데이터 전처리
&
연습문제

최석재
lingua@naver.com

데이터 전처리 예

기초 정보 확인

- bank.csv 파일을 읽고, 기초 정보를 확인하시오

In [1]:
```
# 데이터 읽기
import pandas as pd

path = 'C:/Users/nalang/pytest_basic/'
data = pd.read_csv(path+'bank.csv')
data.head()
```

Out[1]:

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

기초 정보 확인

In [2]:
```
# 기초 정보 확인
# 행과 열의 수 확인
data.shape
```

Out[2]:
```
(7234, 17)
```

In [3]:
```
# 컬럼명, non-null 행의 수, 데이터 타입
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7234 entries, 0 to 7233
Data columns (total 17 columns):
age          7234 non-null int64
job          7190 non-null object
marital      7234 non-null object
education    6961 non-null object
default      7234 non-null object
balance      7234 non-null int64
housing      7234 non-null object
loan         7234 non-null object
contact      5196 non-null object
day          7234 non-null int64
month        7234 non-null object
duration     7234 non-null int64
campaign     7234 non-null int64
pdays        7234 non-null int64
previous     7234 non-null int64
poutcome     1334 non-null object
y            7234 non-null object
dtypes: int64(7), object(10)
memory usage: 960.9+ KB
```

기초 통계 정보

In [4]:
```
# 기초 통계 정보
# 숫자형 데이터에 대해서만 출력된다
data.describe()
```

Out[4]:

	age	balance	day	duration	campaign	pdays	previous
count	7234.000000	7234.000000	7234.000000	7234.000000	7234.000000	7234.000000	7234.000000
mean	40.834808	1374.912911	15.623860	262.875311	2.713989	40.277716	0.565939
std	10.706442	3033.882933	8.307826	268.921065	2.983740	99.188008	1.825100
min	2.000000	-3313.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	74.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	453.500000	16.000000	183.000000	2.000000	-1.000000	0.000000
75%	48.000000	1470.750000	21.000000	321.750000	3.000000	-1.000000	0.000000
max	157.000000	81204.000000	31.000000	3366.000000	44.000000	850.000000	40.000000

결측치 제거

In [5]:
```
# data.info()의 정보를 보았을 때, 모든 컬럼이 7234 행이 아니다
# 결측치가 있는 것으로 예상되며
# 다음을 통해 다시 확인할 수 있다
data.isnull().values.any()
```

Out[5]: True

In [6]:
```
# 열마다 몇 개의 결측치가 있는지 확인해본다
data.isnull().sum()
```

Out[6]:

age	0
job	44
marital	0
education	273
default	0
balance	0
housing	0
loan	0
contact	2038
day	0
month	0
duration	0
campaign	0
pdays	0
previous	0
poutcome	5900
y	0

dtype: int64

결측치 제거

In [7]:
```
# 결측치가 있는 컬럼을 제거하는 기준을 잡아본다
# 전체 행의 수의 1/3에 해당하는 숫자를 찾아,
# 이 숫자를 넘는 컬럼은 불완전한 데이터로 보고, 아예 제거하기로 한다
print("전체 행의 수:", len(data))
print("1/3에 해당하는 행의 수", int(len(data)*0.3))
```

```
전체 행의 수: 7234
1/3에 해당하는 행의 수 2170
```

In [8]:
```
# 결측치 제외하기
# 위에서 job과 education은 결측치가 있는 행의 수가 적다
# 따라서, 이 두 컬럼에서는 결측치가 있는 행만을 제거하기로 한다
data = data.dropna(subset=['job', 'education'])
print(data.shape)
```

```
(6935, 17)
```

결측치 제거

In [9]:
```
# poutcome 컬럼은 기준 2170보다도 결측치 행이 많아 사용하기 어렵다
# 따라서 이 열은 아예 제거하기로 한다
# 먼저, 결측치를 2170행 이상으로 갖고 있는 컬럼을 자동으로 제거하는 방법은 다음과 같다
# (저장은 하지 않음)
data.dropna(thresh=2170, axis=1)
```

Out[9]:

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

In [10]:
```
# 여기서는 사용할 컬럼을 직접 선택하는 방식을 취하기로 한다
# 먼저, 컬럼의 이름을 출력해본다
print(data.columns)
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'y'],
      dtype='object')
```

In [11]:
```
# 위의 컬럼에서 'poutcome'만 제외하고 붙여넣는다
data = data[['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'y']]
data.head()
```

Out[11]:

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

결측치 제거

In [12]:
```
# 처리후 결과를 확인한다
data.isnull().sum()
```

Out[12]:
```
age            0
job            0
marital        0
education      0
default        0
balance        0
housing        0
loan           0
contact     1925
day            0
month          0
duration       0
campaign       0
pdays          0
previous       0
y              0
dtype: int64
```

결측치 치환

In [13]:
```
# 위에서 contact는 2170보다는 적으나,
# 해당 행을 제외하기에는 결측치 행의 수가 많다
# 이 컬럼의 결측치는 다른 값으로 치환하는 방법을 사용하기로 한다
# 여기서는 'unknown'을 치환값으로 사용하기로 한다
data = data.fillna({'contact':'unknown'})
data.head()
```

Out[13]:

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

결측치 치환

In [14]:
```
# 처리 후 결과
# 이제 결측치(NaN)는 없어졌다
data.isnull().sum()
```

Out[14]:
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
y            0
dtype: int64
```

이상치 제거

In [15]:
```
# age가 18세 이상, 100세 미만인 고객만 분석 대상으로 삼기로 한다
data = data[data.age >= 18]
data = data[data.age < 100]
data.shape
```

Out[15]:
```
(6933, 16)
```

문자열 값을 숫자형으로 치환 1

yes를 1, no를 0으로 치환

In [16]:
```
# 바이너리한 문자열은 0, 1로 치환한다
data = data.replace('yes', 1)
data = data.replace('no', 0)
data.head()
```

Out[16]:

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

문자열 값을 숫자형으로 치환 2

원-핫 인코딩

In [17]:
```
# 여러 종류로 된 문자열 값은 모든 종류를 0과 1로만 표현하는 원-핫 인코딩을 한다
# 앞에서 'unknown'으로 치환된 것도 대상으로 한다
data = pd.get_dummies(data, columns=['job', 'marital', 'education', 'contact', 'month'])
data.head()
```

Out[17]:

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

5 rows × 43 columns

결과 저장

In [18]:
```
# 현재 path에 자료를 저장하시오 (result.csv)
# index 열이 생기지 않도록 하시오
data.to_csv(path+'result.csv', index=False)
```

연습문제

준비운동

- mtcars 데이터셋의 qsec 컬럼을
- 최소최대(Min-Max)로 정규화한 후,
- 0.5보다 큰 값을 가지는 레코드 수를 구하시오

※ 정규화란 컬럼별로 단위가 달라 값의 범위가 다른 것을 해결하기 위해
모든 컬럼의 값을 같은 범위 내의 값으로 변환하는 것을 말한다

정규화에는 최소값을 0, 최대값을 1로 하는 민맥스 정규화(MinMaxScaer)와
평균이 0, 표준편차가 1이 되게 하는 표준화 정규화(StandardScaler)가 있다

일반적으로는 민맥스 정규화를 더 많이 사용한다

데이터 확인

In [1]:
```
import pandas as pd
path = 'C:/Users/nalang/pytest_basic/'
data = pd.read_csv(path+'/mtcars.csv')
data.head()
```

Out[1]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

컬럼 정규화

In [2]:
```
# 각 괄호를 하나 쓰면 Series, 둘 쓰면 DataFrame
qsec = data[['qsec']]
```

In [3]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(qsec)
qsec_scaled = scaler.transform(qsec)

print(type(qsec_scaled))
```

```
<class 'numpy.ndarray'>
```

해당 레코드 수 세기

In [4]:
```
qsec_scaled_above = qsec_scaled > 0.5
```

In [5]:
```
print(sum(qsec_scaled_above))  # True의 개수를 센다
```

```
[9]
```

연습문제 SET 1

```
In [9]: age80['crim'].mean()
Out[9]: 5.759386624999999
```

연습문제 1

- 다음은 Boston Housing 데이터셋이다.
- crim 항목의 상위에서 10번째 값 (즉, 상위 10번째 값 중에서 가장 작은 값)으로 상위 10개의 값을 변환하고
- age 80 이상인 값에 대하여 crim 평균을 구하시오

```
In [1]: import pandas as pd
path = 'C:/Users/nalang/pytest_basic/'
data=pd.read_csv(path+'boston_housing.csv')
data.head()
```

Out[1]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	pratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

```
In [12]: print("표준편차 전후의 차이:", total_bedrooms_std_pre-total_bedrooms_std_after)
표준편차 전후의 차이: 1.975147291645726
```

연습문제 2

- 데이터의 첫 번째 행부터 순서대로 80%까지의 데이터를 훈련 데이터로 추출 후
- ‘total_bedrooms’ 변수의 결측값(NA)을 ‘total_bedrooms’ 변수의 중앙값으로 대체하고
- 대체 전의 ‘total_bedrooms’ 변수 표준편차 값의 차이의 절댓값을 구하시오

In [1]:
```
import pandas as pd
path = 'C:/Users/nalang/pytest_basic/'
data=pd.read_csv(path+'california_housing.csv')
data.head()
```

Out[1]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_ho
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	

```
In [12]: outlier_high['charges'].sum()
Out[12]: 5852448.66161
```

연습문제 3

- 다음 데이터셋에서 charges 항목의 이상값을 구하시오
- 이상값은 평균에서 1.5 표준편차 이상인 값으로 한다

```
In [1]: import pandas as pd
path = 'C:/Users/nalang/pytest_basic/'
data=pd.read_csv(path+'insurance.csv')
data.head()
```

Out[1]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

연습문제 4

- 아래 E-Commerce Shipping Data의 train set을 참조하여 고객이 주문한 물품의 정시 도착 여부를 예측하시오
- ID와 예측치는 csv 파일로 저장하시오 (EC_result.csv)

데이터 확인

In [1]:
```
import pandas as pd
path = 'C:/Users/nalang/pytest_basic/'
test = pd.read_csv(path+"EC_X_test.csv")
X_train = pd.read_csv(path+"EC_X_train.csv")
y_train = pd.read_csv(path+"EC_y_train.csv")
```

In [2]:
```
X_train.head()
```

Out[2]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Pr
0	1	B	Ship	3	4	134	3	
1	2	F	Ship	4	3	173	3	
2	3	B	Ship	2	1	192	3	
3	4	F	Ship	6	5	284	4	
4	5	F	Flight	3	1	246	3	

종속변수 확인

In [3]:
```
y_train.head()
```

Out[3]:

	ID	Reached.on.Time_Y.N
0	1	0
1	2	1
2	3	1
3	4	1
4	5	1

In [4]:
```
y_train = y_train['Reached.on.Time_Y.N']
y_train
```

Out[4]:
```
0       0
1       1
2       1
3       1
4       1
       ..
8794    0
8795    1
8796    1
8797    0
8798    0
Name: Reached.on.Time_Y.N, Length: 8799, dtype: int64
```

변수 정보 확인

```
In [5]: X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8799 entries, 0 to 8798
Data columns (total 11 columns):
ID                     8799 non-null int64
Warehouse_block        8799 non-null object
Mode_of_Shipment       8799 non-null object
Customer_care_calls    8799 non-null int64
Customer_rating        8799 non-null int64
Cost_of_the_Product    8799 non-null int64
Prior_purchases        8799 non-null int64
Product_importance     8799 non-null object
Gender                 8799 non-null object
Discount_offered       8799 non-null int64
Weight_in_gms          8799 non-null int64
dtypes: int64(7), object(4)
memory usage: 756.3+ KB
```

숫자형과 문자형을 확인한다

숫자형과 문자형 분리

In [6]:
```
# 훈련데이터의 숫자형과 문자형을 분리한다.    ID 컬럼은 제외
X_train_num = X_train[['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product',
                       'Prior_purchases', 'Discount_offered', 'Weight_in_gms']]
X_train_cat = X_train[['Warehouse_block', 'Mode_of_Shipment', 'Product_importance', 'Gender']]
```

