캐글 코리아 4차 대회

학습내용

- 원핫 인코딩을 해 본다.

대회 링크 : https://www.kaggle.com/c/kakr-4th-competition/overview (https://www.kaggle.com/c/kakr-4th-competition/overview)

In [3]:

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

warnings.filterwarnings('ignore')
```

In [4]:

```
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 [5]:

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

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

In [6]:

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

In [7]:

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

```
(32561, 16)
```

In [8]:

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

Out[8]:

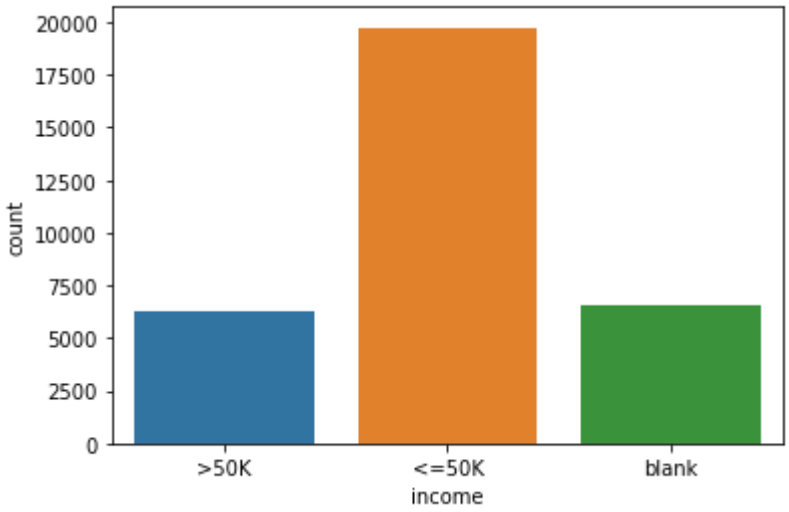
```
<=50K     19744
blank      6512
>50K       6305
Name: income, dtype: int64
```

In [9]:

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

Out[9]:

<AxesSubplot:xlabel='income', ylabel='count'>



In [10]:

```
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 [11]:

```
all_dat.head()
```

Out[11]:

	id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relations
0	0	40	Private	168538	HS-grad	9	Married-civ-spouse	Sales	Husb
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 [12]:

```
all_dat.columns
```

Out[12]:

```
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 [13]:

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

X_cat = all_dat[sel_cat]
y = all_dat['target']
```

In [14]:

```
X_dummy = pd.get_dummies(X_cat)
X_dummy
```

Out[14]:

	workclass_?	workclass_Federal-gov	workclass_Local-gov	workclass_Never-worked	workclass_Private
0	0	0	0	0	1
1	0	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	0	0	0	1
...	...	...	...	...	...
6507	0	0	0	0	1
6508	0	0	0	0	0
6509	0	0	0	0	1
6510	0	0	0	0	1
6511	0	0	0	0	1

32561 rows × 102 columns

In [15]:

```
all_dat_n = pd.concat([all_dat, X_dummy], axis=1)
all_dat_n
```

Out[15]:

	id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	rel
0	0	40	Private	168538	HS-grad	9	Married-civ-spouse	Sales	
1	1	17	Private	101626	9th	5	Never-married	Machine-op-inspct	
2	2	18	Private	353358	Some-college	10	Never-married	Other-service	
3	3	21	Private	151158	Some-college	10	Never-married	Prof-specialty	
4	4	24	Private	122234	Some-college	10	Never-married	Adm-clerical	No
...	...	...	...	...	...	...	...	...	
6507	6507	35	Private	61343	Bachelors	13	Married-civ-spouse	Sales	
6508	6508	41	Self-emp-inc	32185	Bachelors	13	Married-civ-spouse	Tech-support	
6509	6509	39	Private	409189	5th-6th	3	Married-civ-spouse	Other-service	
6510	6510	35	Private	180342	HS-grad	9	Married-civ-spouse	Craft-repair	
6511	6511	28	Private	156819	HS-grad	9	Divorced	Handlers-cleaners	

32561 rows × 119 columns

In [16]:

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

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

In [17]:

```
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 [18]:

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

```
(26049, 110) (6512, 110)
```

In [19]:

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

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

In [20]:

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

```
(26049, 109) (26049,) (6512, 109)
```

