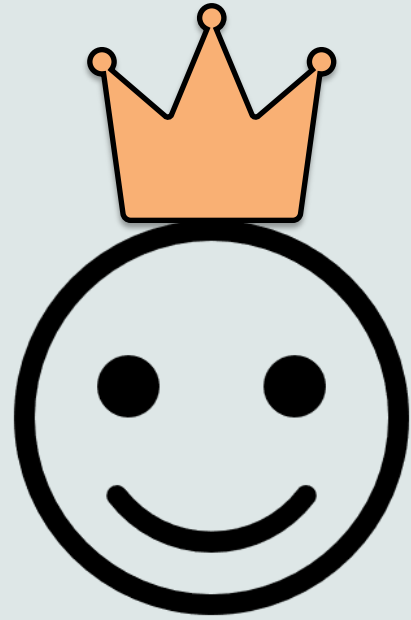


정기 예금 가입 예측 모델

Perfect Bank(김희동, 노태래, 변종우, 이은진)

팀원 소개(Perfect Bank)



김희동

- 팀 큐레이터
- 팀 리더
- PPT작성
- 금융 도메인



노태래

- 메인 코더
- 최종발표
- 금융 도메인



변종우

- 서브 코더
- 중간 발표
- 서기(회의)
- 금융 도메인



이은진

- 서브 코더
- 통계확인
- 금융 도메인

목차

1. 프로젝트 제안배경
2. Timetable & Workflow
3. 데이터 확인(EDA)
4. 데이터 전처리
5. 모델 및 성능 평가
6. Insight 및 한계점
7. 참고문헌

1.프로젝트 제안배경

* 정기예금이란?

고객(예금자)이 금융기관과 사전에 일정기간을 정하여 원금과 이자를 수취하기 위한 목적을 갖는 금융상품


프로젝트 제안배경

- 최근 금융기관은 기존 고객들을 대상으로 정기 예금 유치를 위해 다양한 마케팅 전략을 수립하고 있으며, 효과적인 금융 마케팅을 위해서는 기존의 고객 정보를 바탕으로 예측 모델을 구성한 뒤에 잠재적인 고객을 파악하는 것이 매우 중요
- 금융 기관은 지급준비제도에 따라 예금자의 예금액 지급준비율을 제외한 나머지 금액을 이용하여 정기적금보다 많은 금액을 투자 또는 대출 사업을 위한 자본금으로 운용
- 은행의 사업성 증진을 위해서는 정기예금 상품의 가입자를 예측하는 것이 중요

그린포스트코리아

우리은행, '고객 데이터 플랫폼' 구축으로 만족도 높인다

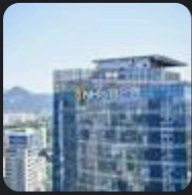
우리은행은 은행권 최초로 고객 접점 채널에서 발생하는 데이터를 실시간으로 수집·통합해 활용하기 위한 '고객 데이터 플랫폼(CDP)' 구축 사업에...



뉴스저널리즘

NH농협은행, 고객 맞춤형 초개인화서비스 '마케팅허브' 오픈

NH농협은행은 '차세대 정보계 프로젝트(프로젝트 通)' 결과물 '마케팅허브'를 오픈한다고 30일 밝혔다. 이번에 선보이는 마케팅허브에서는 마케팅정보...



2. Timetable & Workflow

Time Table

프로젝트 기간 : 5 / 23 ~ 6 / 7

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		24 주제 선정	24 논문 학습	25 논문 학습	26 데이터 수집	27
28	29	30 데이터 전처리 중간 발표	31 데이터 전처리	1 데이터 전처리	2 데이터 모델링	3
4	5 성능 평가	6	7 최종 발표			

Workflow

데이터 수집

데이터 확인 (EDA)



데이터 전처리

컬럼 삭제

이상치 대체

Encoding

Resampling

Scailing

PCA

Train/Test 데이터 분리



데이터 모델링

모델 선정

하이퍼 파라미터 조정

성능 평가

3. 데이터 확인(EDA)

EDA(데이터 요약)

종속변수(20개)

(타겟변수)

- 고객의 정기예금 가입여부(y)

- Data set
- 대상 고객 : 41,188
- 데이터 수집처 : Kaggle, Bank Marketing Data Set

은행 고객 데이터

- 나이(age)
- 직업(jobs)
- 결혼상태(marital)
- 교육상태(education)
- 신용불량 유무(default)
- 주택대출 유무(housing)
- 개인대출 유무(loan)

경기지표

- 고용 변동률 (Emp.var.rate)
- 소비자 가격 지수 (Cons.price.idx)
- 소비자 신뢰 지수 (Cons.conf.idx)
- 리보금리(Euribor3m)
- 취업인원 수 (Nr.employed)

은행 캠페인 연락 데이터

- 연락유형(contact)
- 마지막 연락달(month)
- 마지막 연락요일 (day_of_week)
- 통화시간(duration)

캠페인 관련 데이터

- 연락횟수(campaign)
- 이전연락후 경과일수 (pdays)
- 이전 캠페인 연락횟수 (previous)
- 이전 캠페인 결과 (poutcome)

EDA(변수 확인)

- 데이터 타입 확인
- 숫자형 변수 : 10개
- 범주형 변수 : 11개
- 범주형 변수 인코딩
작업 필요

```
marketing.info()
✓ 0.1s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   age                    41188 non-null  int64
1   job                    41188 non-null  object
2   marital                41188 non-null  object
3   education              41188 non-null  object
4   default                41188 non-null  object
5   housing                41188 non-null  object
6   loan                   41188 non-null  object
7   contact                41188 non-null  object
8   month                  41188 non-null  object
9   day_of_week            41188 non-null  object
10  duration               41188 non-null  int64
11  campaign               41188 non-null  int64
12  pdays                  41188 non-null  int64
13  previous               41188 non-null  int64
14  poutcome               41188 non-null  object
15  emp.var.rate           41188 non-null  float64
16  cons.price.idx         41188 non-null  float64
17  cons.conf.idx          41188 non-null  float64
18  euribor3m              41188 non-null  float64
19  nr.employed            41188 non-null  float64
20  y                       41188 non-null  object
dtypes: float64(5), int64(5), object(11)
```

- 숫자형 데이터의 기술통계량 확인

```
round(marketing.describe(),2)
✓ 0.0s
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00
mean	40.02	258.29	2.57	962.48	0.17	0.08	93.58	-40.50	3.62	5167.04
std	10.42	259.28	2.77	186.91	0.49	1.57	0.58	4.63	1.73	72.25
min	17.00	0.00	1.00	0.00	0.00	-3.40	92.20	-50.80	0.63	4963.60
25%	32.00	102.00	1.00	999.00	0.00	-1.80	93.08	-42.70	1.34	5099.10
50%	38.00	180.00	2.00	999.00	0.00	1.10	93.75	-41.80	4.86	5191.00
75%	47.00	319.00	3.00	999.00	0.00	1.40	93.99	-36.40	4.96	5228.10
max	98.00	4918.00	56.00	999.00	7.00	1.40	94.77	-26.90	5.04	5228.10

EDA(결측치 확인)

- 데이터 프레임 확인
- Sample 수 : 41188
- Columns 수 : 21
- 결측치 수 : 0

