

1조

## 재벌집막내조

대회소개 & 데이터 설명 & 전처리

1

데이터 변수 선택 과정

2

모델링 & 모델링 결과

3

스터디, 프로젝트 소감

4

## 1. 대회소개 & 데이터 설명 & 전처리

## 월간 데이콘 신용카드 사용자 연체 예측 AI 경진대회

알고리즘 | 정형 | 분류 | 금융 | LogLoss

₩ 상금 : 100만원

🕒 2021.04.05 ~ 2021.05.24 17:59 [+ Google Calendar](#)

👤 3,191명 📅 마감

**배경:** 신용점수를 활용해 신청자의 향후 채무 불이행과 신용카드 대금 연체 가능성을 예측하기 위한 금융 서비스 구현

**목적:** 신용카드 사용자 데이터를 보고 사용자의 대금 연체 정도를 측정하는 알고리즘 개발

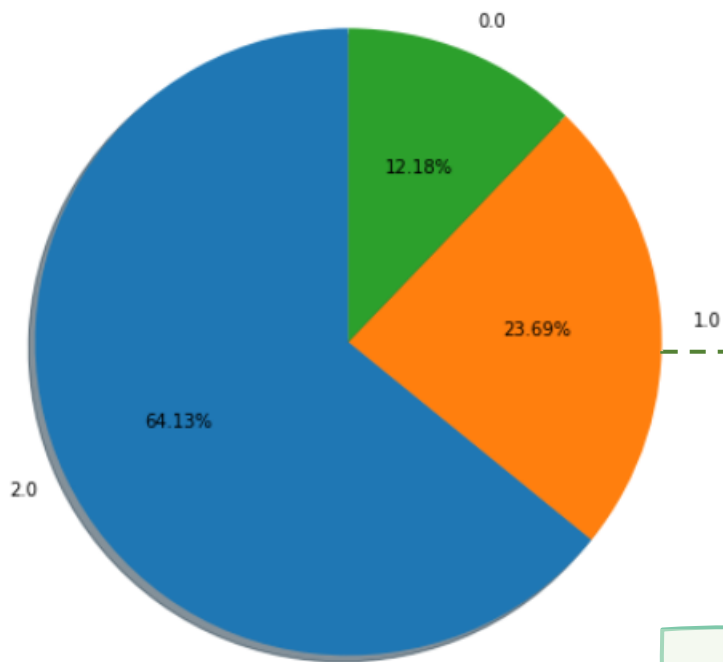
**주최:** DACON

**주관:** DACON

0	index	26457	non-null	int64
1	gender	26457	non-null	object
2	car	26457	non-null	object
3	reality	26457	non-null	object
4	child_num	26457	non-null	int64
5	income_total	26457	non-null	float64
6	income_type	26457	non-null	object
7	edu_type	26457	non-null	object
8	family_type	26457	non-null	object
9	house_type	26457	non-null	object
10	DAYS_BIRTH	26457	non-null	int64
11	DAYS_EMPLOYED	26457	non-null	int64
12	FLAG_MOBIL	26457	non-null	int64
13	work_phone	26457	non-null	int64
14	phone	26457	non-null	int64
15	email	26457	non-null	int64
16	occyp_type	18286	non-null	object
17	family_size	26457	non-null	float64
18	begin_month	26457	non-null	float64
19	credit	26457	non-null	float64

	index	child_num	income_total	DAYS_BIRTH	DAYS_EMPLOYED	work_phone	phone	email
count	26457.00	26457.00	26457.00	26457.00	26457.00	26457.00	26457.00	26457.00
mean	13228.00	0.43	187306.52	15958.05	2198.53	0.22	0.29	0.09
std	7637.62	0.75	101878.37	4201.59	2370.14	0.42	0.46	0.29
min	0.00	0.00	27000.00	7705.00	0.00	0.00	0.00	0.00
25%	6614.00	0.00	121500.00	12446.00	407.00	0.00	0.00	0.00
50%	13228.00	0.00	157500.00	15547.00	1539.00	0.00	0.00	0.00
75%	19842.00	1.00	225000.00	19431.00	3153.00	0.00	1.00	0.00
max	26456.00	19.00	1575000.00	25152.00	15713.00	1.00	1.00	1.00

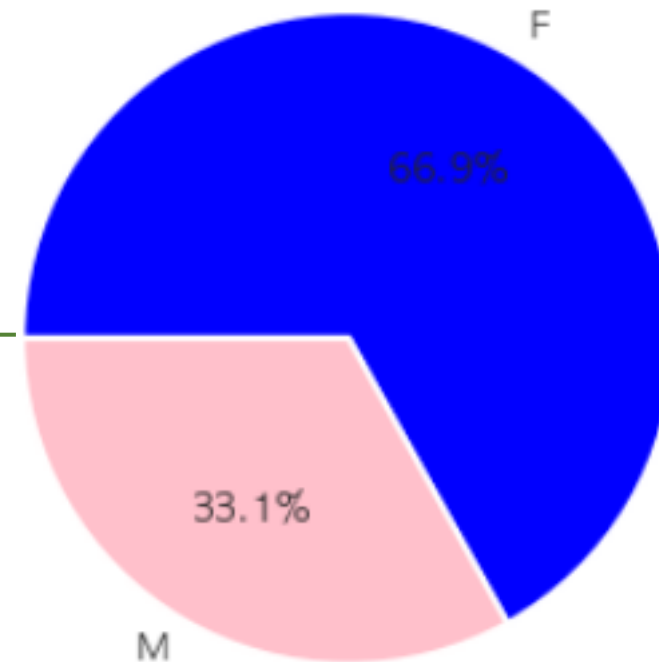
19개의 변수로 구성  
occyp\_type 직업유형에 결측값 존재

**credit - 신용등급**

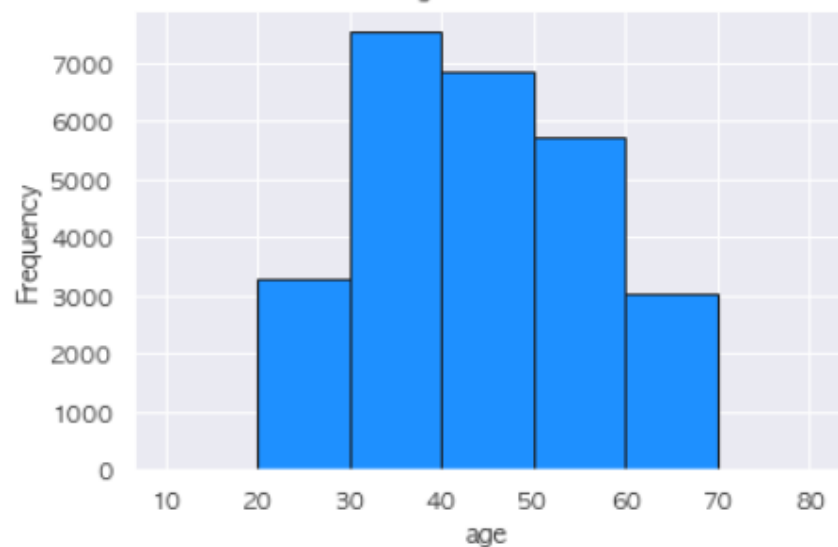
0으로 갈수록 높은 신용등급 의미  
0,1의 분포는 비슷함  
2의 분포가 가장 많음

**gender - 성별**

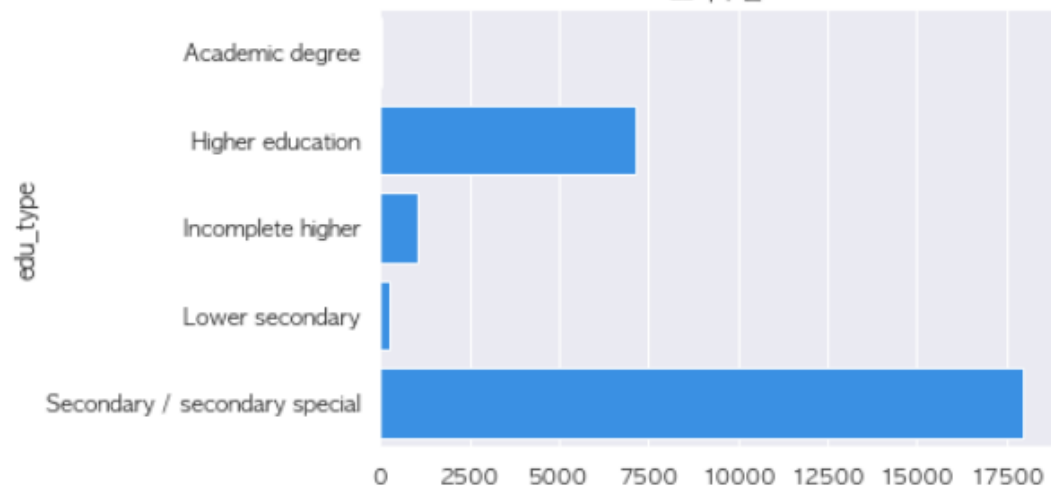
신용카드를 사용하는 성별은  
남성보다 여성이 더 많음



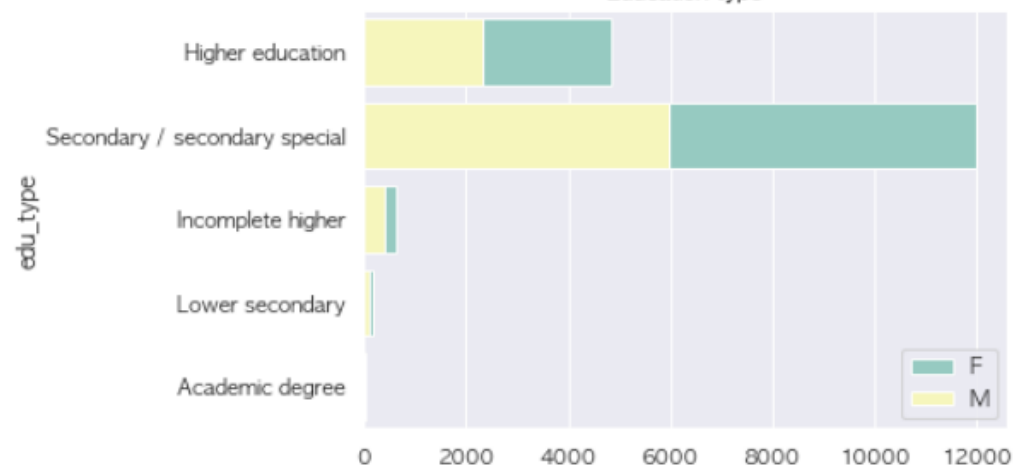
Age distribution



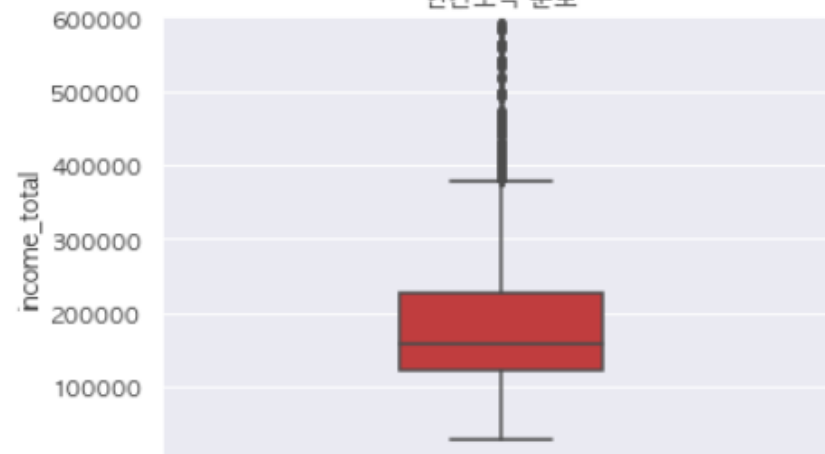
교육수준



Education type



연간소득 분포



```

# 나이 만들기
df['new_age'] = round(abs(df['DAYS_BIRTH'])/365.5,0).astype(np.int32)
df_test['new_age'] = round(abs(df_test['DAYS_BIRTH'])/365.5,0).astype(np.int32)

df['근속연수'] = df['DAYS_EMPLOYED'] // 365 # 근속연수
df['근속월수'] = df['DAYS_EMPLOYED'] // 30 # 근속월수
df['임용 월'] = np.floor(df['DAYS_EMPLOYED'] / 30) - ((np.floor(df['DAYS_EMPLOYED'] / 30) / 12).astype(int) * 12) # 고용된 달
df['임용 주'] = np.floor(df['DAYS_EMPLOYED'] / 7) - ((np.floor(df['DAYS_EMPLOYED'] / 7) / 4).astype(int) * 4) # 고용된 주
df["고용전 날 수"] = df["DAYS_BIRTH"] - df["DAYS_EMPLOYED"]

df_test['근속연수'] = df_test['DAYS_EMPLOYED'] // 365 # 근속연수
df_test['근속월수'] = df_test['DAYS_EMPLOYED'] // 30 # 근속월수
df_test['임용 월'] = np.floor(df_test['DAYS_EMPLOYED'] / 30) - ((np.floor(df_test['DAYS_EMPLOYED'] / 30) / 12).astype(int) * 12) # 고용된 달
df_test['임용 주'] = np.floor(df_test['DAYS_EMPLOYED'] / 7) - ((np.floor(df_test['DAYS_EMPLOYED'] / 7) / 4).astype(int) * 4) # 고용된 주
df_test["고용전 날 수"] = df_test["DAYS_BIRTH"] - df_test["DAYS_EMPLOYED"]

df['고용비율'] = df['DAYS_EMPLOYED'] / df['DAYS_BIRTH'] # 인생 살면서 일한 비율
df['인당 평균 부양비'] = df['income_total'] / df['family_size']

df_test['고용비율'] = df_test['DAYS_EMPLOYED'] / df_test['DAYS_BIRTH'] # 인생 살면서 일한 비율
df_test['인당 평균 부양비'] = df_test['income_total'] / df_test['family_size']

df_test['연봉'] = df_test['income_total'] / (df_test['근속연수'])
df_test['연봉'] = df_test['income_total'] / (df_test['근속연수'])

df["자녀 제외 가족 구성원 수"] = df["family_size"] - df["child_num"]
df_test["자녀 제외 가족 구성원 수"] = df_test["family_size"] - df_test["child_num"]

