2,10):
    k_scores_rfc.append(cross_val_score(RandomForestClassifier(),X_rfc,y_rfc,cv=i).mean())



plt.plot(range(2,10),k_scores_rfc)
plt.xlabel('K-Fold')
plt.ylabel('K-fold scores')
plt.show()
```



In [115]:
```
#Graph k-fold score vs no. of estimators in Random Forest
scores_rfc_es = []
for i in range(10,201,10):
    scores_rfc_es.append(cross_val_score(RandomForestClassifier(n_estimators=i,random_state=0),
                                         X_rfc,y_rfc,cv=10).mean())
```

In [116]:
```
plt.plot(range(10,201,10),scores_rfc_es)
plt.xlabel('No. of DTs in RandomForest')
plt.ylabel('K-fold scores')
plt.show()
```



In [117]:
```
#including other params like max_depth, we will apply gridsearch to fine the best settings for the RF
params_rfc = {
            'n_estimators': [70,80,90,100,110],
            'max_depth': [8,10,12,14,16]
        }
model_rfc = GridSearchCV(RandomForestClassifier(random_state=0), params_rfc,cv=10)
model_rfc.fit(X_rfc,y_rfc)
```

Out[117]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(random_state=0),
             param_grid={'max_depth': [8, 10, 12, 14, 16],
                         'n_estimators': [70, 80, 90, 100, 110]})
```

In [118]:
```
model_rfc.best_params_
```

Out[118]:
```
{'max_depth': 16, 'n_estimators': 80}
```

In [119]:
```
model_rfc.best_score_
```

Out[119]:
```
0.8603137895485412
```

In [120]:
```
best_model_rfc = model_rfc.best_estimator_
```

In [121]:
```
X_train_rfc,X_test_rfc,y_train_rfc,y_test_rfc = train_test_split(X_rfc,y_rfc,random_state=0)
```

In [122]:
```
best_model_rfc.fit(X_train_rfc,y_train_rfc)
```

Out[122]:
```
RandomForestClassifier(max_depth=16, n_estimators=80, random_state=0)
```

In [123]:
```
y_pred_rfc = best_model_rfc.predict(X_test_rfc)
```

In [124]:
```
print(classification_report(y_test_rfc,y_pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.88      0.95      0.91      5655
           1       0.79      0.59      0.68      1880

    accuracy                           0.86      7535
   macro avg       0.83      0.77      0.79      7535
weighted avg       0.85      0.86      0.85      7535
```

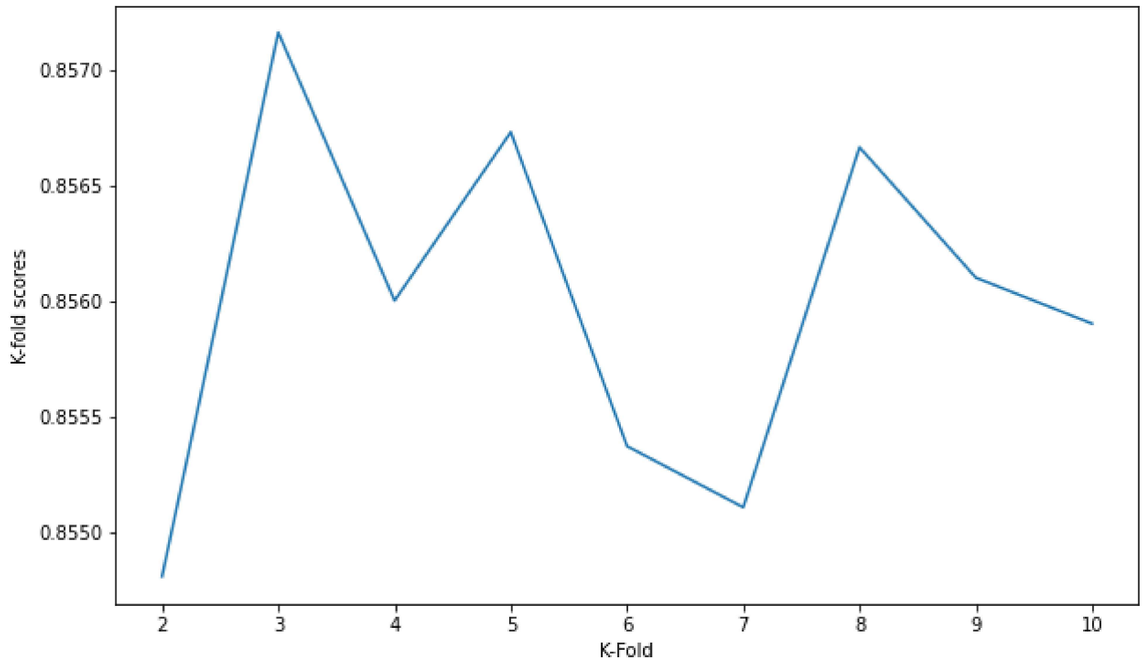
In [226]:
```
print(confusion_matrix(y_test_rfc,y_pred_rfc))
```

```
[[5357  298]
 [ 764 1116]]
```

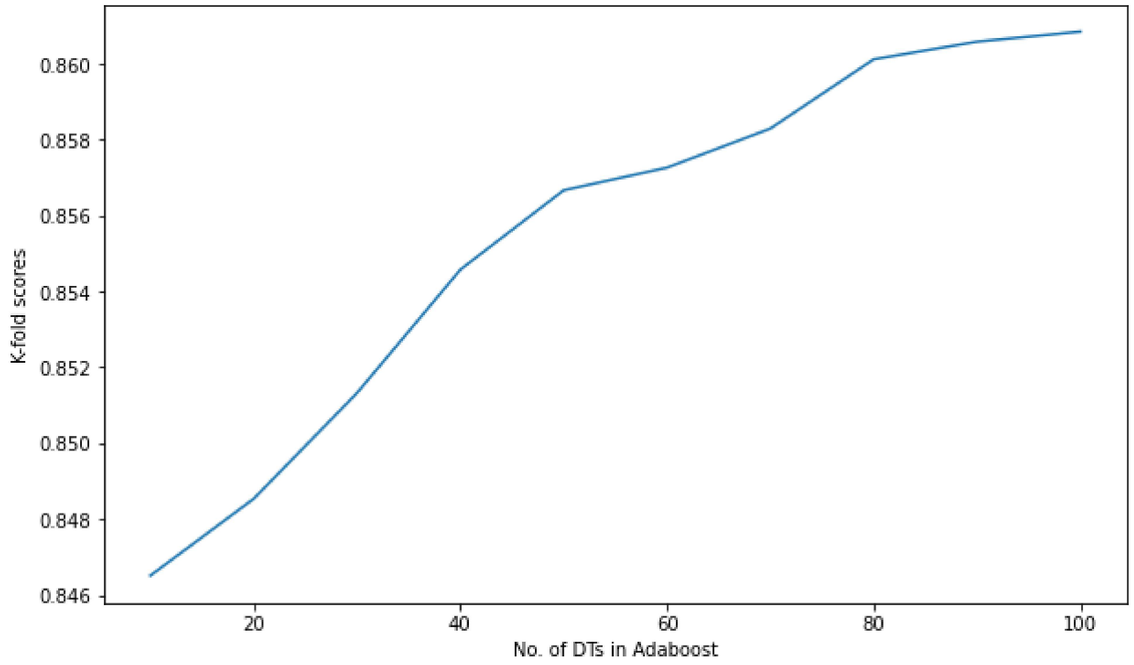
```
######################################## AdaBoost ##############################################################
```

In [125]:
```
X_adac = X_fins
y_adac = y_fin
```

In [127]:
```
#Graph k-fold score for AdaBoost
from sklearn.ensemble import AdaBoostClassifier
k_scores_adac = []
for i in range(2,11):
    k_scores_adac.append(cross_val_score(AdaBoostClassifier(),X_adac,y_adac,cv=i).mean())



plt.plot(range(2,11),k_scores_adac)
plt.xlabel('K-Fold')
plt.ylabel('K-fold scores')
plt.show()
```



In [128]:
```
from sklearn.ensemble import AdaBoostClassifier

#Graph k-fold score vs no. of estimators in Adaboost which uses DT as base estimators
scores_adac = []
for i in range(10,101,10):
    scores_adac.append(cross_val_score(AdaBoostClassifier(n_estimators=i,random_state=0),
                                       X_adac,y_adac,cv=8).mean())
plt.plot(range(10,101,10),scores_adac)
plt.xlabel('No. of DTs in Adaboost')
plt.ylabel('K-fold scores')
plt.show()
```



In [129]:
```
from sklearn.tree import DecisionTreeClassifier
#including other params like max_depth, we will apply gridsearch to fine the best settings
params = {
            'n_estimators': [70,80,90,100],
            'base_estimator': [DecisionTreeClassifier(max_depth=9,random_state=0),
                               DecisionTreeClassifier(max_depth=10,random_state=0),
                               DecisionTreeClassifier(max_depth=11,random_state=0)]
        }
model_adac = GridSearchCV(AdaBoostClassifier(random_state=0), params,cv=8)
model_adac.fit(X_adac,y_adac)
```

Out[129]:
```
GridSearchCV(cv=8, estimator=AdaBoostClassifier(random_state=0),
             param_grid={'base_estimator': [DecisionTreeClassifier(max_depth=9,
                                                                   random_state=0),
                                            DecisionTreeClassifier(max_depth=10,
                                                                   random_state=0),
                                            DecisionTreeClassifier(max_depth=11,
                                                                   random_state=0)],
                         'n_estimators': [70, 80, 90, 100]})
```

In [130]:
```
model_adac.best_params_
```

Out[130]:
```
{'base_estimator': DecisionTreeClassifier(max_depth=10, random_state=0),
 'n_estimators': 100}
```

In [131]:
```
model_adac.best_score_
```

Out[131]:
```
0.8267695734749814
```

In [132]:
```
best_model_adac = model_adac.best_estimator_
```

In [133]:
```
X_train_adac,X_test_adac,y_train_adac,y_test_adac = train_test_split(X_adac,y_adac,random_state=0)
```

In [134]:
```
best_model_adac.fit(X_train_adac,y_train_adac)
```

Out[134]:
```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=10,
                                                         random_state=0),
                   n_estimators=100, random_state=0)
```

In [135]:
```
y_pred_adac = best_model_adac.predict(X_test_adac)
```

In [136]:
```
print(classification_report(y_test_adac,y_pred_adac))
```

```
              precision    recall  f1-score   support

           0       0.87      0.89      0.88      5655
           1       0.66      0.61      0.63      1880

    accuracy                           0.82      7535
   macro avg       0.77      0.75      0.76      7535
weighted avg       0.82      0.82      0.82      7535
```

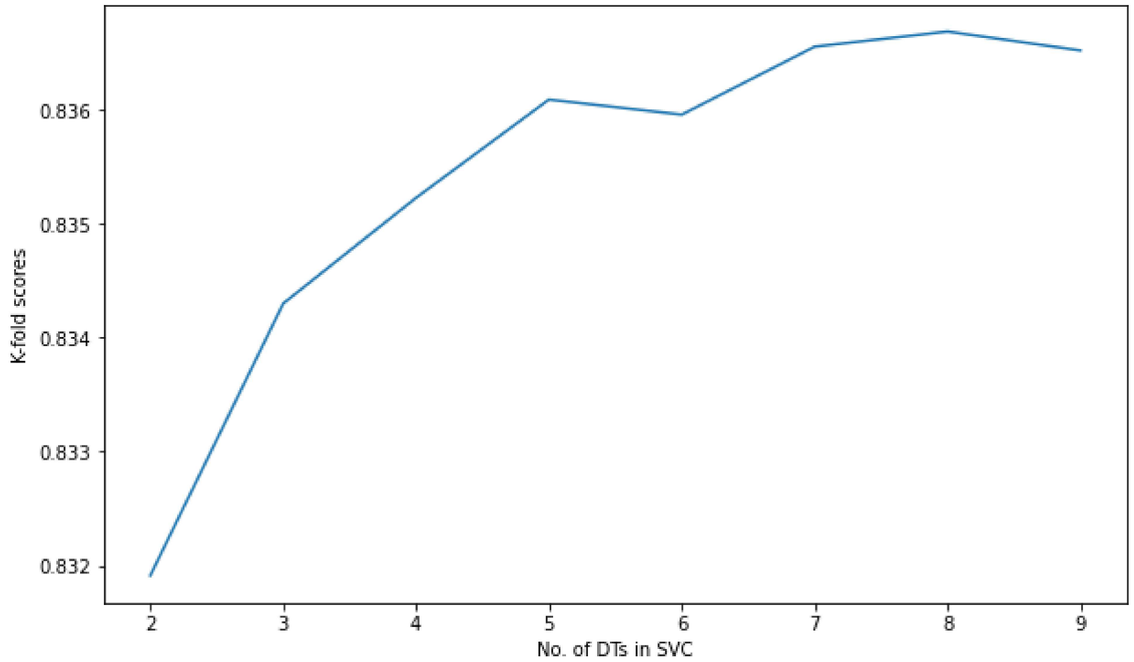
In [233]:
```
print(confusion_matrix(y_test_adac,y_pred_adac))
```

```
[[5058  597]
 [ 736 1144]]
```

```
########################################### SVM ###################################################################
```

