In [1]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import warnings

warnings.filterwarnings('ignore')
```

In [2]:

```
train = pd.read_csv('data/4th_kaggle/train.csv')
test = pd.read_csv('data/4th_kaggle/test.csv')
sub = pd.read_csv('data/4th_kaggle/sample_submission.csv')
```

데이터 탐색

- 컬럼명 : [].columns
- 행열 : [].shape
- 정보 : [].info()
- 수치 데이터 요약정보 : [].describe()
- 결측치 : [].isnull().sum()

데이터 정보

age : 나이
workclass : 고용 형태
fnlwgt : 사람 대표성을 나타내는 가중치 (final weight의 약자)
education : 교육 수준 (최종 학력)
education_num : 교육 수준 수치
marital_status: 결혼 상태
occupation : 업종
relationship : 가족 관계
race : 인종
sex : 성별
capital_gain : 양도 소득
capital_loss : 양도 손실
hours_per_week : 주당 근무 시간
native_country : 국적
income : 수익 (예측해야 하는 값, target variable)

In [3]:

```
print("학습용 데이터 : ", train.shape)
print("테스트용 데이터 : ", test.shape)
```

```
학습용 데이터 :  (26049, 16)
테스트용 데이터 :  (6512, 15)
```

In [4]:

```
y = train['income']
test['income'] = "blank"
```

In [5]:

```
all_dat = pd.concat([train, test], axis=0)
print(all_dat.shape)
```

```
(32561, 16)
```

In [6]:

```
all_dat.income.value_counts()
```

Out[6]:

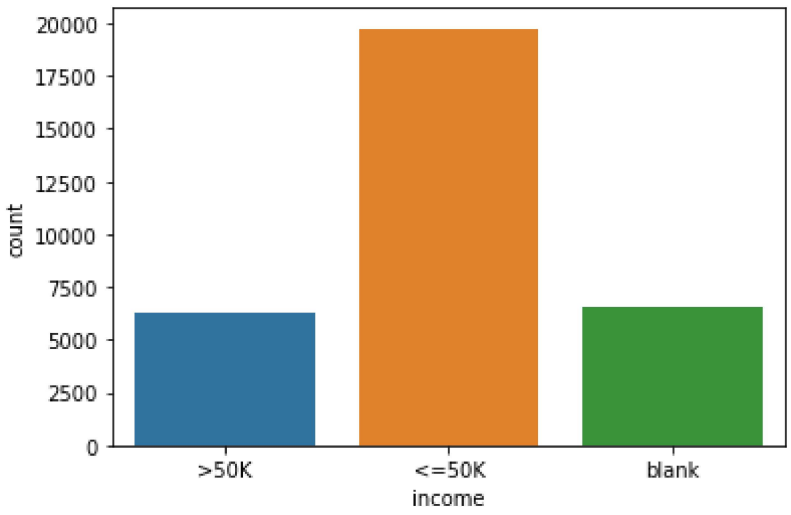
```
<=50K     19744
blank      6512
>50K       6305
Name: income, dtype: int64
```

In [7]:

```
sns.countplot(x="income", data=all_dat)
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f2ef484940>
```



In [8]:

```
all_dat.loc[ all_dat['income']=='>50K' , 'target'] = 1
all_dat.loc[ all_dat['income']=='<=50K' , 'target'] = 0
all_dat.loc[ all_dat['income']=='blank' , 'target'] = 999
all_dat['target'] = all_dat.target.astype("int")
```

In [9]:

```
all_dat.head()
```

Out[9]:

	id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relations
0	0	40	Private	168538	HS-grad	9	Married-civ-spouse	Sales	Husba
1	1	17	Private	101626	9th	5	Never-married	Machine-op-inspct	Own-cl
2	2	18	Private	353358	Some-college	10	Never-married	Other-service	Own-cl
3	3	21	Private	151158	Some-college	10	Never-married	Prof-specialty	Own-cl
4	4	24	Private	122234	Some-college	10	Never-married	Adm-clerical	Not-in-far

In [10]:

```
all_dat.columns
```

Out[10]:

```
Index(['id', 'age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'income', 'target'],
      dtype='object')
```

In [11]:

```
sel_cat = ['workclass', 'education', 'marital_status',
           'occupation', 'relationship', 'race',
           'sex', 'native_country' ]

for i in sel_cat:
    print(all_dat[i].value_counts() )
    print()
```

```
Name: sex, dtype: int64

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

Label Encoding

In [12]:

```
from sklearn.preprocessing import LabelEncoder
```

In [14]:

```
sel_cat = ['workclass', 'education', 'marital_status',
           'occupation', 'relationship', 'race',
           'sex', 'native_country' ]

encoder_x = LabelEncoder()
for i in sel_cat:
    temp = i + "_lbl"
    all_dat[temp] = encoder_x.fit_transform(all_dat[i])
    print()

all_dat.head()
```