```
marketing = pd.read_csv('bank-additional-full.csv', delimiter=';')
marketing
```

Python

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	
...
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0	nonexistent	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0	nonexistent	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1	failure	

41188 rows x 21 columns

marketing.shape

✓ 0.0s

(41188, 21)

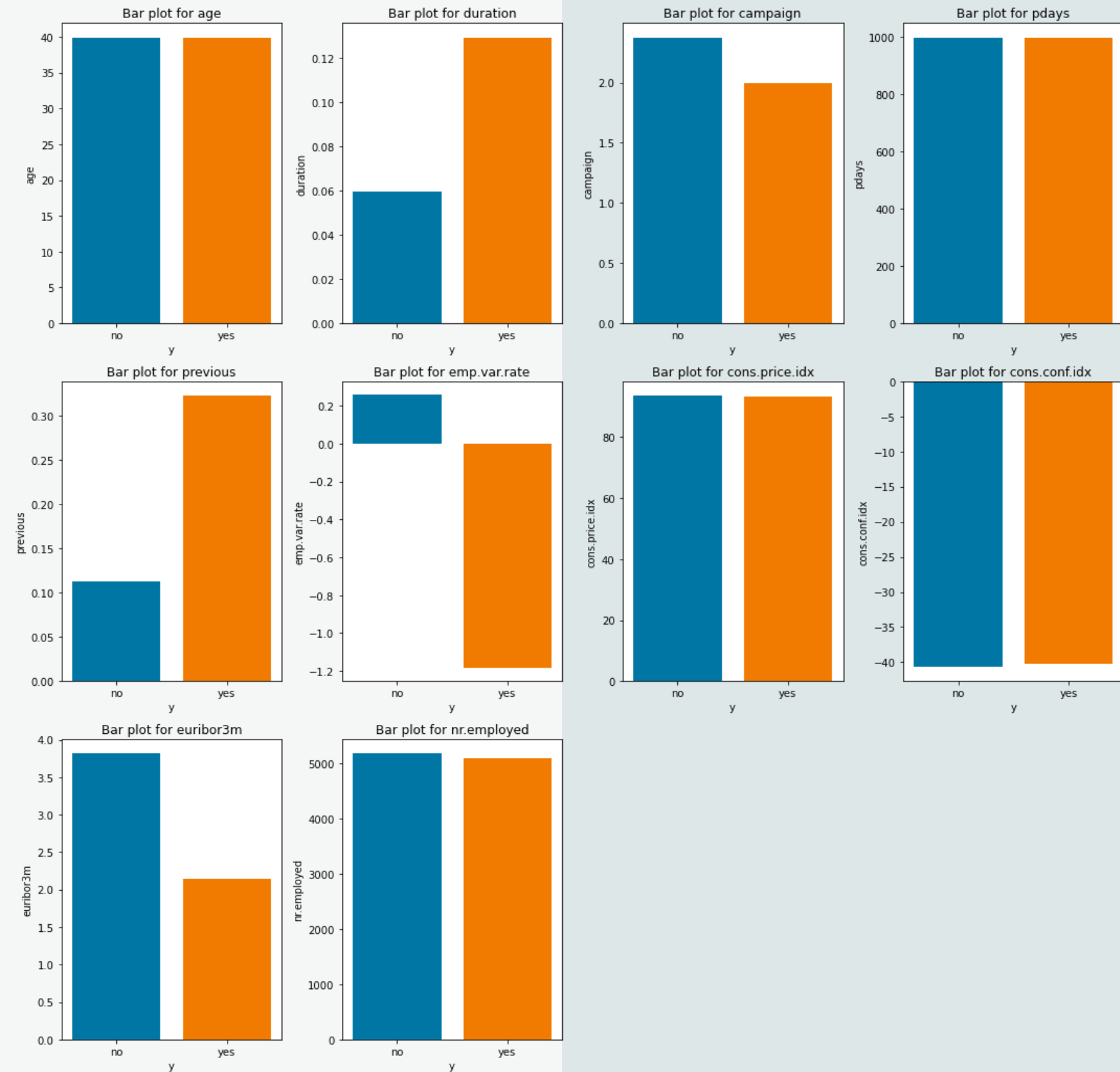
marketing.isna().sum().sum()

✓ 0.0s

0

EDA(변수 시각화 : 전체)

숫자형 종속변수 시각화(10개)



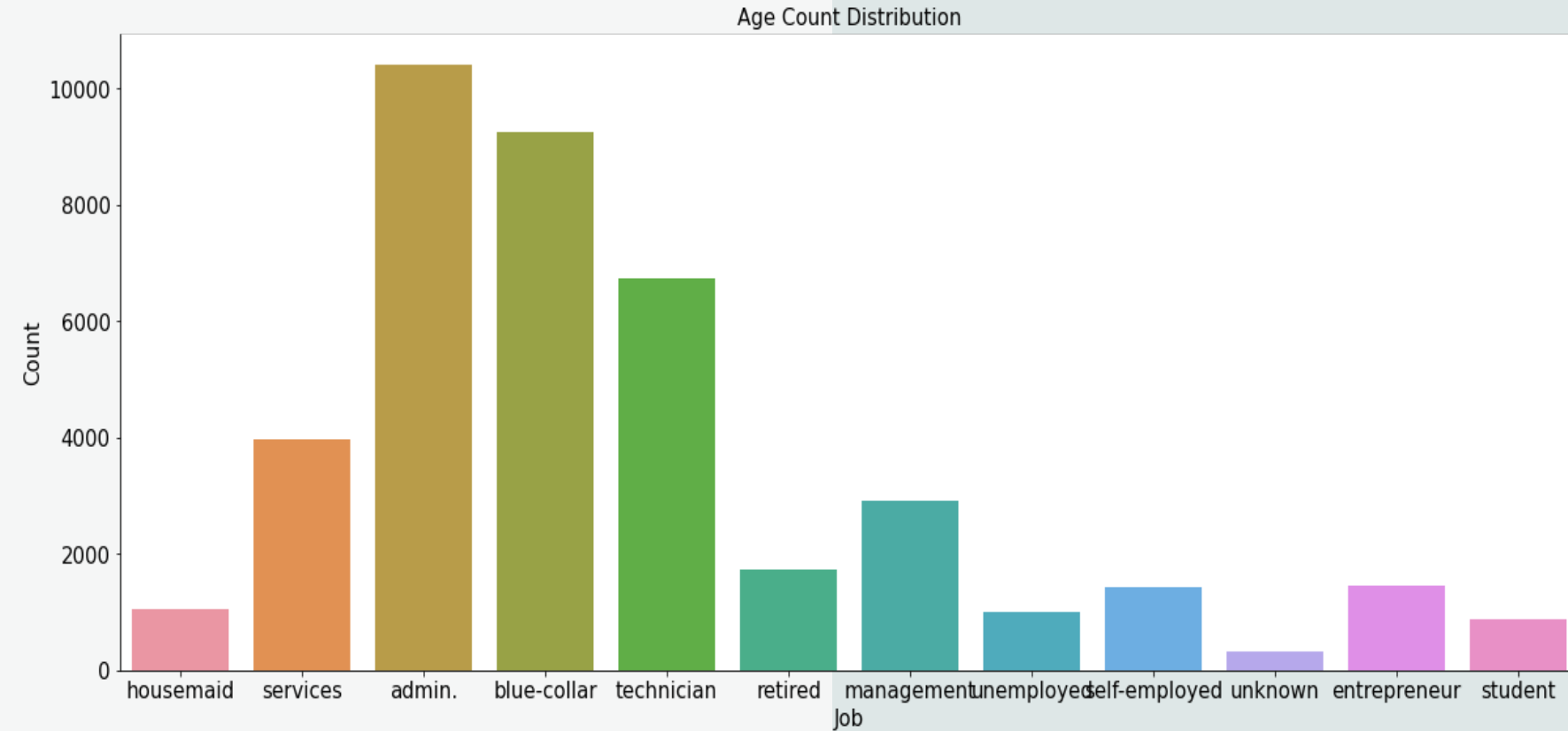
범주형 종속변수, 타겟변수 시각화(11개)



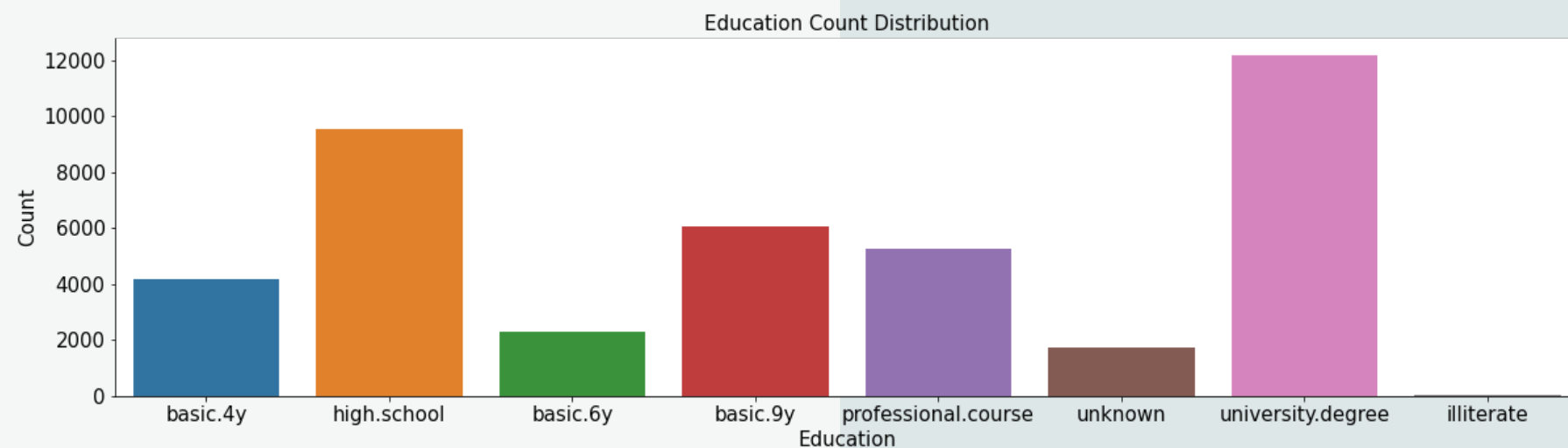
*타겟변수

EDA(범주형 변수 분포 1)

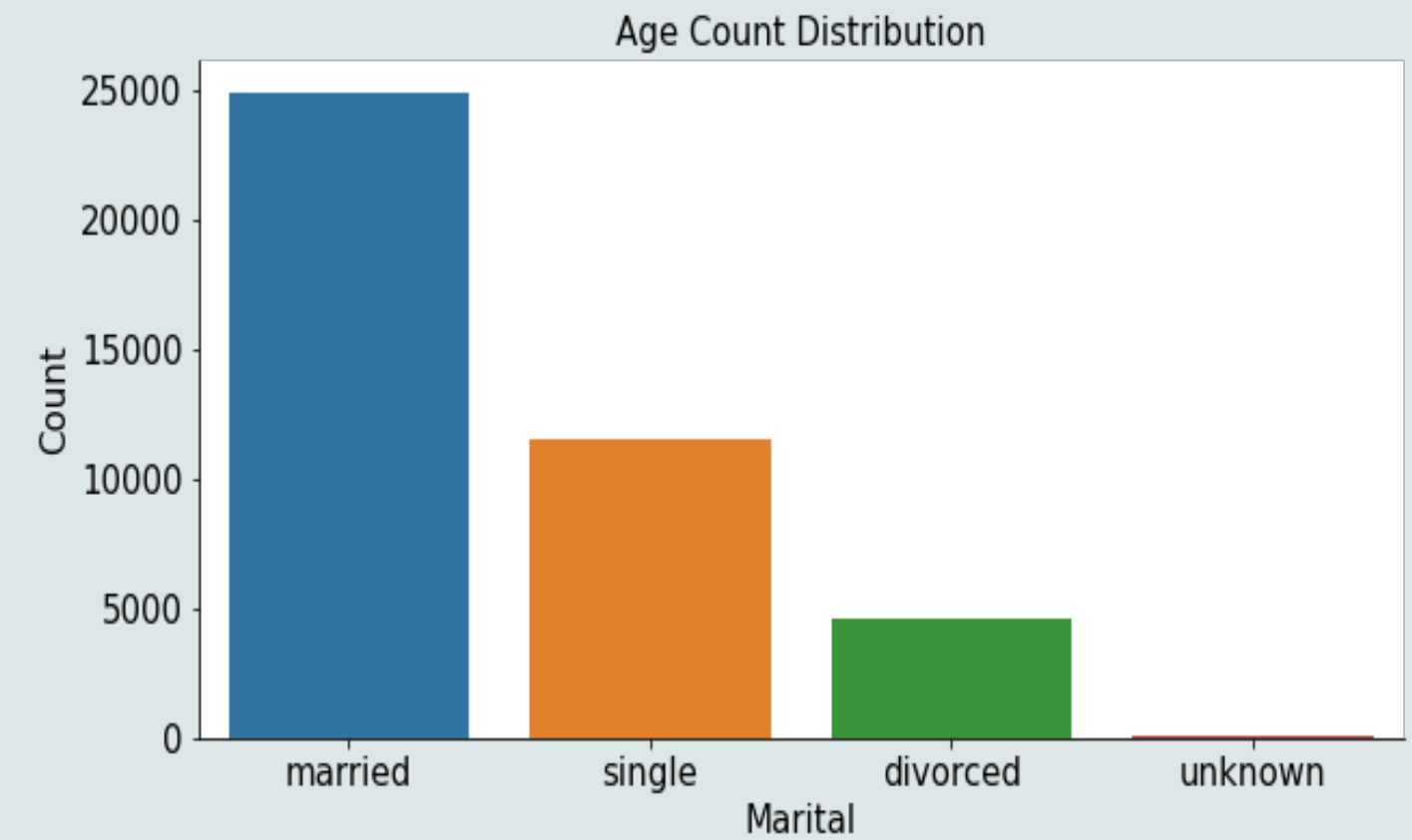
Jobs의 데이터 분포



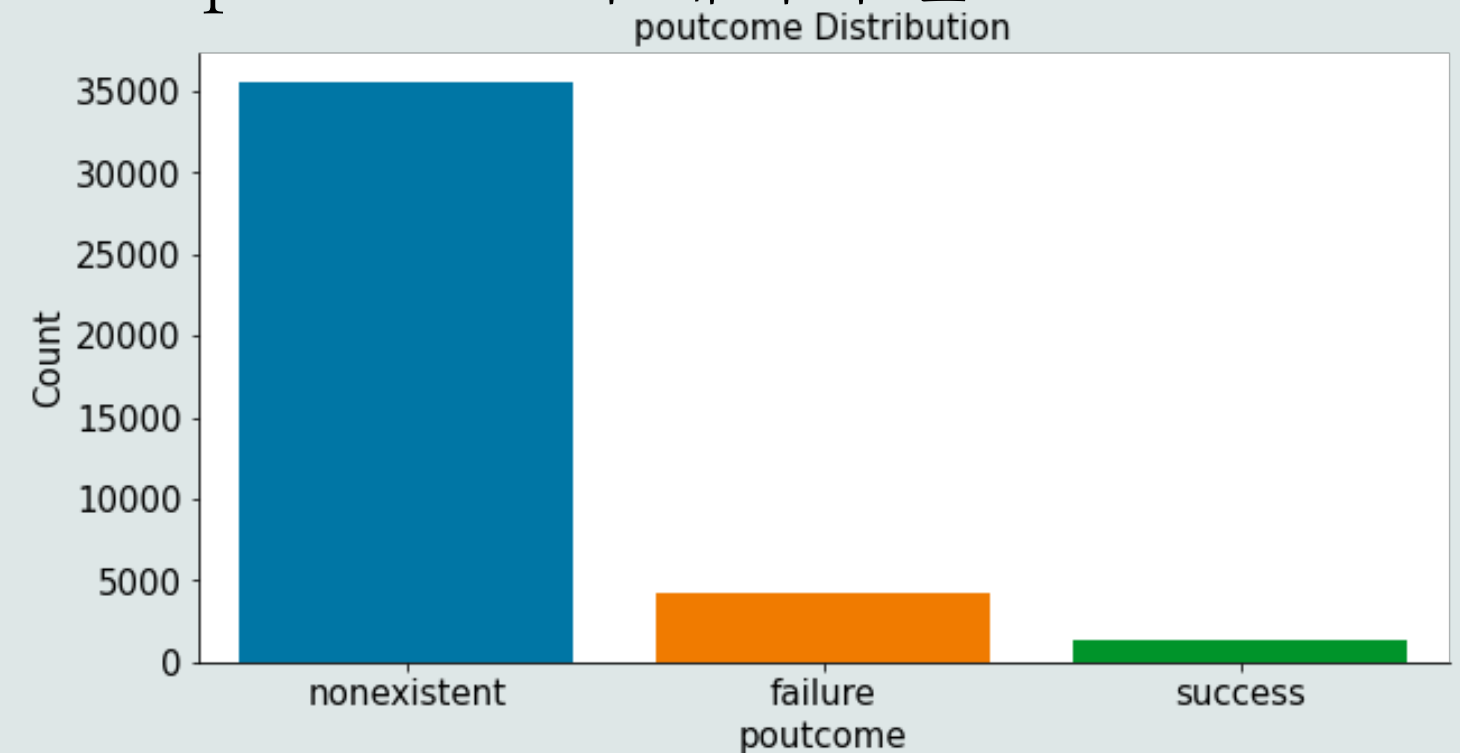
Education의 데이터 분포



Marital의 데이터 분포

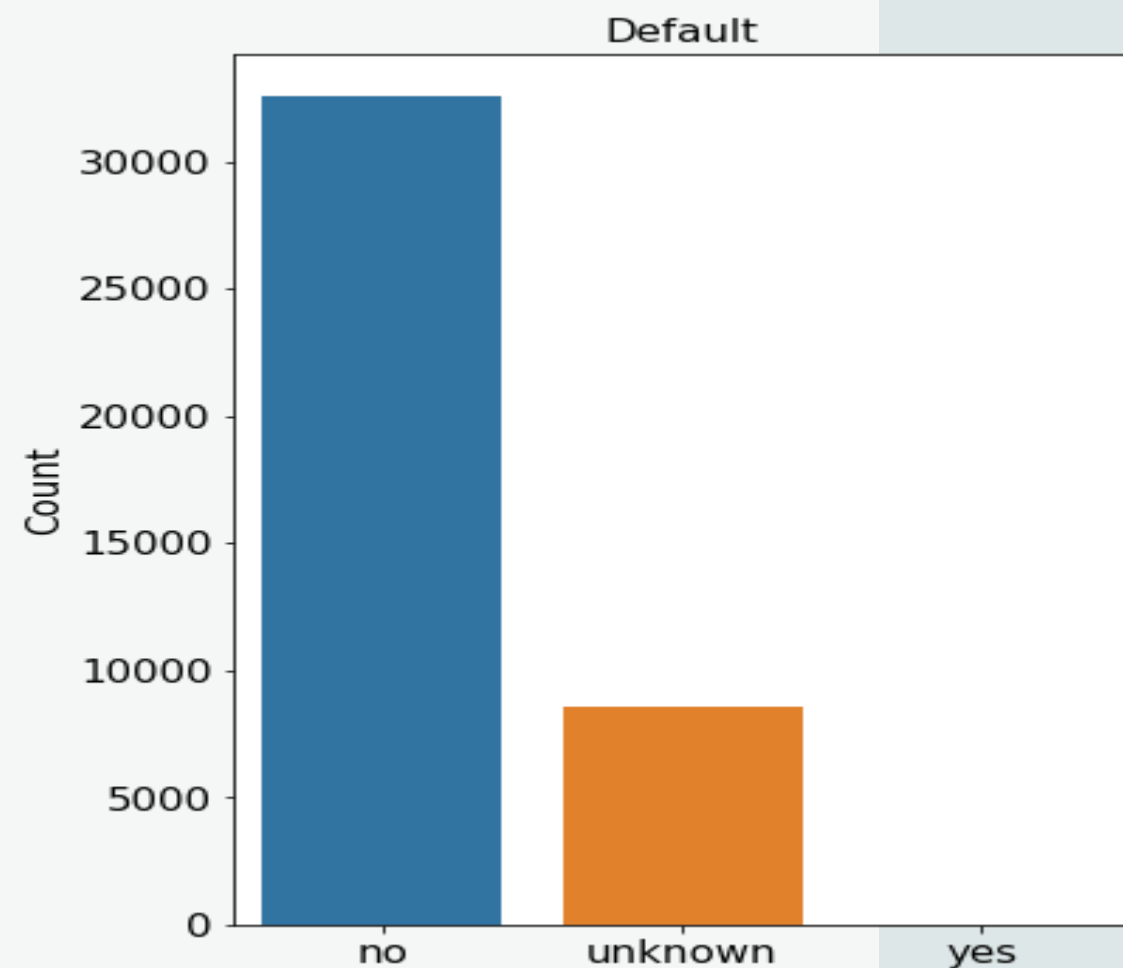


poutcome의 데이터 분포



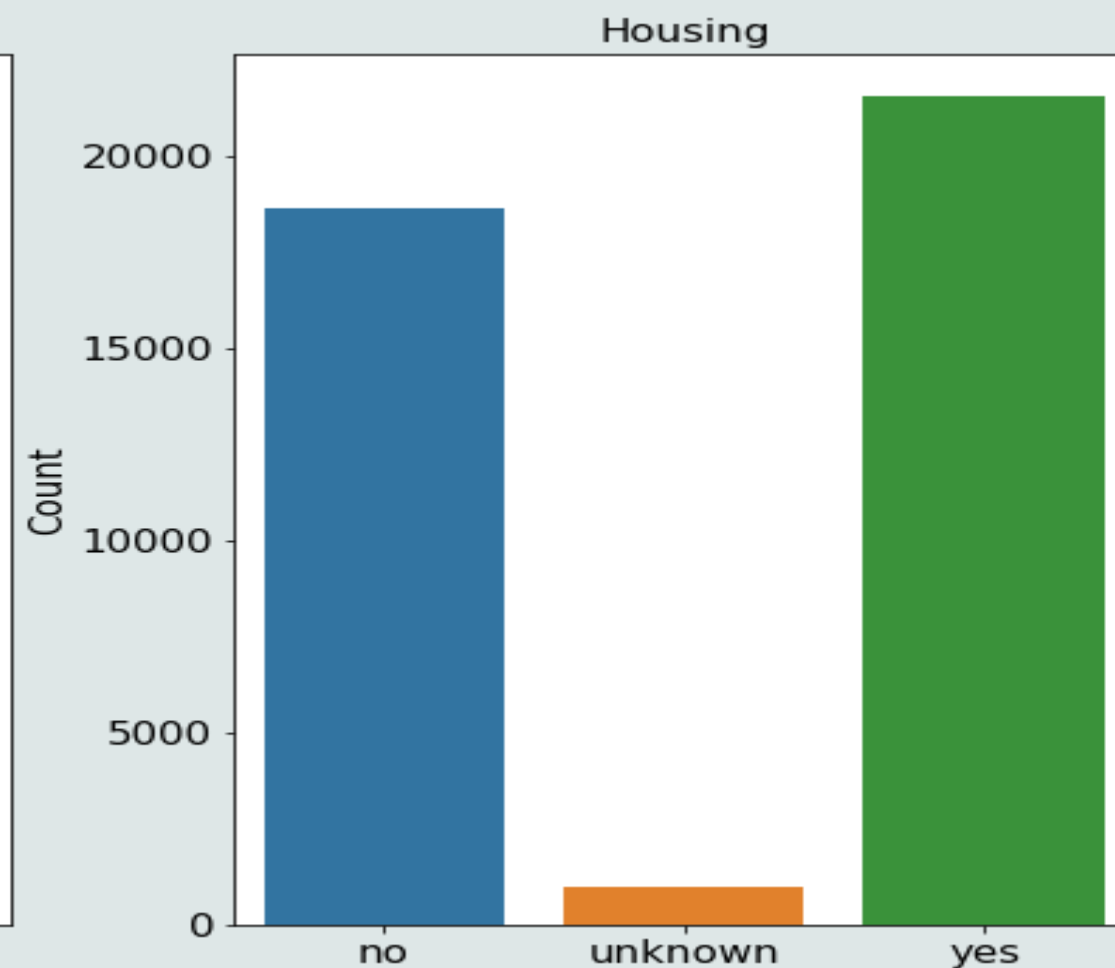
EDA(범주형 변수 분포 2)

Default, Housing, Loan의 데이터 분포



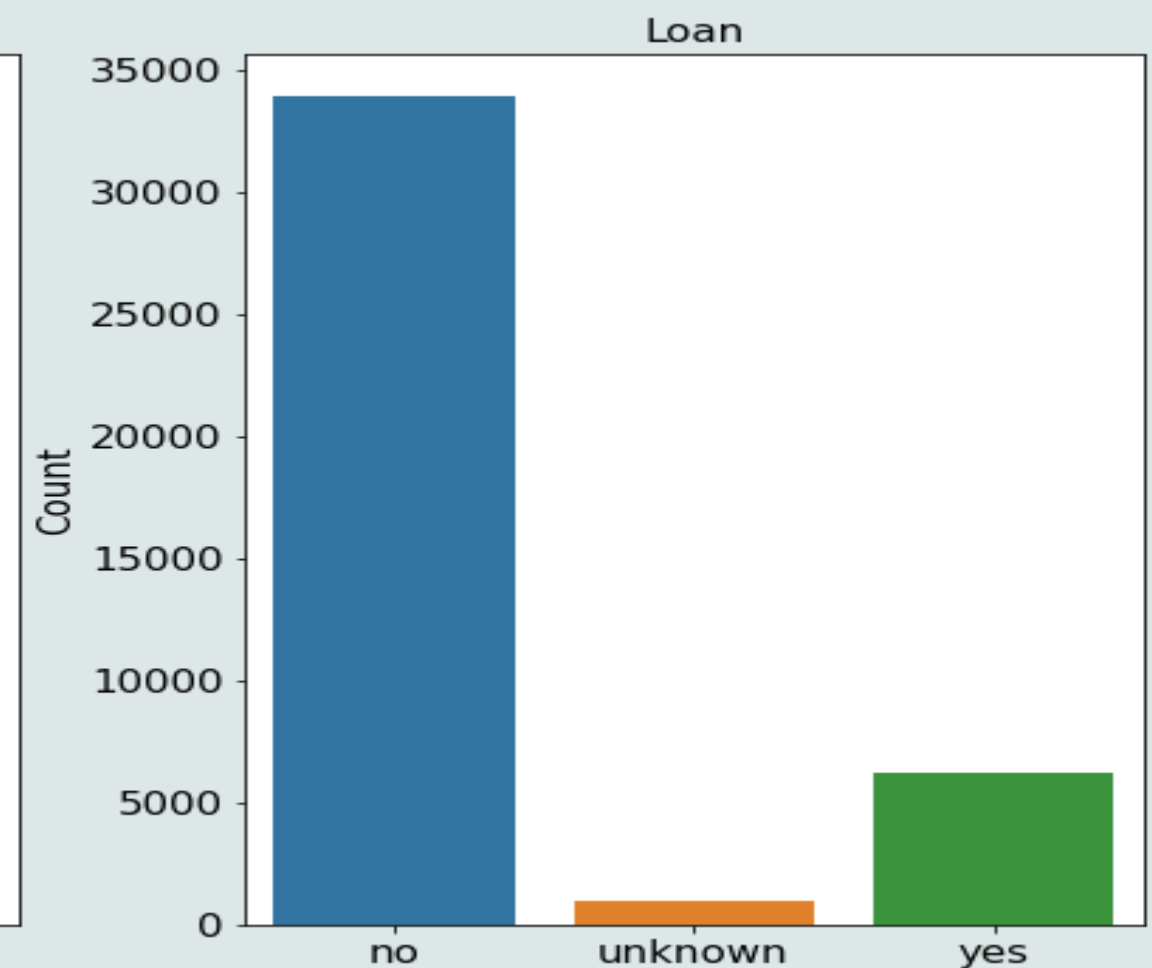
신용불량 유무

- No 응답 수 : 32588
- Unknown 응답 수 : 8597
- Yes 응답 수 : 3



주택대출 유무

- No 응답 수 : 18622
- Unknown 응답 수 : 990
- Yes 응답 수 : 21576

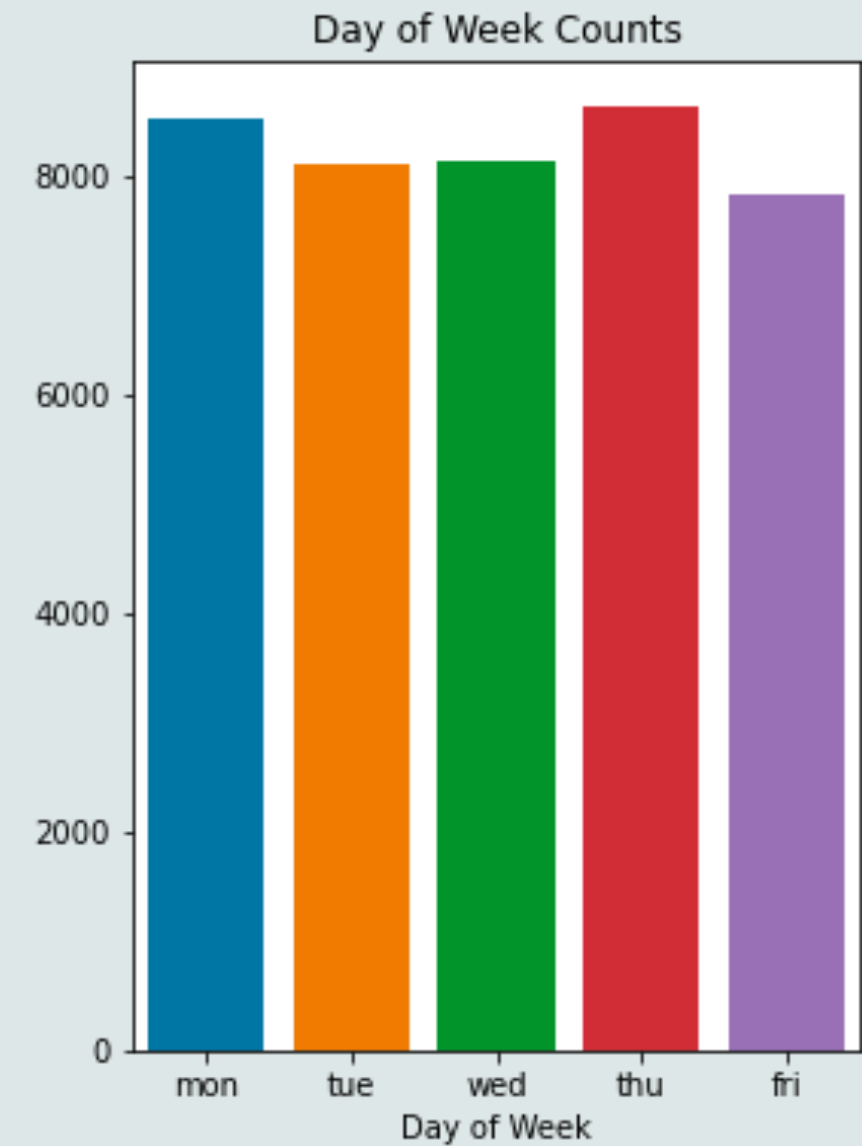
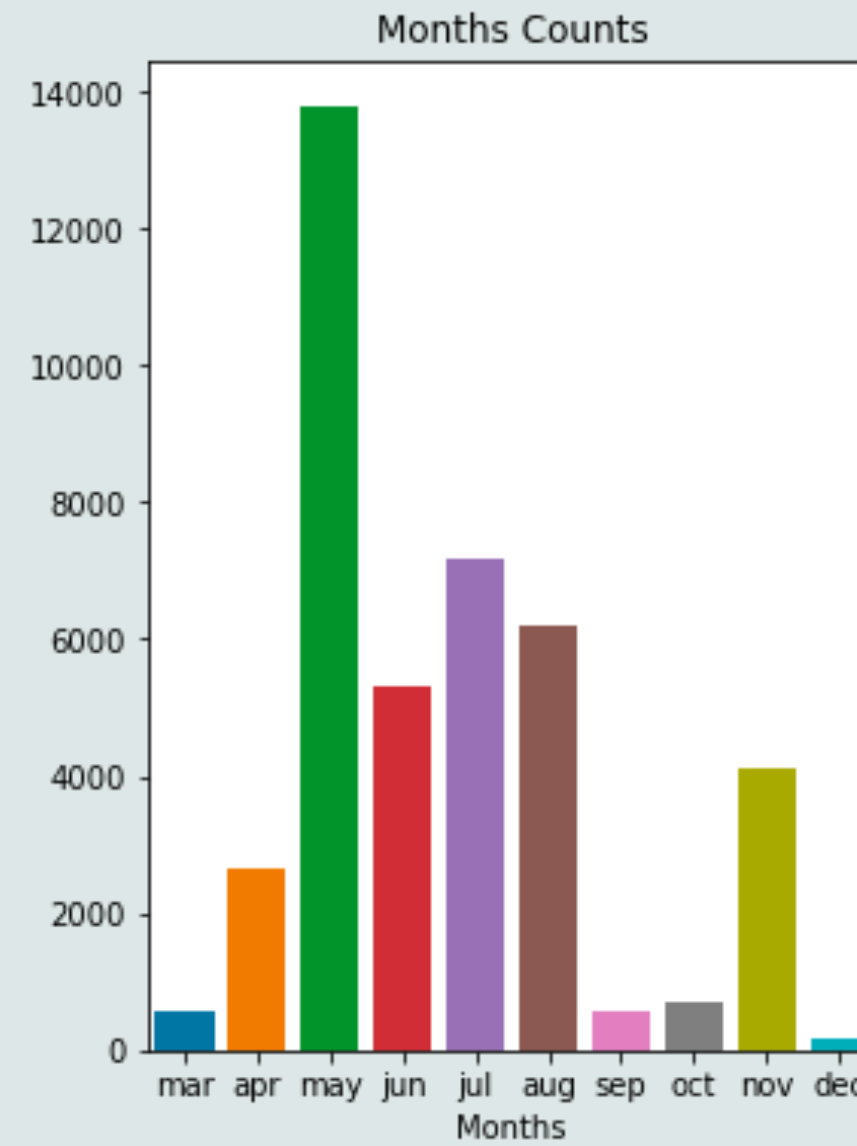
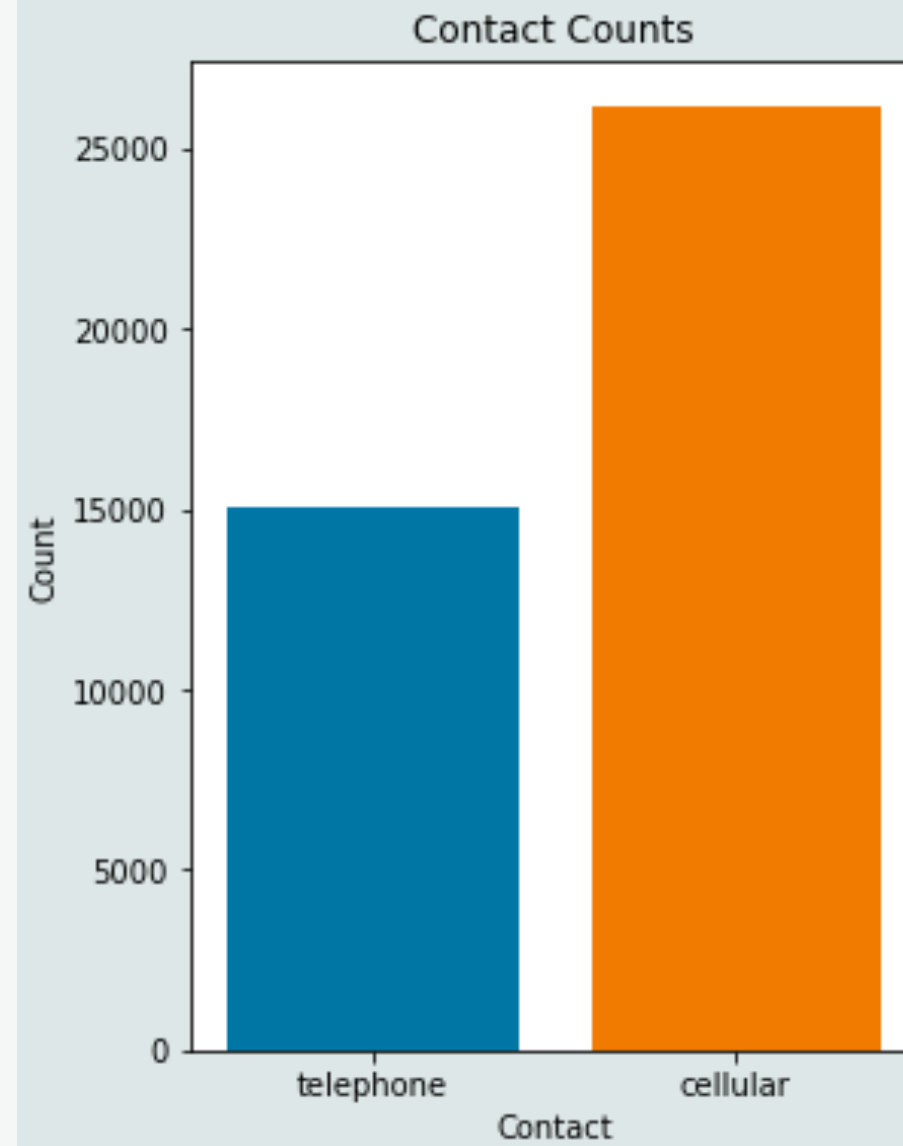


개인대출 유무

- No 응답 수 : 33950
- Unknown 응답 수 : 990
- Yes 응답 수 : 6248

EDA(범주형 변수 분포 3)

- 연락 유형
- 마지막 연락 달
- 마지막 연락 요일



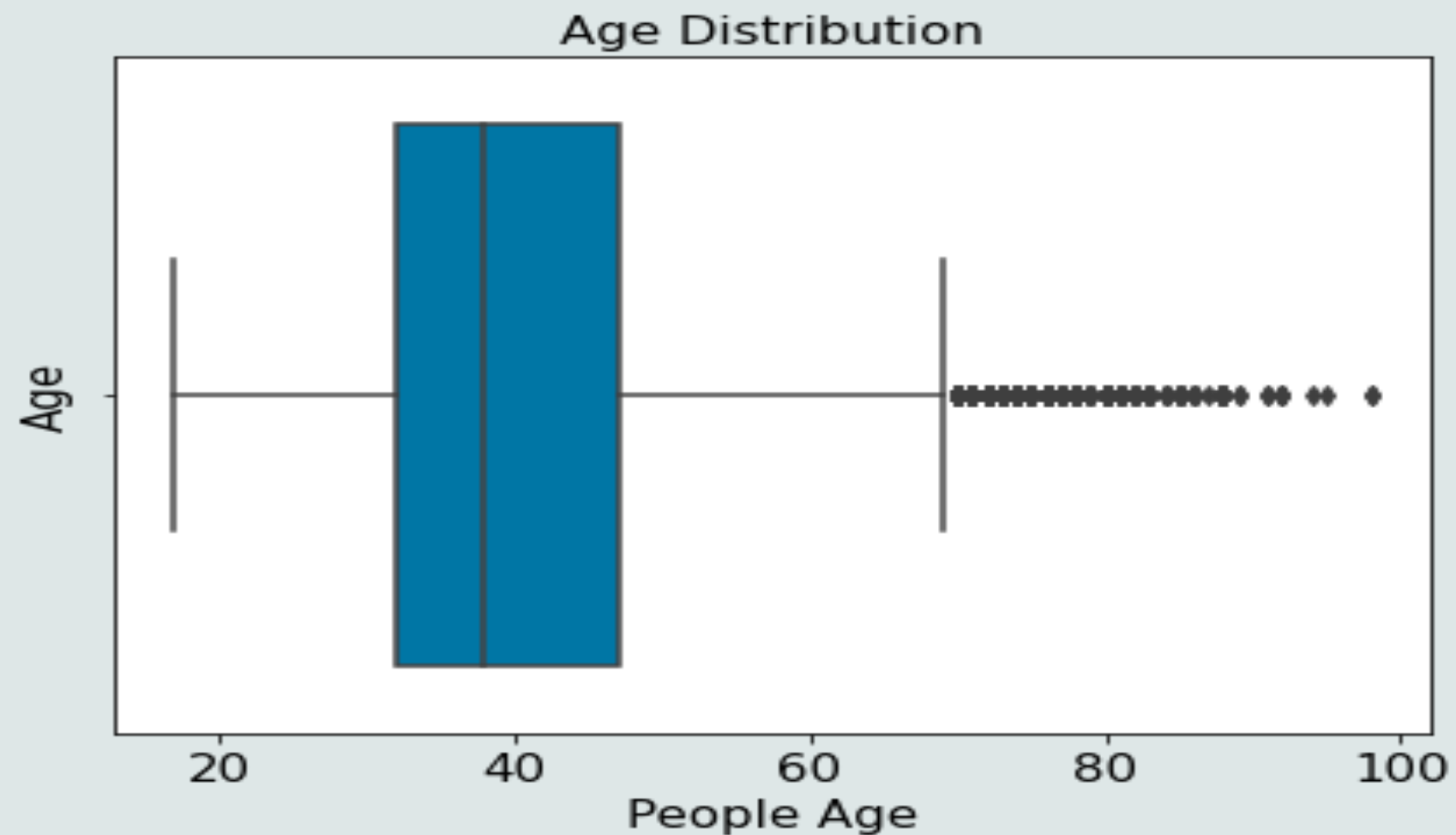
EDA(이상치 확인 : 숫자형 변수 1)

- Age

- 상한치($Q3 + 1.5 * IQR$) = 69.5 이상,
- 하한치 ($Q1 - 1.5 * IQR$) = 9.5 이하의 값은 이상치로 간주
- 이상치 수 : 469
- 이상치 비중 : 1.14%
- 이상치는 원저라이징 통해 해결

```
print('이상치 수: ', marketing[marketing['age'] > 69.6]['age'].count())
print('데이터 수: ', len(marketing))
print('이상치 비중:', round(marketing[marketing['age'] > 69.6]['age'].count()*100/len(marketing),2), '%')
✓ 0.0s
```

이상치 수: 469
데이터 수: 41188
이상치 비중: 1.14 %



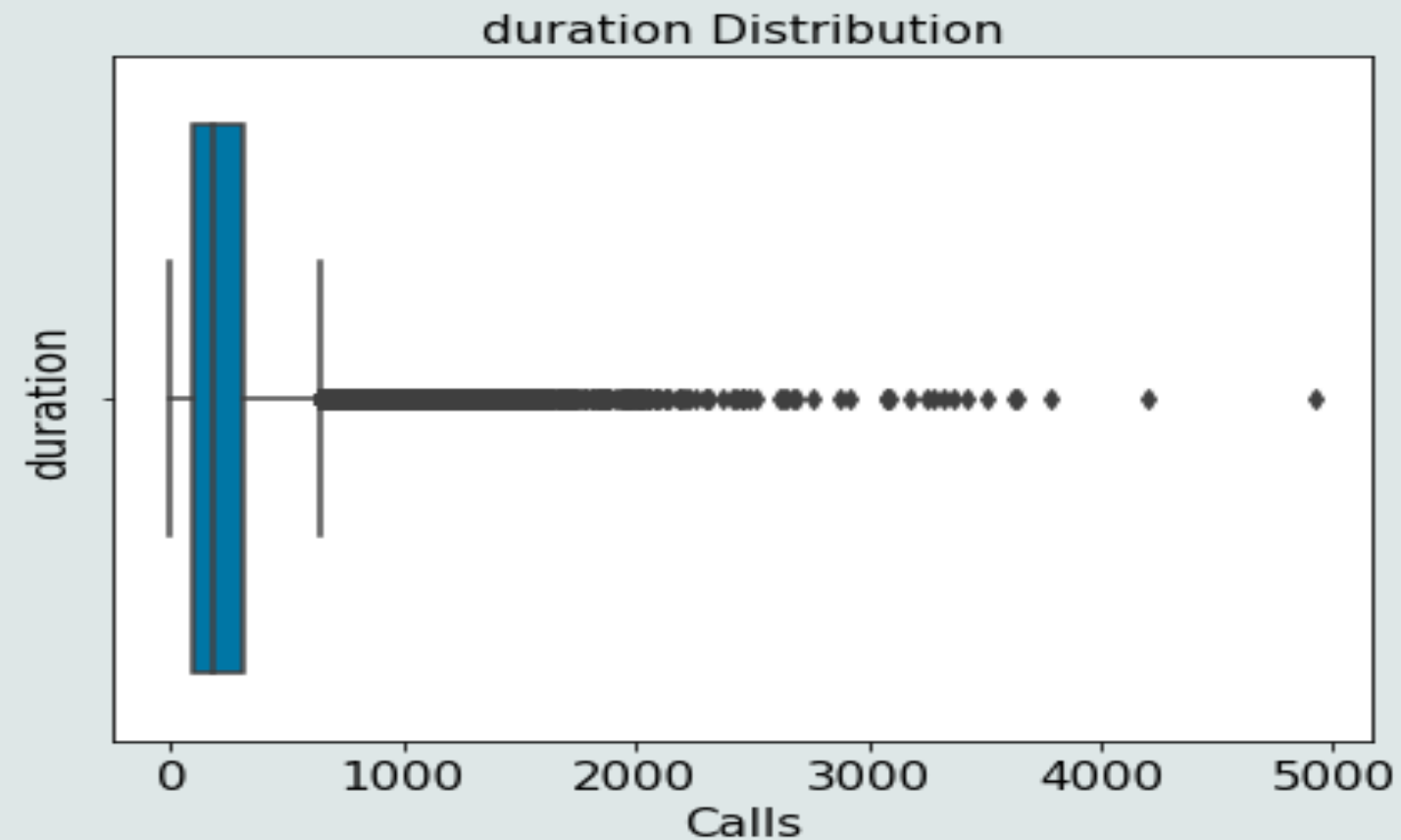
EDA(이상치 확인 : 숫자형 변수 2)

- Duration

- 상한치($Q3 + 1.5 * IQR$)
= 644.5이상,
- 하한치($Q3 - 1.5 * IQR$)
= -223.5이하의 값은 이상치로 간주
- 이상치 수 : 2963
- 이상치 비중 : 7.19%
- 이상치는 원저라이징 통해 해결

```
print('이상치 수: ', marketing[marketing['duration'] > 644.5]['duration'].count())
print('데이터 수: ', len(marketing))
print('이상치 비중:', round(marketing[marketing['duration'] > 644.5]['duration'].count()*100/len(marketing),2), '%')
✓ 0.0s
```

이상치 수: 2963
데이터 수: 41188
이상치 비중: 7.19 %



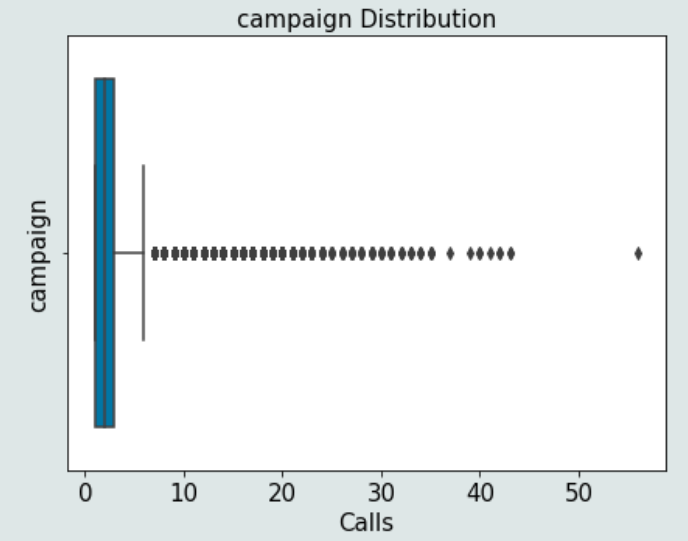
EDA(이상치 확인 : 숫자형 변수 3)

- Campaign

- 상한치 : 6, 하한치 : -2
- 이상치 수 : 2406
- 이상치 비중 : 5.84%

```
print('이상치 수: ', marketing[marketing['campaign'] > 6]['campaign'].count())
print('데이터 수: ', len(marketing))
print('이상치 비중:', round(marketing[marketing['campaign'] > 6]['campaign'].count()*100/len(marketing),2), '%')
✓ 0.0s
```

이상치 수: 2406
데이터 수: 41188
이상치 비중: 5.84 %

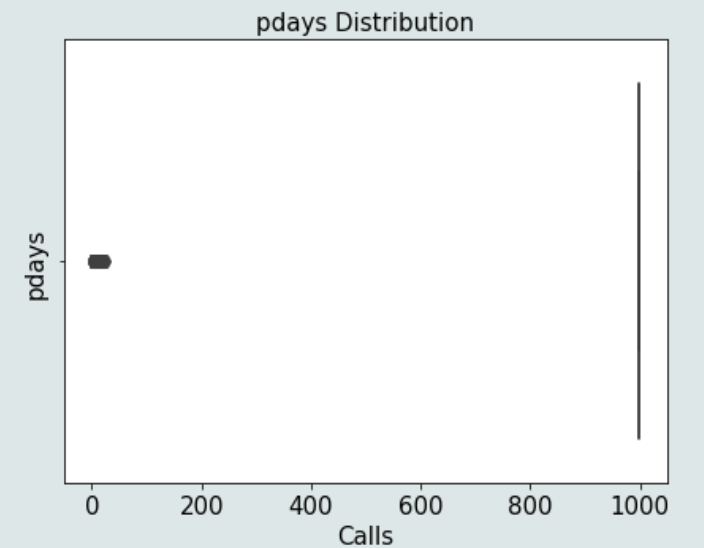


- pdays

- 상한치 : 999, 하한치 : 999
- 이상치 수 : 0
- 이상치 비중 : 0.0%

```
print('이상치 수: ', marketing[marketing['pdays'] > 999]['pdays'].count())
print('데이터 수: ', len(marketing))
print('이상치 비중:', round(marketing[marketing['pdays'] > 999]['pdays'].count()*100/len(marketing),2), '%')
✓ 0.0s
```

이상치 수: 0
데이터 수: 41188
이상치 비중: 0.0 %

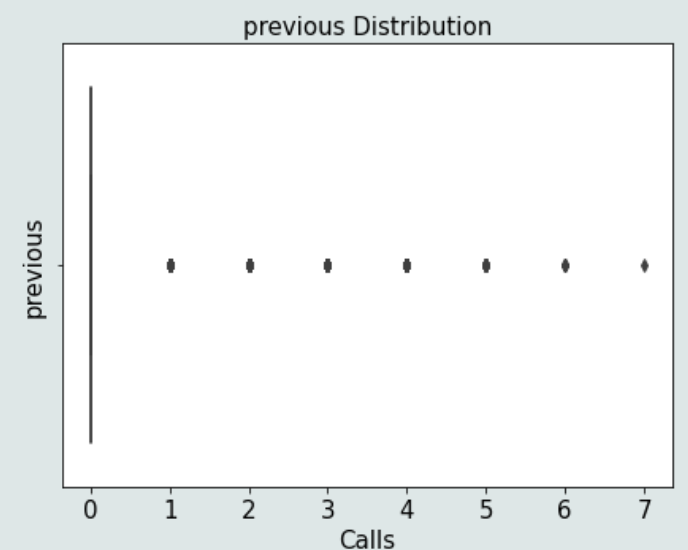


- previous

- 상한치 : 0, 하한치 : 0
- 이상치 수 : 5625
- 이상치 비중 : 13.66%

```
print('이상치 수: ', marketing[marketing['previous'] > 0]['previous'].count())
print('데이터 수: ', len(marketing))
print('이상치 비중:', round(marketing[marketing['previous'] > 0]['previous'].count()*100/len(marketing),2), '%')
✓ 0.0s
```

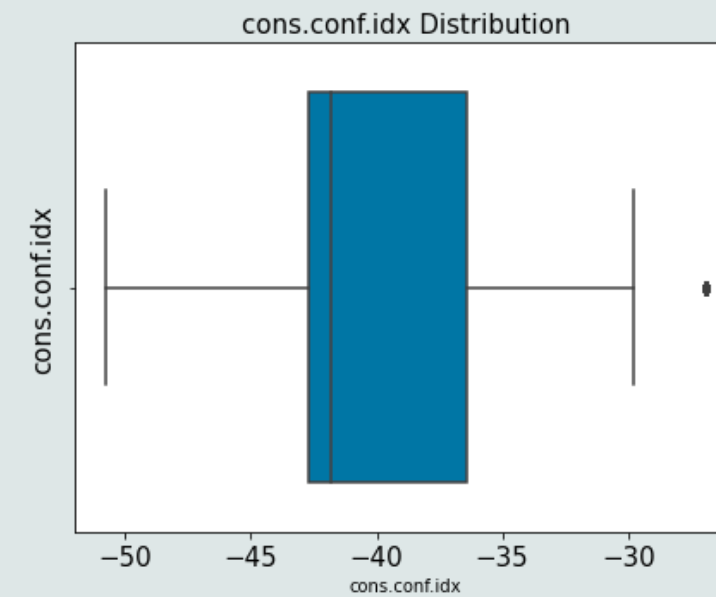
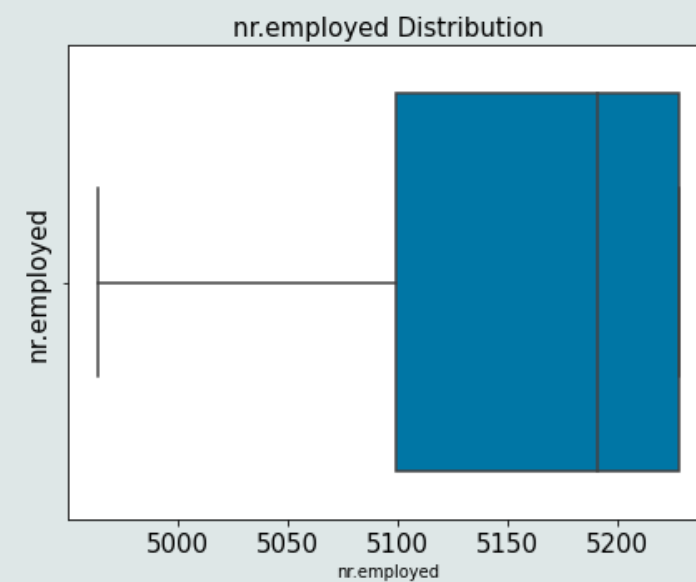
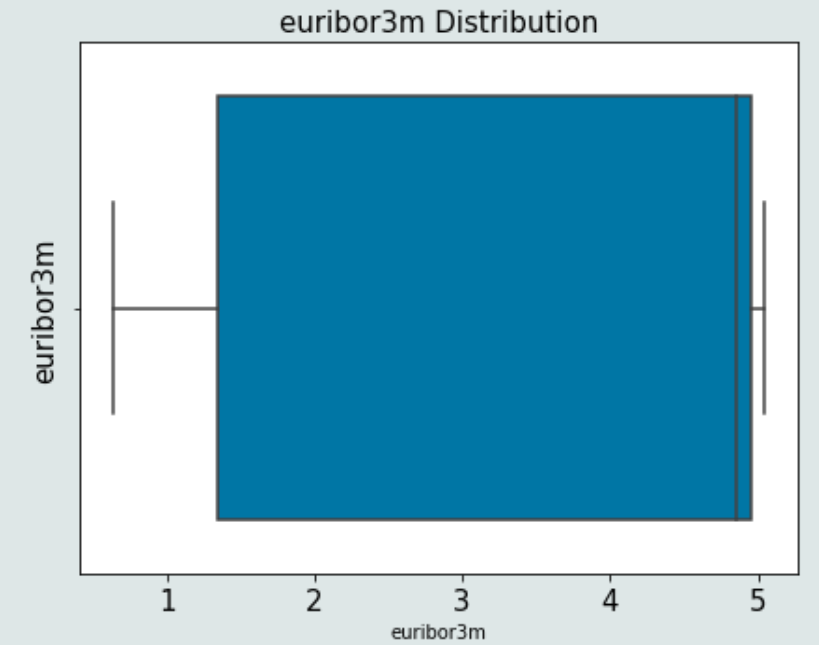
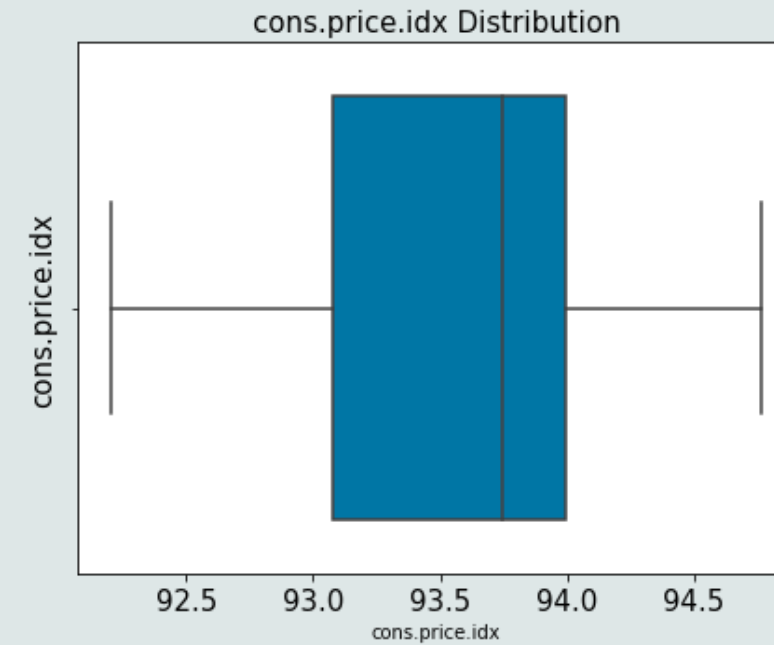
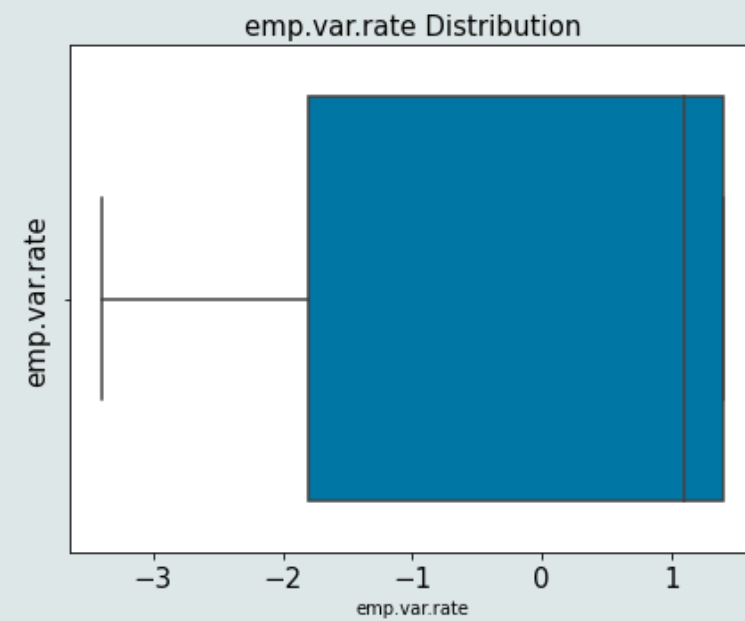
이상치 수: 5625
데이터 수: 41188
이상치 비중: 13.66 %



EDA(이상치 확인 : 숫자형 변수 4)

- Emp.var.rate
- Euribor3m
- Cons.price.idx
- Nr.employed
- 이상치 수 : 0

- Cons.conf.idx
 - 상한치 : -26.94, 하한치 : -52.15
 - 이상치 수 : 447
 - 이상치 비중 : 1.09%

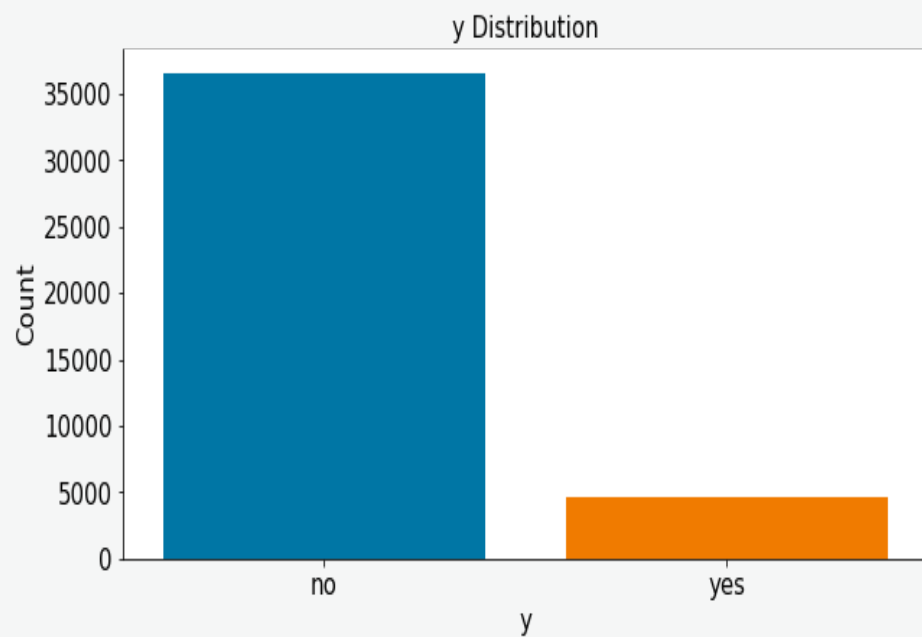


```
print('이상치 수: ', marketing[marketing['cons.conf.idx'] > -26.94]['cons.conf.idx'].count())
print('데이터 수: ', len(marketing))
print('이상치 비중:', round(marketing[marketing['cons.conf.idx'] > -26.94]['cons.conf.idx'].count()*100/len(marketing),2), '%')
✓ 0.0s
```

이상치 수: 447
 데이터 수: 41188
 이상치 비중: 1.09 %

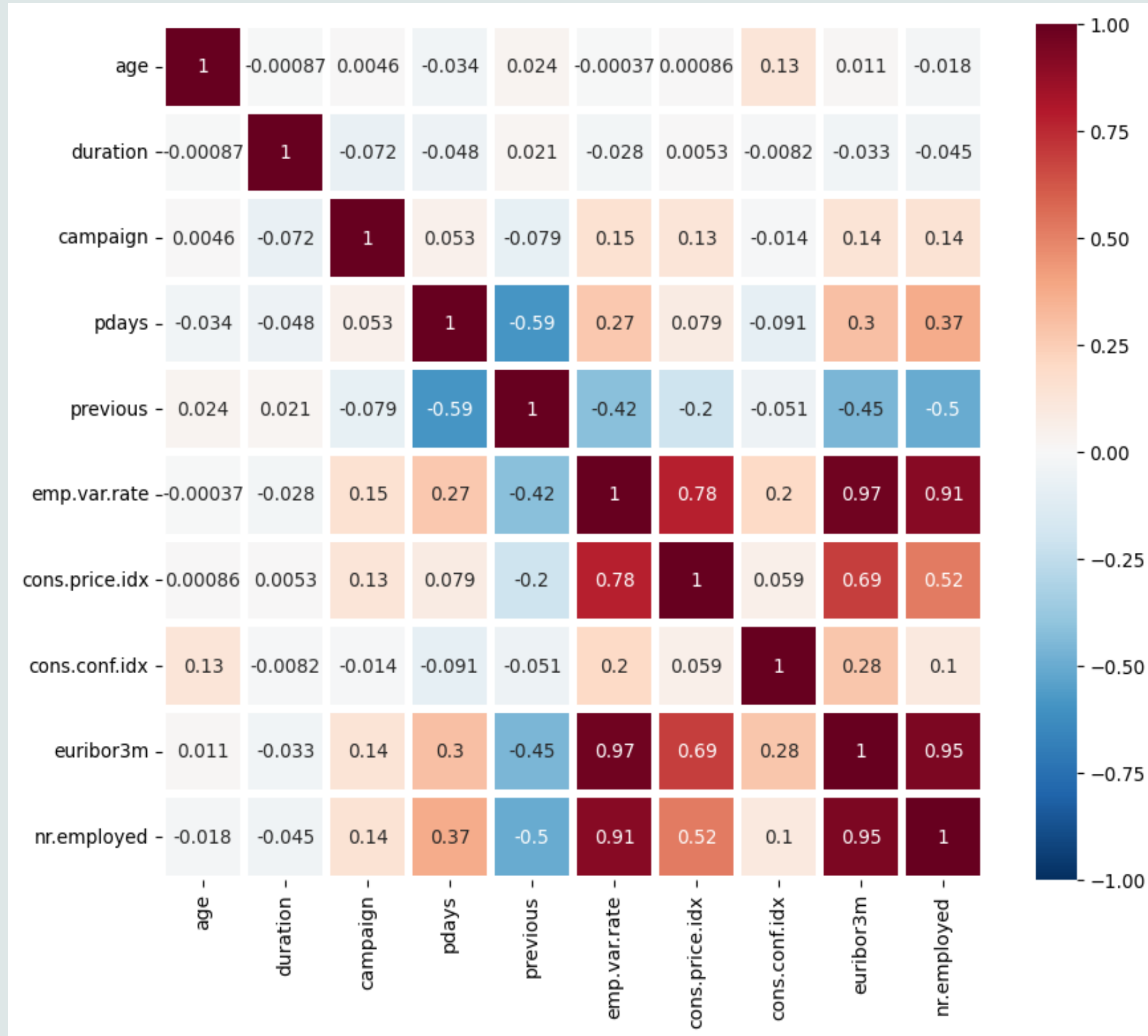
EDA(최초 상관관계)

타겟변수 확인



- No 응답 수 : 36548
- Yes 응답 수 : 4640
- 타겟 변수 불균형이 존재
- 리샘플링을 통해 해결

히트맵을 통한 변수들의 상관관계 확인



경기지표인 고용 변동률, 소비자 가격지수, 리보금리, 취업인원수 서로가 강한 상관관계 존재

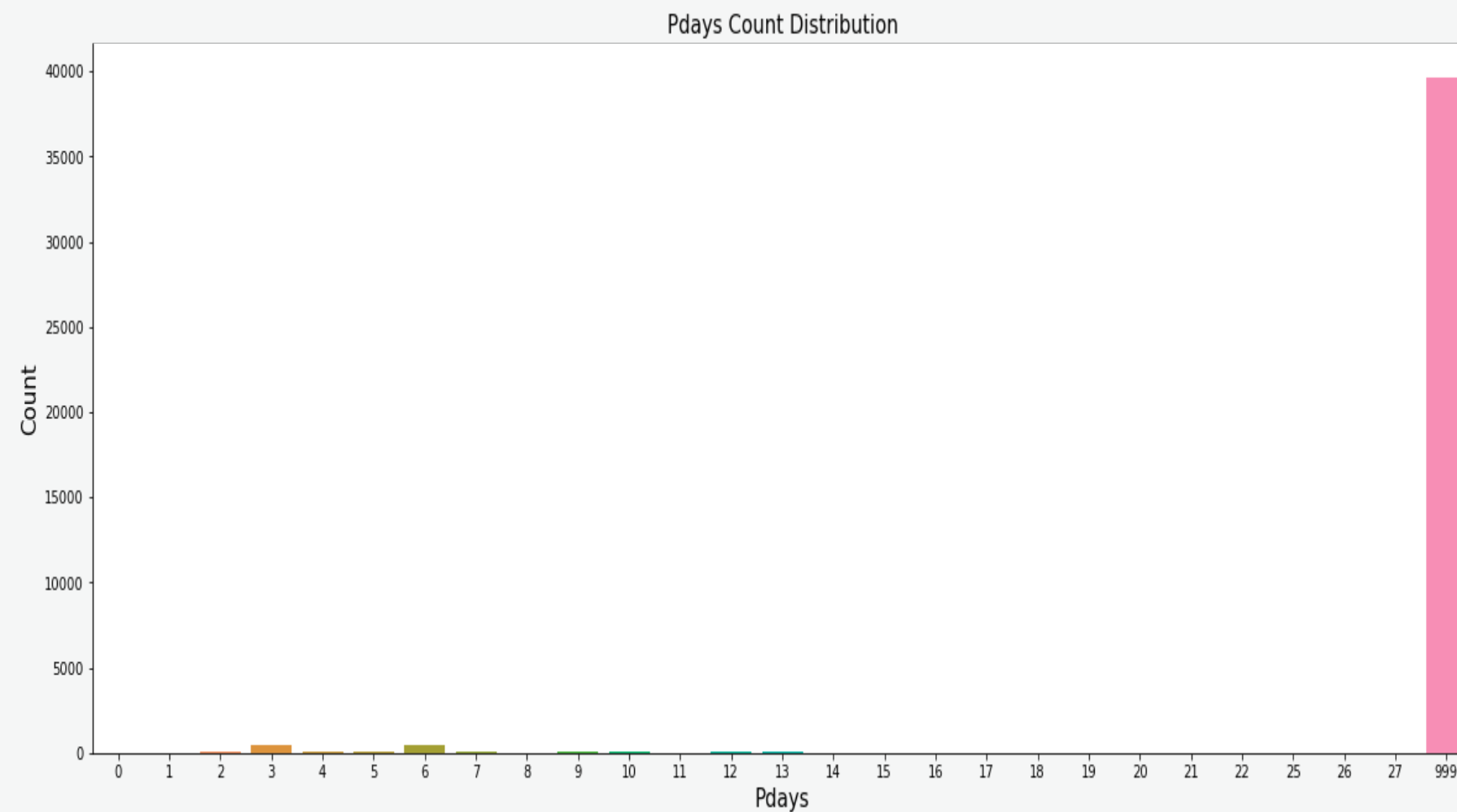
4. 데이터 전처리

데이터 전처리

- 의미 없는 컬럼 삭제
- 원저라이징을 통한 이상치 대체
- 타겟변수 mapping
- 범주형 데이터 인코딩: BinaryEncoding
- Data Resampling : Undersampling
- Scaling : Min_Max , Standard
- 다중공선성 확인 및 PCA
- Data Split: 41,188 고객 데이터 = Train(80%)(32,950) + Test(20%)(8,238)

데이터 전처리(1) : 컬럼 삭제

Pdays의 데이터 분포



- 999라는 데이터(39673개)가 컬럼의 96.32% 차지
- 변수의 분산이 매우 낮음
- 다른 변수와 관계없이 대부분의 데이터가 동일한 값을 가지므로 분석에 유의미한 정보를 가지지 않을 것이라고 판단
- Pdays 컬럼 삭제

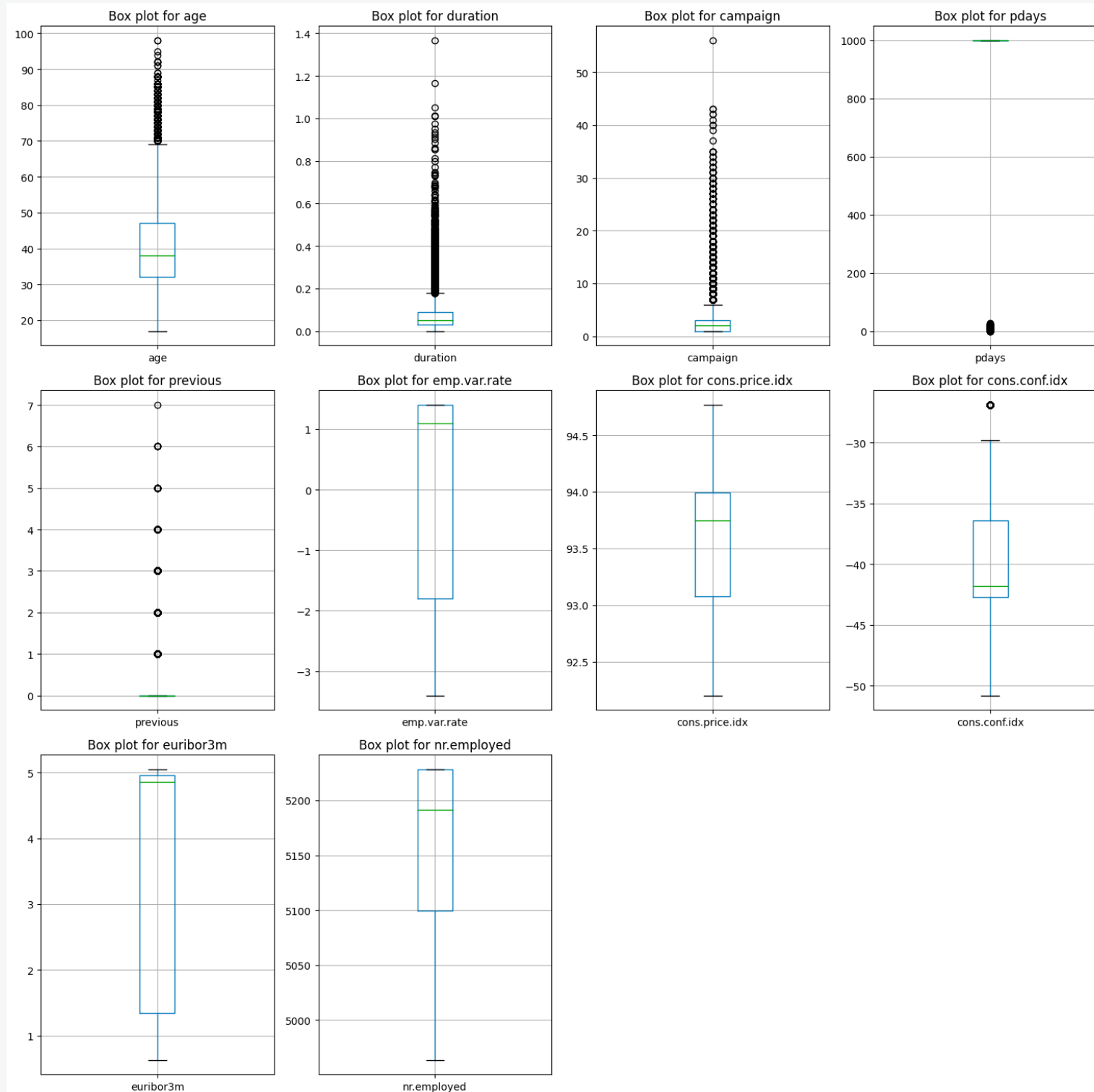
```
print('Pdays:', round(bank[bank['pdays'] == 999]['pdays'].count()*100/len(bank),2), '%')
```

✓ 0.0s

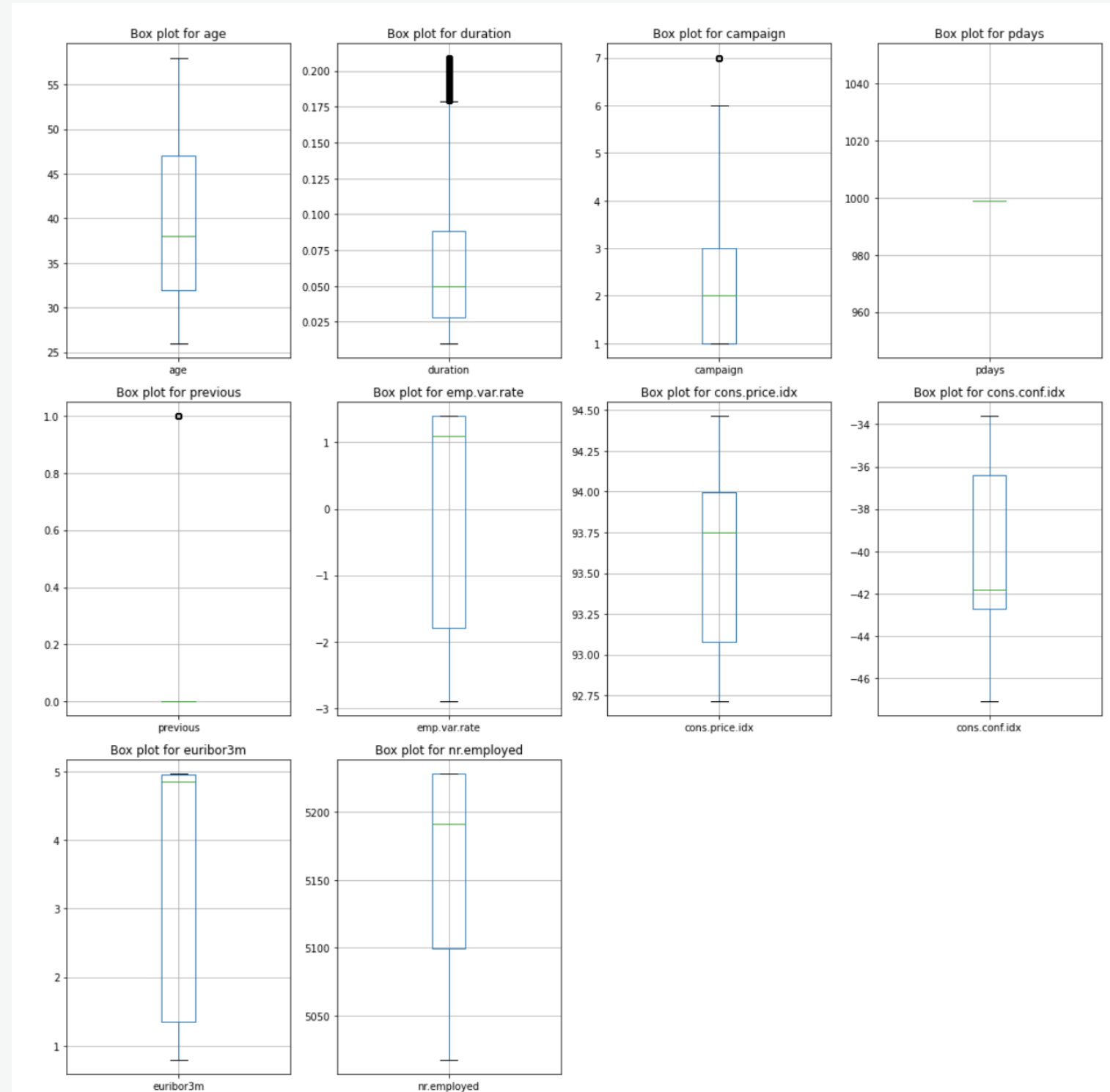
Pdays: 96.32 %

데이터 전처리(2) : 원저라이징

이상치 대체 전



이상치 대체 후(극단치 기준 5% 적용)



데이터 전처리(3) : 타겟변수 mapping

- 예금 가입자는 1, 예금 미가입자는 0으로 표기

```
[ ] # 숫자형 변수들을 따로 추출
num, num_col = hd.getvarcol(marketing, select = 0)
# 범주형 변수들을 따로 추출
obj, obj_col = hd.getvarcol(marketing, select = 1)
# 타겟 변수를 매핑하고 타겟 컬럼을 추출
marketing['y'] = marketing['y'].map(
    {
        'no' : 0,
        'yes' : 1
    }
)
target = marketing['y']
```

```
0      0
1      0
2      0
3      0
4      0
..
41183  1
41184  0
41185  0
41186  1
41187  0
Name: y, Length: 41188, dtype: int64
```

데이터 전처리(4) : 범주형 데이터 인코딩

- Binary Encoding을 이용하여 인코딩 수행
- Binary Encoding은 원-핫 인코딩과는 달리 이진수로 표현되기 때문에 변수의 개수가 많아 지더라도 효과적으로 차원을 줄일 수 있음

```
# 바이너리 인코더로 인코딩을 하고 인코딩된 컬럼과 인코딩 정보가 담긴 딕셔너리를 결과값을 받아온다
obj_encoded, obj_encoded_col, encoded_dict = hd.encoding(marketing)
```

```
obj_encoded
```

	job_0	job_1	job_2	job_3	marital_0	marital_1	marital_2	education_0	education_1	education_2	...	contact_1	month_0	month_1	month_2	month_3	day_of_week_0	day_of_week_1	day_of_week_2	poutcome_0	poutcome_1
0	0	0	0	1	0	0	1	0	0	0	...	1	0	0	0	1	0	0	1	0	1
1	0	0	1	0	0	0	1	0	0	1	...	1	0	0	0	1	0	0	1	0	1
2	0	0	1	0	0	0	1	0	0	1	...	1	0	0	0	1	0	0	1	0	1
3	0	0	1	1	0	0	1	0	0	1	...	1	0	0	0	1	0	0	1	0	1
4	0	0	1	0	0	0	1	0	0	1	...	1	0	0	0	1	0	0	1	0	1
...
41183	0	1	1	0	0	0	1	0	1	0	...	0	0	1	1	0	1	0	1	0	1
41184	0	1	0	0	0	0	1	0	1	0	...	0	0	1	1	0	1	0	1	0	1
41185	0	1	1	0	0	0	1	0	1	1	...	0	0	1	1	0	1	0	1	0	1
41186	0	1	0	1	0	0	1	0	1	0	...	0	0	1	1	0	1	0	1	0	1
41187	0	1	1	0	0	0	1	0	1	0	...	0	0	1	1	0	1	0	1	1	0

```
41188 rows x 28 columns
```

데이터 전처리(5) : Data Resampling

- 먼저 SMOTE로 오버샘플링을 진행했으나 전체 데이터에서 0의 비율이 86%를 차지하여 불균형한 분포로 인해 다수를 예측하는 경향이 발생
- 언더샘플링은 정보 손실의 위험이 존재하지만 다수의 샘플을 제거함으로써 데이터의 균형을 맞추고 소수에 대한 학습과 예측의 정확성을 향상 시킬 수 있음

```
# 언더 샘플링
from imblearn.under_sampling import RandomUnderSampler

undersample = RandomUnderSampler(sampling_strategy='majority')
x_sm, y_sm = undersample.fit_resample(encoded_total[encoded_total.columns.difference(['y'])], encoded_total['y'])

counter_res = Counter(y_sm)
print('Resampled Dataset Shape %s' % counter_res)
```

✓ 0.5s

Resampled Dataset Shape Counter({0: 4640, 1: 4640})

Undersampling 결과 : 0(no) : 4640개, 1(yes) : 4640개 추출

데이터 전처리(6) : Data Scaling

- 값의 편차를 줄이기 위한 정규화, 표준화 스케일링 진행

```
# 스케일러 선택. 숫자형 변수만 넣어야함
def myscaler(x_num, choose=0):
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.preprocessing import StandardScaler
    from sklearn.preprocessing import RobustScaler

    num_col = list(x_num.columns)

    if choose == 0 :

        # Scaler 객체 생성
        scaler = MinMaxScaler()
        x_num_scaled = scaler.fit_transform(x_num)

        # print('\t\t(min, max) (mean, std)')
        print('Scaled (%.2f, %.2f) (%.2f, %.2f)' %(x_num_scaled.min(), x_num_scaled.max(), x_num_scaled.mean(), x_num_scaled.std()))
        x_num_scaled = pd.DataFrame(x_num_scaled, columns=num_col)

    return x_num_scaled
```


데이터 전처리(6) : Data Scaling

- 값의 편차를 줄이기 위한 정규화, 표준화 스케일링 진행

```
elif choose == 1:

    # Scaler 객체 생성
    scaler = StandardScaler()
    x_num_scaled = scaler.fit_transform(x_num)

    # print('\t\t(min, max) (mean, std)')
    print('Train_scaled (%.2f, %.2f) (%.2f, %.2f)' %(x_num_scaled.min(), x_num_scaled.max(), x_num_scaled.mean(), x_num_scaled.std()))
    x_num_scaled = pd.DataFrame(x_num_scaled, columns=num_col)

    return x_num_scaled

elif choose == 2:

    # Scaler 객체 생성
    scaler = RobustScaler()
    x_num_scaled = scaler.fit_transform(x_num)

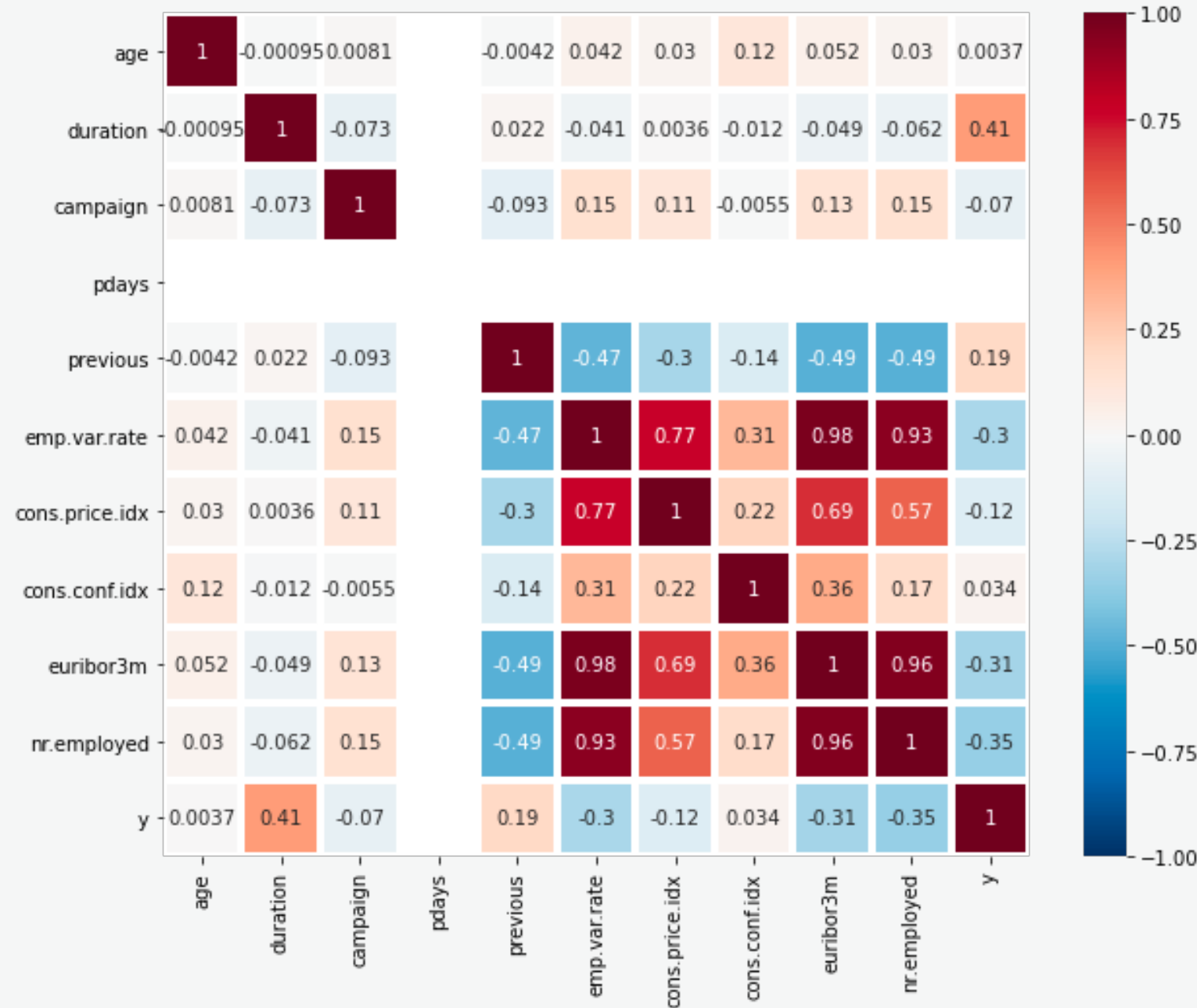
    # print('\t\t(min, max) (mean, std)')
    print('Train_scaled (%.2f, %.2f) (%.2f, %.2f)' %(x_num_scaled.min(), x_num_scaled.max(), x_num_scaled.mean(), x_num_scaled.std()))
    x_num_scaled = pd.DataFrame(x_num_scaled, columns=num_col)

    return x_num_scaled

else:
    x_num_scaled = x_num
    return x_num_scaled
```

데이터 전처리(7) : 다중공선성 확인 및 PCA

히트맵으로 강한 상관관계를 갖는 변수들 확인



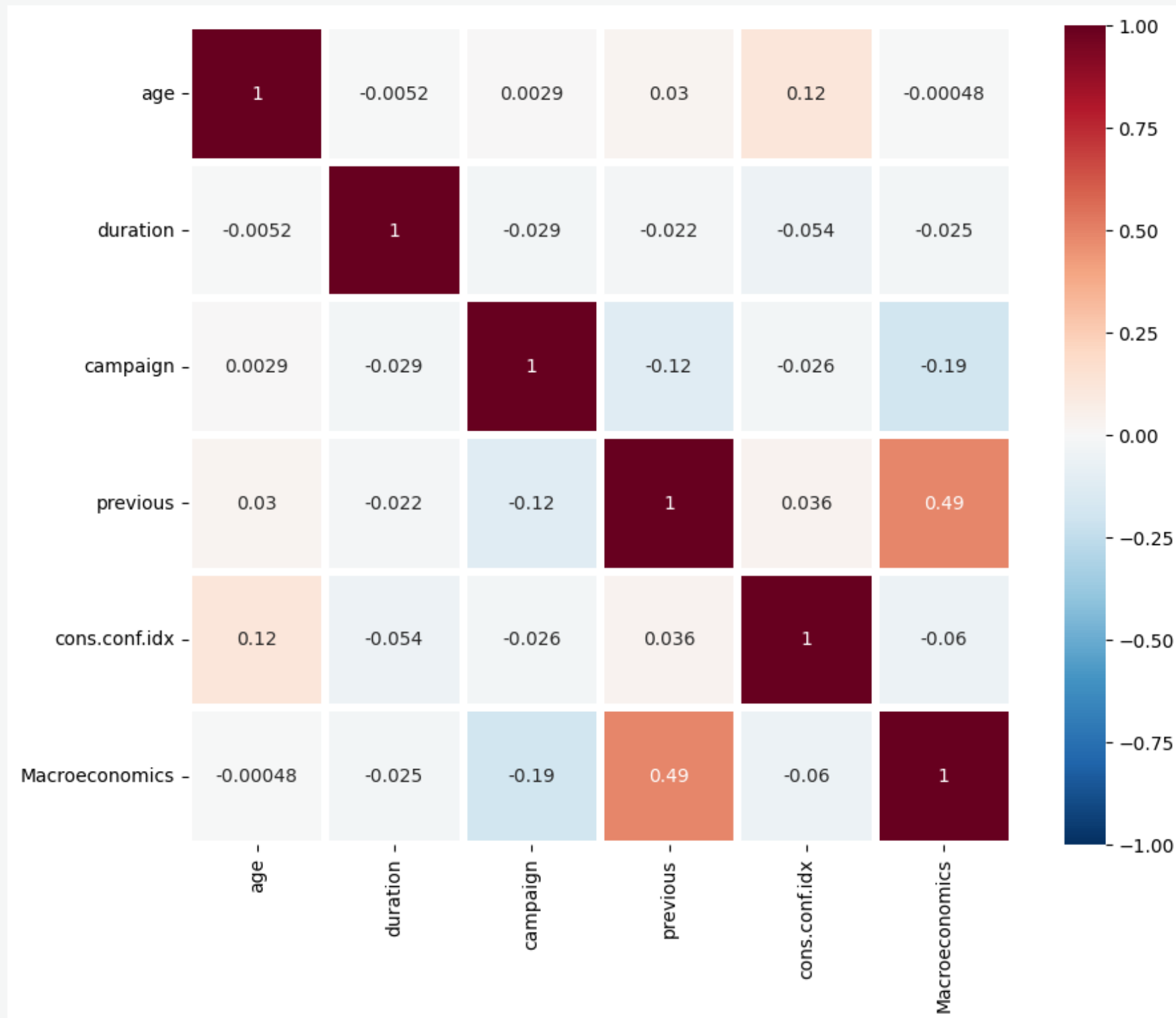
VIF를 이용한 다중공선성 검정

	features	VIF Factor
0	age	2.792743
1	duration	2.568874
2	campaign	1.636500
3	previous	1.704937
4	emp.var.rate	100.943788
5	cons.price.idx	11.905977
6	cons.conf.idx	3.076009
7	euribor3m	69.099913
8	nr.employed	44.484100

- VIF 값이 10이상이면 다중공선성의 문제가 있다고 판단할 수 있음
- 고용변동률, 소비자 가격지수, 리보금리, 취업 인원 수 총 4개의 변수들이 다중공선성이 존재하는 것 확인

데이터 전처리(7) : 다중공선성 확인 및 PCA

PCA 적용후 히트맵, VIF 확인



	features	VIF Factor
0	age	2.376512
1	duration	1.966096
2	campaign	1.455957
3	previous	1.632365
4	cons.conf.idx	2.417111
5	Macroeconomics	1.356126

데이터 전처리(7) : 다중공선성 확인 및 PCA

```
vif_high_columns = ['emp.var.rate', 'cons.price.idx', 'euribor3m', 'nr.employed']  
x_num_pca = hd.doPCA(x_num, vif_high_columns, ['Macroeconomics'], 1)
```

✓ 0.0s

VIF지수가 높은 4개의 피처를 1개의 피처로 PCA 변환했을 때 새롭게 나온 피처가 전체의 변동성을 [86.49035703]%만큼 설명해준다

- VIF지수 높았던 4개의 피처를 PCA기법을 통해 변환한 결과, 새롭게 생성된 피처는 전체 변동성의 86%를 설명해주는 것을 확인
- 다중공선성이 있던 피처들을 대표하는 주성분으로 효과적으로 축소되었음을 알 수 있음
- PCA 기법을 통한 차원 축소

데이터 전처리(8) : Data Split

```
#### PCA 안한거
# 학습 데이터와 테스트 데이터를 분할한다.
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_total, y_sm, train_size=0.8, random_state=1, stratify=y_sm)
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
# 학습 데이터와 테스트 데이터를 나눈 후 인덱스를 재정렬한다.
x_train.reset_index(drop=True, inplace=True)
x_test.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)
y_test.reset_index(drop=True, inplace=True)

#### PCA 한거
x_train_pca, x_test_pca, y_train_pca, y_test_pca = train_test_split(x_total_pca, y_sm, train_size=0.8, random_state=1, stratify=y_sm)
print(x_train_pca.shape, x_test_pca.shape, y_train_pca.shape, y_test_pca.shape)
# 학습 데이터와 테스트 데이터를 나눈 후 인덱스를 재정렬한다.
x_train_pca.reset_index(drop=True, inplace=True)
x_test_pca.reset_index(drop=True, inplace=True)
y_train_pca.reset_index(drop=True, inplace=True)
y_test_pca.reset_index(drop=True, inplace=True)

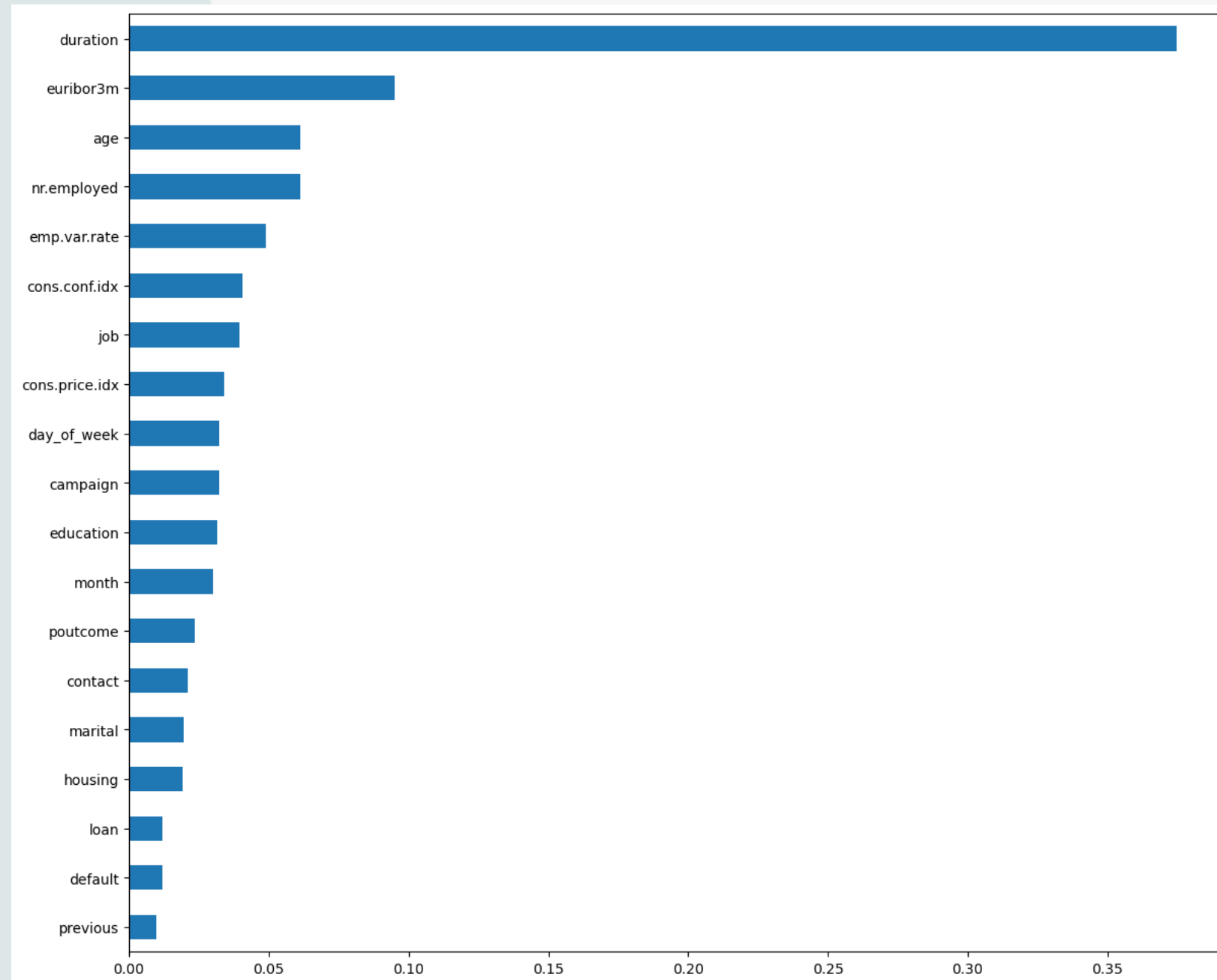
✓ 0.0s
(7424, 37) (1856, 37) (7424,) (1856,)
(7424, 34) (1856, 34) (7424,) (1856,)
```

- 분리한 결과 학습 데이터는 7424, 테스트 데이터는 1856인 데이터셋을 구성
- 학습 데이터(80%)와 테스트 데이터(20%)를 분리

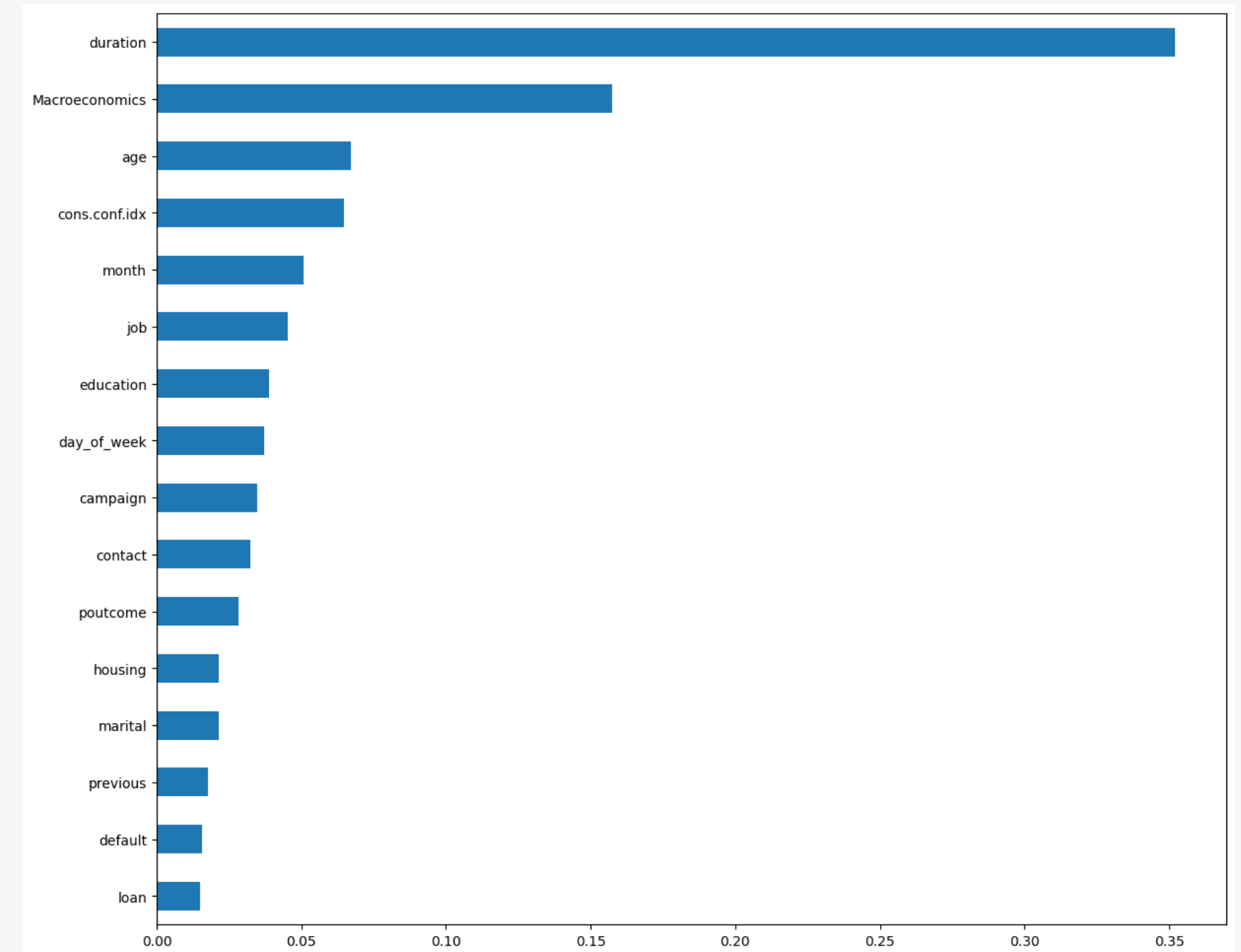
5. 모델 및 성능 평가

모델 및 성능 평가 : 피처 중요도 확인

PCA전

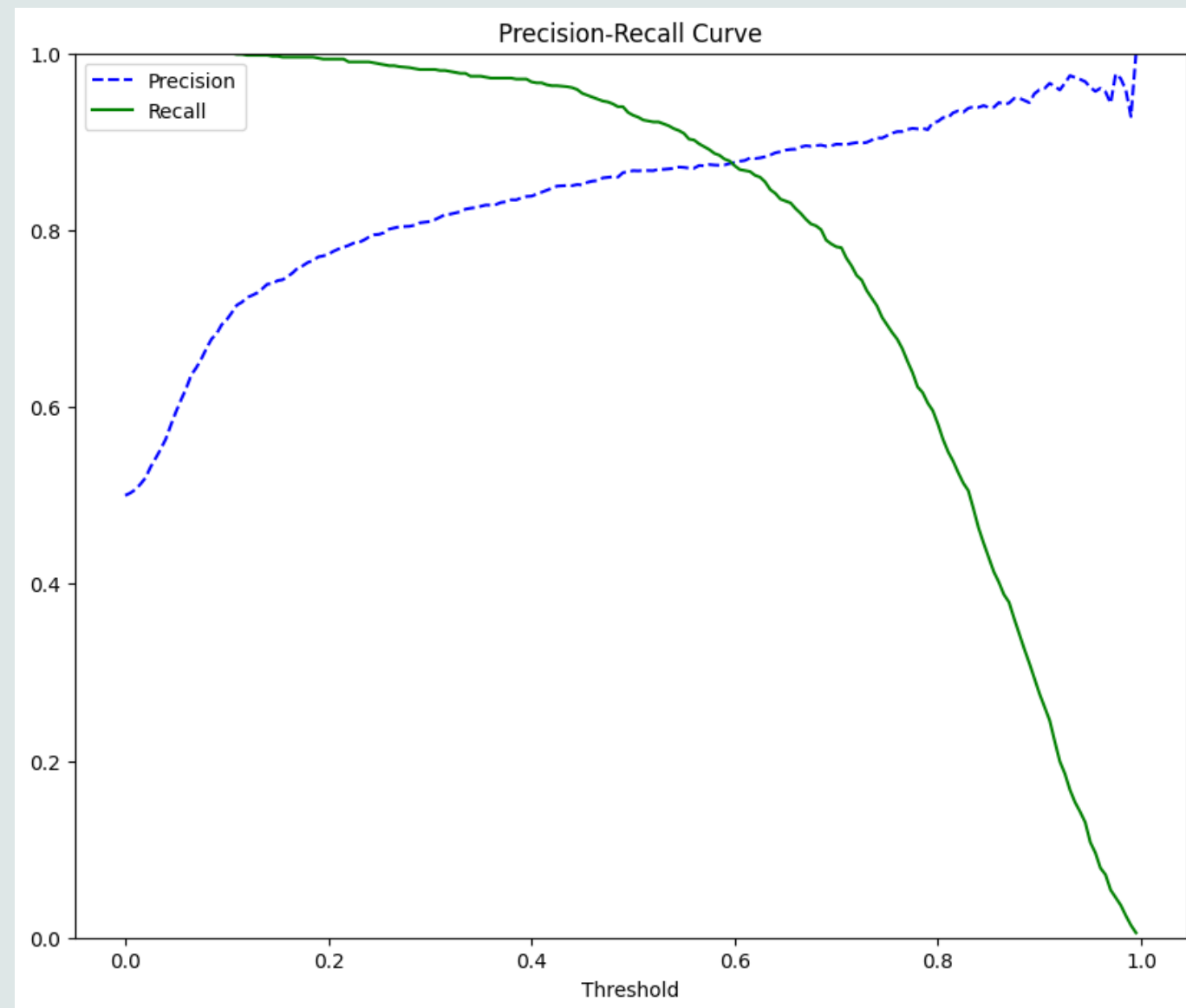


PCA후



데이터 모델링(Random forest)

* Undersampling, MinMax, PCA 적용



Fitting 5 folds for each of 24 candidates, totalling 120 fits

`{'max_depth': 40, 'n_estimators': 200}`

최적의 하이퍼 파라미터 : `{'max_depth': 40, 'n_estimators': 200}`

교차 검증 정확도 : [0.864 0.8828 0.8801 0.8788 0.8956], 평균 정확도 : 0.8803

교차 검증 정밀도 : [0.8309 0.8524 0.8518 0.848 0.8664], 평균 정밀도 : 0.8499

교차 검증 재현율 : [0.9137 0.9259 0.9206 0.9233 0.9353], 평균 재현율 : 0.9238

교차 검증 f1 스코어 : [0.8703 0.8876 0.8849 0.884 0.8995], 평균 f1 스코어 : 0.8853

교차 검증 roc_auc : [0.9229 0.9347 0.9386 0.9389 0.9459], 평균 roc_auc : 0.9362

최적의 Threshold 값 : 0.595

혼돈행렬 : [[812 116]

[113 815]]

정확도 : 0.8766

정밀도 : 0.8754

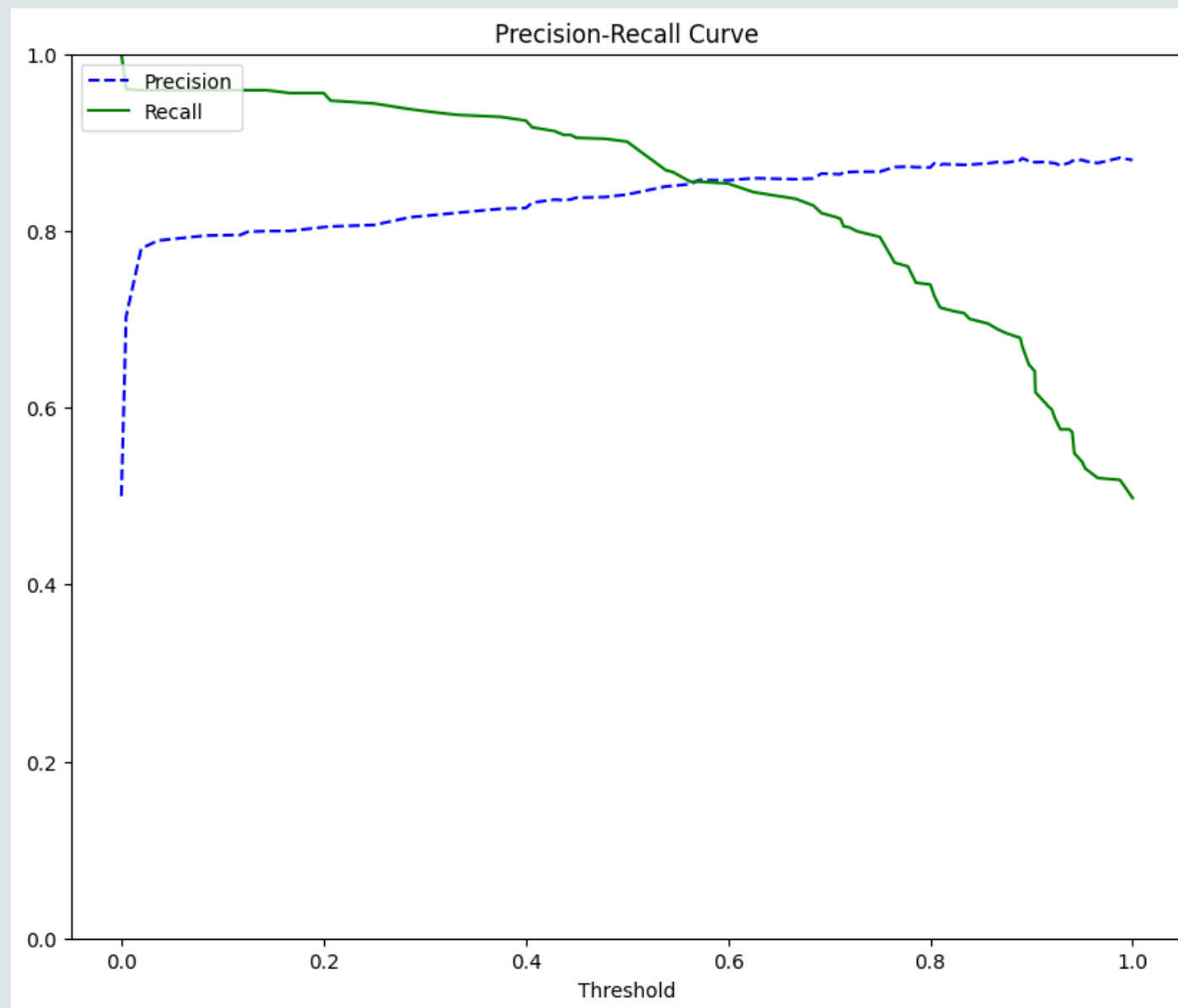
재현율 : 0.8782

roc_auc 스코어 : 0.8766

f1 스코어 : 0.8768

데이터 모델링(Decision tree)

* Undersampling, MinMax, PCA 적용

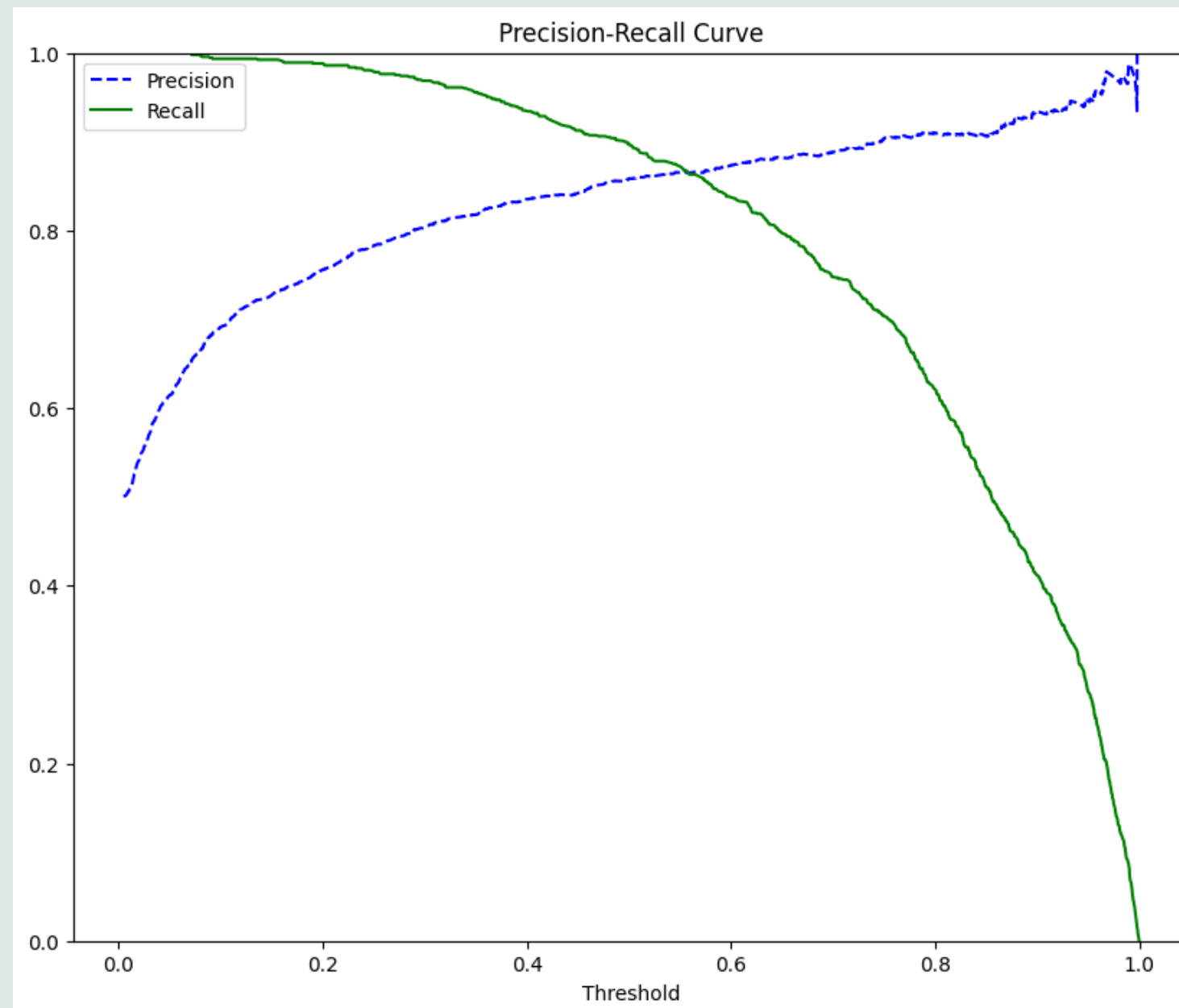


```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
{'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10}
최적의 하이퍼 파라미터 : {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10}
교차 검증 정확도 : [0.8343 0.8478 0.8606 0.8525 0.8605], 평균 정확도 : 0.8512
교차 검증 정밀도 : [0.8238 0.8431 0.8564 0.8493 0.8562], 평균 정밀도 : 0.8458
교차 검증 재현율 : [0.8504 0.8544 0.8668 0.8573 0.8666], 평균 재현율 : 0.8591
교차 검증 f1 스코어 : [0.8369 0.8487 0.8615 0.8533 0.8614], 평균 f1 스코어 : 0.8524
교차 검증 roc_auc : [0.8824 0.8879 0.9073 0.8987 0.9134], 평균 roc_auc : 0.8979
```

```
최적의 Threshold 값 : 0.5714285714285714
혼돈행렬 : [[796 132]
 [134 794]]
정확도 : 0.8567
정밀도 : 0.8575
재현율 : 0.8556
roc_auc 스코어 : 0.8567
f1 스코어 : 0.8565
```

데이터 모델링(Logistic regression)

* Undersampling, MinMax, PCA 적용

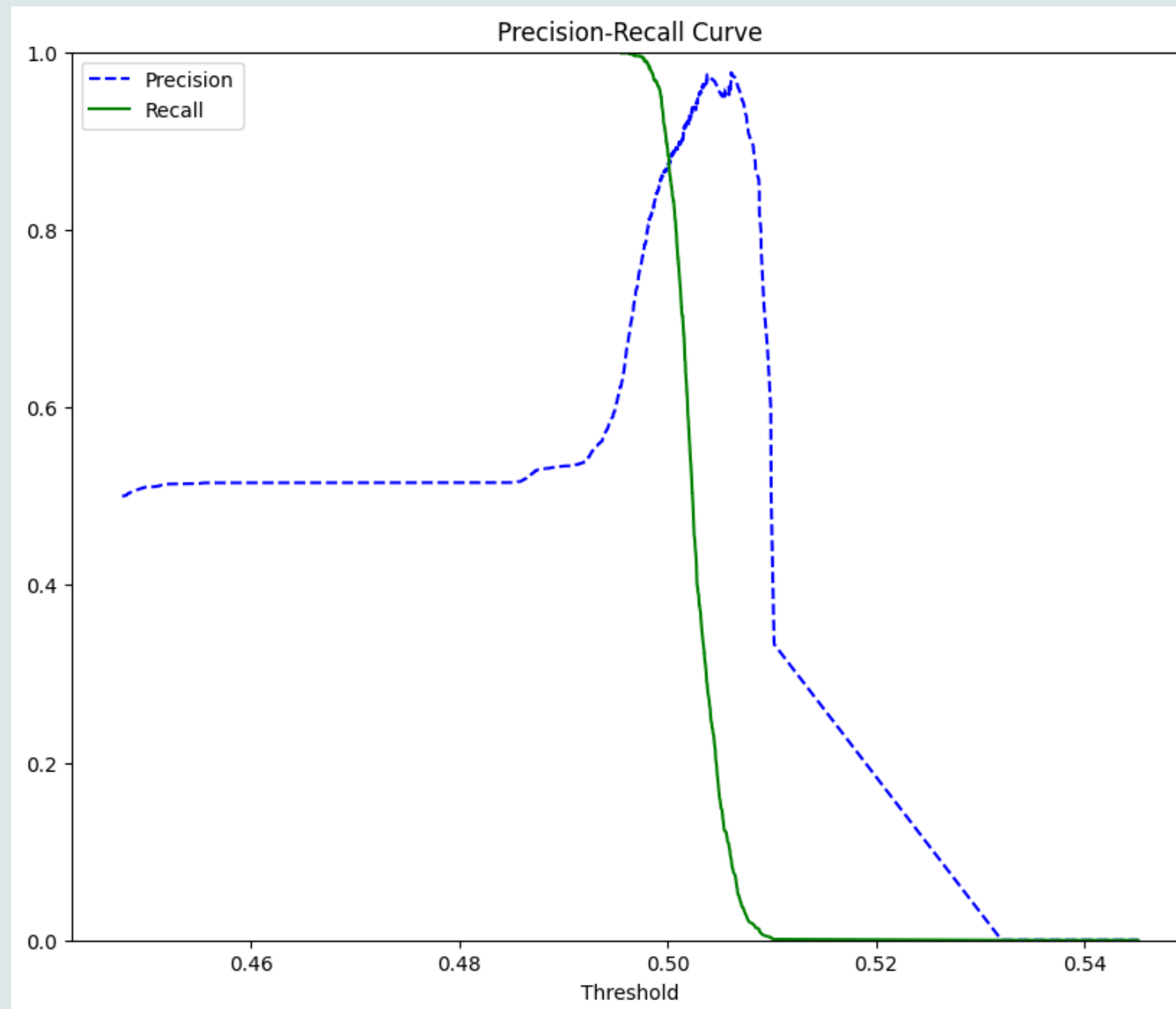


```
Fitting 5 folds for each of 168 candidates, totalling 840 fits
{'C': 0.4306717067640514, 'penalty': 'l2'}
최적의 하이퍼 파라미터 : {'C': 0.4306717067640514, 'penalty': 'l2'}
교차 검증 정확도 : [0.8471 0.8525 0.862 0.8559 0.874 ], 평균 정확도 : 0.8583
교차 검증 정밀도 : [0.8289 0.8436 0.8558 0.8369 0.8618], 평균 정밀도 : 0.8454
교차 검증 재현율 : [0.8747 0.8652 0.8708 0.8843 0.8908], 평균 재현율 : 0.8772
교차 검증 f1 스코어 : [0.8511 0.8543 0.8632 0.8599 0.8761], 평균 f1 스코어 : 0.8609
교차 검증 roc_auc : [0.9149 0.9187 0.9256 0.9301 0.9361], 평균 roc_auc : 0.9251
```

```
최적의 Threshold 값 : 0.5574200438688891
혼돈행렬 : [[803 125]
 [125 803]]
정확도 : 0.8653
정밀도 : 0.8653
재현율 : 0.8653
roc_auc 스코어 : 0.8653
f1 스코어 : 0.8653
```


데이터 모델링(ADABOOST)

* Undersampling, MinMax, PCA 적용

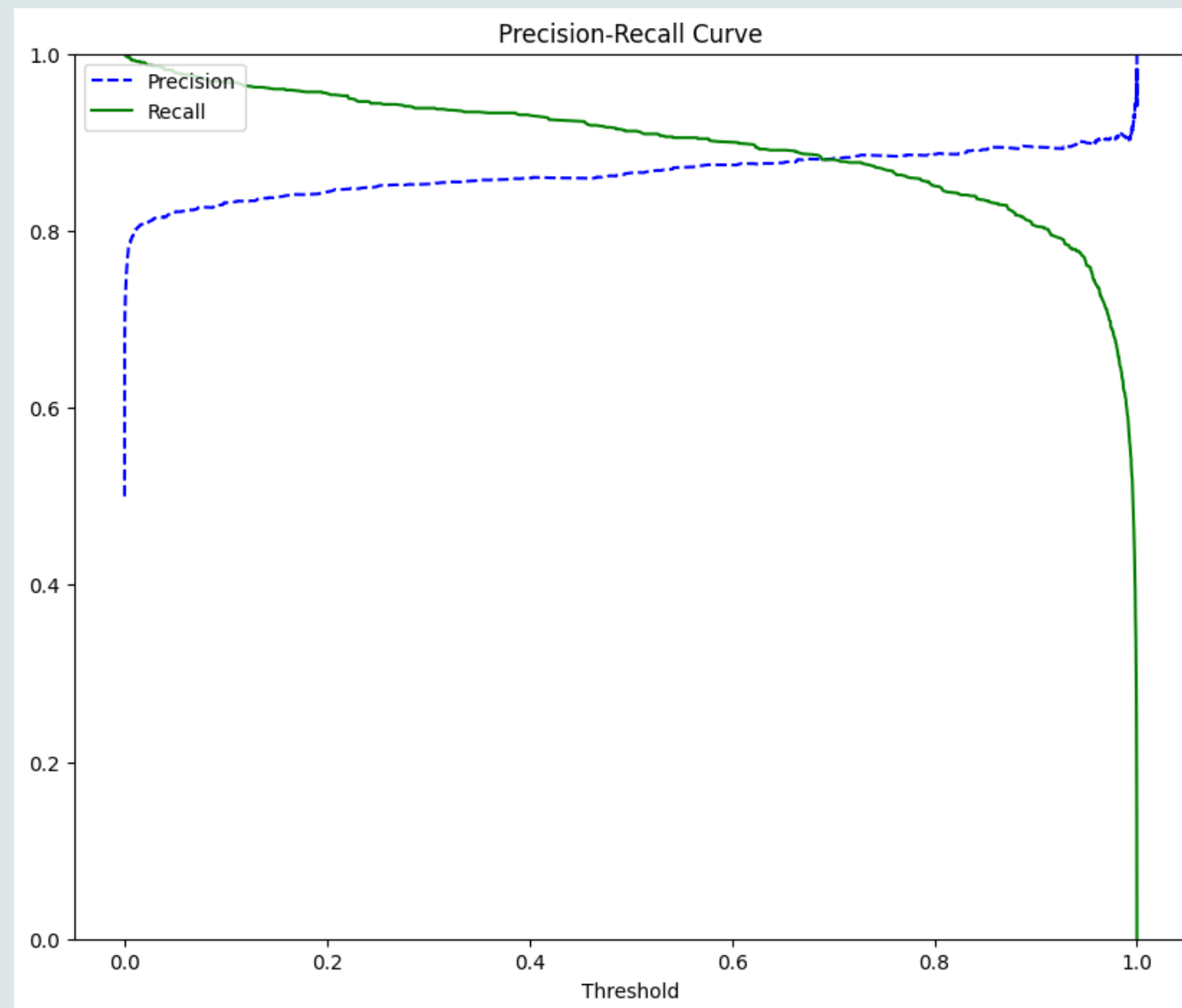


```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
{'learning_rate': 1, 'n_estimators': 200}
최적의 하이퍼 파라미터 : {'learning_rate': 1, 'n_estimators': 200}
교차 검증 정확도 : [0.8539 0.8593 0.868 0.8828 0.8908], 평균 정확도 : 0.871
교차 검증 정밀도 : [0.8422 0.8587 0.8681 0.8719 0.8867], 평균 정밀도 : 0.8655
교차 검증 재현율 : [0.8706 0.8598 0.8681 0.8977 0.8962], 평균 재현율 : 0.8785
교차 검증 f1 스코어 : [0.8562 0.8593 0.8681 0.8846 0.8914], 평균 f1 스코어 : 0.8719
교차 검증 roc_auc : [0.9239 0.9289 0.9428 0.9452 0.9475], 평균 roc_auc : 0.9376
```

```
최적의 Threshold 값 : 0.5001165599393947
혼돈행렬 : [[811 117]
 [117 811]]
정확도 : 0.8739
정밀도 : 0.8739
재현율 : 0.8739
roc_auc 스코어 : 0.8739
f1 스코어 : 0.8739
```

데이터 모델링(XGboost)

* Undersampling, MinMax, PCA 적용



```
Fitting 5 folds for each of 64 candidates, totalling 320 fits
{'learning_rate': 0.1, 'max_depth': 10, 'n_estimators': 1000}
최적의 하이퍼 파라미터 : {'learning_rate': 0.1, 'max_depth': 10, 'n_estimators': 1000}
교차 검증 정확도 : [0.8545 0.8727 0.8855 0.8781 0.8868], 평균 정확도 : 0.8755
교차 검증 정밀도 : [0.8312 0.8549 0.8697 0.8603 0.8708], 평균 정밀도 : 0.8574
교차 검증 재현율 : [0.8895 0.8976 0.9071 0.9031 0.9084], 평균 재현율 : 0.9011
교차 검증 f1 스코어 : [0.8594 0.8757 0.888 0.8812 0.8892], 평균 f1 스코어 : 0.8787
교차 검증 roc_auc : [0.9182 0.9307 0.9404 0.9322 0.9432], 평균 roc_auc : 0.9329
```

```
최적의 Threshold 값 : 0.68915594
혼돈행렬 : [[817 111]
 [111 817]]
정확도 : 0.8804
정밀도 : 0.8804
재현율 : 0.8804
roc_auc 스코어 : 0.8804
f1 스코어 : 0.8804
```

데이터 모델링 : 예측성능 종합 결과 1

* PCA전에는 MinMax를 적용한 XGBoost의 precision지표가 가장 높게 나옴

		PCA 전				
		Accuracy	Precision	Recall	ROC-AUC	F-1 score
Standard	Random Forest	0.8722	0.8763	0.8782	0.8772	0.8773
	의사결정나무	0.8448	0.8478	0.8405	0.8448	0.8442
	로지스틱 회귀	0.8685	0.8685	0.8685	0.8685	0.8685
	ADABOost	0.8761	0.8761	0.8761	0.8761	0.8761
	XGBoost	0.8825	0.8825	0.8825	0.8825	0.8825
Minmax	Random Forest	0.8836	0.8836	0.8836	0.8836	0.8836
	의사결정나무	0.8481	0.8519	0.8427	0.8481	0.8472
	로지스틱 회귀	0.875	0.875	0.875	0.875	0.875
	ADABOost	0.8804	0.8804	0.8804	0.8804	0.8804
	XGBoost	0.8879	0.8879	0.8879	0.8879	0.8879

데이터 모델링 : 예측성능 종합 결과 2

결론 : 종합적으로 PCA, Standard를 적용한 XGBoost의 Recall과 Precision지표를 채택하여 모델 결정 및 평가

		PCA 후				
		Accuracy	Precision	Recall	ROC-AUC	F-1 score
Standard	Random Forest	0.8788	0.8792	0.8782	0.8788	0.8787
	의사결정나무	0.8319	0.8312	0.833	0.8319	0.8321
	로지스틱 회귀	0.8599	0.8599	0.8599	0.8599	0.8599
	ADABOOST	0.8793	0.8793	0.8793	0.8793	0.8793
	XGBoost	0.8825	0.8825	0.8825	0.8825	0.8825
Minmax	Random Forest	0.8766	0.8754	0.8782	0.8766	0.8768
	의사결정나무	0.8567	0.8575	0.8556	0.8567	0.8565
	로지스틱 회귀	0.8653	0.8653	0.8653	0.8653	0.8653
	ADABOOST	0.8739	0.8739	0.8739	0.8739	0.8739
	XGBoost	0.8804	0.8804	0.8804	0.8804	0.8804

* PCA후에는 Standard를 적용한 XGBoost의 precision지표가 가장 높게 나옴

6. Insight 및 한계점

Insight 및 한계점

- 기여

- 프로파일 분석을 통한 개인 맞춤형 정기예금 가입 예측 가능
- 은행의 정기예금 가입자 예측 가능
- 향후 은행의 투자 사업, 대출 사업 의사결정에 도움을 줄 것으로 기대됨

- 한계

- 가상 데이터를 이용한 모델 구축
- 언더샘플링으로 인한 데이터의 정보 손실
- 해외 가상 데이터를 활용으로 인한 국내 경기지표 불일치

7. 참고문헌

참고 문헌

- 김고운, 금리민감도와 고객이탈의도에 관한 실증분석 : 이자율 유형과 성별의 조절효과, 2022
- 김경태·이지형, 딥 러닝과 Boosted Decision Tree를 활용한 고객 이탈 예측 모델, 2018
- 박지훈·이현상, 딥러닝기반의 금융회사 고객이탈 예측모형에 관한 연구 : 중소기업 금융에 대한 시사점 도출, 2020
- 나광택,이진영, 증권 금융 상품 거래 고객의 이탈 예측 및 원인 추론, 2015
- 정광수, 빅데이터를 활용한 은행권 고객 세분화 기법 연구, 2013
- 오은택, 설명가능한 정기예금 가입여부 예측을 위한 앙상블 학습 기법 분석, 2009
- 유민한, 은행고객 세분화를 통한 이탈고객 관리분석, 가계성예금을 중심으로, 2018
- 이세희, 이지형 RNN을 이용한 고객 이탈 예측 및 분석, 2014
- 김석제, 이차원 고객충성도 세그먼트 기반의 고객이탈예측 방법론, 2015

감사합니다! :)