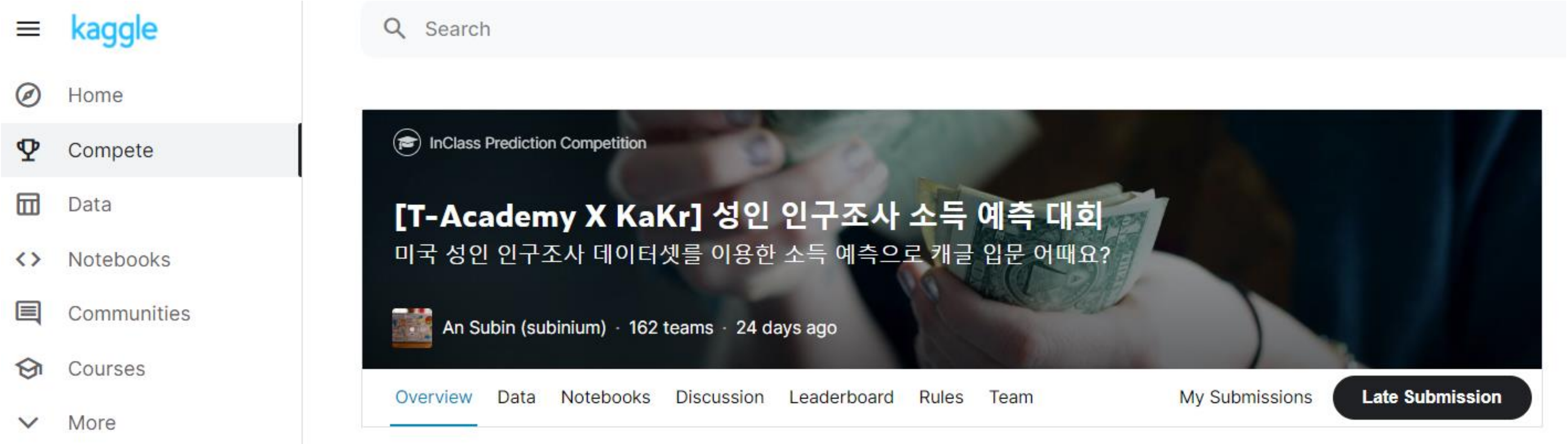
inBig

머신러닝 3팀

박병현 이은성 최영효 한영동 김아연



미국 성인 인구조사 소득 예측 대회 – 목차

데이터 정의 파악

데이터 EDA

Feature Engineering

Modeling

미국 성인 인구조사 소득 예측 대회

Train.csv

id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
0	40	Private	168538	HS-grad	9	Married-civ-spou	Sales	Husband	White	Male	0	0	60	United-States	>50K
1	17	Private	101626	9th	5	Never-married	Machine-op-ins	Own-child	White	Male	0	0	20	United-States	<=50K
2	18	Private	353358	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	16	United-States	<=50K
3	21	Private	151158	Some-college	10	Never-married	Prof-specialty	Own-child	White	Female	0	0	25	United-States	<=50K
4	24	Private	122234	Some-college	10	Never-married	Adm-clerical	Not-in-family	Black	Female	0	0	20	?	<=50K
5	43	Private	236985	HS-grad	9	Married-civ-spou	Craft-repair	Husband	Black	Male	0	0	40	United-States	<=50K
6	43	State-gov	206139	Bachelors	13	Married-civ-spou	Adm-clerical	Husband	White	Male	0	0	50	United-States	>50K
7	37	Private	340599	11th	7	Separated	Other-service	Unmarried	Black	Female	0	0	40	United-States	<=50K
8	47	Private	230136	HS-grad	9	Married-civ-spou	Other-service	Husband	Black	Male	0	0	60	United-States	>50K
9	41	Private	153031	Some-college	10	Married-civ-spou	Sales	Husband	White	Male	0	0	65	United-States	>50K
10	34	Private	238376	1st-4th	2	Married-civ-spou	Craft-repair	Husband	White	Male	0	0	40	Mexico	<=50K

- age: 나이
- workclass: 고용 형태
- fnlwgt: 사람 대표성을 나타내는 가중치 (final weight의 약자)
- education: 교육 수준
- 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)
  - `>50K`
  - `<=50K`

미국 성인 인구조사 소득 예측 대회

Train.csv

id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
0	40	Private	168538	HS-grad	9	Married-civ-spou	Sales	Husband	White	Male	0	0	60	United-States	>50K
1	17	Private	101626	9th	5	Never-married	Machine-op-ins	Own-child	White	Male	0	0	20	United-States	<=50K
2	18	Private	353358	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	16	United-States	<=50K
3	21	Private	151158	Some-college	10	Never-married	Prof-specialty	Own-child	White	Female	0	0	25	United-States	<=50K
4	24	Private	122234	Some-college	10	Never-married	Adm-clerical	Not-in-family	Black	Female	0	0	20	?	<=50K
5	43	Private	236985	HS-grad	9	Married-civ-spou	Craft-repair	Husband	Black	Male	0	0	40	United-States	<=50K
6	43	State-gov	206139	Bachelors	13	Married-civ-spou	Adm-clerical	Husband	White	Male	0	0	50	United-States	>50K
7	37	Private	340599	11th	7	Separated	Other-service	Unmarried	Black	Female	0	0	40	United-States	<=50K
8	47	Private	230136	HS-grad	9	Married-civ-spou	Other-service	Husband	Black	Male	0	0	60	United-States	>50K
9	41	Private	153031	Some-college	10	Married-civ-spou	Sales	Husband	White	Male	0	0	65	United-States	>50K
10	34	Private	238376	1st-4th	2	Married-civ-spou	Craft-repair	Husband	White	Male	0	0	40	Mexico	<=50K