```

## 총 9개의 파생변수 생성

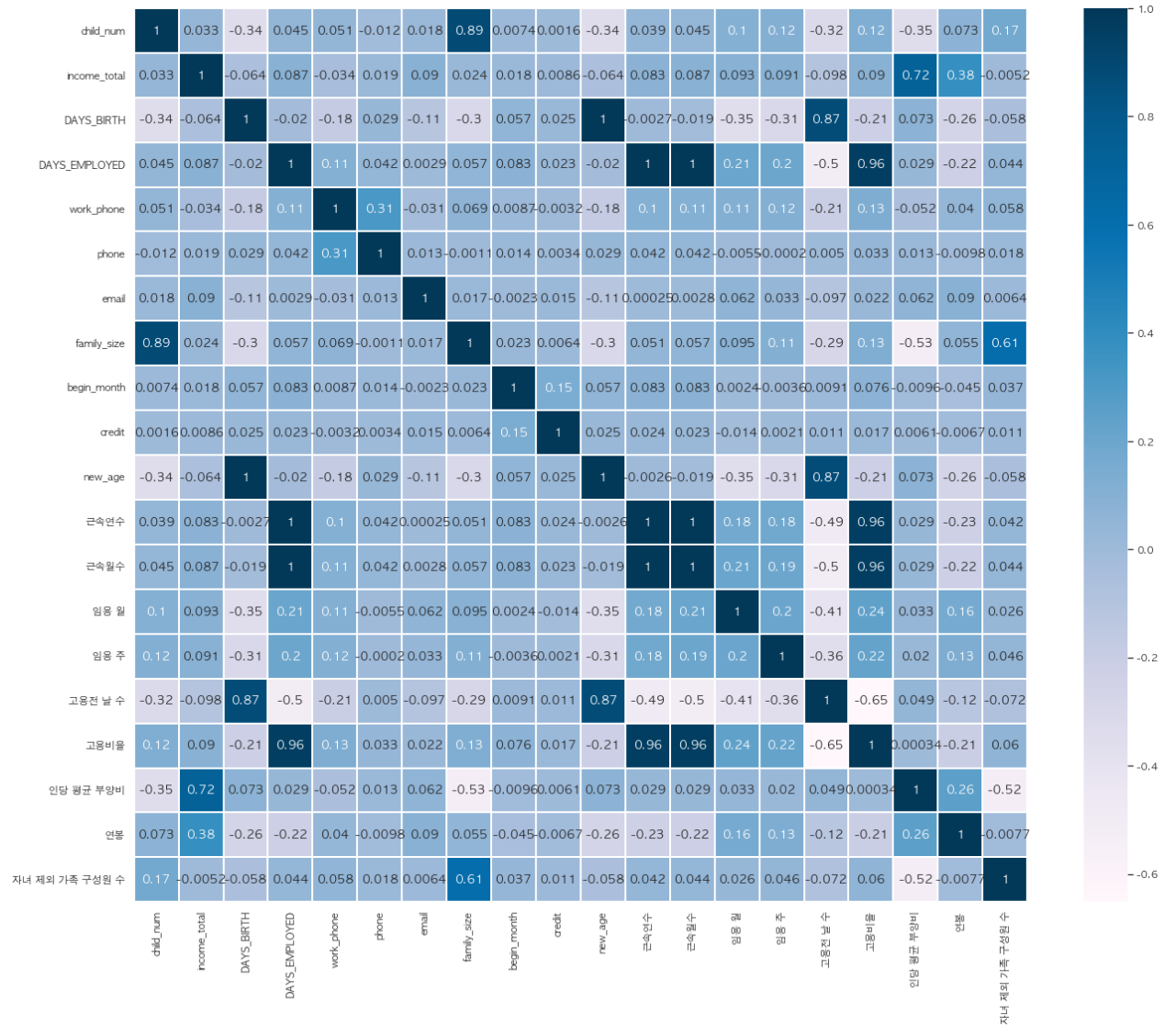
근속연수, 임용 월, 임용 주

고용 전 날 수, 근속 연수, 근속 월수, 고용비율,  
인당 평균 부양비, 연봉, 자녀 제외 가족 구성원수

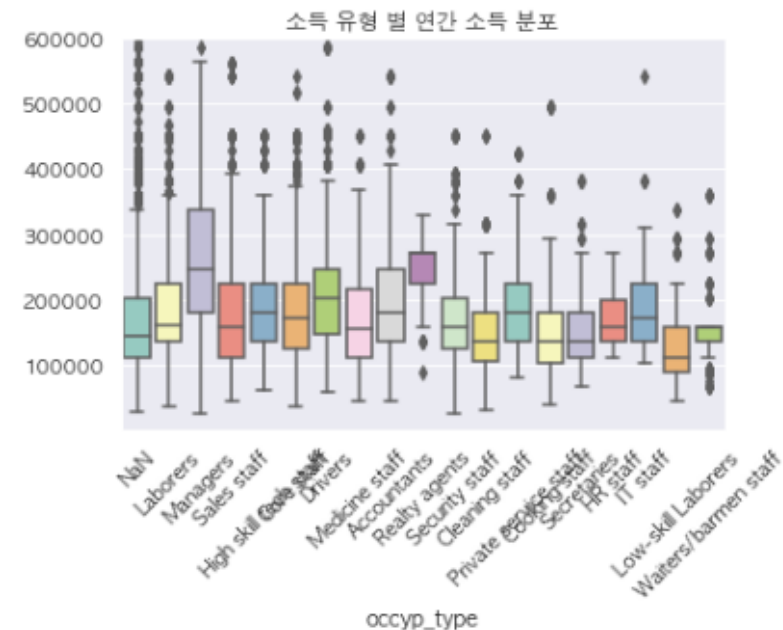
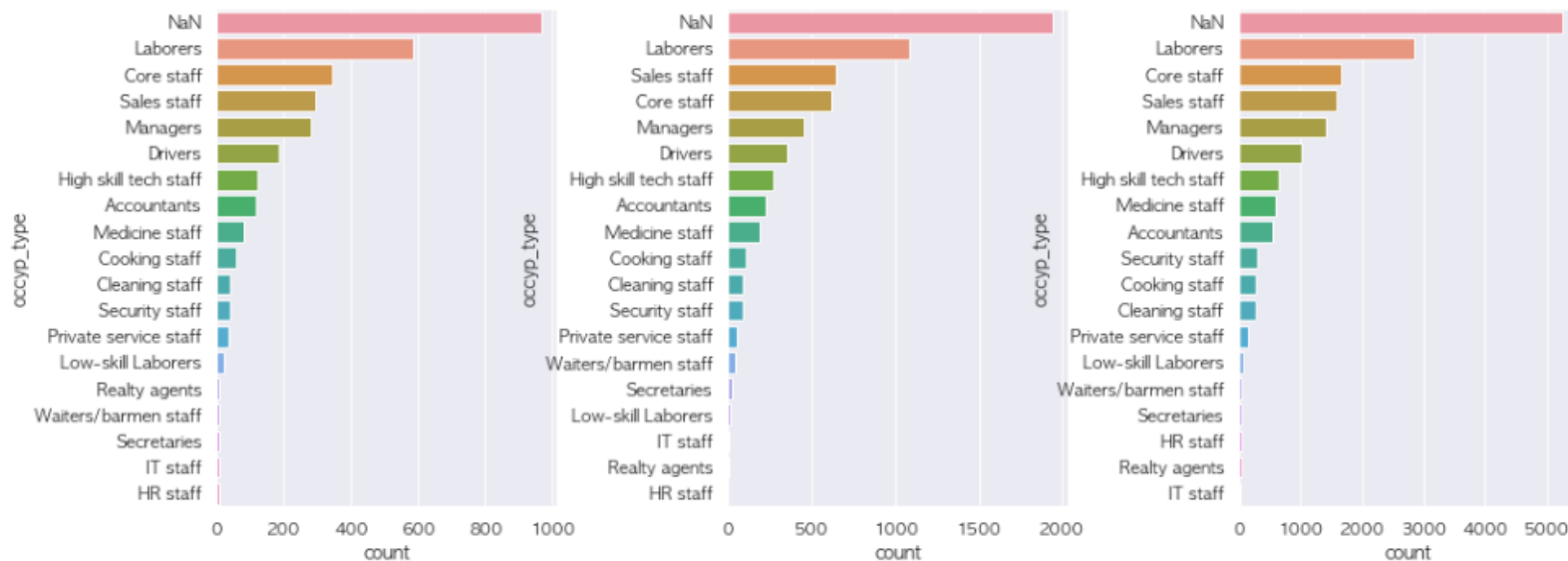


0	gender	26451	non-null	object
1	car	26451	non-null	object
2	reality	26451	non-null	object
3	child_num	26451	non-null	int64
4	income_total	26451	non-null	float64
5	income_type	26451	non-null	object
6	edu_type	26451	non-null	object
7	family_type	26451	non-null	object
8	house_type	26451	non-null	object
9	DAYS_BIRTH	26451	non-null	int64
10	DAYS_EMPLOYED	26451	non-null	int64
11	work_phone	26451	non-null	int64
12	phone	26451	non-null	int64
13	email	26451	non-null	int64
14	occyp_type	18280	non-null	object
15	family_size	26451	non-null	float64
16	begin_month	26451	non-null	float64
17	credit	26451	non-null	float64
18	new_age	26451	non-null	int64
19	근속연수	26451	non-null	int64
20	근속월수	26451	non-null	int64
21	임용 월	26451	non-null	float64
22	임용 주	26451	non-null	float64
23	고용전 날 수	26451	non-null	int64
24	고용비율	26451	non-null	float64
25	인당 평균 부양비	26451	non-null	float64
26	연봉	26451	non-null	float64
27	자녀 제외 가족 구성원 수	26451	non-null	float64