In [23]:

```
X.columns
```

Out[23]:

```
Index(['id', 'age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss',
       'hours_per_week', 'workclass_?', 'workclass_Federal-gov',
       'workclass_Local-gov',
       ...
       'native_country_Portugal', 'native_country_Puerto-Rico',
       'native_country_Scotland', 'native_country_South',
       'native_country_Taiwan', 'native_country_Thailand',
       'native_country_Trinadad&Tobago', 'native_country_United-States',
       'native_country_Vietnam', 'native_country_Yugoslavia'],
      dtype='object', length=109)
```

In [25]:

```
type(X)
```

Out[25]:

```
pandas.core.frame.DataFrame
```

In [29]:

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

In [28]:

```
sel = ['age', 'fnlwgt', 'capital_gain']

X_tr_all = X[sel]
y_tr_all = y
X_test_all = test_X[sel]

X_train, X_test, y_train, y_test = train_test_split(X_tr_all,
                                                    y_tr_all,
                                                    test_size=0.3,
                                                    random_state=77)
```

In [32]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
import lightgbm as lgb
import numpy as np
import time
```

In [33]:

```
model_list = ["RandomForestRegressor",  "xgb_basic", "lightgbm-model", "GradientBoostingClassifier",
model_score = []
model_time = []
```

In [20]:

```
# print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

로지스틱 모델

In [36]:

```
now_time = time.time()

model = RandomForestRegressor(random_state=30)
model.fit(X_train, y_train)
score = cross_val_score(model, X_train, y_train, cv=5, scoring="roc_auc")
print(score)

pro_time = time.time() - now_time

print("걸린 시간 :", pro_time)  # 걸린 시간
print("RandomForestRegressor Score : {}".format(np.mean( score ) ))  # 점수
```

```
[0.7204471  0.73955972 0.73611203 0.72815306 0.71192845]
걸린 시간 : 18.650969743728638
RandomForestRegressor Score : 0.7272400712073421
```

Xgboost 모델

In [50]:

```
now_time = time.time()

xg_reg = xgb.XGBRegressor(objective ='reg:logistic',
            colsample_bytree = 0.3, # 각나무마다 사용하는 feature 비율
            learning_rate = 0.1,
            max_depth = 3,
            alpha = 0.1,
            n_estimators = 100)  # n_estimators=100

xg_reg.fit(X_train, y_train)
score = cross_val_score(xg_reg, X_train, y_train, cv=5, scoring="roc_auc")
print(score)

pro_time = time.time() - now_time

print("걸린 시간 :", pro_time)  # 걸린 시간
print("xgboosting Score : {}".format(np.mean( score ) ))  # 점수
```

```
[0.79390866 0.79191892 0.78006093 0.77233925 0.76897191]
걸린 시간 : 2.395632266998291
xgboosting Score : 0.7814399372194389
```

LightGBM 모델

In [41]:

```
now_time = time.time()

m_lgbm1 = lgb.LGBMRegressor()
m_lgbm1.fit(X_train, y_train)
score = cross_val_score(m_lgbm1, X_train, y_train, cv=5, scoring="roc_auc")
print(score)

pro_time = time.time() - now_time

print("걸린 시간 :", pro_time)  # 걸린 시간
print("LightGBM 모델 Score : {}".format(np.mean( score ) ))  # 점수
```

```
[0.77682267 0.78360496 0.77297815 0.76017757 0.75535851]
걸린 시간 : 1.3089540004730225
LightGBM 모델 Score : 0.7697883701739261
```

최종 모델 예측

In [94]:

```
model = xgb.XGBRegressor(objective ='reg:logistic',
            colsample_bytree = 0.3, # 각나무마다 사용하는 feature 비율
            learning_rate = 0.1,
            max_depth = 3,
            alpha = 0.1,
            n_estimators = 100)  # n_estimators=100

model.fit(X_train, y_train)

pred = model.predict(X_test_all)
pred
```

Out[94]:

```
array([0.1073395 , 0.28856385, 0.01607929, ..., 0.31839067, 0.23990224,
       0.12781414], dtype=float32)
```

In [95]:

```
pred = np.where(pred > 0.32, 1, 0)
np.sum(pred==1)
```

Out[95]:

```
1451
```

In [96]:

```
sub['prediction'] = pred
sub.to_csv("thirdSub4th_xgb2.csv", index=False)
```

In [ ]: