python™

데이터 전처리

- 데이터 전처리
  데이터 분석을 하기 위해 분석하기 좋은 형태로 데이터를 가공하는 일

  모델을 만들기 전에 하는 매우매우매우매우매우매우 중요한 부분
  전처리에 따라 같은 데이터를 같은 모델을 넣어도 결과가 아예 달라질 수 있다!

- 결측치 제거
- 데이터 타입 변환
- 누락 feature 추가
- 부적합 feature 삭제 등등...

데이터 전처리

- 데이터 전처리를 하기 전에 확인해야 할 것
  - 전반적인 데이터 확인
  - 독립 및 종속 변수의 데이터 형태 확인
  - 독립 및 종속 변수의 데이터 분포 확인
  - 결측치(null) 및 이상치(outlier) 확인 등 ...
- 데이터 전처리 때 해야 할 것
  - 데이터 타입 및 변수 변환
  - 인코딩
  - 이상치 및 결측치 처리
  - 누락변수 추가 및 부적합 변수 제거
  - 데이터 스케일링 등 ...

전반적인 데이터 확인

	Unnamed: 0	year	metro	id	sex	age	number	education	marriage	asset	debt	income	income_d	industry	job	house	education_year
0	0	2020	G1	10000112	1	34	3	6	2	112000	54500	6593	4599	F	3	2	16
1	1	2020	G1	10000132	2	45	2	8	2	42500	17500	17720	15257	J	2	3	21
2	2	2020	G1	10000162	2	73	1	2	3	5712	0	908	725	T	4	2	6
3	3	2020	G1	10000182	1	58	2	4	2	14870	0	2748	2431	C	5	2	12
4	4	2020	G1	10000192	2	27	1	4	1	814	0	1015	893	R	2	3	12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18059	18059	2020	G2	993800291	1	42	3	6	2	6705	2200	179	-390	NaN	NaN	3	16
18060	18060	2020	G2	994800251	1	49	5	6	2	130180	22000	15454	11784	O	3	1	16
18061	18061	2020	G2	994800261	1	57	5	6	2	265226	29000	15098	11560	NaN	NaN	1	16
18062	18062	2020	G2	994800291	2	53	2	7	4	109695	25000	9114	7074	O	2	2	18
18063	18063	2020	G2	995800261	1	44	5	6	2	71223	13000	7857	5404	C	3	1	16

- 데이터의 개수는 18064개
- target : “income”
- 필요 없는 데이터 : “Unnamed: 0”, “year”, “id”

독립 및 종속 변수의 데이터 형태 확인

```
df.metro.value_counts()
```
✓ 0.0s

```
G2    12170
G1     5894
Name: metro, dtype: int64
```

```
df.education.value_counts()
```
✓ 0.0s

```
4    5681
6    4120
2    2392
3    2024
5    1918
1     928
7     797
8     204
Name: education, dtype: int64
```

```
df.age.value_counts()
```
✓ 0.0s

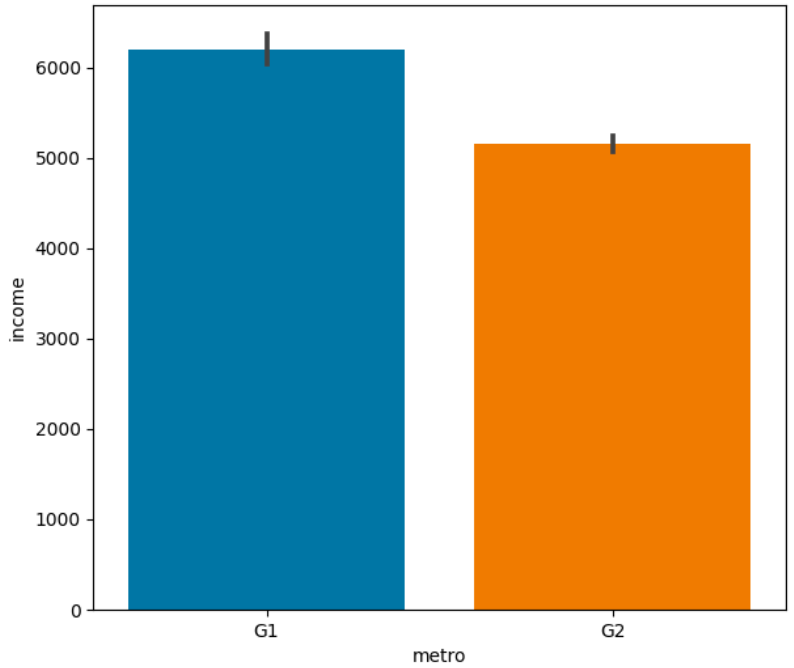
```
60     504
59     504
58     445
62     443
61     443
      ...
96       3
97       2
20       2
98       1
102      1
Name: age, Length: 80, dtype: int64
```

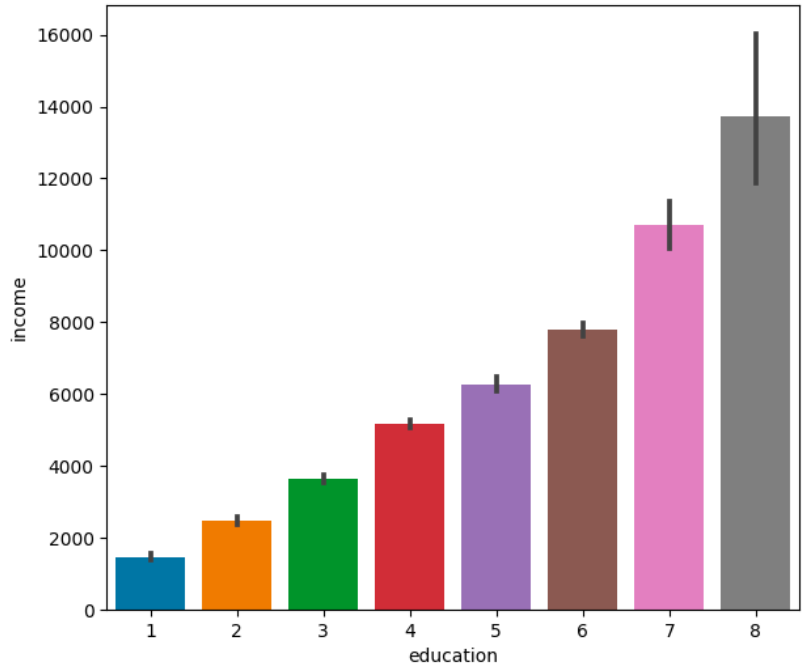
숫자라고 numerical 데이터가 아님
8개 정도면 categorical 데이터
age처럼 80개나 되는 데이터는
numerical 데이터라고 볼 수 있음

- 일반적으로 value_counts를 통해서 확인
- categorical data는 barplot
  numerical data는 distplot으로 시각화를 하는게 일반적이다

python™

독립 및 종속 변수의 데이터 형태 확인

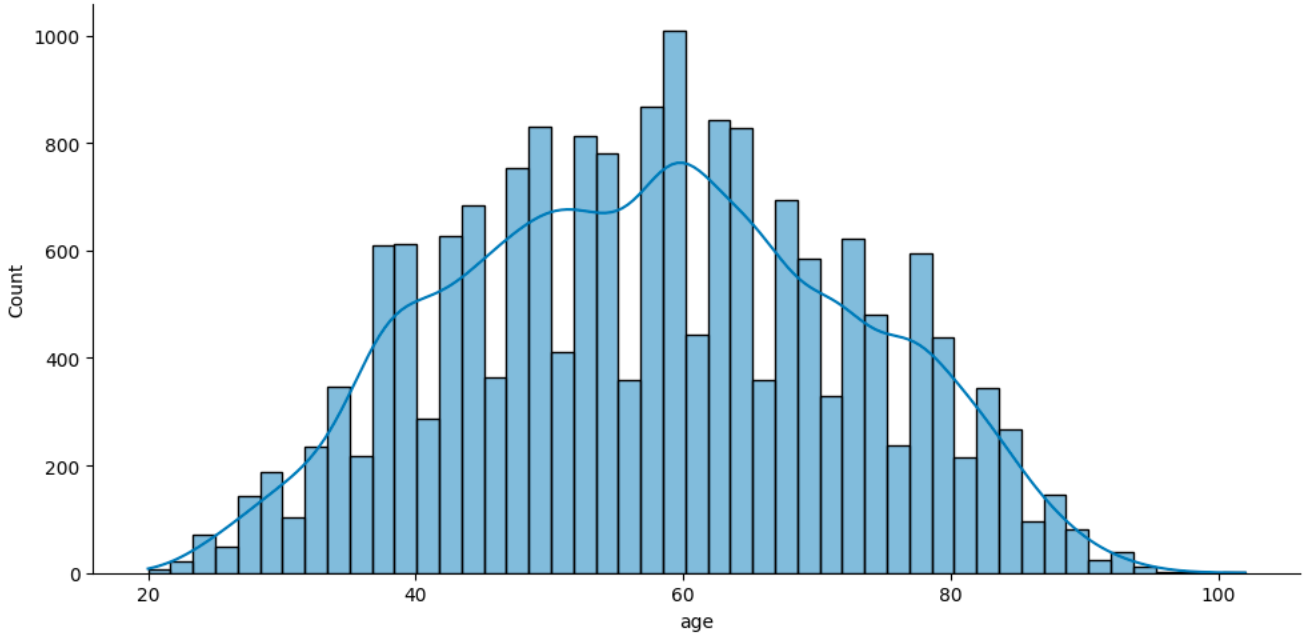




- import seaborn as sns
  sns.barplot(x='metro', y='income', data=df)
- categorical 데이터를 barplot으로 시각화 하면 위와 같이 나타난다

python™

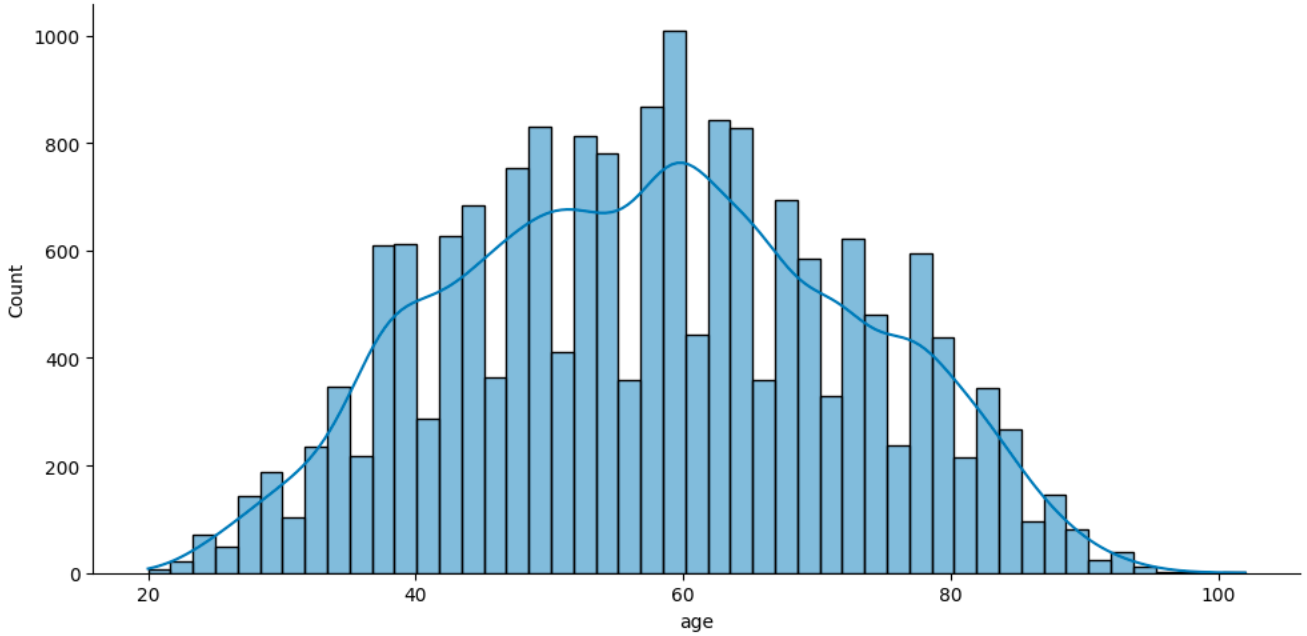
독립 및 종속 변수의 데이터 형태 확인



- import seaborn as sns
  sns.displot(df.age, height=5, aspect=2, kde=True)
- numeric 데이터를 distplot으로 시각화 하면 위와 같이 나타난다