총 27개의 변수가 생성된 것을 확인



## occyp\_type 직업유형에 Nan 값으로 채울 시



신용 등급 별 직업유형 분포의 차이가 크지 않음

직업 유형 별 연간 소득 분포가 비슷함



Nan값으로 채워도 괜찮다고 판단

```
def change_type(data):  
    data['child_num'] = data['child_num'].astype('object')  
    data['FLAG_MOBIL'] = data['FLAG_MOBIL'].astype('object')  
    data['work_phone'] = data['work_phone'].astype('object')  
    data['phone'] = data['phone'].astype('object')  
    data['email'] = data['email'].astype('object')  
    data['occyp_type'] = data['occyp_type'].astype('object')  
    data['family_size '] = data['family_size'].astype('object')  
    #data['credit'] = data['credit'].astype('object')  
    data['week_ap'] = data['week_ap'].astype('int')  
    data['week_ap'] = data['week_ap'].astype('object')  
    data['month_ap'] = data['month_ap'].astype('int')  
    data['month_ap'] = data['month_ap'].astype('object')  
    data['family-child'] = data['family-child'].astype('object')  
    return data
```

데이터 타입 변경

## 2. 데이터 변수 선택

## 데이터 변수선택 - 범주형 변수 (카이제곱 검정)

**카이제곱 독립성 검정** : 두가지 범주형 또는 명목형 변수가 관련될 가능성 여부를 확인하는데 사용되는 통계적 가설 검정

	독립변수	종속변수
t검정	범주형	수치형
분산분석 (일원 분산분석)	범주형	수치형
카이제곱검정	범주형	범주형
상관분석 (피어슨)	수치형	수치형
회귀분석 (단순 선형)	수치형	수치형
로지스틱 회귀분석	수치형 (or 범주형)	범주형

**H0: 독립변수와 종속변수는 독립이다**

**H1: 독립변수와 종속변수는 독립이 아니다**



**H0: 독립변수와 종속변수는 관련성이 없다**

**H1: 독립변수와 종속변수는 관련성이 있다**

**카이제곱 독립성 검정** : 두가지 범주형 또는 명목형 변수가 관련될 가능성 여부를 확인하는데 사용되는 통계적 가설 검정

	chi_2	p-value	df
gender	0.742683	6.898085e-01	2
car	9.366187	9.250354e-03	2
reality	11.230277	3.642304e-03	2
child_num	19.978358	2.945805e-02	10
income_type	23.800389	2.475172e-03	8
edu_type	8.886748	3.519398e-01	8
family_type	46.383397	2.009568e-07	8
house_type	37.725432	4.236293e-05	10
FLAG_MOBIL	0.000000	1.000000e+00	0
work_phone	0.385865	8.245374e-01	2
phone	7.995643	1.835558e-02	2
email	6.107595	4.717942e-02	2
occyp_type	90.398801	1.422022e-06	36
month_ap	66.017750	2.756842e-06	22
week_ap	7.283921	2.953866e-01	6
family-child	25.970734	2.254547e-04	6
family_size	39.212899	9.706224e-05	12

	chi_2	p-value	df
reality	11.230277	3.642304e-03	2
child_num	19.978358	2.945805e-02	10
income_type	23.800389	2.475172e-03	8
family_type	46.383397	2.009568e-07	8
house_type	37.725432	4.236293e-05	10
occyp_type	90.398801	1.422022e-06	36
month_ap	66.017750	2.756842e-06	22
family-child	25.970734	2.254547e-04	6
family_size	39.212899	9.706224e-05	12

**P-value < 0.05**

**통계량 > 10**

**전체 범주형 변수: 16개**

**변수 선택 후 범주형 변수: 9개**

```
def cat_selection(data):
    #H0 : 독립변수의 범주형 변수(i)에 따른 종속변수 (credit)의 비율은 차이가 있다
    #H1 : 차이가 없다

    h0,h1,chi,c_list = [],[],[],[]

    for i in cat_list:
        chi_list = []
        cross_df = pd.crosstab(data[i],data['credit'],margins=False)
        result = chi2_contingency(observed=cross_df, correction=False)

        c_list.append(i)
        chi_list.append(result[0])
        chi_list.append(result[1])
        chi_list.append(result[2])
        chi.append(chi_list)

    chi_df = pd.DataFrame(columns=['chi_2','p-value','df'],data = chi)
    chi_df.index = c_list

    chi_df1 = chi_df[chi_df['p-value']<0.05]
    chi_df2= chi_df1[chi_df1['chi_2']>10]

    return chi_df, chi_df2
```

로지스틱 검정  $\rightarrow P > |z|$  가 0.05보다 작으면 변수가 유의하다는 의미

```
def logis_glm_one(data):
    data['credit'] = data['credit'].astype('int')
    output = data['credit']
    for i in int_list:
        feature = data[i]
        model = sm.formula.glm('output ~ feature', data, family = sm.families.Binomial()).fit()
```

logistic begin\_month Generalized Linear Model Regression Results

```
Dep. Variable:      output    No. Observations:      26451
Model:              GLM      Df Residuals:            26449
Model Family:       Binomial Df Model:                1
Link Function:      logit    Scale:                  1.0000
Method:             IRLS     Log-Likelihood:        -inf
Date:               Wed, 25 Jan 2023    Deviance:      1.5665e+06
Time:               08:05:44    Pearson chi2:   9.09e+19
No. Iterations:     4
Covariance Type:    nonrobust
```

	coef	std err	z	P> z	[0.025	0.975]
Intercept	4.768e+15	7.71e+05	6.19e+09	0.000	4.77e+15	4.77e+15
feature	1.519e+13	2.49e+04	6.1e+08	0.000	1.52e+13	1.52e+13

logistic income\_total Generalized Linear Model Regression Results

```
Dep. Variable:      output    No. Observations:      26451
Model:              GLM      Df Residuals:            26449
Model Family:       Binomial Df Model:                1
Link Function:      logit    Scale:                  1.0000
Method:             IRLS     Log-Likelihood:        -inf
Date:               Wed, 25 Jan 2023    Deviance:      1.5665e+06
Time:               08:05:44    Pearson chi2:   9.09e+19
No. Iterations:     3
Covariance Type:    nonrobust
```

	coef	std err	z	P> z	[0.025	0.975]
Intercept	2.29e+15	8.64e+05	2.65e+09	0.000	2.29e+15	2.29e+15
feature	2.658e+08	4.050	6.56e+07	0.000	2.66e+08	2.66e+08

logistic new\_age Generalized Linear Model Regression Results

```
Dep. Variable:      output    No. Observations:      26451
Model:              GLM      Df Residuals:            26449
Model Family:       Binomial Df Model:                1
Link Function:      logit    Scale:                  1.0000
Method:             IRLS     Log-Likelihood:        -inf
Date:               Wed, 25 Jan 2023    Deviance:      1.5665e+06
Time:               08:05:44    Pearson chi2:   9.09e+19
No. Iterations:     3
Covariance Type:    nonrobust
```

	coef	std err	z	P> z	[0.025	0.975]
Intercept	2.037e+15	1.62e+06	1.26e+09	0.000	2.04e+15	2.04e+15
feature	1.898e+10	98.203	1.93e+08	0.000	1.9e+10	1.9e+10

logistic credit Generalized Linear Model Regression Results

```
Dep. Variable:      output    No. Observations:      26451
Model:              GLM      Df Residuals:            26449
Model Family:       Binomial Df Model:                1
Link Function:      logit    Scale:                  1.0000
Method:             IRLS     Log-Likelihood:        nan
Date:               Wed, 25 Jan 2023    Deviance:      1.2698e+06
Time:               08:05:45    Pearson chi2:   7.64e+19
No. Iterations:     3
Covariance Type:    nonrobust
```

	coef	std err	z	P> z	[0.025	0.975]
Intercept	-1.387e+15	9.83e+05	-1.41e+09	0.000	-1.39e+15	-1.39e+15
feature	2.814e+15	5.88e+05	4.79e+09	0.000	2.81e+15	2.81e+15

&lt; 개별 값을 기준으로 로지스틱 검정 &gt;

개별 값을 기준으로 로지스틱 검정 시  
p-value가 모두 0에 가깝게 나오고  
통계량이 크게 나옴

odds 비가 0 or inf로 나옴

 $\rightarrow$  변수선택의 의미가 없다

로지스틱 검정  $\rightarrow P > |z|$  가 0.05보다 작으면 변수가 유의하다는 의미

```
def logis_glm(data):
    output = data['credit'].astype('int')
    formula = 'output ~ income_total + DAYS_BIRTH + DAYS_EMPLOYED + family_size + begin_month + new_age + year_os + month_os + bef_hire + RATIO_EMPLOYED + aver_dep + year_income'
    model = sm.formula.glm(formula, data, family = sm.families.Binomial()).fit()
    return model.summary(), np.exp(model.params)
```

### 수치형 변수 여러 개 넣고 로지스틱 검정

Generalized Linear Model Regression Results

Dep. Variable:	output	No. Observations:	26451
Model:	GLM	Df Residuals:	26439
Model Family:	Binomial	Df Model:	11
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-inf
Date:	Wed, 25 Jan 2023	Deviance:	1.5665e+06
Time:	08:05:46	Pearson chi2:	9.09e+19
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	std err	z	P> z	[0.025	0.975]
Intercept	4.203e+15	3.89e+06	1.08e+09	0.000	4.2e+15	4.2e+15
income_total	-6.426e+08	8.220	-7.82e+07	0.000	-6.43e+08	-6.43e+08
DAYS_BIRTH	3.152e+11	1.64e+04	1.92e+07	0.000	3.15e+11	3.15e+11
DAYS_EMPLOYED	6.552e+11	3.26e+04	2.01e+07	0.000	6.55e+11	6.55e+11
family_size	7.016e+13	7.66e+05	9.16e+07	0.000	7.02e+13	7.02e+13
begin_month	2.226e+13	2.51e+04	8.87e+08	0.000	2.23e+13	2.23e+13
new_age	1.3e+13	1.41e+06	9.2e+06	0.000	1.3e+13	1.3e+13
year_os	1.551e+14	1.47e+06	1.05e+08	0.000	1.55e+14	1.55e+14
month_os	-4.188e+13	1.48e+06	-2.83e+07	0.000	-4.19e+13	-4.19e+13
bef_hire	-3.399e+11	1.63e+04	-2.08e+07	0.000	-3.4e+11	-3.4e+11
RATIO_EMPLOYED	-2.513e+14	1.52e+07	-1.66e+07	0.000	-2.51e+14	-2.51e+14
aver_dep	9.845e+08	13.052	7.54e+07	0.000	9.85e+08	9.85e+08
year_income	3.311e+08	8.102	4.09e+07	0.000	3.31e+08	3.31e+08

```
""", Intercept      inf
income_total      0.0
DAYS_BIRTH        inf
DAYS_EMPLOYED      inf
family_size        inf
begin_month        inf
new_age            inf
year_os            inf
month_os           0.0
bef_hire           0.0
RATIO_EMPLOYED     0.0
aver_dep           inf
year_income         inf
dtype: float64)
```

변수 모두가 0.05 보다 작음

수치형 변수의 로지스틱은 의미가 없음



로지스틱 검정  $\rightarrow P > |z|$  가 0.05보다 작으면 변수가 유의하다는 의미

```
dt = data[data['credit'] != 0.0]
logis_glm(dt)
```

타겟변수 = credit (신용등급)

0,1,2 로 구분/ 0으로 갈수록 높은 신용등급

(로지스틱 검정은 0,1 로 하는 회귀라서  $Y \neq 0.0$ )

Generalized Linear Model Regression Results

Dep. Variable:	output	No. Observations:	23229
Model:	GLM	Df Residuals:	23217
Model Family:	Binomial	Df Model:	11
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-inf
Date:	Wed, 25 Jan 2023	Deviance:	1.2698e+06
Time:	08:05:47	Pearson chi2:	7.64e+19
No. Iterations:	4		
Covariance Type:	nonrobust		

	coef	std err	z	P> z	[0.025	0.975]
Intercept	5.717e+15	4.15e+06	1.38e+09	0.000	5.72e+15	5.72e+15
income_total	2.512e+08	8.751	2.87e+07	0.000	2.51e+08	2.51e+08
DAYS_BIRTH	-1.245e+12	1.75e+04	-7.12e+07	0.000	-1.25e+12	-1.25e+12
DAYS_EMPLOYED	-2.381e+12	3.46e+04	-6.88e+07	0.000	-2.38e+12	-2.38e+12
family_size	8.102e+13	8.13e+05	9.96e+07	0.000	8.1e+13	8.1e+13
begin_month	2.971e+13	2.69e+04	1.11e+09	0.000	2.97e+13	2.97e+13
new_age	4.102e+13	1.51e+06	2.72e+07	0.000	4.1e+13	4.1e+13
year_os	7.852e+13	1.57e+06	4.99e+07	0.000	7.85e+13	7.85e+13
month_os	1.025e+14	1.57e+06	6.51e+07	0.000	1.02e+14	1.02e+14
bef_hire	1.136e+12	1.74e+04	6.53e+07	0.000	1.14e+12	1.14e+12
RATIO_EMPLOYED	-1.755e+15	1.62e+07	-1.08e+08	0.000	-1.76e+15	-1.76e+15
aver_dep	6.279e+08	13.819	4.54e+07	0.000	6.28e+08	6.28e+08
year_income	2.17e+08	8.745	2.48e+07	0.000	2.17e+08	2.17e+08

```

****, Intercept      inf
income_total         inf
DAYS_BIRTH           0.0
DAYS_EMPLOYED         0.0
family_size           inf
begin_month           inf
new_age               inf
year_os               inf
month_os              inf
bef_hire              inf
RATIO_EMPLOYED        0.0
aver_dep              inf
year_income           inf
dtype: float64)
```

<  $Y \neq 0.0$  인 (2.0, 1.0) 으로 로지스틱 회귀 >

p-value < 0.05

Odds > 1 :

이후 VIF 결과와 비교해

변수 선택 필요

로지스틱 검정  $\rightarrow P > |z|$  가 0.05보다 작으면 변수가 유의하다는 의미

```
dt = data[data['credit'] != 2.0]
logis_glm(dt)
```

#### Generalized Linear Model Regression Results

Dep. Variable:	output	No. Observations:	9489
Model:	GLM	Df Residuals:	9477
Model Family:	Binomial	Df Model:	11
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-6033.6
Date:	Wed, 25 Jan 2023	Deviance:	12067.
Time:	08:05:46	Pearson chi2:	9.48e+03
No. Iterations:	4		
Covariance Type:	nonrobust		

	coef	std err	z	P> z	[0.025	0.975]
Intercept	0.9775	0.203	4.822	0.000	0.580	1.375
income_total	-1.311e-06	4.38e-07	-2.994	0.003	-2.17e-06	-4.53e-07
DAYS_BIRTH	0.0020	0.001	2.332	0.020	0.000	0.004
DAYS_EMPLOYED	0.0041	0.002	2.361	0.018	0.001	0.007
family_size	0.0016	0.041	0.039	0.969	-0.079	0.083
begin_month	-0.0094	0.001	-7.441	0.000	-0.012	-0.007
new_age	0.0012	0.075	0.017	0.987	-0.145	0.147
year_os	0.1007	0.078	1.285	0.199	-0.053	0.254
month_os	-0.1941	0.078	-2.477	0.013	-0.348	-0.040
bef_hire	-0.0020	0.001	-2.357	0.018	-0.004	-0.000
RATIO_EMPLOYED	1.2059	0.800	1.507	0.132	-0.363	2.775
aver_dep	6.024e-07	7.33e-07	0.822	0.411	-8.35e-07	2.04e-06
year_income	6.363e-08	4.17e-07	0.153	0.879	-7.53e-07	8.8e-07

타겟변수 = credit ( 신용등급 )

0,1,2 로 구분/ 0으로 갈수록 높은 신용등급

( 로지스틱 검정은 0,1 로 하는 회귀라서  $Y \neq 2.0$  )

```
""", Intercept          2.657676
income_total          0.999999
DAYS_BIRTH            1.002039
DAYS_EMPLOYED         1.004084
family_size           1.001626
begin_month           0.990657
new_age               1.001243
year_os               1.105979
month_os              0.823611
bef_hire              0.997963
RATIO_EMPLOYED        3.339672
aver_dep              1.000001
year_income           1.000000
dtype: float64)
```

<  $Y \neq 2.0$  인 (0.0, 1.0) 으로 로지스틱 회귀 >

p-value < 0.05

Odds > 1 :

RATIO\_EMPLOYED

이후 VIF 결과와 비교해

변수 선택 필요

### < VIF 다중공산성 >

: 독립 변수간 상관 관계를 보이는 것

다중공산성이 있으면 부정확한 회귀 결과가 도출되기  
때문에 확인 필요

VIF 지수 10 이상이면 다중공산성 보유 가능성 높음

VIF 지수 10 이상의 변수 삭제

오히려 모델의 정확도가 낮아지는 경우 발생

	VIF_Factor	Feature
0	inf	child_num
1	1.771380e+01	income_total
2	2.395629e+04	DAYS_BIRTH
3	1.466627e+05	DAYS_EMPLOYED
4	inf	family_size
5	3.582473e+00	begin_month
6	5.629168e+00	credit
7	2.392278e+04	new_age
8	1.374594e+03	year_os
9	1.467077e+05	month_os
10	4.662239e+00	month_ap
11	2.543796e+00	week_ap
12	1.366302e+01	aver_dep
13	2.103375e+00	year_income
14	inf	family-child

수치형 변수의 VIF  
값을 내림차순  
으로 산출

```
def vif(x):
    # vif 10 초과시 drop을 위한 임계값 설정
    thresh = 10
    # Filter method로 feature selection 진행 후 최종 도출 될 데이터 프레임 형성
    output = pd.DataFrame()
    # 데이터의 컬럼 개수 설정
    k = x.shape[1]
    # VIF 측정
    vif = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
    for i in range(1,k):
        print(f'{i}번째 VIF 측정')
        # VIF 최대 값 선정
        a = np.argmax(vif) # np.argmax -> 가장 큰 값이 있는 인덱스 값을 반환하는 메서드
        print(f'Max VIF feature & value : {x.columns[a]}, {vif[a]}')
        # VIF 최대 값이 임계치를 넘지 않는 경우 break
        if (vif[a] <= thresh):
            print('\n')
            for q in range(output.shape[1]):
                print(f'{output.columns[q]}의 vif는 {np.round(vif[q],2)}입니다.')
            break
        # VIF 최대 값이 임계치를 넘는 경우, + 1번째 시도인 경우 : if 문으로 해당 feature 제거 후 다시 vif 측정
        if (i == 1):
            output = x.drop(x.columns[a], axis = 1)
            vif = [variance_inflation_factor(output.values, j) for j in range(output.shape[1])]
        # VIF 최대 값이 임계치를 넘는 경우, + 1번째 이후 시도인 경우 : if 문으로 해당 feature 제거 후 다시 vif 측정
        elif (i > 1):
            output = output.drop(output.columns[a], axis = 1)
            vif = [variance_inflation_factor(output.values, j) for j in range(output.shape[1])]
    return (output)
```

## 필터메서드

다중공선성이 높은 변수를 하나씩 제거할 때마다 다시 VIF 값을 산출해서 변수를 하나씩 제거하는 방법을 통해 좀 더 정확하게 제거 할 변수를 판별

1번째 VIF 측정

Max VIF feature & value : child\_num, inf

2번째 VIF 측정

Max VIF feature & value : year\_os, 146707.72625933637

3번째 VIF 측정

Max VIF feature & value : income\_total, 23942.481549781103

4번째 VIF 측정

Max VIF feature & value : income\_total, 1459.4633487743724

5번째 VIF 측정

Max VIF feature & value : month\_ap, 29.389384525806967

6번째 VIF 측정

Max VIF feature & value : child\_num, 16.2543931141145

7번째 VIF 측정

Max VIF feature & value : DAYS\_EMPLOYED, 8.831385159073534

family\_size의 vif는 6.74입니다.

begin\_month의 vif는 3.56입니다.

credit의 vif는 5.48입니다.

new\_age의 vif는 8.83입니다.

year\_os의 vif는 2.09입니다.

month\_ap의 vif는 2.75입니다.

week\_ap의 vif는 2.38입니다.

aver\_dep의 vif는 3.92입니다.

year\_income의 vif는 1.93입니다.

## 수치형 변수

famiy\_size

begin\_month

credit

new\_age

year\_os

month\_ap

week\_ap

aver\_dep

year\_income

변수 선택을 통해 적합한 칼럼만 분석에 사용

	reality	child_num	income_type	family_type	house_type	occyp_type	month_ap	family-child	family_size	income_total	DAYS_EMPLOYED	begin_month	new_age	aver_dep	year_income
0	N	0	Commercial associate	Married	Municipal apartment	Nan	0	2.0	2.0	202500.0	4709	6.0	38	101250.0	16875.0
1	Y	1	Commercial associate	Civil marriage	House / apartment	Laborers	3	2.0	3.0	247500.0	1540	5.0	31	82500.0	61875.0
2	Y	0	Working	Married	House / apartment	Managers	3	2.0	2.0	450000.0	4434	22.0	52	225000.0	37500.0
3	Y	0	Commercial associate	Married	House / apartment	Sales staff	9	2.0	2.0	202500.0	2092	37.0	41	101250.0	40500.0
4	Y	0	State servant	Married	House / apartment	Managers	10	2.0	2.0	157500.0	2105	26.0	41	78750.0	31500.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26446	N	2	State servant	Married	House / apartment	Core staff	6	2.0	4.0	225000.0	1984	2.0	33	56250.0	45000.0
26447	Y	1	Working	Separated	House / apartment	Nan	10	1.0	2.0	180000.0	2475	47.0	42	90000.0	30000.0
26448	N	0	Working	Civil marriage	With parents	Core staff	7	2.0	2.0	292500.0	2015	25.0	28	146250.0	58500.0
26449	Y	0	Working	Single / not married	House / apartment	Laborers	3	1.0	1.0	171000.0	107	59.0	28	171000.0	0.0
26450	N	0	Working	Civil marriage	House / apartment	Security staff	9	2.0	2.0	81000.0	1013	9.0	54	40500.0	40500.0

### 3. 모델링 전처리 & 결과

```
[ ] X = df_train  
    y = data['credit']
```

```
[ ] #train = df_train.copy()  
    #X = train.drop(['credit'],axis=1)  
    #y = df_train['credit']
```

```
[ ] from sklearn.model_selection import train_test_split  
  
X_train, X_valid, y_train, y_valid = train_test_split(X,y,test_size=0.2, random_state = 2023, shuffle=False)  
print('X_train : ', X_train.shape)  
print('y_train : ', y_train.shape)  
print('X_valid : ', X_valid.shape)  
print('y_valid : ', y_valid.shape)
```

```
X_train : (21160, 15)  
y_train : (21160,)  
X_valid : (5291, 15)  
y_valid : (5291,)
```

**train**  
**test**  
**split**



### 범주형 변수 인코딩 ( OrdinalEncoder )

```
from sklearn.preprocessing import OrdinalEncoder

Encoder = OrdinalEncoder()
X_train_encoder = Encoder.fit_transform(X_train[train_cat], y_train)

print('X_train_encoder.shape : ', X_train_encoder.shape)
print('len(train_cat) : ', len(train_cat))

X_train_encoder.shape : (21160, 9)
len(train_cat) : 9
```

### 수치형 변수 스케일링 ( StandardScaler )

```
from sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()
X_train_Scaled = X_train[train_int] = Scaler.fit_transform(X_train[train_int], y_train)
print('X_train_Scaled.shape : ', X_train_Scaled.shape)
print('len(train_int) : ', len(train_int))

X_train_Scaled.shape : (21160, 6)
len(train_int) : 6
```

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=23)
pre_X_train_over, y_train_over = smote.fit_resample(pre_X_train, y_train)

print('X_train : ', pre_X_train.shape)
print('y_train : ', y_train.shape)
print('X_train_over : ', pre_X_train_over.shape)
print('y_train_over : ', y_train_over.shape)

X_train : (21160, 15)
y_train : (21160,)
X_train_over : (40851, 15)
y_train_over : (40851,)
```

## SMOTE

적은 레이블을 가진 데이터 세트를 많은 레이블을 가진 세트 수준으로 늘리는 방법

적은 데이터 세트에 있는 개별 데이터들의 K 최근 점 이웃을 찾아서 이 데이터와 K개 이웃들의 차이를 일정 값으로 만들어서 기존 데이터와 차이나는 새로운 데이터들을 생성하는 것

```

print('모델링 시작.....')
print('-----LGBMClassifier-----')
lgbm = LGBMClassifier(n_estimators=400, learning_rate= 0.05)
lgbm.fit(X_train,y_train,eval_metric='logloss')
y_pred = lgbm.predict(X_valid)
y_pred_proba = lgbm.predict_proba(X_valid)
print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

print('-----XGBClassifier-----')
xgb = XGBClassifier(n_estimators=400, learning_rate= 0.05)
xgb.fit(X_train,y_train)
y_pred = xgb.predict(X_valid)
y_pred_proba = xgb.predict_proba(X_valid)
print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

print('-----GradientBoostingClassifier-----')
gbc = GradientBoostingClassifier(n_estimators=400, learning_rate=0.05)
gbc.fit(X_train,y_train)
y_pred = gbc.predict(X_valid)
y_pred_proba = gbc.predict_proba(X_valid)
print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

print('-----RandomForestClassifier-----')
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_valid)
y_pred_proba = rfc.predict_proba(X_valid)
print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

print('-----CatBoostClassifier-----')
catb = CatBoostClassifier(verbose=0)
catb.fit(X_train, y_train, verbose=0)
y_pred = catb.predict(X_valid)
y_pred_proba = catb.predict_proba(X_valid)
print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

end_time = time.time()
print("Model execution time: {:.2f}s".format(end_time - start_time))

modeling(X_train , X_valid , y_train , y_valid)

```

## 순정 모델링을 한번에 돌려주는 함수 생성

LGBM

XGB

GBC

RF

CBC

Encoder	Scaler	Sampling	Model	Log_loss
Ordinal	Standard	SMOTE	LGBM Classifier	0.7825
			XGB Classifier	0.9022
			Catboost Classifier	0.7884
			GradientBoosting Classifier	0.8871
			RandomForest Classifier	1.1059

순정 모델링 - LGBM Classifier 가 적합

## Run all model Non Hyperparameter Optimization

```
def modeling(X_train , X_valid , y_train , y_valid):
    start_time = time.time()

    print('모델링 시작.....')
    print('-----LGBMClassifier-----')
    lgbm = LGBMClassifier(n_estimators=400, learning_rate= 0.05)
    lgbm.fit(X_train,y_train,eval_metric='logloss')
    y_pred = lgbm.predict(X_valid)
    y_pred_proba = lgbm.predict_proba(X_valid)
    print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

    print('-----XGBClassifier-----')
    xgb = XGBClassifier(n_estimators=400, learning_rate= 0.05)
    xgb.fit(X_train,y_train)
    y_pred = xgb.predict(X_valid)
    y_pred_proba = xgb.predict_proba(X_valid)
    print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

    print('-----LGBMClassifier-----')
    gbc = GradientBoostingClassifier(n_estimators=400, learning_rate= 0.05)
    gbc.fit(X_train,y_train)
    y_pred = gbc.predict(X_valid)
    y_pred_proba = gbc.predict_proba(X_valid)
    print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

    print('-----XGBClassifier-----')
    [18:15:37] WARNING: /Users/runner/miniforge3/conda-bld/xgboost-split_1645117948562/work/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
    [[ 26  60  524]
     [ 16 361  901]
     [ 12  58 3333]] 70.31 0.7444729107898345

    print('-----GradientBoostingClassifier-----')
    catb = CatBoostClassifier(verbose=0)
    catb.fit(X_train, y_train, verbose=0)
    y_pred = catb.predict(X_valid)
    y_pred_proba = catb.predict_proba(X_valid)
    print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

    print('-----RandomForestClassifier-----')
    rfc = RandomForestClassifier()
    rfc.fit(X_train, y_train)
    y_pred = rfc.predict(X_valid)
    y_pred_proba = rfc.predict_proba(X_valid)
    print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

    print('-----CatBoostClassifier-----')
    catb = CatBoostClassifier(verbose=0)
    catb.fit(X_train, y_train, verbose=0)
    y_pred = catb.predict(X_valid)
    y_pred_proba = catb.predict_proba(X_valid)
    print(confusion_matrix(y_valid, y_pred), round(accuracy_score(y_valid, y_pred) * 100, 2), log_loss(y_valid, y_pred_proba))

    end_time = time.time()
    print("Model execution time: {:.2f}s".format(end_time - start_time))

modeling(X_train , X_valid , y_train , y_valid)
```

순정 모델링을 한번에 돌려주는 함수 생성

변수를 선택하지 않고 파생변수만 만들며

SMOTE 하지 않은 데이터셋을 최종으로

보고 하이퍼파라미터 실시

시간이 너무 오래걸려서(5시간 이상) 부득이하게 LGBM 기반  
RandomizedSearchCV를 사용 / best 파라미터로 모델링 진행

```
[32]: n_fold = 17
cv = StratifiedKFold(n_splits=n_fold, shuffle=True, random_state=40)

print(feat_train.shape)

lgb_p_val = np.zeros((feat_train.shape[0], n_class))
lgb_p_tst = np.zeros((feat_test.shape[0], n_class))
for i, (i_trn, i_val) in enumerate(cv.split(feat_train, target), 1):
    print(f'training model for CV #{i}')
    lgb_clf = LGBMClassifier(max_depth=24,
                             num_leaves=110,
                             colsample_bytree=0.3,
                             n_estimators=230,
                             min_child_samples=2,
                             subsample=0.9,
                             subsample_freq=2,
                             learning_rate=0.09,
                             random_state=2021,
                             verbose = 0)

    lgb_clf.fit(feat_train[i_trn], target[i_trn],
                eval_set=[(feat_train[i_val], target[i_val])], verbose = 0)

    lgb_p_val[i_val, :] = lgb_clf.predict_proba(feet_train[i_val])
    lgb_p_tst += lgb_clf.predict_proba(feet_test) / n_fold

print(f'{log_loss(target, lgb_p_val)}')
print(f'{confusion_matrix(target, np.argmax(lgb_p_val, axis=1))}%')
```

StratifiedKFold 를 사용

이유: SMOTE를 하지 않았기에 폴드별 target이  
불균형할 것을 우려해 S-Kfold를 이용함

Fold는 10 ~ 17까지 돌려본 결과 16이 Public  
기준으로 logloss가 가장 낮았음

자체적으로 돌린 모델은 17일때 가장 logloss가  
낮았으나 데이콘에 제출했을때는 오히려 public기  
준으로는 16보다 높아짐 (오버피팅 예상)

Encoder	Scaler	Sampling	Model	Log_loss
Ordinal	Standard	X	LGBM Classifier (RandomizedSearchCV)	0.6961
			XGB Classifier	0.7444
			Catboost Classifier	0.7387
			GradientBoosting Classifier	0.7773
			RandomForest Classifier	0.9929

LGBM Classifier (RandomizedSearchCV)가 적합

	변수선택 과정	SMOTE	LGBM 순정 Log_loss	LGBM Classifier (RandomizedSearchCV)
1	O	O	0.7301	-
2	O	X	0.7444	-
3	X	O	0.75(726등)	0.71(477등)
4	X	X	0.71(450등)	0.6961(279등)

-lgbm\_hyperop\_submission.csv

2023-01-30 0.7106779558  
02:28:21 0.6927148478

-lgbm\_hyperop\_nosmote\_nfold\_16\_submission.csv

2023-01-31 0.6961195973  
00:39:22 0.6755429504



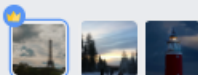

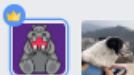
PUBLIC

PRIVATE

AWARDS

RANKING CHART

● WINNER ● 1% ● 4% ● 10%

#	팀	팀 멤버	최종점수	제출수
1	소회의실		0.6581	77
2	dswook		0.65862	66
3	js4756		0.65913	77
4	초보산님	<div><div>_lgbm_hyperop_nosmote_nfold_16_submission.csv</div><div>2023-01-31 00:39:22</div><div>0.6961195973 0.6755429504</div><div><input type="checkbox"/></div></div>		

279 등으로 마무리

## 4. *STUDY*, 소감

## 파이썬 공부

전체 보기   강의 번호별   LIT 455   리스트 보기   +

필터   정렬   Q   ...   **새로 만들기**

유형   ☒ 복습   필요할 때   + 필터 추가

<input checked="" type="checkbox"/> 복습	Aa 강의명	유형	필요할 때	자료	설명	+ ...
<input type="checkbox"/>	 SVM	모델링	모델링 이진분류	<a href="https://blog.na...">https://blog.na...</a>		
<input type="checkbox"/>	 VIF 다중공산성	전처리	변수선택			
<input type="checkbox"/>	 스케일링	전처리	전처리	<a href="https://blog.na...">https://blog.na...</a>		
<input type="checkbox"/>	 RandomForest	모델링	모델링 분류/회귀			
<input type="checkbox"/>	 LGBM	모델링		<a href="https://open.sp...">https://open.sp...</a>		
<input type="checkbox"/>	 KNN	모델링	모델링 분류			
<input type="checkbox"/>	 Logistic Regression	모델링	모델링 이진분류	<a href="https://www.ge...">https://www.ge...</a>		
<input type="checkbox"/>	 SGDClassifier	모델링	모델링 이진분류			
<input type="checkbox"/>	 DecisionTreeClassifier	모델링				
<input type="checkbox"/>	 Cat BoostClassifier	모델링			범주형 변수 모델링 가능, 전처리	
<input type="checkbox"/>	 XGBClassifier	모델링				

Notion을 활용한 STUDY !!

## 파이썬 공부

전체 보기 ▾












필터 정렬 🔍 ...

새로 만들기 ▾

☒ 유형 ▾
 ☒ 복습 ▾
 ☐ 필요할 때 ▾
 + 필터 추가
☒ 복습

Aa 강의명

유형 ▾

<input type="checkbox"/>	 SVM	모델링	모델
<input type="checkbox"/>	 VIF 다중공산성	전처리	변수
<input type="checkbox"/>	 스케일링	전처리	전처
<input type="checkbox"/>	 RandomForest	모델링	모델
<input type="checkbox"/>	 LGBM	모델링	
<input type="checkbox"/>	 KNN	모델링	모델
<input type="checkbox"/>	 Logistic Regression	모델링	모델
<input type="checkbox"/>	 SGDClassifier	모델링	모델
<input type="checkbox"/>	 DecisionTreeClassifier	모델링	
<input type="checkbox"/>	 Cat BoostClassifier	모델링	
<input type="checkbox"/>	 XGBClassifier	모델링	

+ 새로 만들기

계산 ▾

개수 11

## SVM

🕒 작성일시

2023년 1월 28일 오전 2:08

📄 유형

모델링

📎 자료

<https://blog.na...>☒ 복습☐

≡ 설명

비어 있음

≡ 필요할 때

모델링 이진분류

+ 속성 추가

🗨️ 댓글 추가

## SVM

SVM(Support Vector Machine)은 분류 또는 회귀 작업에 사용할 수 있는 감독 기계 학습 알고리즘

```
from sklearn import svm
from sklearn.datasets import make_classification

# Generate sample data
X, y = make_classification(n_features=4, random_state=0)
```

## CatBoost

```
# Load the iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Binarize the target variable
y = np.where(y == 0, 0, 1)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the CatBoostClassifier
cb = CatBoostClassifier(iterations=500, learning_rate=0.1, depth=6, logging_level='Silent')
cb.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = cb.predict(X_test)

# Evaluate the model's accuracy
accuracy = cb.score(X_test, y_test)
print("Accuracy:", accuracy)
```

분류 및 회귀 문제용 의사 결정 트리 라이브러리의 오픈 소스 그래디언트 부스팅

**범주형 데이터를** 처리하도록 설계

많은 수의 범주형 기능, 누락된 값 및 대규모 데이터 세트가 있는 문제를 해결하기 위한 강력한 도구

➡ 코드 예제

## XGBoost

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split

# Load the iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Binarize the target variable
y = np.where(y == 0, 0, 1)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the XGBClassifier
xgb = XGBClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, silent=True)
xgb.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = xgb.predict(X_test)

# Evaluate the model's accuracy
accuracy = xgb.score(X_test, y_test)
print("Accuracy:", accuracy)
```

회귀, 분류 및 순위 문제를 포함한 다양한 기계 학습 작업에 널리 사용

**대규모 데이터 세트를** 처리하고 멀티 코어 머신에서 잘 작동하므로 대규모 머신 러닝 문제에 적합

모델 성능을 개선하고 과적합을 줄이는 데 도움이 되는 **정규화**, **조기 중지** 및 **누락된 값 처리**와 같은 몇 가지 고급 기능을 제공



코드 예제

이번 대회를 진행하면서 EDA 전처리 모델링 등을 각각 함수 하나로 통합하면서 시간은 좀 걸려도 코딩능력이 향상되었고 SMOTE, StratifiedKFold, 스텀킹 등 파이썬 머신러닝 완벽가이드' 책에서 배운 내용을 적용해 볼 수 있어서 좋았다. 또한 유의성 검정 다중공선성 제거 등 학과에서 배우는 내용을 이용해 볼 수 있는것도 좋았다.

최지혁

이번 프로젝트는 전공 지식과 동아리에서 배운 스킬을 적절히 잘 사용한 프로젝트였다. 전공시간에 배운 변수의 유의성 검정을 통해 변수 선택을 진행했다. 함수화를 통해 반복적인 코드를 간소화했다. 동아리에서 학습한 EDA, 모델링 전처리, 모델링 코드를 사용해 프로젝트를 마무리했다. 또한 모델링 시 전처리의 중요성에 대해 다시 느끼는 계기가 되었다.

신주연

EDA과정, 파생변수를 다루는 과정에서 많은 아이디어와 경험이 필요하다고 느꼈다. 데이터의 종류에 따라 적용하는 알고리즘이 다르기 때문에 데이터의 종류별 분석 방법들을 다양하게 알아야 하며 변수 선택 과정을 원활하게 하기 위해 VIF, 다중공선성과 같이 다양한 방법들을 알아야 한다고 생각했다. 앞으로 해야 할 게 많을 거 같다. 또한 각각의 단계마다 데이터 분석이 어떤 흐름으로 진행되는지 대략적으로 길을 잡을 수 있었다.

김진호

너무나도 멋있고 완벽하신 DNA 선배님들이 하나부터 열까지 다 알려주셔서 너무 감사했고 배울 점이 굉장히 많다고 느꼈다. 지금까지 이렇게 대회에 깊이 파고든 적이 없었는데 이번 대회를 진행하면서 데이터 전처리 과정과 모델링 전처리 과정이 엄청 중요하다는 것을 다시 한번 느꼈다. 아직도 모르는 것이 아주 많기 때문에 이번 프로젝트에서 새롭게 알게 된 내용을 복습하고 더 많이 공부해야겠다고 생각했다.

박은비



GitHub: 프로그래머 면접에서 GitHub 계정이 일종의 포트폴리오 역할을 할 수 있는 개인사이트  
다양한 기록이 있으니 궁금하신 분들 놀러오세요! 😊

최지혁 <https://github.com/weed0328>

신주연 <https://github.com/juyeon-shin>

박은비 <https://github.com/eunbili> ( 준비 중 )

김진호 <https://github.com/kxino> ( 준비 중 )

*To be continue...*