In [137]:
```
X_svc = X_fins
y_svc = y_fin
```

In [138]:
```
from sklearn.svm import SVC
#Graph k-fold score vs no. of estimators in Adaboost which uses DT as base estimators
scores_svc = []
for i in range(2,10):
    scores_svc.append(cross_val_score(SVC(),
                                      X_svc,y_svc,cv=i).mean())
plt.plot(range(2,10),scores_svc)
plt.xlabel('No. of DTs in SVC')
plt.ylabel('K-fold scores')
plt.show()
```



In [140]:
```
from sklearn.model_selection import GridSearchCV
```

In [146]:
```
params_dictionary = {
                        'C' : [0.1, 1, 10],
                        'kernel': ['linear', 'rbf'],

                    }

model_svc = GridSearchCV(SVC(random_state=0, verbose=True),param_grid=params_dictionary,cv=8)
```

In [147]:
```
model_svc.fit(X_svc,y_svc)
```

```
[LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSV
M][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][Lib
SVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM][LibSVM]
```

Out[147]:
```
GridSearchCV(cv=8, estimator=SVC(random_state=0, verbose=True),
             param_grid={'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']})
```

In [148]:
```
model_svc.best_params_
```

Out[148]:
```
{'C': 10, 'kernel': 'rbf'}
```

In [149]:
```
model_svc.best_score_
```

Out[149]:
```
0.847871795771413
```

In [150]:
```
best_model_svm =model_svc.best_estimator_
```

In [151]:
```
from sklearn.model_selection import train_test_split
X_train_svm,X_test_svm,y_train_svm,y_test_svm = train_test_split(X_svc,y_svc,random_state=0)
```

In [152]:
```
best_model_svm.fit(X_train_svm,y_train_svm)
```

```
[LibSVM]
```

Out[152]:
```
SVC(C=10, random_state=0, verbose=True)
```

In [153]:
```
best_model_svm.score(X_test_svm,y_test_svm)
```

Out[153]:
```
0.8477770404777704
```

In [154]:
```
y_pred_svm = best_model_svm.predict(X_test_svm)
```

In [155]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test_svm,y_pred_svm))
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      5655
           1       0.74      0.60      0.66      1880

    accuracy                           0.85      7535
   macro avg       0.81      0.77      0.78      7535
weighted avg       0.84      0.85      0.84      7535
```

In [234]:
```
print(confusion_matrix(y_test_svm,y_pred_svm))
```

```
[[5255  400]
 [ 747 1133]]
```

Classification Model Summary

Result	Logistic Regressio Classification	KNN Classification	Random Forest Classification	AdaBoost Classification	SVM Classification
F1-Accuracy	0.84	0.83	0.86	0.82	0.85
Precision-Macro	0.8	0.78	0.83	0.77	0.81
Precision-Weighted	0.84	0.82	0.85	0.82	0.84

- The result suggests that Random Forest Classification performed the best
- AdaBoost performed a bit low of the models
- Scores of all models were close