python™

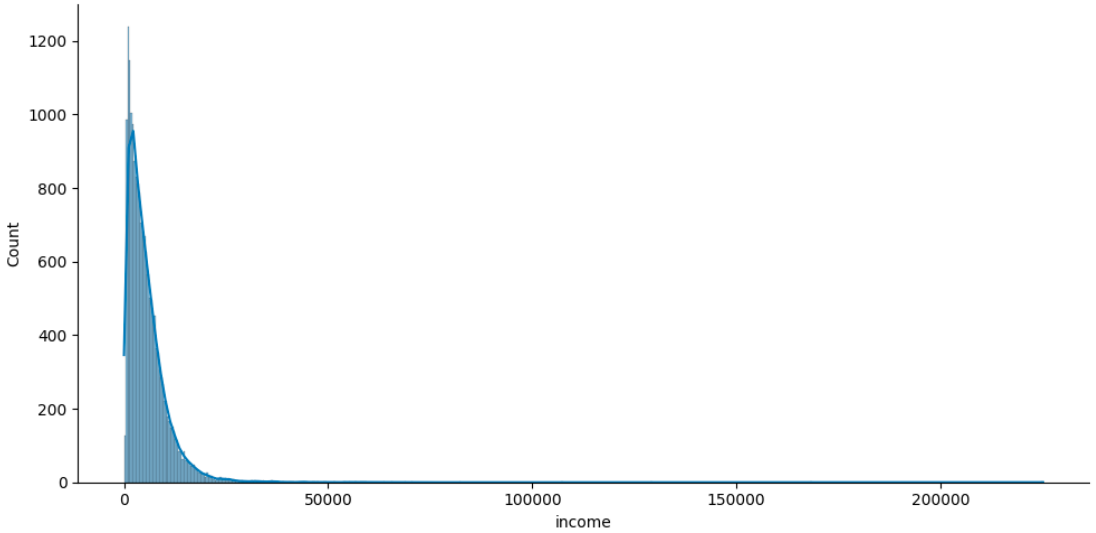
독립 및 종속 변수의 데이터 분포 확인

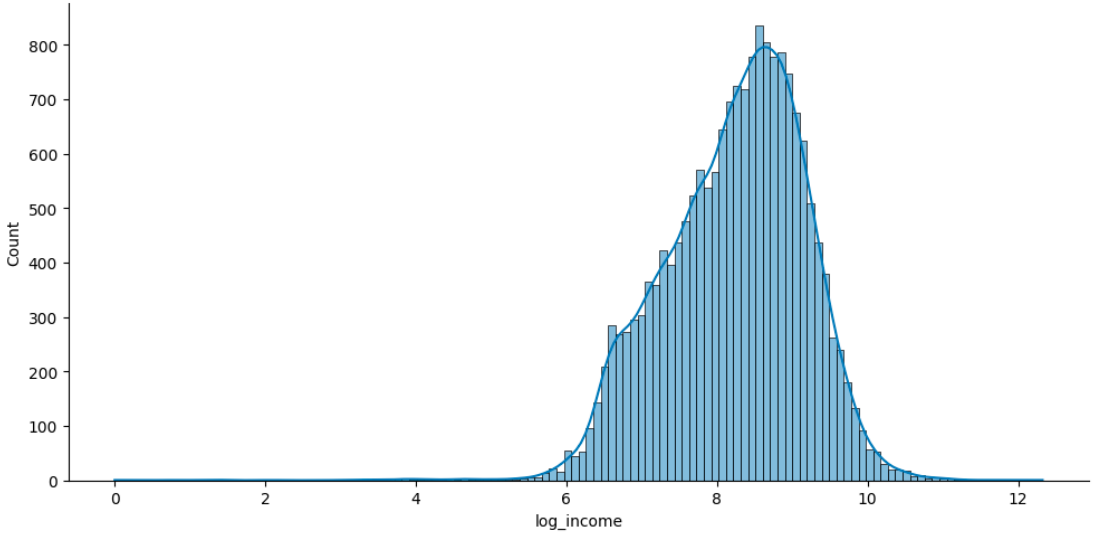


- 데이터들은 정규분포에 근사해야한다
- 정규분포에 근사하지 않는 데이터들은 데이터 변환을 시켜주는 것이 일반적이다

python™

독립 및 종속 변수의 데이터 분포 확인





- import numpy as np
  df['log_income']=np.log(df['income'])
- 정규분포에 근사하지 않는 데이터들은 log 변환을 통해서 근사하게 만든다

python™

독립 및 종속 변수의 데이터 분포 확인

```
df['log_debt']=np.log(df['debt'])
df[df['debt']<=0]
```