test.csv

Income : 예측하고자 하는 종속변수

id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gai	capital_los	hours_per_	native_country	income
0	28	Private	67661	Some-college	10	Never-married	Adm-clerical	Other-relative	White	Female	0	0	40	United-States	?
1	40	Self-emp-inc	37869	HS-grad	9	Married-civ-spou	Exec-manageria	Husband	White	Male	0	0	50	United-States	?
2	20	Private	109952	Some-college	10	Never-married	Handlers-clean	Own-child	White	Male	0	0	25	United-States	?
3	40	Private	114537	Assoc-voc	11	Married-civ-spou	Exec-manageria	Husband	White	Male	0	0	50	United-States	?
4	37	Private	51264	Doctorate	16	Married-civ-spou	Prof-specialty	Husband	White	Male	0	0	99	France	?
5	36	Private	279615	Bachelors	13	Divorced	Sales	Own-child	White	Female	0	0	40	United-States	?
6	49	Private	87928	HS-grad	9	Married-civ-spou	Adm-clerical	Husband	White	Male	0	0	40	United-States	?
7	26	Private	130620	Assoc-acdm	12	Married-spouse-	Craft-repair	Other-relative	Asian-Pac-	Female	0	0	40	?	?
8	45	Private	28119	HS-grad	9	Married-civ-spou	Adm-clerical	Husband	White	Male	0	0	7	United-States	?
9	39	Private	236136	Assoc-voc	11	Divorced	Adm-clerical	Unmarried	Black	Female	0	0	40	United-States	?
10	57	Private	133126	Some-college	10	Never-married	Craft-repair	Not-in-family	Black	Female	0	0	40	United-States	?
11	36	?	216256	HS-grad	9	Married-civ-spou	?	Husband	White	Male	3464	0	30	United-States	?
12	52	Private	190333	Some-college	10	Married-civ-spou	Adm-clerical	Husband	White	Male	0	0	40	United-States	?

미국 성인 인구조사 소득 예측 대회 EDA

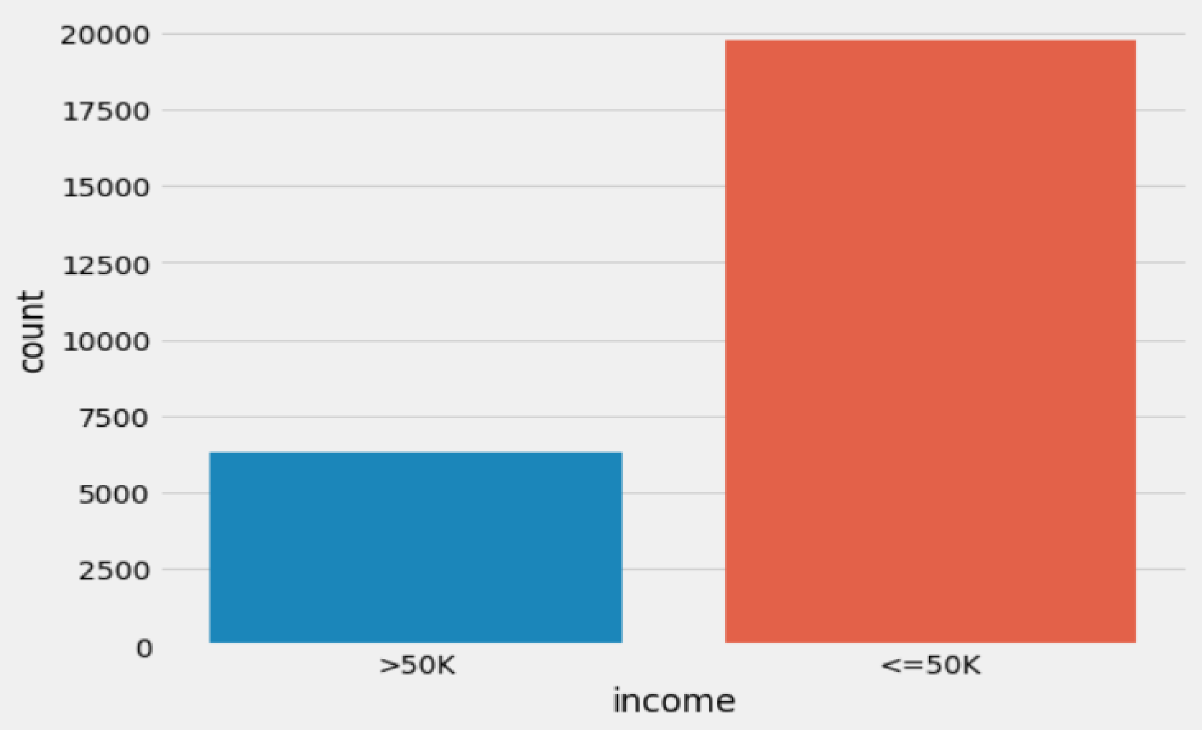
In [5]:

```
# 대략적인 데이터 살펴보기
tmp_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26049 entries, 0 to 26048
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              26049 non-null  int64
 1   age             26049 non-null  int64
 2   workclass       26049 non-null  object
 3   fnlwgt          26049 non-null  int64
 4   education       26049 non-null  object
 5   education_num   26049 non-null  int64
 6   marital_status  26049 non-null  object
 7   occupation      26049 non-null  object
 8   relationship    26049 non-null  object
 9   race            26049 non-null  object
 10  sex             26049 non-null  object
 11  capital_gain    26049 non-null  int64
 12  capital_loss    26049 non-null  int64
 13  hours_per_week  26049 non-null  int64
 14  native_country  26049 non-null  object
 15  income          26049 non-null  object
dtypes: int64(7), object(9)
memory usage: 3.2+ MB
```

```
sns.countplot('income', data=tmp_train)
print(tmp_train['income'].value_counts())
```

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



미국 성인 인구조사 소득 예측 대회 EDA (변수 탐색) - age

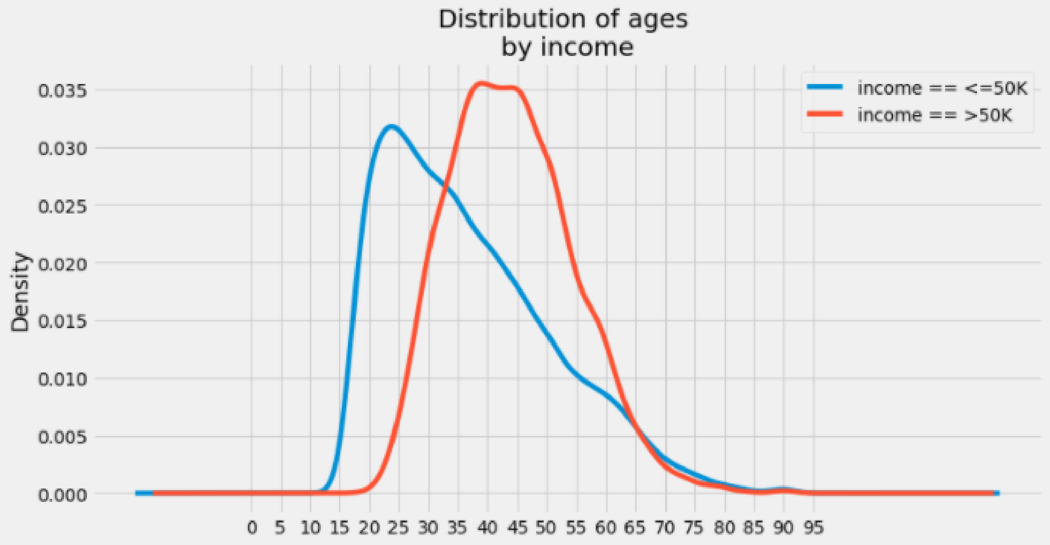
age

- 낮은 연령대에서는 Target이 0, 높은 연령대에서는 Target이 1인 경향을 보인다.
- Titanic 대회처럼 `age_band` 변수를 만들어 연령대 범위를 나눠서 적용해봤지만 그대로 사용할 때 Feature Importance가 높게 나왔다. 모델 스코어에 대한 결과는 급하게 적용하느라 제대로 확인을 못했다.

```python
print('최소 연령: ', tmp_train['age'].min())
print('최고 연령: ', tmp_train['age'].max())
print('평균 연령: {:.2f}'.format(tmp_train['age'].mean()))
```

```
최소 연령:  17
최고 연령:  90
평균 연령: 38.57
```

```python
# 소득에 따른 연령 구분

plt.figure(figsize=(12, 6))

tmp_train.loc[tmp_train['income'] == '<=50K', 'age'].plot(kind='kde', label='income == <=50K')
tmp_train.loc[tmp_train['income'] == '>50K', 'age'].plot(kind='kde', label='income == >50K')

plt.title('Distribution of ages \nby income')
plt.xticks(range(0, 100, 5))
plt.legend()
plt.show()
```

