[계속]정형데이터의 전처리(1)

결손값 제외

```
bank_df = bank_df.dropna(subset=['job','education'])
print(bank_df.shape)
```

```
(6935, 17)
```

결손값의 대상 행이나 열이 여러 개 있지만 우선, job이나 education의 경우 결손값의 갯수가 각각 44, 273개 입니다. 따라서 결손값이 있는 행을 데이터세트에서 제외해도 크게 문제되지 않을 것입니다.

```
bank_df = bank_df.dropna(thresh=2400, axis=1)
bank_df.shape
```

```
(6935, 16)
```

또한, contact와 poutcome을 보면, contact은 2038개라서 조금 애매하고 poutcome은 5900개라 결손값이 너무 많습니다. 데이터세트에서 제외하는 것이 차라리 나을 것 같습니다.

결손값 보완

```
bank_df = bank_df.fillna({'contact':'unknown'})
bank_df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	no
1	36	technician	single	secondary	no	265	yes	yes	unknown	5	may	348	1	-1	0	no
2	25	blue-collar	married	secondary	no	-7	yes	no	unknown	5	may	365	1	-1	0	no
3	53	technician	married	secondary	no	-3	no	no	unknown	5	may	1666	1	-1	0	no
4	24	technician	single	secondary	no	-103	yes	yes	unknown	5	may	145	1	-1	0	no

contact의 경우 데이터세트에 남겨 놓되 NaN을 'unknown'으로 변경하도록 합니다.

특이값 제외

```
bank_df = bank_df[bank_df['age'] >= 18]
```

```
bank_df = bank_df[bank_df['age'] < 100]
bank_df.shape
```

```
(6933, 16)
```

나이의 경우 최소 2, 최대 157이 들어 있습니다. 고객과 직접 접촉하여 얻은 자료인데 2살, 157살은 너무 비현실적이죠? 그래서 이러한 자료를 현실화하였습니다.

문자열을 수치로 변환

문자열로 된 데이터를 수치로 변환해야 분석하기가 좋습니다. 그런데, default, housing, loan, deposit은 yes와 no라는 값 두 개를 가지고 있으며 job, marital, education, contact, month는 다수의 값을 가지고 있습니다. 그래서 우선 값 두 개를 가지는 데이터부터 변환하겠습니다. yes를 1로, no는 0으로 바꿉니다.

```
bank_df = bank_df.replace('yes', '1')
bank_df = bank_df.replace('no', '0')
bank_df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	y
0	58	management	married	tertiary	0	2143	1	0	unknown	5	may	261	1	-1	0	0
1	36	technician	single	secondary	0	265	1	1	unknown	5	may	348	1	-1	0	0
2	25	blue-collar	married	secondary	0	-7	1	0	unknown	5	may	365	1	-1	0	0
3	53	technician	married	secondary	0	-3	0	0	unknown	5	may	1666	1	-1	0	0
4	24	technician	single	secondary	0	-103	1	1	unknown	5	may	145	1	-1	0	0

다음으로 다수 값을 가지는 자료를 변환하도록 하죠. 다수값의 경우 원핫인코딩을 사용하여 더미 변수로 만드는 방법이 좋습니다.

```
bank_df_job = pd.get_dummies(bank_df['job']) # one-hot encoding
bank_df_job.head()
```

	admin.	blue-collar	entrepreneur	housemaid	management	retired	self-employed	services	student	technician	unemployed
0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	1	0

get_dummies를 사용하여 job을 더미 변수로 만들었습니다.

```
bank_df_marital = pd.get_dummies(bank_df['marital'])
bank_df_education = pd.get_dummies(bank_df['education'])
bank_df_contact = pd.get_dummies(bank_df['contact'])
bank_df_month = pd.get_dummies(bank_df['month'])
print(bank_df_marital.head())
print(bank_df_education.head())
print(bank_df_contact.head())
bank_df_month.head()
```

marital, education, contact, month도 더미변수화합니다.

	divorced	married	single
0	0	1	0
1	0	0	1
2	0	1	0
3	0	1	0
4	0	0	1

	primary	secondary	tertiary
0	0	0	1
1	0	1	0
2	0	1	0
3	0	1	0
4	0	1	0

	cellular	telephone	unknown
0	0	0	1
1	0	0	1
2	0	0	1
3	0	0	1
4	0	0	1

	apr	aug	dec	feb	jan	jul	jun	mar	may	nov	oct	sep
0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0