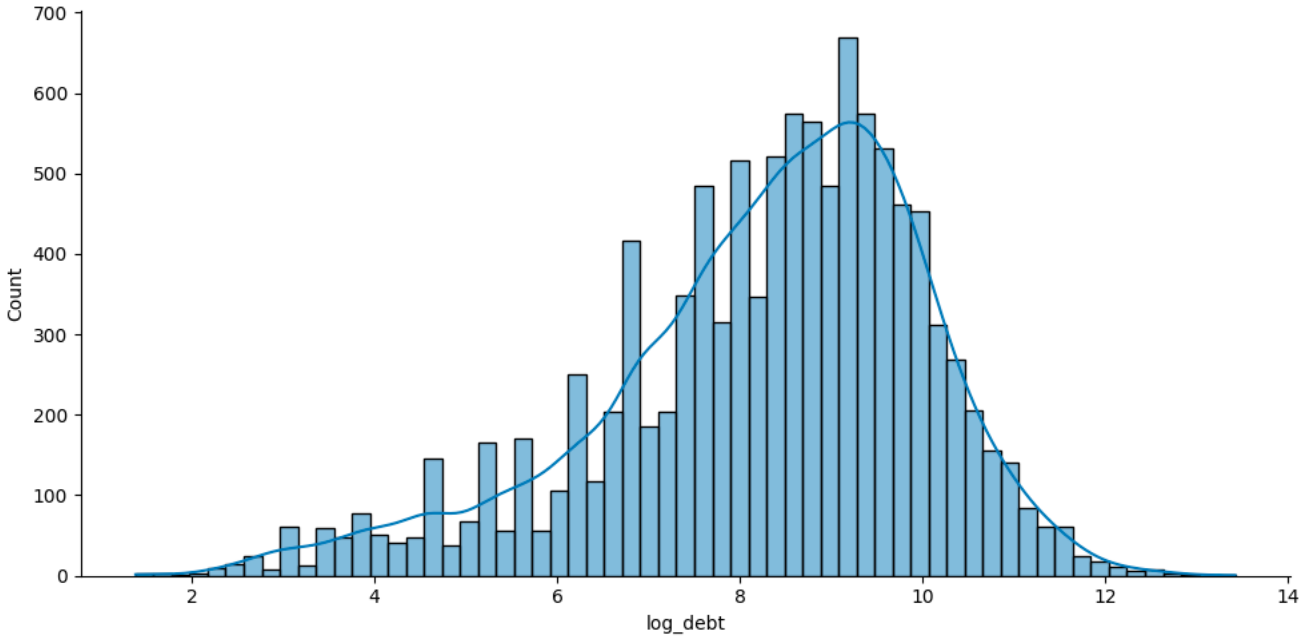
✓ 0.0s

	metro	sex	age	number	education	marriage	asset	debt	income	income_d	industry	job	house	education_year	log_income	log_asset	log_debt
2	G1	2	73	1	2	3	5712	0	908	725	T	4	2	6	6.811244	8.650325	-inf
3	G1	1	58	2	4	2	14870	0	2748	2431	C	5	2	12	7.918629	9.607101	-inf
4	G1	2	27	1	4	1	814	0	1015	893	R	2	3	12	6.922644	6.701960	-inf
9	G2	2	66	2	4	2	57615	0	3805	3236	NaN	NaN	1	12	8.244071	10.961538	-inf
10	G2	1	56	4	5	2	32425	0	11162	8111	O	3	1	14	9.320270	10.386685	-inf
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18050	G2	2	87	1	1	3	10130	0	1112	1057	NaN	NaN	1	0	7.013915	9.223257	-inf
18051	G2	1	65	2	3	2	56490	0	3495	3076	A	6	1	9	8.159089	10.941819	-inf
18052	G2	1	51	2	4	1	14110	0	3087	2686	C	7	1	12	8.034955	9.554639	-inf
18054	G2	2	66	3	4	4	15000	0	5181	4677	P	4	1	12	8.552753	9.615805	-inf
18056	G2	1	68	2	3	2	27428	0	2356	903	S	4	1	9	7.764721	10.219320	-inf

7220 rows × 17 columns

- log변환을 할 때 주의할 점은, 0 이하의 값은 log 변환을 할 수 없다는 것이다
  또한 -inf 값으로 나타낸 데이터는 시각화에 반영되지 못한다

독립 및 종속 변수의 데이터 분포 확인



- log변환을 할 때 주의할 점은, 0 이하의 값은 log 변환을 할 수 없다는 것이다
  또한 -inf 값으로 나타낸 데이터는 시각화에 반영되지 못한다

python™

독립 및 종속 변수의 데이터 분포 확인

```
df['log_debt']=np.log(df['debt']+abs(df['debt'].min()+1))
df[df['debt']<=0]
```

✓ 0.0s

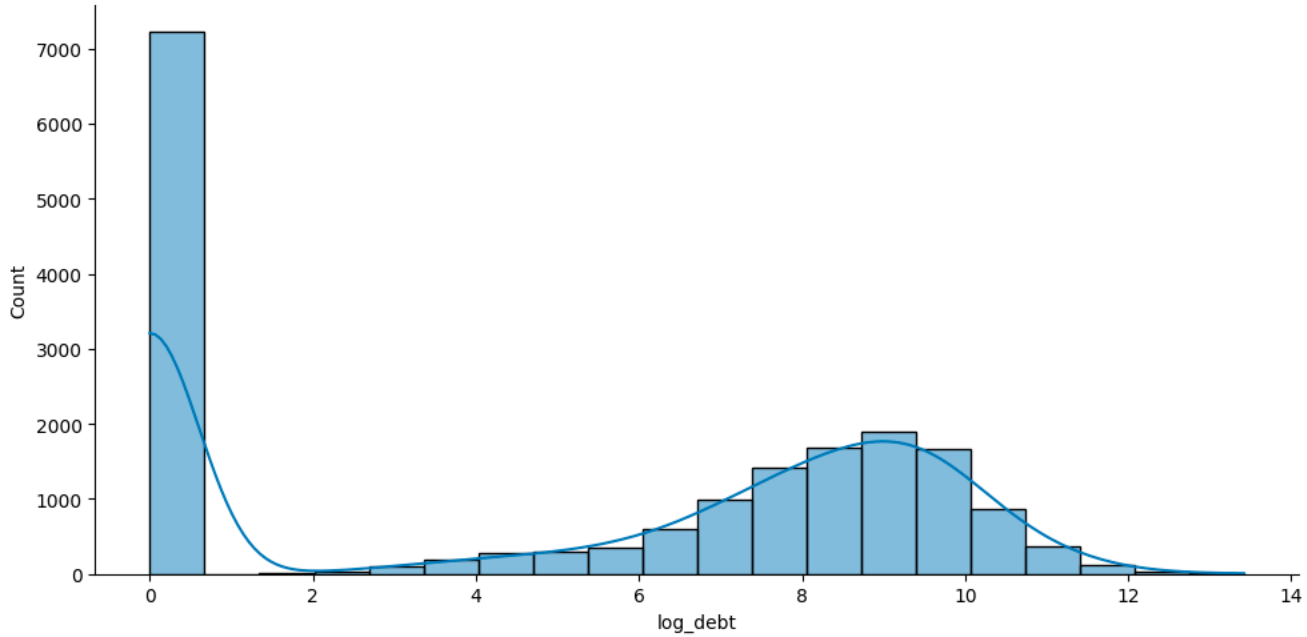
	metro	sex	age	number	education	marriage	asset	debt	income	income_d	industry	job	house	education_year	log_income	log_asset	log_debt
2	G1	2	73	1	2	3	5712	0	908	725	T	4	2	6	6.811244	8.650325	0.0
3	G1	1	58	2	4	2	14870	0	2748	2431	C	5	2	12	7.918629	9.607101	0.0
4	G1	2	27	1	4	1	814	0	1015	893	R	2	3	12	6.922644	6.701960	0.0
9	G2	2	66	2	4	2	57615	0	3805	3236	NaN	NaN	1	12	8.244071	10.961538	0.0
10	G2	1	56	4	5	2	32425	0	11162	8111	O	3	1	14	9.320270	10.386685	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18050	G2	2	87	1	1	3	10130	0	1112	1057	NaN	NaN	1	0	7.013915	9.223257	0.0
18051	G2	1	65	2	3	2	56490	0	3495	3076	A	6	1	9	8.159089	10.941819	0.0
18052	G2	1	51	2	4	1	14110	0	3087	2686	C	7	1	12	8.034955	9.554639	0.0
18054	G2	2	66	3	4	4	15000	0	5181	4677	P	4	1	12	8.552753	9.615805	0.0
18056	G2	1	68	2	3	2	27428	0	2356	903	S	4	1	9	7.764721	10.219320	0.0

7220 rows × 17 columns

- df['log_debt']=np.log(df['debt']+abs(df['debt'].min()+1))
- 이에 대해서는 일반적으로 최소값 +1을 더한 이후 log 변환을 해준다

python™

독립 및 종속 변수의 데이터 분포 확인



- 이에 대해서는 일반적으로 최소값 +1을 더한 이후 log 변환을 해준다
  해당 데이터들은 $\log 1$이라는 값으로 반영되며, 시각화에 반영된다

python™

독립 및 종속 변수의 데이터 분포 확인

```
df['h_debt']=0
df.loc[df['debt']>0, 'h_debt']=1
df
```

✓ 0.0s

	metro	sex	age	number	education	marriage	asset	debt	income	income_d	industry	job	house	education_year	log_income	log_asset	log_debt	h_debt
0	G1	1	34	3	6	2	112000	54500	6593	4599	F	3	2	16	8.793764	11.626254	10.905974	1
1	G1	2	45	2	8	2	42500	17500	17720	15257	J	2	3	21	9.782449	10.657259	9.770013	1
2	G1	2	73	1	2	3	5712	0	908	725	T	4	2	6	6.811244	8.650325	0.000000	0
3	G1	1	58	2	4	2	14870	0	2748	2431	C	5	2	12	7.918629	9.607101	0.000000	0
4	G1	2	27	1	4	1	814	0	1015	893	R	2	3	12	6.922644	6.701960	0.000000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18059	G2	1	42	3	6	2	6705	2200	179	-390	NaN	NaN	3	16	5.187386	8.810609	7.696667	1
18060	G2	1	49	5	6	2	130180	22000	15454	11784	O	3	1	16	9.645623	11.776673	9.998843	1
18061	G2	1	57	5	6	2	265226	29000	15098	11560	NaN	NaN	1	16	9.622318	12.488338	10.275086	1
18062	G2	2	53	2	7	4	109695	25000	9114	7074	O	2	2	18	9.117567	11.605459	10.126671	1
18063	G2	1	44	5	6	2	71223	13000	7857	5404	C	3	1	16	8.969160	11.173571	9.472782	1

- 해당 데이터들은 무시할 수 없을 만큼의 데이터이다
  하지만 빚이 없다는 특징은 target 변수에 꽤 큰 영향을 미칠 수 있다
  따라서 새로운 feature을 추가하여 차별점을 만들어준다

결측치(null) 및 이상치(outlier) 확인

```
df.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18064 entries, 0 to 18063
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   metro           18064 non-null  object
 1   sex             18064 non-null  int64
 2   age             18064 non-null  int64
 3   number          18064 non-null  int64
 4   education       18064 non-null  int64
 5   marriage        18064 non-null  int64
 6   asset           18064 non-null  int64
 7   debt            18064 non-null  int64
 8   income          18064 non-null  int64
 9   income_d        18064 non-null  int64
 10  industry        13214 non-null  object
 11  job             13214 non-null  object
 12  house           18064 non-null  int64
 13  education_year  18064 non-null  int64
 14  log_income      18064 non-null  float64
 15  log_asset       18064 non-null  float64
 16  log_debt        18064 non-null  float64
 17  h_debt          18064 non-null  int64
dtypes: float64(3), int64(12), object(3)
memory usage: 2.5+ MB
```

```
# 데이터 결측치 확인
df.isnull().sum()
```
✓ 0.0s

```
metro                0
sex                  0
age                  0
number               0
education            0
marriage             0
asset                0
debt                 0
income               0
income_d             0
industry          4850
job               4850
house                0
education_year       0
log_income           0
log_asset            0
log_debt             0
h_debt               0
dtype: int64
```

```
(df.values==' ').sum()
```
✓ 0.0s

```
0
```

- 데이터의 형태를 확인해준 이후 isnull함수를 통해 NaN을 살펴본다.
  결측치는 주로 isnull을 통해 찾아낼 수 있지만 때때로 다른 형태로 되어있을 때 있다
  숫자데이터 : NaN, 문자데이터 : None 그 외 : 주로 ‘ ’

python™

결측치(null) 및 이상치(outlier) 확인

```
df[df.income_d<=0]
```

✓ 0.0s

	metro	sex	age	number	education	marriage	asset	debt	income	income_d	industry	job	house	education_year	log_income	log_asset	log_debt	h_debt
122	G2	1	60	1	3	4	130	1000	31	-216	F	9	3	9	3.433987	4.867534	6.908755	1
197	G1	2	75	1	6	3	85490	10000	1371	-211	NaN	NaN	1	16	7.223296	11.356155	9.210440	1
346	G1	2	53	1	4	1	2005	4431	107	-1363	NaN	NaN	3	12	4.672829	7.603399	8.396606	1
367	G1	1	68	3	4	2	224270	173940	4041	-1323	A	6	1	12	8.304247	12.320606	12.066471	1
438	G1	1	37	3	4	2	43220	6600	469	-20	P	8	1	12	6.150603	10.674059	8.794976	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17497	G2	1	65	1	4	1	23300	4032	295	-29	NaN	NaN	1	12	5.686975	10.056209	8.302266	1
17708	G2	2	71	2	3	2	8293	3500	274	-750	NaN	NaN	1	9	5.613128	9.023167	8.160804	1
17992	G2	1	66	4	4	2	252702	200000	7139	-2579	A	6	5	12	8.873328	12.439966	12.206078	1
18001	G2	1	40	1	5	4	49120	0	283	-151	G	9	1	14	5.645447	10.802022	0.000000	0
18059	G2	1	42	3	6	2	6705	2200	179	-390	NaN	NaN	3	16	5.187386	8.810609	7.696667	1

109 rows × 18 columns

- 이상치 데이터에 대해서 확인해본다.
  이상치는 '내가 이상하다고 여기는 경우' 와 IQR기반으로 찾아내는 방법이 있다.
  outlier는 일반적으로 제거해주는 것이 일반적이다.

결측치(null) 및 이상치(outlier) 확인

```
df=df[df.income_d>=0]
df
```

✓ 0.0s

	metro	sex	age	number	education	marriage	asset	debt	income	income_d	industry	job	house	education_year	log_income	log_asset	log_debt	h_debt
0	G1	1	34	3	6	2	112000	54500	6593	4599	F	3	2	16	8.793764	11.626254	10.905974	1
1	G1	2	45	2	8	2	42500	17500	17720	15257	J	2	3	21	9.782449	10.657259	9.770013	1
2	G1	2	73	1	2	3	5712	0	908	725	T	4	2	6	6.811244	8.650325	0.000000	0
3	G1	1	58	2	4	2	14870	0	2748	2431	C	5	2	12	7.918629	9.607101	0.000000	0
4	G1	2	27	1	4	1	814	0	1015	893	R	2	3	12	6.922644	6.701960	0.000000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18058	G2	1	37	4	6	2	13720	4100	7453	6083	C	2	3	16	8.916372	9.526610	8.318986	1
18060	G2	1	49	5	6	2	130180	22000	15454	11784	O	3	1	16	9.645623	11.776673	9.998843	1
18061	G2	1	57	5	6	2	265226	29000	15098	11560	NaN	NaN	1	16	9.622318	12.488338	10.275086	1
18062	G2	2	53	2	7	4	109695	25000	9114	7074	O	2	2	18	9.117567	11.605459	10.126671	1
18063	G2	1	44	5	6	2	71223	13000	7857	5404	C	3	1	16	8.969160	11.173571	9.472782	1

17956 rows × 18 columns

- 당연히 이상치를 제거할 땐, 이상치의 양이 전체 데이터에 대비하여 크지 않아야 한다.
  양이 매우 많다면, 이상치가 아니라 '특징이 있는' 데이터가 된다

인코딩

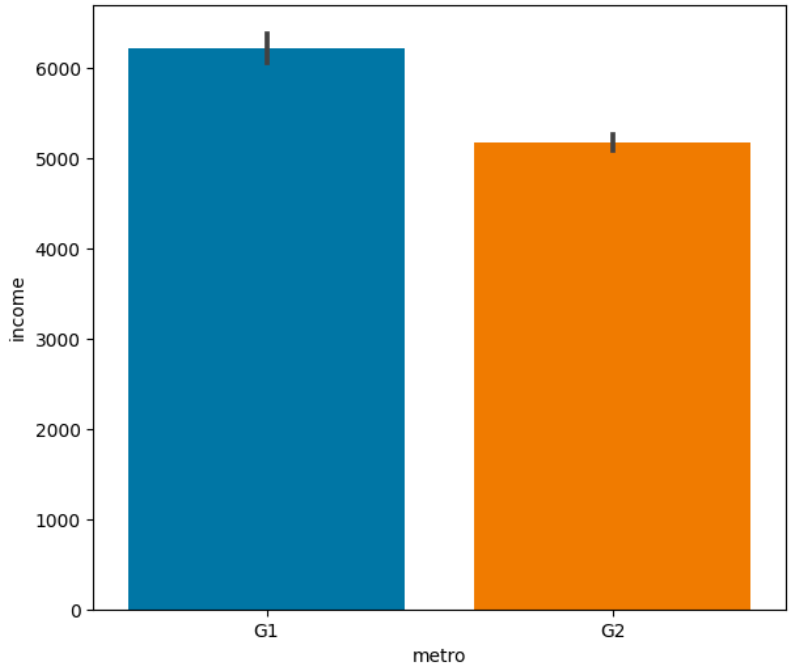
- Categorical 데이터에 대해서는 encoding을 따로 해주어야 한다
  크게 label encoding / one-hot encoding 두 가지 방식을 사용한다

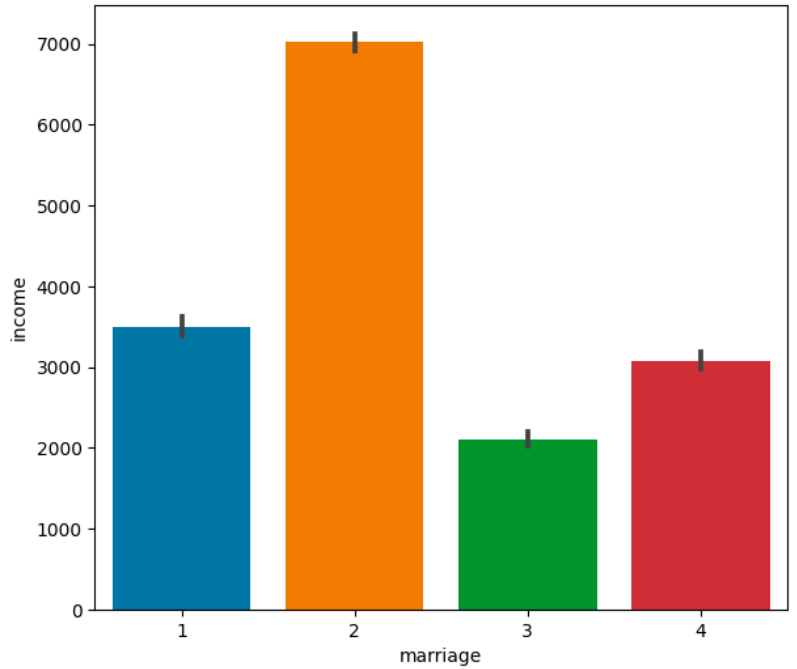
- Label Encoding
  categorical 데이터의 문자열을 그대로 숫자형으로 변환

- One-Hot Encoding
  feature값의 유형에 따라 새로운 feature(더미변수)을 추가하여
  고유값에 해당하는 column에만 1을 부여하고 나머지에는 0을 부여하는 방법

- 다만, Label Encoding이 되어 있어도 One-Hot Encoding을 하는 경우가 잦다

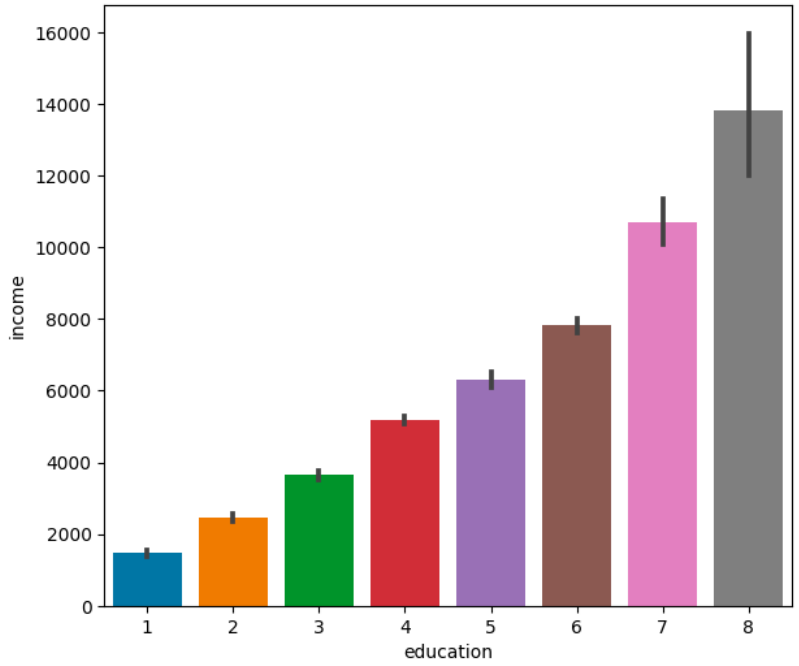
인코딩

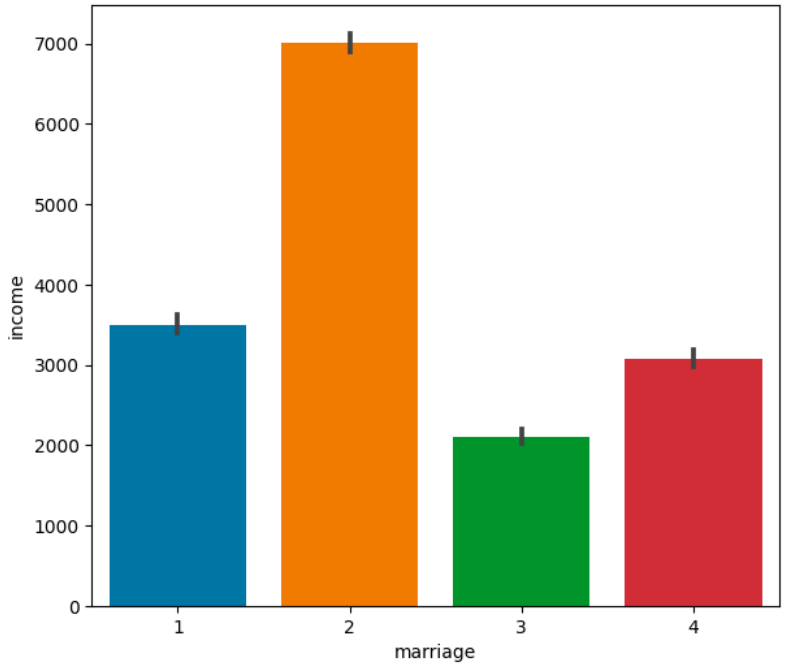




- metro의 경우 label encoding이 되어있지 않고, marriage는 label encoding이 되어있다.

인코딩





- education은 label encoding만으로도 충분하겠지만 marriage는 one-hot encoding을 할 필요성이 있다

python™

인코딩

```
# label encoding
df=df.applymap(lambda x : 0 if x == 'G1' else(1 if x=='G2' else x))
df.sex=df.sex.apply(lambda x : 0 if x == 1 else(1 if x==2 else x))
df
```

✓ 0.1s Python

	metro	sex	age	number	education	marriage	asset	debt	income	income_d	industry	job	house	education_year	log_income	log_asset	log_debt	h_debt
0	0	0	34	3	6	2	112000	54500	6593	4599	F	3	2	16	8.793764	11.626254	10.905974	1
1	0	1	45	2	8	2	42500	17500	17720	15257	J	2	3	21	9.782449	10.657259	9.770013	1
2	0	1	73	1	2	3	5712	0	908	725	T	4	2	6	6.811244	8.650325	0.000000	0
3	0	0	58	2	4	2	14870	0	2748	2431	C	5	2	12	7.918629	9.607101	0.000000	0
4	0	1	27	1	4	1	814	0	1015	893	R	2	3	12	6.922644	6.701960	0.000000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18058	1	0	37	4	6	2	13720	4100	7453	6083	C	2	3	16	8.916372	9.526610	8.318986	1
18060	1	0	49	5	6	2	130180	22000	15454	11784	O	3	1	16	9.645623	11.776673	9.998843	1
18061	1	0	57	5	6	2	265226	29000	15098	11560	NaN	NaN	1	16	9.622318	12.488338	10.275086	1
18062	1	1	53	2	7	4	109695	25000	9114	7074	O	2	2	18	9.117567	11.605459	10.126671	1
18063	1	0	44	5	6	2	71223	13000	7857	5404	C	3	1	16	8.969160	11.173571	9.472782	1

17956 rows × 18 columns

- apply함수를 이용하여 label encoding을 실시.
  metro에 대해선 필수, sex에 대해선 필수x

인코딩

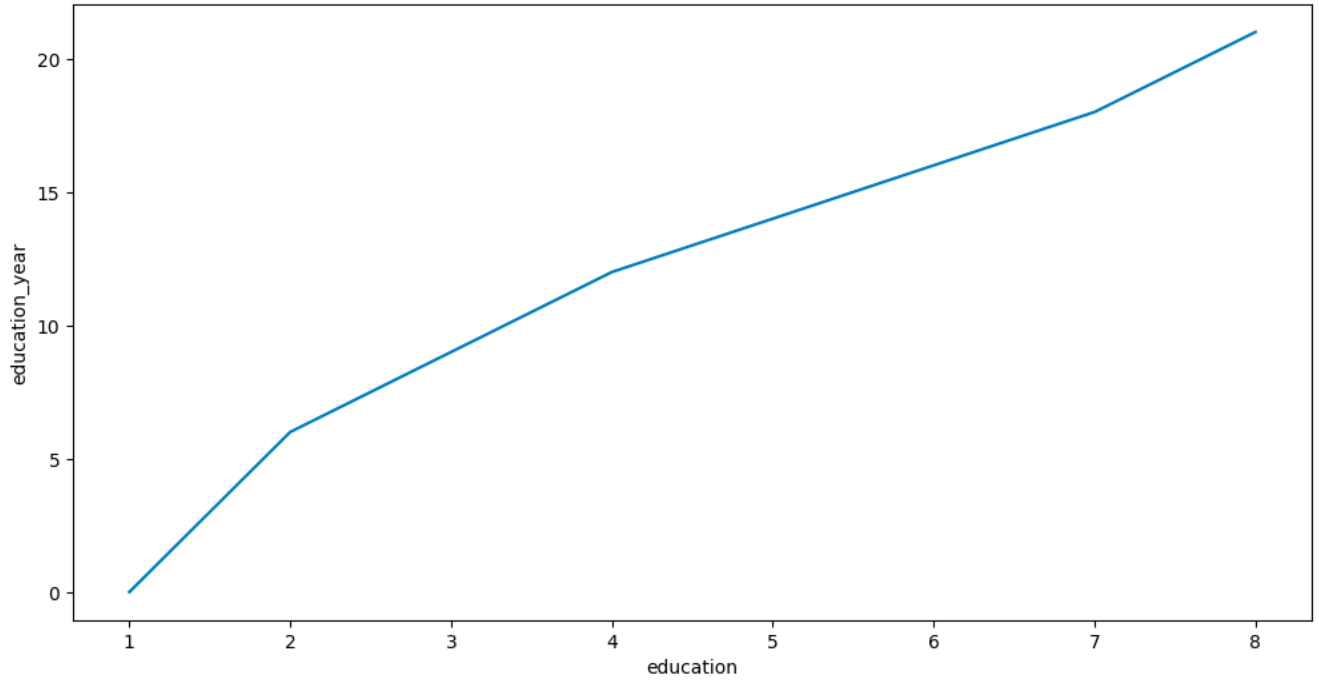
```
df=pd.get_dummies(data=df, columns=["marriage","industry","job","house"], drop_first=True)
df.columns
```

✓ 0.0s

```
Index(['metro', 'sex', 'age', 'number', 'education', 'asset', 'debt', 'income',
       'income_d', 'education_year', 'log_income', 'log_asset', 'log_debt',
       'h_debt', 'marriage_2', 'marriage_3', 'marriage_4', 'industry_B',
       'industry_C', 'industry_D', 'industry_E', 'industry_F', 'industry_G',
       'industry_H', 'industry_I', 'industry_J', 'industry_K', 'industry_L',
       'industry_M', 'industry_N', 'industry_NaN', 'industry_O', 'industry_P',
       'industry_Q', 'industry_R', 'industry_S', 'industry_T', 'industry_U',
       'job_2', 'job_3', 'job_4', 'job_5', 'job_6', 'job_7', 'job_8', 'job_9',
       'job_A', 'job_NaN', 'house_2', 'house_3', 'house_4', 'house_5'],
      dtype='object')
```

- one_hot encoding을 통하여 더미변수들을 생성
  drop_first 파라미터는 첫번째 더미변수를 제거하는 역할을 한다

누락 변수 추가 및 부적합 변수 제거



- education과 education_year은 매우 큰 상관관계를 갖는다
  같은 걸 설명하는 변수는 하나면 충분하기 때문에 하나를 제거한다
- df=df.drop(['education'],axis=1)

python™

데이터 스케일링

- 데이터의 범위를 재정의 하는 것
- Standard Scaler : 평균과 분산을 이용하여 데이터를 정규화

$$x_{new} = \frac{x - \mu}{\sigma}$$

- Robust Scaler : 중간값(median)과 사분위값을 이용하여 데이터를 정규화

$$x_{new} = \frac{x - x_{median}}{x_{0.75} - x_{0.25}}$$

- MinMax Scaler : 최대값과 최소값을 이용하여 데이터를 정규화

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

데이터 스케일링

- 데이터 스케일링이 필요한 모델
  - PCA
  - Clustering
  - KNN
  - SVM
  - Regression (변수의 중요도 및 계수들 간의 비교가 필요할 때)
  - Ridge / Lasso
- 데이터 스케일링이 필요하지 않는 모델
  - Logistic Regression
  - Decision Tree

python™