```
##################################### Random Forest Classification in pyspark ##############################################
```

Classification Model with PySpark in Anaconda Jupyter Notebook on Local Machine

In [170]:
```
# Preparing data for using in pyspark

# For joining X and y and bring back to a .CSV file to be used in pyspark
result = df_class
result.head(10)
```

Out[170]:

	age	type_employer	fnlwgt	education_num	marital	sex	capital_gain	capital_loss	hr_per_week	income	...	relationship_Unmarried	relationship_Wife	country_Europe	country_Latin.and.South
0	39	2	77516	13	1	1	2174	0	40	0	...	0	0	0	
1	50	1	83311	13	0	1	0	0	13	0	...	0	0	0	
2	38	0	215646	9	2	1	0	0	40	0	...	0	0	0	
3	53	0	234721	7	0	1	0	0	40	0	...	0	0	0	
4	28	0	338409	13	0	0	0	0	40	0	...	0	1	0	
5	37	0	284582	14	0	0	0	0	40	0	...	0	1	0	
6	49	0	160187	5	0	0	0	0	16	0	...	0	0	0	
7	52	1	209642	9	0	1	0	0	45	1	...	0	0	0	
8	31	0	45781	14	1	0	14084	0	50	1	...	0	0	0	
9	42	0	159449	13	0	1	5178	0	40	1	...	0	0	0	

10 rows × 36 columns

In [171]:
```
result.shape
```

Out[171]: (30139, 36)

In [172]:
```
result.to_csv('classification_pyspark.csv', index = False)
```

In [173]:
```
import findspark
findspark.init()
findspark.find()
```

Out[173]:
```
'C:\\spark-3.1.1-bin-hadoop2.7'
```

In [174]:
```
from pyspark.sql import SparkSession
import pyspark
```

In [175]:
```
spark= SparkSession.builder.appName("Random Forest Classification").getOrCreate()
```

In [176]:
```
spark
```

Out[176]:

SparkSession - in-memory

SparkContext

Spark UI (http://DinarTriOSEducation:4043)

Version

v3.1.1

Master

local[*]

AppName

Random Forest Classification

In [177]:
```
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.ml.classification import *
from pyspark.ml.evaluation import *
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
```

In [178]:
```
df_spc = spark.read.csv('classification_pyspark.csv',inferSchema=True, header=True)
```

In [179]:
```
df_spc.limit(10).toPandas()
```

Out[179]:

	age	type_employer	fnlwgt	education_num	marital	sex	capital_gain	capital_loss	hr_per_week	income	...	relationship_Unmarried	relationship_Wife	country_Europe	country_Latin.and.South
0	39	2	77516	13	1	1	2174	0	40	0	...	0	0	0	
1	50	1	83311	13	0	1	0	0	13	0	...	0	0	0	
2	38	0	215646	9	2	1	0	0	40	0	...	0	0	0	
3	53	0	234721	7	0	1	0	0	40	0	...	0	0	0	
4	28	0	338409	13	0	0	0	0	40	0	...	0	1	0	
5	37	0	284582	14	0	0	0	0	40	0	...	0	1	0	
6	49	0	160187	5	0	0	0	0	16	0	...	0	0	0	
7	52	1	209642	9	0	1	0	0	45	1	...	0	0	0	
8	31	0	45781	14	1	0	14084	0	50	1	...	0	0	0	
9	42	0	159449	13	0	1	5178	0	40	1	...	0	0	0	

10 rows × 36 columns

In [180]:
```
df_spc.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- type_employer: integer (nullable = true)
 |-- fnlwgt: integer (nullable = true)
 |-- education_num: integer (nullable = true)
 |-- marital: integer (nullable = true)
 |-- sex: integer (nullable = true)
 |-- capital_gain: integer (nullable = true)
 |-- capital_loss: integer (nullable = true)
 |-- hr_per_week: integer (nullable = true)
 |-- income: integer (nullable = true)
 |-- occupation_Armed-Forces: integer (nullable = true)
 |-- occupation_Craft-repair: integer (nullable = true)
 |-- occupation_Exec-managerial: integer (nullable = true)
 |-- occupation_Farming-fishing: integer (nullable = true)
 |-- occupation_Handlers-cleaners: integer (nullable = true)
 |-- occupation_Machine-op-inspct: integer (nullable = true)
 |-- occupation_Other-service: integer (nullable = true)
 |-- occupation_Priv-house-serv: integer (nullable = true)
 |-- occupation_Prof-specialty: integer (nullable = true)
 |-- occupation_Protective-serv: integer (nullable = true)
 |-- occupation_Sales: integer (nullable = true)
 |-- occupation_Tech-support: integer (nullable = true)
 |-- occupation_Transport-moving: integer (nullable = true)
 |-- relationship_Not-in-family: integer (nullable = true)
 |-- relationship_Other-relative: integer (nullable = true)
 |-- relationship_Own-child: integer (nullable = true)
 |-- relationship_Unmarried: integer (nullable = true)
 |-- relationship_Wife: integer (nullable = true)
 |-- country_Europe: integer (nullable = true)
 |-- country_Latin.and.South.America: integer (nullable = true)
 |-- country_North.America: integer (nullable = true)
 |-- country_South: integer (nullable = true)
 |-- race_Asian-Pac-Islander: integer (nullable = true)
 |-- race_Black: integer (nullable = true)
 |-- race_Other: integer (nullable = true)
 |-- race_White: integer (nullable = true)
```

In [181]:
```
print(df_spc.count())
print(len(df_spc.columns))
```

```
30139
36
```

In [182]:
```
df_spc.columns
```

Out[182]:
```
['age',
 'type_employer',
 'fnlwgt',
 'education_num',
 'marital',
 'sex',
 'capital_gain',
 'capital_loss',
 'hr_per_week',
 'income',
 'occupation_Armed-Forces',
 'occupation_Craft-repair',
 'occupation_Exec-managerial',
 'occupation_Farming-fishing',
 'occupation_Handlers-cleaners',
 'occupation_Machine-op-inspct',
 'occupation_Other-service',
 'occupation_Priv-house-serv',
 'occupation_Prof-specialty',
 'occupation_Protective-serv',
 'occupation_Sales',
 'occupation_Tech-support',
 'occupation_Transport-moving',
 'relationship_Not-in-family',
 'relationship_Other-relative',
 'relationship_Own-child',
 'relationship_Unmarried',
 'relationship_Wife',
 'country_Europe',
 'country_Latin.and.South.America',
 'country_North.America',
 'country_South',
 'race_Asian-Pac-Islander',
 'race_Black',
 'race_Other',
 'race_White']
```

In [207]:
```
df_spark = df_spc.toDF('age','type_employer','fnlwgt','education_num','marital','sex','capital_gain', 'capital_loss', 'hr_per_week','label',
 'occupation_Armed_Forces', 'occupation_Craft_repair','occupation_Exec_managerial', 'occupation_Farming_fishing',
 'occupation_Handlers_cleaners', 'occupation_Machine_op_inspct', 'occupation_Other_service', 'occupation_Priv_house_serv',
 'occupation_Prof_specialty', 'occupation_Protective_serv', 'occupation_Sales', 'occupation_Tech_support',
 'occupation_Transport_moving', 'relationship_Not_in_family', 'relationship_Other_relative', 'relationship_Own_child',
 'relationship_Unmarried', 'relationship_Wife', 'country_Europe', 'country_Latin_and_South_America', 'country_North_America',
 'country_South', 'race_Asian_Pac_Islander', 'race_Black', 'race_Other', 'race_White')
```

In [208]:
```
df_spark.columns
```

Out[208]:
```
['age',
 'type_employer',
 'fnlwgt',
 'education_num',
 'marital',
 'sex',
 'capital_gain',
 'capital_loss',
 'hr_per_week',
 'label',
 'occupation_Armed_Forces',
 'occupation_Craft_repair',
 'occupation_Exec_managerial',
 'occupation_Farming_fishing',
 'occupation_Handlers_cleaners',
 'occupation_Machine_op_inspct',
 'occupation_Other_service',
 'occupation_Priv_house_serv',
 'occupation_Prof_specialty',
 'occupation_Protective_serv',
 'occupation_Sales',
 'occupation_Tech_support',
 'occupation_Transport_moving',
 'relationship_Not_in_family',
 'relationship_Other_relative',
 'relationship_Own_child',
 'relationship_Unmarried',
 'relationship_Wife',
 'country_Europe',
 'country_Latin_and_South_America',
 'country_North_America',
 'country_South',
 'race_Asian_Pac_Islander',
 'race_Black',
 'race_Other',
 'race_White']
```

In [209]:
```
input_columns = ['age',
 'type_employer',
 'fnlwgt',
 'education_num',
 'marital',
 'sex',
 'capital_gain',
 'capital_loss',
 'hr_per_week',
 'occupation_Armed_Forces',
 'occupation_Craft_repair',
 'occupation_Exec_managerial',
 'occupation_Farming_fishing',
 'occupation_Handlers_cleaners',
 'occupation_Machine_op_inspct',
 'occupation_Other_service',
 'occupation_Priv_house_serv',
 'occupation_Prof_specialty',
 'occupation_Protective_serv',
 'occupation_Sales',
 'occupation_Tech_support',
 'occupation_Transport_moving',
 'relationship_Not_in_family',
 'relationship_Other_relative',
 'relationship_Own_child',
 'relationship_Unmarried',
 'relationship_Wife',
 'country_Europe',
 'country_Latin_and_South_America',
 'country_North_America',
 'country_South',
 'race_Asian_Pac_Islander',
 'race_Black',
 'race_Other',
 'race_White' ]
```

In [210]:
```
dependent_var = 'label'
```

In [211]:
```
assembler = VectorAssembler(inputCols=input_columns, outputCol="features")
feature_vec=assembler.transform(df_spark).select('features',dependent_var)
feature_vec.show(5)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(35,[0,1,2,3,4,5,...|    0|
|(35,[0,1,2,3,5,8,...|    0|
|(35,[0,2,3,4,5,8,...|    0|
|(35,[0,2,3,5,8,13...|    0|
|(35,[0,2,3,8,17,2...|    0|
+--------------------+-----+
only showing top 5 rows
```

In [212]:
```
# Split the data into train and test sets
train_data, test_data = feature_vec.randomSplit([.80,.20],seed=0)
```

In [ ]:
```
```

In [213]:
```
# Let us add the parameters of choice
classifier_rfc = RandomForestClassifier()
paramGridRfc = (ParamGridBuilder() \
                .addGrid(classifier_rfc.maxDepth, [2, 5, 10]) \
                .build())

crossval_rfc = CrossValidator(estimator=classifier_rfc,
                              estimatorParamMaps=paramGridRfc,
                              evaluator=BinaryClassificationEvaluator(),
                              numFolds=2)
fitModel_rfc = crossval_rfc.fit(train_data)
bestModel_rfc = fitModel_rfc.bestModel
featureImportances = bestModel_rfc.featureImportances.toArray()
print("Feature Importances:\n ", featureImportances)
```

```
Feature Importances:
  [7.69336628e-02 8.39917794e-03 1.19365926e-02 1.81885692e-01
 2.50049744e-01 2.14331244e-02 1.61140535e-01 4.41334030e-02
 5.99494910e-02 0.00000000e+00 2.29683006e-03 2.77719634e-02
 2.50194588e-03 1.67385283e-03 1.40505112e-03 9.22067202e-03
 3.84047187e-05 1.20476613e-02 1.21550541e-03 1.82074208e-03
 2.08666555e-03 1.87715376e-03 3.42998641e-02 3.48814734e-03
 4.45704181e-02 1.11757483e-02 1.63918310e-02 1.23708213e-03
 8.55019194e-04 2.36010528e-03 6.13594221e-04 7.76931127e-04
 1.65417120e-03 3.75586822e-04 2.38363025e-03]
```

In [220]:
```
predictions = fitModel_rfc.transform(test_data)
predictions
```

Out[220]:
```
DataFrame[features: vector, label: int, rawPrediction: vector, probability: vector, prediction: double]
```

In [224]:
```
accuracy = BinaryClassificationEvaluator(metricName='areaUnderROC').evaluate(predictions)
print("\nAccuracy: ", accuracy)
```

```
Accuracy:  0.9022708533213222
```

As the AUC value in 0.9023 the project performed really good.