In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree  import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
salary=pd.read_csv('salary.csv')
```

In [3]:

```
salary
```

Out[3]:

	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_gain	capital_loss
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0
...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0
32557	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0
32558	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0
32559	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0
32560	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0

32561 rows × 14 columns

In [4]:

```
salary.describe()
```

Out[4]:

	age	education_num	capital_gain	capital_loss	hours_per_week
count	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	10.080679	1077.648844	87.303830	40.437456
std	13.640433	2.572720	7385.292085	402.960219	12.347429

	age	education_num	capital_gain	capital_loss	hours_per_week
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	48.000000	12.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	4356.000000	99.000000

In [5]:

```
salary.dtypes
```

Out[5]:

```
age                int64
workclass         object
education         object
education_num      int64
marital_status    object
occupation        object
relationship      object
race              object
gender            object
capital_gain       int64
capital_loss       int64
hours_per_week     int64
native_country    object
income_bracket    object
dtype: object
```

In [6]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list1=['workclass','education','marital_status','occupation','relationship','race','gender','native
_country','income_bracket']
for val in list1:
    salary[val]=le.fit_transform(salary[val].astype(str))
```

In [7]:

```
salary
```

Out[7]:

	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_gain	capital_loss	h
0	39	7	9	13	4	1	1	4	1	2174	0	
1	50	6	9	13	2	4	0	4	1	0	0	
2	38	4	11	9	0	6	1	4	1	0	0	
3	53	4	1	7	2	6	0	2	1	0	0	
4	28	4	9	13	2	10	5	2	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	4	7	12	2	13	5	4	0	0	0	
32557	40	4	11	9	2	7	0	4	1	0	0	
32558	58	4	11	9	6	1	4	4	0	0	0	
32559	22	4	11	9	4	1	3	4	1	0	0	
32560	52	5	11	9	2	4	5	4	0	15024	0	

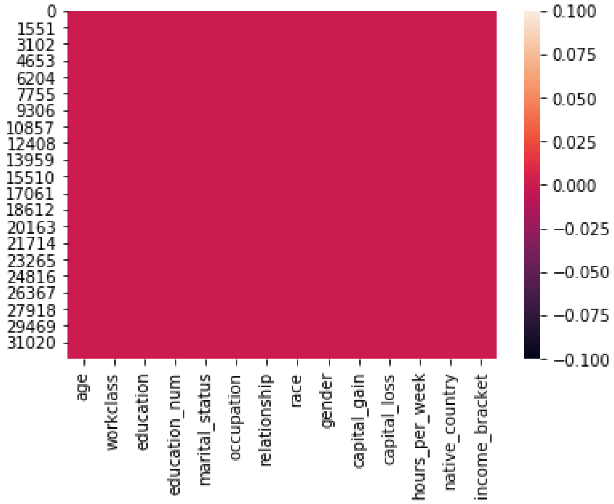
32561 rows × 14 columns

In [8]:

```
sns.heatmap(salary.isnull())
```

Out[8]:

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x20061492f88&gt;
```



In [9]:

```
salary.isnull().sum()
```

Out[9]:

```
age               0
workclass         0
education         0
education_num     0
marital_status    0
occupation        0
relationship      0
race              0
gender            0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
income_bracket    0
dtype: int64
```

In [10]:

```
salary.corr()
```

Out[10]:

	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_
age	1.000000	0.003787	-0.010508	0.036527	-0.266288	-0.020947	-0.263698	0.028718	0.088832	0.07
workclass	0.003787	1.000000	0.023513	0.052085	-0.064731	0.254892	-0.090461	0.049742	0.095981	0.03
education	- 0.010508	0.023513	1.000000	0.359153	-0.038407	-0.021260	-0.010876	0.014131	- 0.027356	0.03
education_num	0.036527	0.052085	0.359153	1.000000	-0.069304	0.109697	-0.094153	0.031838	0.012280	0.12
marital_status	- 0.266288	-0.064731	-0.038407	-0.069304	1.000000	-0.009654	0.185451	- 0.068013	- 0.129314	-0.04
occupation	- 0.020947	0.254892	-0.021260	0.109697	-0.009654	1.000000	-0.075607	0.006763	0.080296	0.02
relationship	- 0.263698	-0.090461	-0.010876	-0.094153	0.185451	-0.075607	1.000000	- 0.116055	- 0.582454	-0.05
race	0.028718	0.049742	0.014131	0.031838	-0.068013	0.006763	-0.116055	1.000000	0.087204	0.01
gender	0.088832	0.095981	-0.027356	0.012280	-0.129314	0.080296	-0.582454	0.087204	1.000000	0.04
capital_gain	0.077674	0.033835	0.030046	0.122630	-0.043393	0.025505	-0.057919	0.011145	0.048480	1.00
capital_loss	0.057775	0.012216	0.016746	0.079923	-0.034187	0.017987	-0.061062	0.018899	0.045567	-0.03
hours_per_week	0.068756	0.138962	0.055510	0.148123	0.190519	0.080383	0.248974	0.041910	0.229309	0.07

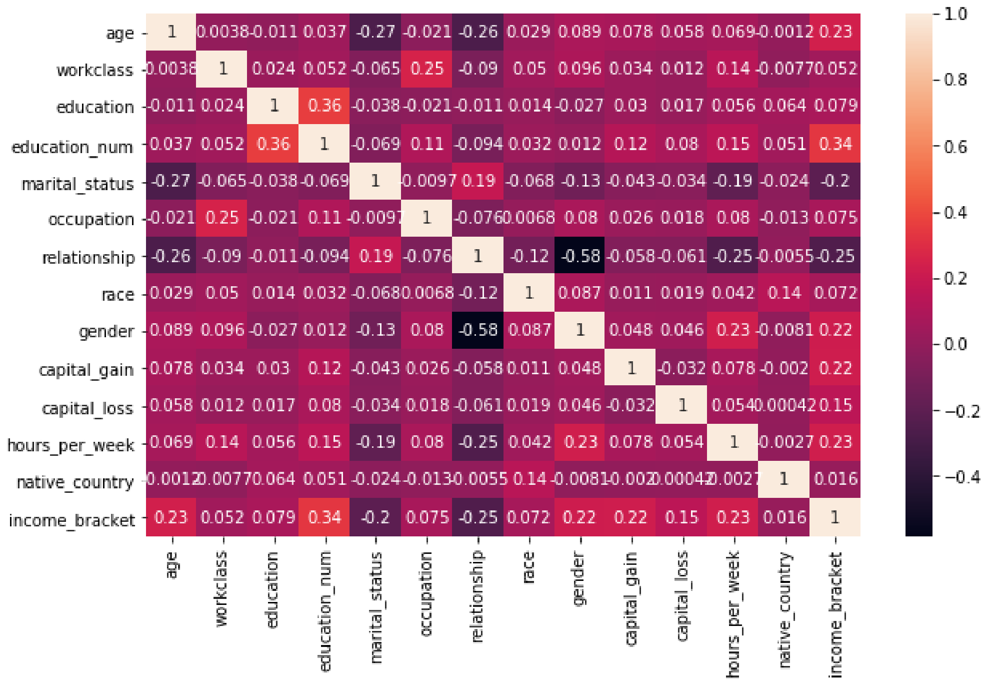
	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_
native_country	0.001151	-0.007690	0.064288	0.050840	-0.023819	-0.012543	-0.005507	0.137852	0.008119	-0.00
income_bracket	0.234037	0.051604	0.079317	0.335154	-0.199307	0.075468	-0.250918	0.071846	0.215980	0.22

In [11]:

```
plt.figure(figsize=(10,6))
sns.heatmap(salary.corr(),annot=True)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20063700f48>
```



In [12]:

```
salary.skew()
```

Out[12]:

```
age                0.558743
workclass         -0.752024
education         -0.934042
education_num     -0.311676
marital_status    -0.013508
occupation         0.114583
relationship       0.786818
race              -2.435386
gender            -0.719293
capital_gain      11.953848
capital_loss       4.594629
hours_per_week     0.227643
native_country    -3.658303
income_bracket     1.212430
dtype: float64
```

In [13]:

```
from scipy.stats import zscore
z_score=abs(zscore(salary))
print(salary.shape)
sal=salary.loc[(z_score<3).all(axis=1)]
print(sal.shape)
```

```
(32561, 14)
```

(27722, 14)

In [14]:

```
sal
```

Out[14]:

	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_gain	capital_loss	h
0	39	7	9	13	4	1	1	4	1	2174	0	
1	50	6	9	13	2	4	0	4	1	0	0	
2	38	4	11	9	0	6	1	4	1	0	0	
3	53	4	1	7	2	6	0	2	1	0	0	
5	37	4	12	14	2	4	5	4	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	4	7	12	2	13	5	4	0	0	0	
32557	40	4	11	9	2	7	0	4	1	0	0	
32558	58	4	11	9	6	1	4	4	0	0	0	
32559	22	4	11	9	4	1	3	4	1	0	0	
32560	52	5	11	9	2	4	5	4	0	15024	0	

27722 rows × 14 columns

In [15]:

```
x=sal.iloc[:,0:-1]
```

In [16]:

```
x
```

Out[16]:

	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_gain	capital_loss	h
0	39	7	9	13	4	1	1	4	1	2174	0	
1	50	6	9	13	2	4	0	4	1	0	0	
2	38	4	11	9	0	6	1	4	1	0	0	
3	53	4	1	7	2	6	0	2	1	0	0	
5	37	4	12	14	2	4	5	4	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	4	7	12	2	13	5	4	0	0	0	
32557	40	4	11	9	2	7	0	4	1	0	0	
32558	58	4	11	9	6	1	4	4	0	0	0	
32559	22	4	11	9	4	1	3	4	1	0	0	
32560	52	5	11	9	2	4	5	4	0	15024	0	

27722 rows × 13 columns

In [17]:

```
x.shape
```

Out[17]:

(27722, 13)

In [18]:

```
y=sal.iloc[:,-1]
```

In [19]:

```
y
```

Out[19]:

```
0        0
1        0
2        0
3        0
5        0
        ..
32556    0
32557    1
32558    0
32559    0
32560    1
Name: income_bracket, Length: 27722, dtype: int32
```

In [20]:

```
y.shape
```

Out[20]:

```
(27722,)
```

In [21]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.22,random_state=42)
```

In [22]:

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
lr.score(x_train,y_train)
pred=lr.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.8166912608624365
[[4476  244]
 [ 874  505]]
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      4720
           1       0.67      0.37      0.47      1379

    accuracy                           0.82      6099
   macro avg       0.76      0.66      0.68      6099
weighted avg       0.80      0.82      0.80      6099
```

In [23]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
predknn=knn.predict(x_test)
print(accuracy_score(y_test,predknn))
print(confusion_matrix(y_test,predknn))
print(classification_report(y_test,predknn))
```

```
0.8316117396294475
[[4276  444]
 [ 583  796]]
```

```
              precision    recall  f1-score   support

           0       0.88      0.91      0.89      4720
           1       0.64      0.58      0.61      1379

    accuracy                           0.83      6099
   macro avg       0.76      0.74      0.75      6099
weighted avg       0.83      0.83      0.83      6099
```

In [24]:

```
gnb=GaussianNB()
gnb.fit(x_train,y_train)
gnb.score(x_train,y_train)
predgnb=gnb.predict(x_test)
print(accuracy_score(y_test,predgnb))
print(confusion_matrix(y_test,predgnb))
print(classification_report(y_test,predgnb))
```

```
0.7868503033284145
[[3777  943]
 [ 357 1022]]
              precision    recall  f1-score   support

           0       0.91      0.80      0.85      4720
           1       0.52      0.74      0.61      1379

    accuracy                           0.79      6099
   macro avg       0.72      0.77      0.73      6099
weighted avg       0.82      0.79      0.80      6099
```

In [25]:

```
svc=SVC(kernel='rbf')
svc.fit(x_train,y_train)
svc.score(x_train,y_train)
predsvc=svc.predict(x_test)
print(accuracy_score(y_test,predsvc))
print(confusion_matrix(y_test,predsvc))
print(classification_report(y_test,predsvc))
```

```
0.8102967699622889
[[4703   17]
 [1140  239]]
              precision    recall  f1-score   support

           0       0.80      1.00      0.89      4720
           1       0.93      0.17      0.29      1379

    accuracy                           0.81      6099
   macro avg       0.87      0.58      0.59      6099
weighted avg       0.83      0.81      0.76      6099
```

In [26]:

```
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
dtc.score(x_train,y_train)
preddtc=dtc.predict(x_test)
print(accuracy_score(y_test,preddtc))
print(confusion_matrix(y_test,preddtc))
print(classification_report(y_test,preddtc))
```

```
0.8134120347597967
[[4190  530]
 [ 608  771]]
              precision    recall  f1-score   support

           0       0.87      0.89      0.88      4720
```

```
           1       0.59      0.56      0.58      1379

    accuracy                           0.81      6099
   macro avg       0.73      0.72      0.73      6099
weighted avg       0.81      0.81      0.81      6099
```

In [27]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.8442367601246106
[[4354  366]
 [ 584  795]]
              precision    recall  f1-score   support

           0       0.88      0.92      0.90      4720
           1       0.68      0.58      0.63      1379

    accuracy                           0.84      6099
   macro avg       0.78      0.75      0.76      6099
weighted avg       0.84      0.84      0.84      6099
```

In [28]:

```
from sklearn.ensemble import AdaBoostClassifier
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
ad.score(x_train,y_train)
predad=ad.predict(x_test)
print(accuracy_score(y_test,predad))
print(confusion_matrix(y_test,predad))
print(classification_report(y_test,predad))
```

```
0.8552221675684538
[[4447  273]
 [ 610  769]]
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      4720
           1       0.74      0.56      0.64      1379

    accuracy                           0.86      6099
   macro avg       0.81      0.75      0.77      6099
weighted avg       0.85      0.86      0.85      6099
```

In [29]:

```
#AdaBoostClassifier is the best model among all models

import joblib
joblib.dump(ad,'salary.pkl')
```

Out[29]:

```
['salary.pkl']
```

In [ ]:

```

```