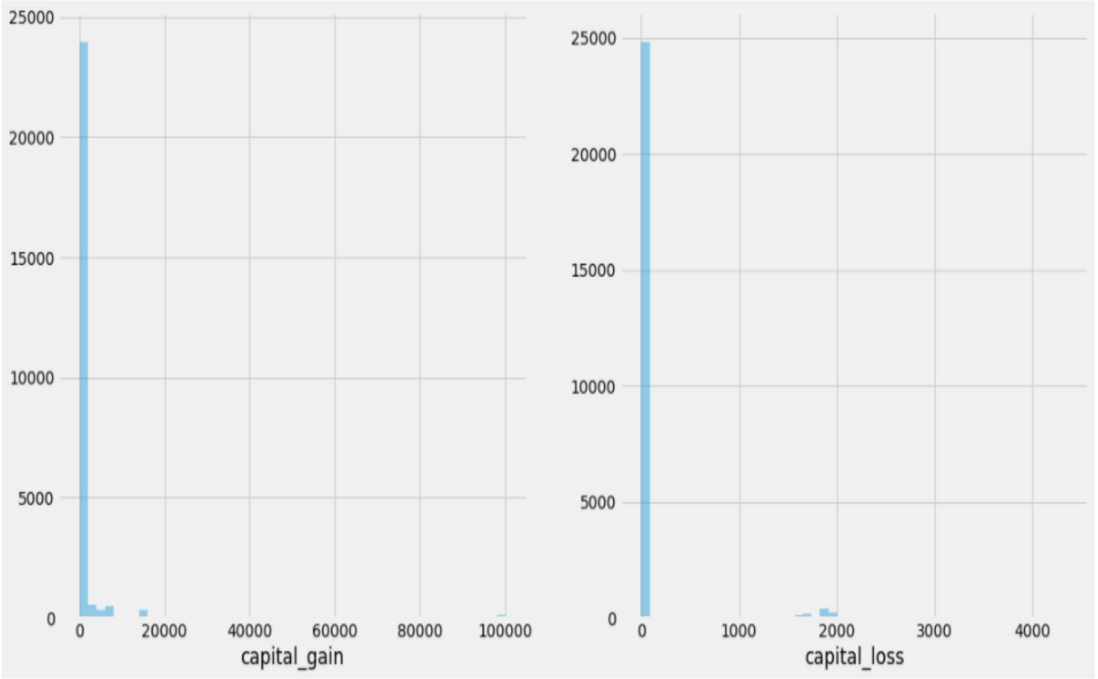


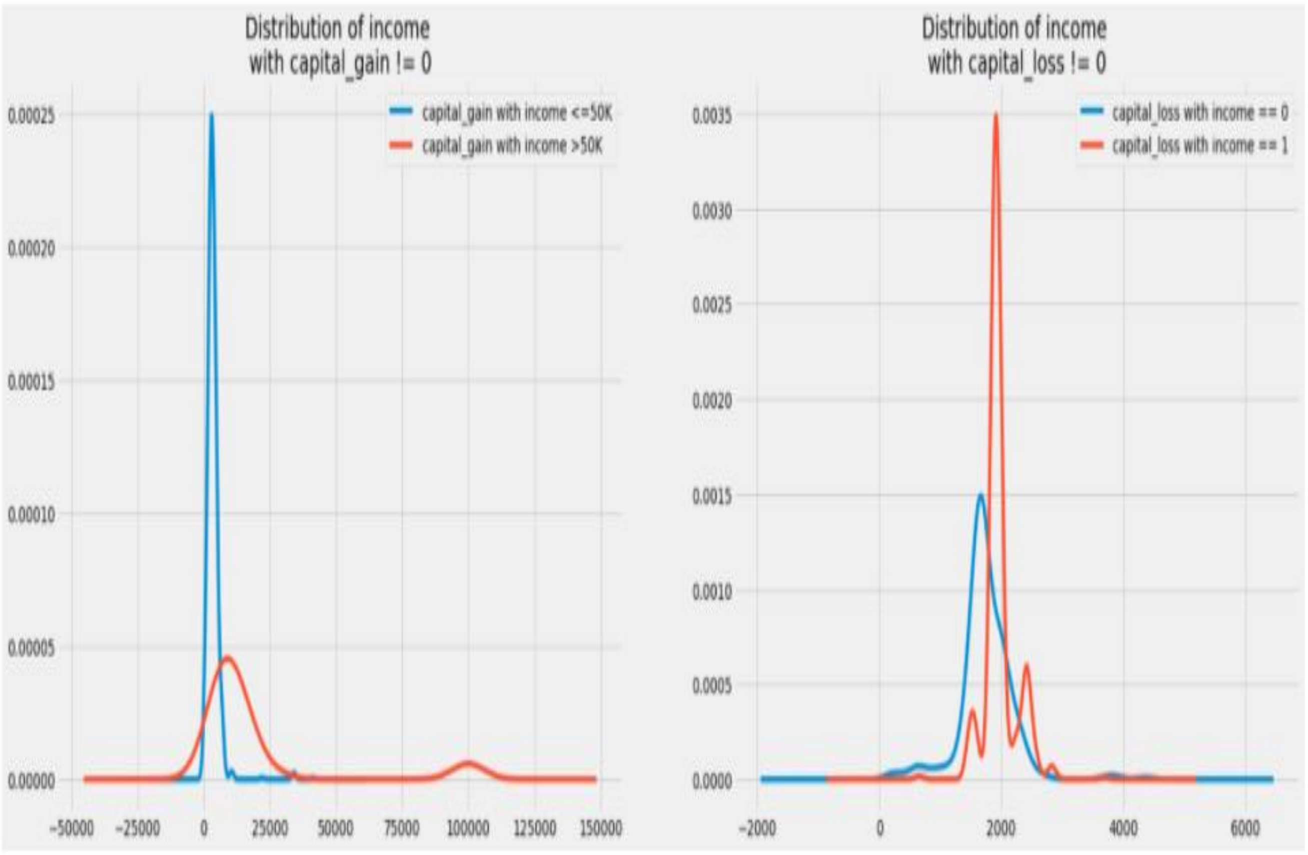

```python
def make_age_band(df):
    df['age_band'] = 0
    df.loc[df['age'] < 20, 'age_band'] = 10
    df.loc[(df['age'] >= 20) & (df['age'] < 30), 'age_band'] = 20
    df.loc[(df['age'] >= 30) & (df['age'] < 40), 'age_band'] = 30
    df.loc[(df['age'] >= 40) & (df['age'] < 50), 'age_band'] = 40
    df.loc[(df['age'] >= 50) & (df['age'] < 60), 'age_band'] = 50
    df.loc[(df['age'] >= 60) & (df['age'] < 70), 'age_band'] = 60
    df.loc[(df['age'] >= 70) & (df['age'] < 80), 'age_band'] = 70
    df.loc[(df['age'] >= 80) & (df['age'] < 90), 'age_band'] = 80
    df.loc[(df['age'] >= 90), 'age_band'] = 90

    return df

tmp_train = make_age_band(tmp_train)
```

미국 성인 인구조사 소득 예측 대회 EDA (변수 탐색) – Capital_gain & loss

capital_gain & capital_los (양도 소득, 양도 손실)

```
fig, ax = plt.subplots(1, 2, figsize=(16, 8))

sns.distplot(tmp_train['capital_gain'],kde=False, ax=ax[0])
sns.distplot(tmp_train['capital_loss'],kde=False, ax=ax[1])

plt.show()
```