분석 데이터 세트 만들기

이제 정리하도록 하죠. 분석할 데이터 세트를 구성하기 위해 수치로된 데이터 항목을 취합합니다.

```
# 수치자료만 추출
tmp1 = bank_df[['age', 'default', 'balance', 'housing',
                'loan', 'day', 'duration', 'campaign',
                'pdays', 'previous', 'y']]
tmp1.head()
```

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	y
0	58	0	2143	1	0	5	261	1	-1	0	0
1	36	0	265	1	1	5	348	1	-1	0	0
2	25	0	-7	1	0	5	365	1	-1	0	0
3	53	0	-3	0	0	5	1666	1	-1	0	0
4	24	0	-103	1	1	5	145	1	-1	0	0

수치자료만의 tmp1에 더미변수 자료들을 결합하겠습니다. concat 함수를 사용하도록 합니다.

```
# 더미변수와 결합
tmp2 = pd.concat([tmp1, bank_df_marital], axis=1)
tmp3 = pd.concat([tmp2, bank_df_education], axis=1)
tmp4 = pd.concat([tmp3, bank_df_contact], axis=1)
bank_df_new = pd.concat([tmp4, bank_df_month], axis=1)
bank_df_new.head()
```

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...	dec	feb	jan	jul	jun	mar	may	nov	oct	sep
0	58	0	2143	1	0	5	261	1	-1	0	...	0	0	0	0	0	0	1	0	0	0
1	36	0	265	1	1	5	348	1	-1	0	...	0	0	0	0	0	0	1	0	0	0
2	25	0	-7	1	0	5	365	1	-1	0	...	0	0	0	0	0	0	1	0	0	0
3	53	0	-3	0	0	5	1666	1	-1	0	...	0	0	0	0	0	0	1	0	0	0
4	24	0	-103	1	1	5	145	1	-1	0	...	0	0	0	0	0	0	1	0	0	0

5 rows × 32 columns

이렇게 해서 만들어진 자료를 엑셀에 저장합니다.

```
# csv 파일로 보관
bank_df_new.to_csv('bank-prep.csv', index=False)
```

불균형 데이터의 균형화

자료의 한 쪽값이 너무 기울면 분석에 많은 지장을 초래합니다. 그래서 가급적이면 자료의 균형을 맞추는 시도를 하게 됩니다. 은행이 수차례에 걸친 캠페인을 하여 예금 가입을 했는지를 나타내는 y 변수가

대상입니다. 그런데 가입한 사람이 안한 사람보다 훨씬 적습니다. 상식적으로 생각해도 예금을 권유한다고 덥석하는 사람이 드물겠죠? 이러한 불균형 자료를 같은 비율로 맞추는 작업을 해보겠습니다.
여기서 잠깐! 바로 위에서 작업하여 변경된 파일을 쓰지 말고, 자료 파일을 조금 더 보완하도록 하겠습니다.
자료 원본인 bank.csv를 다시 불러옵니다. 다음 작업을 수행해 주세요:

```
# job, educationd의 결손값 제거
bank_df = bank_df.dropna(subset=['job', 'education'])
bank_df.shape
```

```
(6935, 17)
```

같은 작업을 왜 다시 하냐구요? 조금만 참고 따라해보세요.

```
# 필드(컬럼)의 결손값이 2400 이상이면 변수를 제거
bank_df = bank_df.dropna(thresh=2400, axis=1)
bank_df.shape
```

```
(6935, 16)
```

```
# contact의 결손값을 unknown으로 변경
bank_df = bank_df.fillna({'contact':'unknown'})
bank_df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	no
1	36	technician	single	secondary	no	265	yes	yes	unknown	5	may	348	1	-1	0	no
2	25	blue-collar	married	secondary	no	-7	yes	no	unknown	5	may	365	1	-1	0	no
3	53	technician	married	secondary	no	-3	no	no	unknown	5	may	1666	1	-1	0	no
4	24	technician	single	secondary	no	-103	yes	yes	unknown	5	may	145	1	-1	0	no

```
# 비현실적인 나이값 배제
bank_df = bank_df[bank_df['age'] >= 18]
bank_df = bank_df[bank_df['age'] < 100]
bank_df.shape
```

```
(6933, 16)
```

job을 2그룹으로 나누고 job2라는 컬럼을 만들겠습니다.

```
# job의 실제 일하는 사람을 worker, 직장이 없는 사람을 non-worker로 변경
bank_df.loc[(bank_df['job'] == 'management') |
            (bank_df['job'] == 'technician') |
            (bank_df['job'] == 'blue-collar') |
            (bank_df['job'] == 'admin.') |
            (bank_df['job'] == 'services') |
            (bank_df['job'] == 'self-employed') |
            (bank_df['job'] == 'entrepreneur') |
            (bank_df['job'] == 'housemaid'), 'job2'] = 'worker'
```

```
bank_df.loc[(bank_df['job'] == 'retired') |
            (bank_df['job'] == 'unemployed') |
            (bank_df['job'] == 'student'), 'job2'] = 'non-worker'
```

```
bank_df.head(10)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	y	job2
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	no	worker
1	36	technician	single	secondary	no	265	yes	yes	unknown	5	may	348	1	-1	0	no	worker
2	25	blue-collar	married	secondary	no	-7	yes	no	unknown	5	may	365	1	-1	0	no	worker
3	53	technician	married	secondary	no	-3	no	no	unknown	5	may	1666	1	-1	0	no	worker
4	24	technician	single	secondary	no	-103	yes	yes	unknown	5	may	145	1	-1	0	no	worker
5	60	retired	married	tertiary	no	100	no	no	unknown	5	may	528	1	-1	0	no	non-worker
6	55	technician	married	secondary	no	1205	yes	no	unknown	5	may	158	2	-1	0	no	worker
7	54	management	married	secondary	no	282	yes	yes	unknown	5	may	154	1	-1	0	no	worker
8	55	services	divorced	secondary	no	91	no	no	unknown	5	may	349	1	-1	0	no	worker
9	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	yes	worker

```
# 마지막으로 접촉한 달을 분기로 그룹화
bank_df.loc[(bank_df['month'] == 'jan') |
            (bank_df['month'] == 'feb') |
            (bank_df['month'] == 'mar'), 'month2'] = '1Q'
```

```
bank_df.loc[(bank_df['month'] == 'apr') |
            (bank_df['month'] == 'may') |
            (bank_df['month'] == 'jun'), 'month2'] = '2Q'
```

```
bank_df.loc[(bank_df['month'] == 'jul') |
            (bank_df['month'] == 'aug') |
            (bank_df['month'] == 'sep'), 'month2'] = '3Q'
```

```
bank_df.loc[(bank_df['month'] == 'oct') |
            (bank_df['month'] == 'nov') |
            (bank_df['month'] == 'des'), 'month2'] = '4Q'
```

```
bank_df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	y	job2	month2
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	no	worker	2Q
1	36	technician	single	secondary	no	265	yes	yes	unknown	5	may	348	1	-1	0	no	worker	2Q
2	25	blue-collar	married	secondary	no	-7	yes	no	unknown	5	may	365	1	-1	0	no	worker	2Q
3	53	technician	married	secondary	no	-3	no	no	unknown	5	may	1666	1	-1	0	no	worker	2Q
4	24	technician	single	secondary	no	-103	yes	yes	unknown	5	may	145	1	-1	0	no	worker	2Q

```
# 마지막 접촉일이 10이하: early, 20이하: middle, 20이상: late
bank_df.loc[bank_df['day'] <= 10, 'day2'] = 'early'
bank_df.loc[(bank_df['day'] > 10) & (bank_df['day'] <= 20), 'day2'] = 'middle'
bank_df.loc[bank_df['day'] > 20, 'day2'] = 'late'
```

```
# 접촉한 시간이 300 미만: short, 이상: long
bank_df.loc[bank_df['duration'] < 300, 'duration2'] = 'short'
bank_df.loc[bank_df['duration'] >= 300, 'duration2'] = 'long'
```

```
# 이번 캠페인 이전에 접촉한 횟수 previous < 1 : zero, previous >= 1 : one-more
bank_df.loc[bank_df['previous'] < 1, 'previous2'] = 'zero'
bank_df.loc[bank_df['previous'] >= 1, 'previous2'] = 'one-more'
```

```
# 지난 캠페인으로 접촉한 후 경과시간이 0미만: less, 0이상: more
bank_df.loc[bank_df['pdays'] < 0, 'pdays2'] = 'less'
bank_df.loc[bank_df['pdays'] >= 0, 'pdays2'] = 'more'
```

이제 다시 문자열을 숫자로 변환하겠습니다.

```
bank_df = bank_df.replace('yes', 1)
bank_df = bank_df.replace('no', 0)
```

직업, 결혼여부, 교육수준, 접촉형태, 접촉한 마지막 달 등을 더미로 변환합니다.

```
# 아래 5변수를 더미로 변환
bank_df_job = pd.get_dummies(bank_df['job'])
bank_df_marital = pd.get_dummies(bank_df['marital'])
bank_df_education = pd.get_dummies(bank_df['education'])
bank_df_contact = pd.get_dummies(bank_df['contact'])
bank_df_month = pd.get_dummies(bank_df['month'])
```

```
tmp1 = bank_df[['age', 'default', 'balance', 'housing', 'loan', 'day',
                'duration', 'campaign', 'pdays', 'previous', 'y']]
tmp2 = pd.concat([tmp1, bank_df_marital], axis=1)
tmp3 = pd.concat([tmp2, bank_df_education], axis=1)
tmp4 = pd.concat([tmp3, bank_df_contact], axis=1)
bank_df_new = pd.concat([tmp4, bank_df_month], axis=1)
```

```
# 추가된 컬럼을 더미로 변환
bank_df_job2 = pd.get_dummies(bank_df['job2'])
bank_df_month2 = pd.get_dummies(bank_df['month2'])
bank_df_day2 = pd.get_dummies(bank_df['day2'])
bank_df_duration2 = pd.get_dummies(bank_df['duration2'])
bank_df_previous2 = pd.get_dummies(bank_df['previous2'])
bank_df_pdays2 = pd.get_dummies(bank_df['pdays2'])
```

```
tmp5 = pd.concat([bank_df_new, bank_df_job2], axis=1)
tmp6 = pd.concat([tmp5, bank_df_month2], axis=1)
tmp7 = pd.concat([tmp6, bank_df_day2], axis=1)
tmp8 = pd.concat([tmp7, bank_df_duration2], axis=1)
tmp9 = pd.concat([tmp8, bank_df_previous2], axis=1)
bank_df_new2 = pd.concat([tmp9, bank_df_pdays2], axis=1)
```

```
# 결과 저장
bank_df_new2.to_csv('bank-prep2.csv', index=False)
```

```
# 불균형 데이터 세트에 대한 언더 샘플링
import numpy as np
from imblearn.under_sampling import RandomUnderSampler
```

```
X = np.array(bank_df_new2.drop('y', axis=1))
Y = np.array(bank_df_new2['y'])
print(np.sum(Y == 1), np.sum(Y == 0))
sampler = RandomUnderSampler(random_state=42)
X, Y = sampler.fit_resample(X, Y)
print(np.sum(Y == 1), np.sum(Y == 0))
```

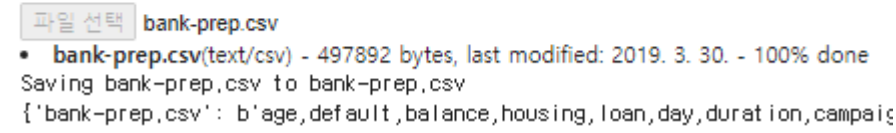
```
820 6113
820 820
```

정형 데이터의 전처리(2)

bank 데이터 세트는 나이, 시간, 날짜로 표현된 기간 등 모든 항목의 스케일이 달라 표준화가 필요합니다. 즉 정규화하여 평균이 0, 표준편차가 1인 자료로 변경할 필요가 있습니다.

bank-prep.csv를 불러옵니다.

```
from google.colab import files
files.upload()
```




```
import pandas as pd
bank_df = pd.read_csv('bank-prep.csv', sep=',')
```

스케일링을 위해 MinMaxScaler를 가져옵니다.

```
# MinMaxScaler의 로드
from sklearn.preprocessing import MinMaxScaler

# 데이터 세트에서 목적변수를 제외
bank_df = bank_df.drop('y', axis=1)
```

데이터 세트에 스케일러를 피트합니다.

```
# 범위변환의 인스턴스를 생성
mc = MinMaxScaler()
mc.fit(bank_df)
print(mc.n_samples_seen_, mc.data_min_, mc.data_max_, mc.feature_range)
```

```
6933 [ 1,900e+01  0,000e+00 -3,313e+03  0,000e+00  0,000e+00  1,000e+00
  0,000e+00  1,000e+00 -1,000e+00  0,000e+00  0,000e+00  0,000e+00
  0,000e+00  0,000e+00  0,000e+00  0,000e+00  0,000e+00  0,000e+00
  0,000e+00  0,000e+00  0,000e+00  0,000e+00  0,000e+00  0,000e+00
  0,000e+00  0,000e+00  0,000e+00  0,000e+00  0,000e+00  0,000e+00
  0,000e+00] [9,0000e+01 1,0000e+00 8,1204e+04 1,0000e+00 1,0000e+00 3,1000e+01
 3,3660e+03 4,4000e+01 8,5000e+02 4,0000e+01 1,0000e+00 1,0000e+00
 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00
 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00
 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00 1,0000e+00
 1,0000e+00] (0, 1)
```

피트된 스케일러를 사용하여 데이터 세트를 정규화합니다.

```
# bank_df의 각항목을 정규화
bank_df_mc = pd.DataFrame(mc.transform(bank_df), columns=bank_df.columns)
bank_df_mc.head()
```

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...	dec	feb	jan	jul	jun	mar	may	nov	oct	sep
0	0.549296	0.0	0.064555	1.0	0.0	0.133333	0.077540	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
1	0.239437	0.0	0.042335	1.0	1.0	0.133333	0.103387	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
2	0.084507	0.0	0.039116	1.0	0.0	0.133333	0.108437	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
3	0.478873	0.0	0.039164	0.0	0.0	0.133333	0.494949	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
4	0.070423	0.0	0.037981	1.0	1.0	0.133333	0.043078	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

5 rows × 31 columns

age를 비롯한 balance, day, duration 등이 0과 1사이의 범위에 속한 값으로 변경된 것을 확인할 수 있습니다.

```
bank_df_mc.mean()
```

```
age          0.305136
default      0.018462
balance      0.055470
housing      0.565412
loan         0.165008
day          0.485648
duration     0.078067
campaign     0.039763
pdays        0.048986
```

그러나 변환된 데이터 세트의 평균이 아직 0이 된 것은 아닙니다. 또다른 스케일러를 사용해보겠습니다.

```
# StandardScaler의 로드
from sklearn.preprocessing import StandardScaler

# Z변환 인스턴스 생성
sc = StandardScaler()
sc.fit(bank_df)

# bank_df의 각항목을 정규화
bank_df_sc = pd.DataFrame(sc.transform(bank_df), columns=bank_df.columns)
bank_df_sc.head()
```

	age	default	balance	housing	loan	day	duration	campaign	pdays	previous	...
0	1.647908	-0.137148	0.250618	0.876711	-0.444540	-1.275382	-0.006613	-0.574193	-0.418664	-0.310149	...
1	-0.443424	-0.137148	-0.362335	0.876711	2.249514	-1.275382	0.317659	-0.574193	-0.418664	-0.310149	...
2	-1.489090	-0.137148	-0.451112	0.876711	-0.444540	-1.275382	0.381022	-0.574193	-0.418664	-0.310149	...
3	1.172605	-0.137148	-0.449807	-1.140627	-0.444540	-1.275382	5.230180	-0.574193	-0.418664	-0.310149	...
4	-1.584150	-0.137148	-0.482445	0.876711	2.249514	-1.275382	-0.438974	-0.574193	-0.418664	-0.310149	...

5 rows × 31 columns

StandardScaler를 써보니 -1 ~ 1 사이의 범위로 데이터를 스케일링하는 것을 알 수 있습니다.

```
bank_df_sc.mean()
```

```
age          -3,085501e-16
default      -2,955983e-15
balance       8,058845e-17
housing       1,263524e-14
loan         -6,665694e-15
day          -1,745907e-15
```

모든 항목의 평균은 0이 된 것을 알 수 있죠?

```
bank_df_sc.std()
```

```
age           1,000072
default       1,000072
balance       1,000072
housing       1,000072
loan          1,000072
day           1,000072
```

모든 항목의 표준편차는 1로 확인되고 있습니다.