In [7]:
```
# 테스트데이터의 숫자형과 문자형을 분리한다. ID 컬럼은 제외
X_test_num = X_test[['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product',
                       'Prior_purchases', 'Discount_offered', 'Weight_in_gms']]
X_test_cat = X_test[['Warehouse_block', 'Mode_of_Shipment', 'Product_importance', 'Gender']]
```

문자형 원-핫 인코딩

In [8]:
```
# 문자형에 대해서만 원-핫 인코딩을 한다
X_train_cat = pd.get_dummies(X_train_cat)
X_test_cat = pd.get_dummies(X_test_cat)

# 원-핫 인코딩 결과는 데이터프레임
print(type(X_train_cat))
print(type(X_test_cat))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
```

숫자형 정규화

In [9]:
```
# 숫자형을 대상으로 정규화한다
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X_train_num)
```

Out[9]:
```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

In [10]:
```
X_train_scaled = scaler.transform(X_train_num)
X_test_scaled = scaler.transform(X_test_num)

# 정규화 결과는 넘파이 배열
print(type(X_train_scaled))
print(type(X_test_scaled))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

숫자형과 문자형 결합

In [11]:
```
# 원-핫 인코딩한 문자형과, 정규화한 숫자형을 합친다
X_train_final = pd.concat([pd.DataFrame(X_train_scaled), X_train_cat], axis=1)
X_test_final = pd.concat([pd.DataFrame(X_test_scaled), X_test_cat], axis=1)
```

모델 생성 및 성능 확인

In [12]:
```
# 모델 생성 및 성능 확인 1
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train_final, y_train)

model.score(X_train_final, y_train)
```

Out[12]: 0.6401863848164564

또는,

In [13]:
```
# 모델 생성 및 성능 확인 2
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train_final, y_train)

model.score(X_train_final, y_train)
```

Out[13]: 0.9859074894874418

훈련데이터 결측치 확인

In [13]:
```
# Null 값 check 1
X_train.isnull().values.any()
```

Out[13]:
```
False
```

In [14]:
```
# Null 값 check 2
X_train.isnull().sum()
```

Out[14]:
```
ID                     0
Warehouse_block        0
Mode_of_Shipment       0
Customer_care_calls    0
Customer_rating        0
Cost_of_the_Product    0
Prior_purchases        0
Product_importance     0
Gender                 0
Discount_offered       0
Weight_in_gms          0
dtype: int64
```

테스트 데이터 결측치 확인

In [15]:
```
# Null 값 check 1
X_test.isnull().values.any()
```

Out[15]:
```
False
```

In [16]:
```
# Null 값 check 2
X_test.isnull().sum()
```

Out[16]:
```
ID                     0
Warehouse_block        0
Mode_of_Shipment       0
Customer_care_calls    0
Customer_rating        0
Cost_of_the_Product    0
Prior_purchases        0
Product_importance     0
Gender                 0
Discount_offered       0
Weight_in_gms          0
dtype: int64
```

테스트 데이터 예측

In [17]:
```
# 테스트 데이터 예측
pred_test = model.predict(X_test_final)
```

결과 정리

In [18]:
```
# 결과 정리
# pd.concat() 에는 Series나 DataFrame만 들어갈 수 있다
# 현재 X_test['ID']는 Series이나, pred_test는 Numpy 배열이므로, DataFrame으로 변환해야 한다
# 둘 모두 DataFrame으로 변환한다
# 열 병합이므로 axis=1 을 해줘야 한다. 반드시!
result = pd.concat([pd.DataFrame(X_test['ID']), pd.DataFrame(pred_test, columns=['predicted'])], axis=1)
```

In [19]:
```
result.head()
```

Out[19]:

	ID	predicted
0	8800	1
1	8801	1
2	8802	0
3	8803	1
4	8804	1

저장

In [20]:
```
# 저장
result.to_csv(path+"EC_result.csv", index=False)
```

참고 – 컬럼명 변경하기

In [21]:
```
# 컬럼명 변경
# df.rename(columns={"A": "a", "B": "c"})
# 아래에서는 0 이 숫자이므로 따옴표를 쓰면 안된다
result = result.rename(columns={0:'predicted'})
```

연습문제 SET 2

연습문제 1

In [7]:
```
answer=data70['tax'].quantile(q=0.25)
print(answer)
```

```
264.0
```

- 다음은 Boston Housing 데이터셋이다
- 데이터를 처음부터 순서대로 70%를 추출하여
- 변수 중 'tax'의 사분위수 Q1값을 구하시오

In [1]:
```
import pandas as pd

path = 'C:/Users/nalang/pytest_basic/'
data=pd.read_csv(path+'boston_housing_new.csv')
data.head()
```

Out[1]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	pratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

연습문제 2

In [6]:
```
# 0번 행의 모든 컬럼에 대하여 mean2000 보다 큰 것을 구하고
# 그에 해당하는 것(True)의 개수를 센다
sum(data2000.iloc[0, :] > mean2000)
```

Out[6]: 7

- 다음은 국가별 국내 입국자 수 데이터이다
- 2000년도 행만 선택한 후, 그 해의 전체 입국자 평균보다 국내에 많이 입국한 국가의 수를 산출하시오

In [1]:
```
import pandas as pd

path = 'C:/Users/nalang/pytest_basic/'
data=pd.read_csv(path+'tour.csv', encoding='euc-kr')
data.head()
```

Out[1]:

	year	중국	일본	대만	홍콩	필리핀	인도네시아	태국	베트남	인도	...	그리스	불가리아	덴마크	아일랜드	유럽기타	오스트레일리아
0	1995	178359	1667203	130147	100407	163228	37723	73770	16720	35668	...	14591	3408	5409	1741	7216	27251
1	1996	199604	1526559	114729	77958	178045	46570	74162	16750	38585	...	14142	3447	5797	2066	9688	30694
2	1997	214244	1676434	104144	96650	187235	53204	62374	14102	42934	...	12255	3870	5479	2230	9079	34640
3	1998	210662	1954416	108880	229072	165272	36486	49973	11561	40606	...	8730	3315	5661	2071	8181	31028
4	1999	316639	2184121	110563	234087	198583	47019	63679	16121	43829	...	10878	2371	5607	2131	8829	33378

연습문제 3

In [7]:
```
# 인덱스 정보를 얻기 위해 .index[n] 를 사용한다
na_ratio.sort_values(ascending=False).index[0]   # 0번 행의 index를 가져온다
```

Out[7]: 'Age'

- 다음 데이터셋의 컬럼 중 결측치의 비율이 가장 높은 컬럼명을 출력하시오

In [1]:
```
import pandas as pd

path = 'C:/Users/nalang/pytest_basic/'
data=pd.read_csv(path+'titanic.csv')
data.head()
```

Out[1]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	S

연습문제 4

- 아래 데이터셋의 여행객 정보를 기반으로 여행보험 상품 가입 여부를 예측하시오
- (train 데이터의 TravelInsurance 컬럼이 상품 가입 여부이며, 독립변수로 되어 있는 test 데이터로 예측)
- ID와 예측치를 csv 파일로 저장하시오 (Travel_result.csv)

In [1]:
```
import pandas as pd

path = 'C:/Users/nalang/pytest_basic/'

train = pd.read_csv(path+"TravelInsurancePrediction_train.csv")
test = pd.read_csv(path+"TravelInsurancePrediction_test.csv")
```

In [2]:
```
train.head()
```

Out[2]:

	ID	Age	Employment Type	GraduateOrNot	AnnualIncome	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad	TravelInsurance
0	0	31	Government Sector	Yes	400000	6	1	No	No	0
1	1	31	Private Sector/Self Employed	Yes	1250000	7	0	No	No	0
2	2	34	Private Sector/Self Employed	Yes	500000	4	1	No	No	1
3	3	28	Private Sector/Self Employed	Yes	700000	3	1	No	No	0
4	4	28	Private Sector/Self Employed	Yes	700000	8	1	Yes	No	0

Out[19]:

	ID	predicted
0	1501	0
1	1502	0
2	1503	0
3	1504	0
4	1505	0

In [20]:
```
# 저장
id_predicted.to_csv(path+"Travel_result.csv", index=False)
```