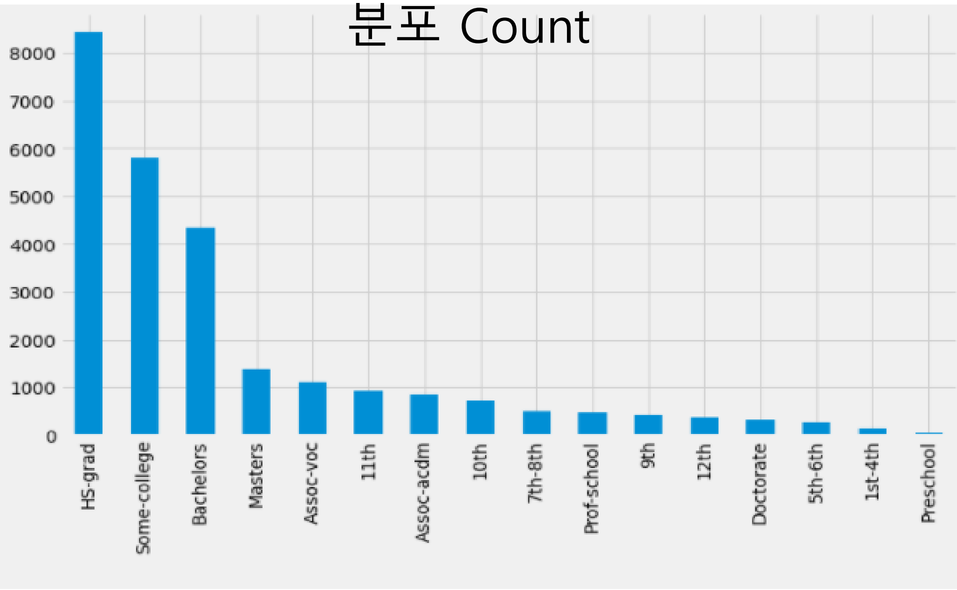
- 0값이 상당히 많음
- **양도소득 + 양도손실로 새로운 Column 생성 => Log**

> 양도세는 중요한 변수로 판단 가능

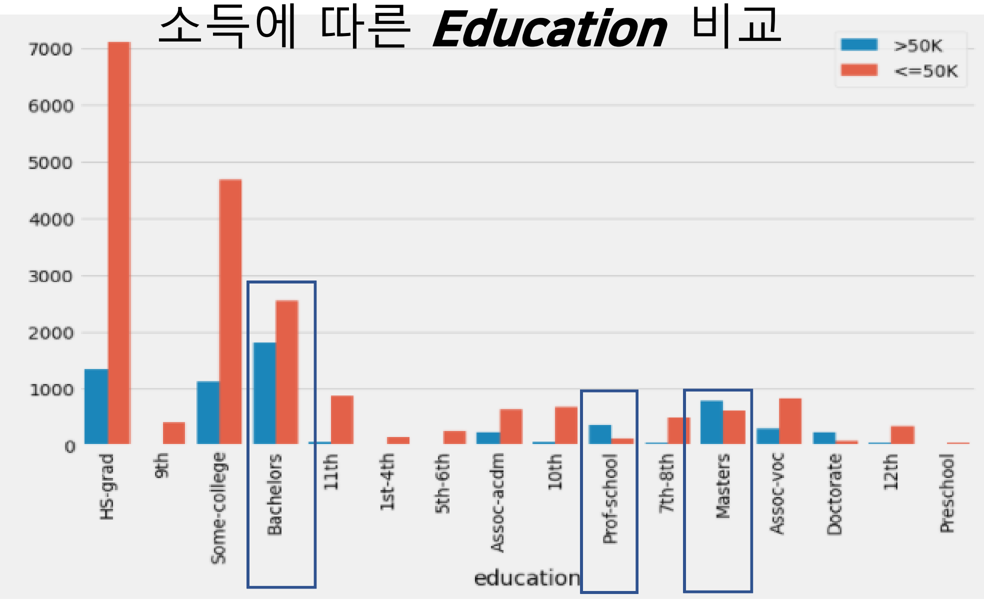
미국 성인 인구조사 소득 예측 대회 EDA (변수 탐색) - Education

```python
# education 분포 확인
fig, ax = plt.subplots(1, 2, figsize=(24, 6))

tmp_train['education'].value_counts().plot(kind='bar', ax=ax[0])

# 타겟에 따른 분포 확인
sns.countplot(x='education', hue='income', data=tmp_train, ax=ax[1])
ax[1].tick_params(axis='x', labelrotation=90)
ax[1].legend(loc='upper right')
ax[1].set_ylabel('')

plt.show()
```

분포 Count



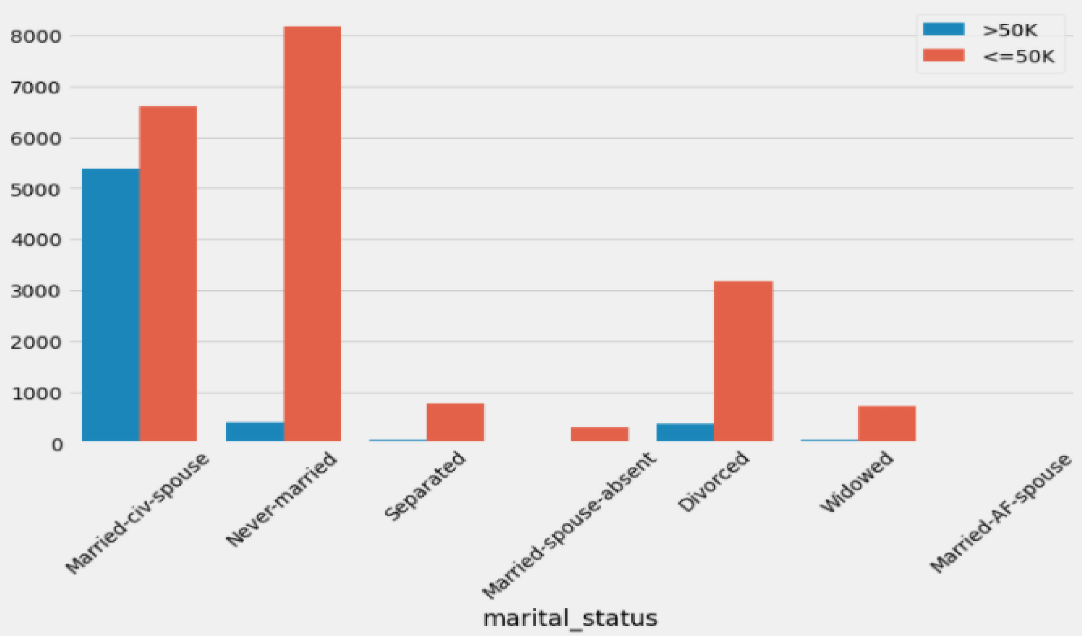
소득에 따른 ***Education*** 비교

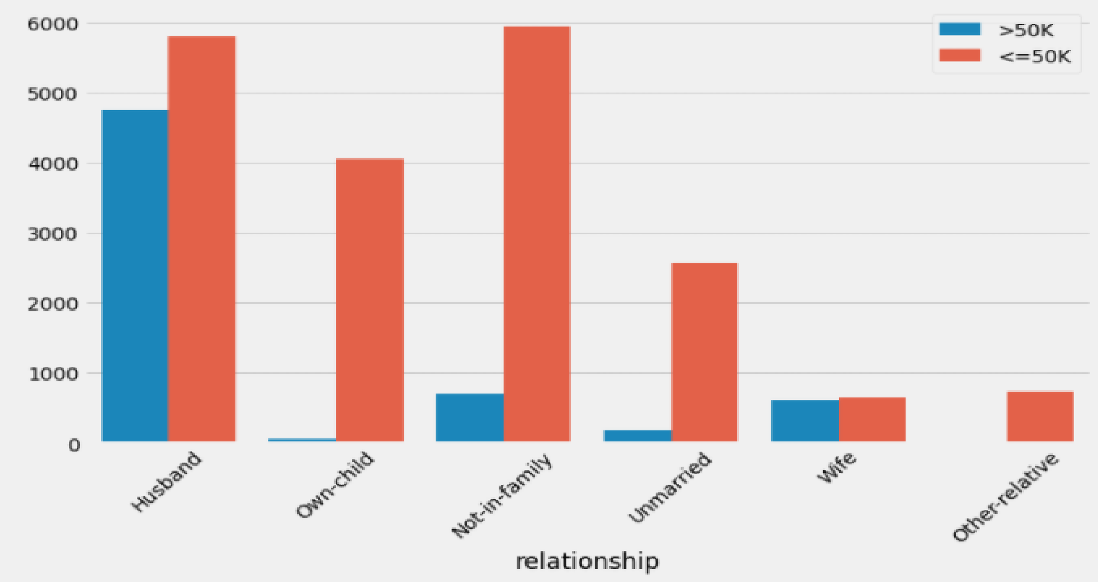


Education 특정값에서는 고소득자가 많음을 확인 가능

미국 성인 인구조사 소득 예측 대회 EDA (변수 탐색) – 결혼 여부

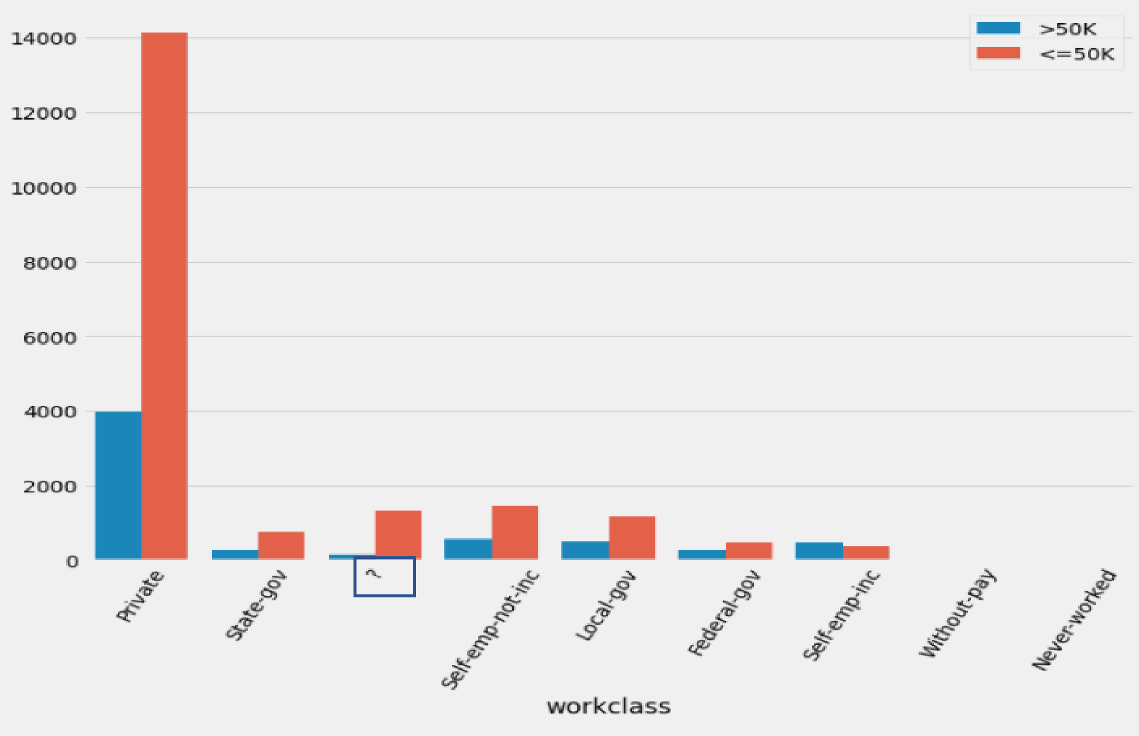
```
fig, ax = plt.subplots(1, 2, figsize=(24, 6))

sns.countplot(x='marital_status', hue='income', data=tmp_train, ax=ax[0])
ax[0].tick_params(axis='x', labelrotation=45)
ax[0].legend()
ax[0].set_ylabel('')

sns.countplot(x='relationship', hue='income', data=tmp_train, ax=ax[1])
ax[1].tick_params(axis='x', labelrotation=45)
ax[1].legend()
ax[1].set_ylabel('')

plt.show()
```



