

Machine Learning with Python

Life is too short, You need Python



실습 내용

- Mobile 데이터로 모델링합니다.
- Logistic Regression 알고리즘으로 모델링합니다.

1.환경 준비

- 기본 라이브러리와 대상 데이터를 가져와 이후 과정을 준비합니다.

```
In [1]: # 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings(action='ignore')
%config InlineBackend.figure_format='retina'
```

```
In [2]: # 데이터 읽어오기
path = 'https://raw.githubusercontent.com/jangrae/csv/master/mobile_cust_churn.csv'
data = pd.read_csv(path)
```

2.데이터 이해

- 분석할 데이터를 충분히 이해할 수 있도록 다양한 탐색 과정을 수행합니다.

```
In [3]: # 상위 몇 개 행 확인
data.head()
```

```
Out[3]:
```

	id	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDSET_PRICE	OVER_15MINS_CALLS_PER_I
0	1	0	31953	0	6	313378		161
1	2	1	36147	0	13	800586		244
2	3	1	27273	230	0	305049		201
3	4	0	120070	38	33	788235		780
4	5	1	29215	208	85	224784		241

데이터 설명

- COLLEGE: 대학 졸업여부
- INCOME: 연수입
- OVERAGE: 월평균 초과사용 시간(분)
- LEFTOVER: 월평균 잔여시간비율(%)
- HOUSE: 집값
- HANDSET_PRICE: 스마트폰 가격
- OVER_15MINS_CALLS_PER_MONTH: 월평균 장기통화(15분이상) 횟수
- AVERAGE_CALL_DURATION: 평균 통화 시간
- REPORTED_SATISFACTION: 만족도 설문조사 결과
- REPORTED_USAGE_LEVEL: 사용도 자가진단 결과
- CONSIDERING_CHANGE_OF_PLAN: 향후 변경계획 설문조사 결과
- CHURN: 이탈(번호이동) 여부 (Target 변수)

```
In [4]: # 변수 확인
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                     20000 non-null  int64
1   COLLEGE                              20000 non-null  int64
2   INCOME                                20000 non-null  int64
3   OVERAGE                              20000 non-null  int64
4   LEFTOVER                             20000 non-null  int64
5   HOUSE                                20000 non-null  int64
6   HANDSET_PRICE                        20000 non-null  int64
7   OVER_15MINS_CALLS_PER_MONTH          20000 non-null  int64
8   AVERAGE_CALL_DURATION                20000 non-null  int64
9   REPORTED_SATISFACTION                 20000 non-null  object
10  REPORTED_USAGE_LEVEL                  20000 non-null  object
11  CONSIDERING_CHANGE_OF_PLAN            20000 non-null  object
12  CHURN                                 20000 non-null  object
dtypes: int64(9), object(4)
memory usage: 2.0+ MB
```

```
In [5]: # 기술통계 확인
data.describe()
```

```
Out[5]:
```

	id	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDS
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000
mean	10000.500000	0.502400	80281.447750	85.979550	23.898650	493155.264250	30.000000
std	5773.647028	0.500007	41680.586319	85.992324	26.816645	252407.884692	20.000000
min	1.000000	0.000000	20007.000000	-2.000000	0.000000	150002.000000	1.000000
25%	5000.750000	0.000000	42217.000000	0.000000	0.000000	263714.250000	2.000000
50%	10000.500000	1.000000	75366.500000	59.000000	14.000000	452259.500000	3.000000
75%	15000.250000	1.000000	115881.750000	179.000000	41.000000	702378.000000	5.000000
max	20000.000000	1.000000	159983.000000	335.000000	89.000000	999996.000000	8.000000

```
In [6]: # target 값 개수 확인
data['CHURN'].value_counts()
```

```
Out[6]: CHURN
STAY      10148
LEAVE      9852
Name: count, dtype: int64
```

```
In [7]: # 상관관계 확인
data.corr(numeric_only=True)
```

Out[7]:

	id	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	H
id	1.000000	-0.005557	0.003686	-0.006050	0.006069	0.011347	
COLLEGE	-0.005557	1.000000	0.011122	-0.003091	-0.003925	-0.000217	
INCOME	0.003686	0.011122	1.000000	0.000458	0.006515	-0.010964	
OVERAGE	-0.006050	-0.003091	0.000458	1.000000	-0.003123	0.002412	
LEFTOVER	0.006069	-0.003925	0.006515	-0.003123	1.000000	0.006530	
HOUSE	0.011347	-0.000217	-0.010964	0.002412	0.006530	1.000000	
HANDSET_PRICE	-0.007838	0.009950	0.727200	0.000324	0.004004	-0.007756	
OVER_15MINS_CALLS_PER_MONTH	0.001254	-0.007205	0.002136	0.770557	-0.010411	0.007410	
AVERAGE_CALL_DURATION	-0.005830	-0.001490	-0.007219	0.000653	-0.660285	-0.009359	

3.데이터 준비

- 전처리 과정을 통해 머신러닝 알고리즘에 사용할 수 있는 형태의 데이터를 준비합니다.

1) 변수 제거

```
In [8]: # 제거 대상: id
drop_cols = ['id']

# 변수 제거
data = data.drop(drop_cols, axis=1)

# 확인
data
```

Out[8]:

	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDSET_PRICE	OVER_15MINS_CALLS_PER_
0	0	31953	0	6	313378		161
1	1	36147	0	13	800586		244
2	1	27273	230	0	305049		201
3	0	120070	38	33	788235		780
4	1	29215	208	85	224784		241
...
19995	0	153252	0	23	368403		597
19996	1	107126	71	82	237397		609
19997	0	78529	0	66	172589		275
19998	0	78674	47	41	572406		288
19999	0	124697	0	0	845575		808

20000 rows × 12 columns

2) x, y 분리

```
In [9]: # Target 설정
target = 'CHURN'

# 데이터 분리
x = data.drop(target, axis=1)
y = data.loc[:, target]
```

3) 가변수화

```
In [10]: # 가변수화 대상: REPORTED_SATISFACTION, REPORTED_USAGE_LEVEL, CONSIDERING_CHANGE_OF_PLAN
dumm_cols = ['REPORTED_SATISFACTION', 'REPORTED_USAGE_LEVEL', 'CONSIDERING_CHANGE_OF_PLAN']

# 가변수화
x = pd.get_dummies(x, columns=dumm_cols, drop_first=True, dtype=int)

# 확인
data
```

Out[10]:

	COLLEGE	INCOME	OVERAGE	LEFTOVER	HOUSE	HANDSET_PRICE	OVER_15MINS_CALLS_PER_
0	0	31953	0	6	313378		161
1	1	36147	0	13	800586		244
2	1	27273	230	0	305049		201
3	0	120070	38	33	788235		780
4	1	29215	208	85	224784		241
...
19995	0	153252	0	23	368403		597
19996	1	107126	71	82	237397		609
19997	0	78529	0	66	172589		275
19998	0	78674	47	41	572406		288
19999	0	124697	0	0	845575		808

20000 rows × 12 columns

4) 학습용, 평가용 데이터 분리

```
In [11]: # 모듈 불러오기
from sklearn.model_selection import train_test_split

# 7:3으로 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

4.모델링

- 본격적으로 모델을 선언하고 학습하고 평가하는 과정을 진행합니다.

```
In [12]: # 1단계: 불러오기
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
```

```
In [13]: # 2단계: 선언하기
model = LogisticRegression()
```

```
In [14]: # 3단계: 학습하기
model.fit(x_train, y_train)
```

```
Out[14]: LogisticRegression
LogisticRegression()
```

```
In [15]: # 4단계: 예측하기
y_pred = model.predict(x_test)
```

```
In [16]: # 5단계: 평가하기
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1795 1140]
 [1104 1961]]
```

	precision	recall	f1-score	support
LEAVE	0.62	0.61	0.62	2935
STAY	0.63	0.64	0.64	3065
accuracy			0.63	6000
macro avg	0.63	0.63	0.63	6000
weighted avg	0.63	0.63	0.63	6000

```
In [21]: # 예측 확률
p = model.predict_proba(x_test)
p = p[:,[1]]

# 임계값 : 0.4
y_pred2 = ['STAY' if x > 0.4 else 'LEAVE' for x in p] # 0.4보다 크면 1 아니면 0

# 성능 평가
print(classification_report(y_test, y_pred2))
```

	precision	recall	f1-score	support
LEAVE	0.72	0.41	0.53	2935
STAY	0.60	0.85	0.70	3065
accuracy			0.63	6000
macro avg	0.66	0.63	0.61	6000
weighted avg	0.66	0.63	0.62	6000

```
In [ ]:
```