Out[14]:

	id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relations
0	0	40	Private	168538	HS-grad	9	Married-civ-spouse	Sales	Husba
1	1	17	Private	101626	9th	5	Never-married	Machine-op-inspct	Own-ch
2	2	18	Private	353358	Some-college	10	Never-married	Other-service	Own-ch
3	3	21	Private	151158	Some-college	10	Never-married	Prof-specialty	Own-ch
4	4	24	Private	122234	Some-college	10	Never-married	Adm-clerical	Not-in-fam

5 rows × 33 columns

In [15]:

```
all_dat_n = all_dat.drop(sel_cat, axis=1)
all_dat_n
```

Out[15]:

	id	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week	income
0	0	40	168538	9	0	0	60	>50K
1	1	17	101626	5	0	0	20	<=50K
2	2	18	353358	10	0	0	16	<=50K
3	3	21	151158	10	0	0	25	<=50K
4	4	24	122234	10	0	0	20	<=50K
...	...	...	...	...	...	...	...	...
6507	6507	35	61343	13	0	0	40	blank
6508	6508	41	32185	13	0	0	40	blank
6509	6509	39	409189	3	0	0	40	blank
6510	6510	35	180342	9	0	0	40	blank
6511	6511	28	156819	9	0	0	36	blank

32561 rows × 25 columns

In [16]:

```
train_n = all_dat_n.loc[ (all_dat_n['target']==0) | (all_dat_n['target']==1)  , : ]
test_n = all_dat_n.loc[ all_dat_n['target']==999  ,  : ]
```

In [17]:

```
print(train_n.shape, test_n.shape)
```

```
(26049, 25) (6512, 25)
```

In [18]:

```
X = train_n.drop(['target', 'income'], axis=1)
y = train_n['target']

test_X = test_n.drop(['target', 'income'], axis=1)
```

In [19]:

```
print(X.shape, y.shape, test_X.shape)
```

```
(26049, 23) (26049,) (6512, 23)
```

로지스틱 모델

In [22]:

```
from sklearn.linear_model import LogisticRegression
```

In [23]:

```
model = LogisticRegression()
model.fit(X, y)
pred = model.predict(test_X)
```

In [24]:

```
sub.columns
```

Out[24]:

```
Index(['id', 'prediction'], dtype='object')
```

In [25]:

```
print( sub.shape )
print( pred.shape )
```

```
(6512, 2)
(6512,)
```

In [26]:

```
sub['prediction'] = pred
sub.to_csv("thirdSub4th.csv", index=False)
```

여러가지 모델 확인해 보기

In [28]:

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

In [40]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
import numpy as np
```

In [34]:

```
X = train_n.drop(['target', 'income'], axis=1)
y = train_n['target']

print(X.shape, y.shape)
```

```
(26049, 23) (26049,)
```

In [35]:

```
test_X = test_n.drop(['target', 'income'], axis=1)


X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                    stratify=y,
                                                    random_state=42)
```

In [37]:

```
model_list = ["LogisticRegression",  "DecisionTreeClassifier", "KNeighborsClassifier",
              "RandomForestClassifier",  "AdaBoostClassifier"]
model_score = []
```

In [38]:

```
model = LogisticRegression()
model.fit(X_train, y_train)
score = cross_val_score(model, X_train, y_train,
                        cv=5, scoring="accuracy")
print(score)
print("MSE 평균 :", score.mean())
m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)
```

```
[0.78428864 0.7896084  0.78756079 0.78781674 0.79242385]
MSE 평균 : 0.7883396832025241
```

In [41]:

```
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
score = cross_val_score(model, X_train, y_train,
                        cv=5, scoring="accuracy")
print(score)
print("MSE 평균 :", score.mean())
m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)
```

```
[0.8024565  0.80496545 0.81238802 0.8067571  0.79626312]
MSE 평균 : 0.8045660375480169
```

In [42]:

```
model = KNeighborsClassifier()
model.fit(X_train, y_train)
score = cross_val_score(model, X_train, y_train,
                        cv=5, scoring="accuracy")
print(score)
print("MSE 평균 :", score.mean())
m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)
```

```
[0.77456499 0.76068595 0.77732275 0.76375736 0.76785257]
MSE 평균 : 0.7688367256209427
```

In [44]:

```
model = RandomForestClassifier()
model.fit(X_train, y_train)
score = cross_val_score(model, X_train, y_train,
                        cv=5, scoring="accuracy")
print(score)
print("MSE 평균 :", score.mean())
m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)
```

```
[0.84800409 0.8505247  0.86255439 0.85564372 0.85257231]
MSE 평균 : 0.8538598411008873
```

In [45]:

```
model = AdaBoostClassifier()
model.fit(X_train, y_train)
score = cross_val_score(model, X_train, y_train,
                        cv=5, scoring="accuracy")
print(score)
print("MSE 평균 :", score.mean())
m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)
```

```
[0.85337769 0.84873304 0.8651139  0.85845918 0.85436396]
MSE 평균 : 0.8560095532282161
```

최종 모델

In [46]:

```
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
pred = model.predict(test_X)
```

In [47]:

```
sub['prediction'] = pred
sub.to_csv("multiModelFourSub4th.csv", index=False)
```

0.85546