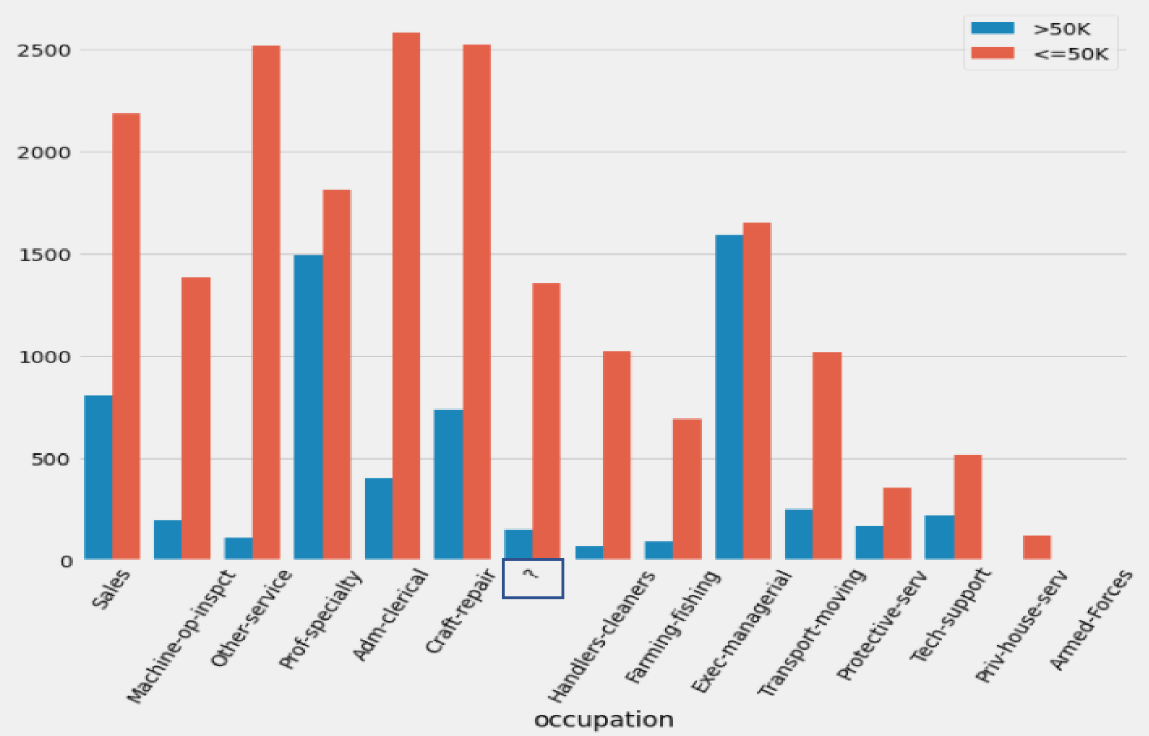


미국 성인 인구조사 소득 예측 대회 EDA (변수 탐색) – Work Class & Occupation

```
# workclass와 occupation 데이터 분포 확인
fig, ax = plt.subplots(1, 2, figsize=(24, 8))

sns.countplot('workclass', hue='income', data=tmp_train, ax=ax[0])
ax[0].tick_params(axis='x', labelrotation=60)
ax[0].set_ylabel('')
ax[0].legend(loc='upper right')

sns.countplot('occupation', hue='income', data=tmp_train, ax=ax[1])
ax[1].tick_params(axis='x', labelrotation=60)
ax[1].set_ylabel('')
ax[1].legend(loc='upper right')

plt.show()
```



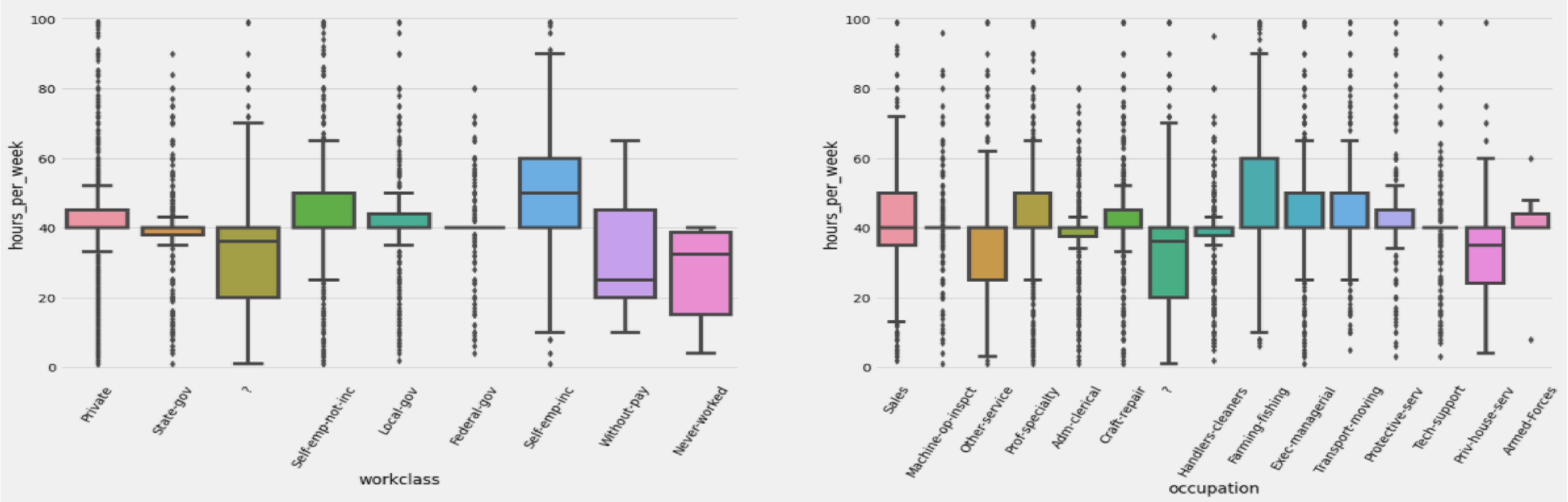
?



결측치 처리

```
# workclass 및 occupation과의 관계를 보기 위해 그려봄

fig, ax = plt.subplots(1, 2, figsize=(24, 8))

sns.boxplot(x='workclass', y='hours_per_week', data=tmp_train, ax=ax[0])
ax[0].tick_params(axis='x', labelrotation=60)

sns.boxplot(x='occupation', y='hours_per_week', data=tmp_train, ax=ax[1])
ax[1].tick_params(axis='x', labelrotation=60)

plt.show()
```

Y 축 : 주 평균 노동시간



Feature Engineering

<적용 사항>

- **income**
- **age_band** 변수
- **education_num** 삭제
- **capital_log** 변수
- **native_country_bin** 변수

- LabelEncoding (Ordinal Encoding 이라는 개념이 있었는데 다음번에 적용해볼만 할 것 같다.) **- workclass**, **education**, **marital_status**, **occupation**, **relationship**, **race**

- Scaling **- fnlwgt**

- '?' 값 **- workclass**와 **occupation**의 6개의 값을 제외하고는 그대로 사용했다.

Modeling – Classification (이진 분류)

- RandomForest (66.5%)
- LightGBM (87.5%)
- CatBoost (87.4%)

Random Forest

```
rf = RandomForestClassifier(n_jobs=-1, random_state=seed)

rf.fit(X_train, y_train)
y_pred = rf.predict(X_valid)
y_pred_proba = rf.predict_proba(X_valid)

rf_score = f1_score(y_valid, y_pred)
print("RF 기본 성능 확인 스코어: {:.4f}".format(rf_score))
```

```
RF 기본 성능 확인 스코어: 0.6650
```

```
get_clf_eval(y_valid, y_pred, y_pred_proba[:, 1])
```

```
오차 행렬
[[4596  342]
 [ 620  955]]
정확도: 0.8523, 정밀도: 0.7363, 재현율: 0.6063, F1: 0.6650, AUC: 0.9022
```

LightGBM

```
lgbm = LGBMClassifier(n_jobs=-1, random_state=seed)

lgbm.fit(X_train, y_train)
y_pred_lgbm = lgbm.predict(X_valid)
y_pred_proba_lgbm = lgbm.predict_proba(X_valid)[:, 1]

get_clf_eval(y_valid, y_pred_lgbm, y_pred_proba_lgbm)
```

```
오차 행렬
[[4671  267]
 [ 546 1029]]
정확도: 0.8752, 정밀도: 0.7940, 재현율: 0.6533, F1: 0.7168, AUC: 0.9306
```

CatBoost (Categorical Boost)

```
cb = CatBoostClassifier(random_seed=seed, verbose=0)

cb.fit(X_train, y_train)
y_pred_cb = cb.predict(X_valid)
y_pred_proba_cb = cb.predict_proba(X_valid)[:, 1]

get_clf_eval(y_valid, y_pred_cb, y_pred_proba_cb)
```

```
오차 행렬
[[4659  279]
 [ 539 1036]]
정확도: 0.8744, 정밀도: 0.7878, 재현율: 0.6578, F1: 0.7170, AUC: 0.9323
```

Modeling – 하이퍼 파라미터 튜닝 (GirdSearch CV)

LGBM 하이퍼 파라미터 튜닝 - GridSearchCV (0.87353)

https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html (공식 문서 참조)

- **num_leaves**: 가장 메인 파라미터 (일반적으로 **num_leaves** = 2^(**max_depth**)로 깊이별 트리와 동일한 수의 잎을 얻을 수 있다. 실제로는 **num_leaves** < 2^(**max_depth**) 이렇게 두어야함. leaf-wise tree 이기 때문)
- **min_data_in_leaf**: overfitting 방지 매개변수. 최적값은 훈련 샘플 수와 num_leaves에 따라 다른데, 대규모 데이터에서 수백 ~ 수천으로 두면 충분하다.
- **max_depth**: 트리 깊이 조절

<정확도를 위한 변수>

- **max_bin**: 크게 적용(속도는 느려진다.)
- 작은 **learning_rate**, 큰 **num_iterations**
- 큰 **num_leaves** (overfitting이 일어날 수 있다.)
- 데이터 세트 늘리기
- **boosting_type** 을 dart(Dropouts meet Multiple Additive Regression Trees)로 적용 (defalt: 'gbdt'(gradient boosting decision tree))

```
lgbm = LGBMClassifier(random_state=seed)

params = {
    'boosting_type': ['gbdt', 'dart'], # defalt 'gbdt'
    'num_leaves': [20, 31, 50, 70], # default 31
    'max_depth': [-1, 5, 7, 10, 15, 20, 30], # default -1. 끝까지 만드는 것
    'learning_rate': [0.001, 0.01, 0.05, 0.1], # defaut 0.1
}

lgbm_grid_cv = GridSearchCV(lgbm, param_grid=params, scoring='f1', n_jobs=-1, cv=5, verbose=1)
lgbm_grid_cv.fit(X, y)
```