고객의 행동을 예측하는 테크닉

파이썬을 이용한 실무 데이터 분석

데이터 가공

분석의 전제조건

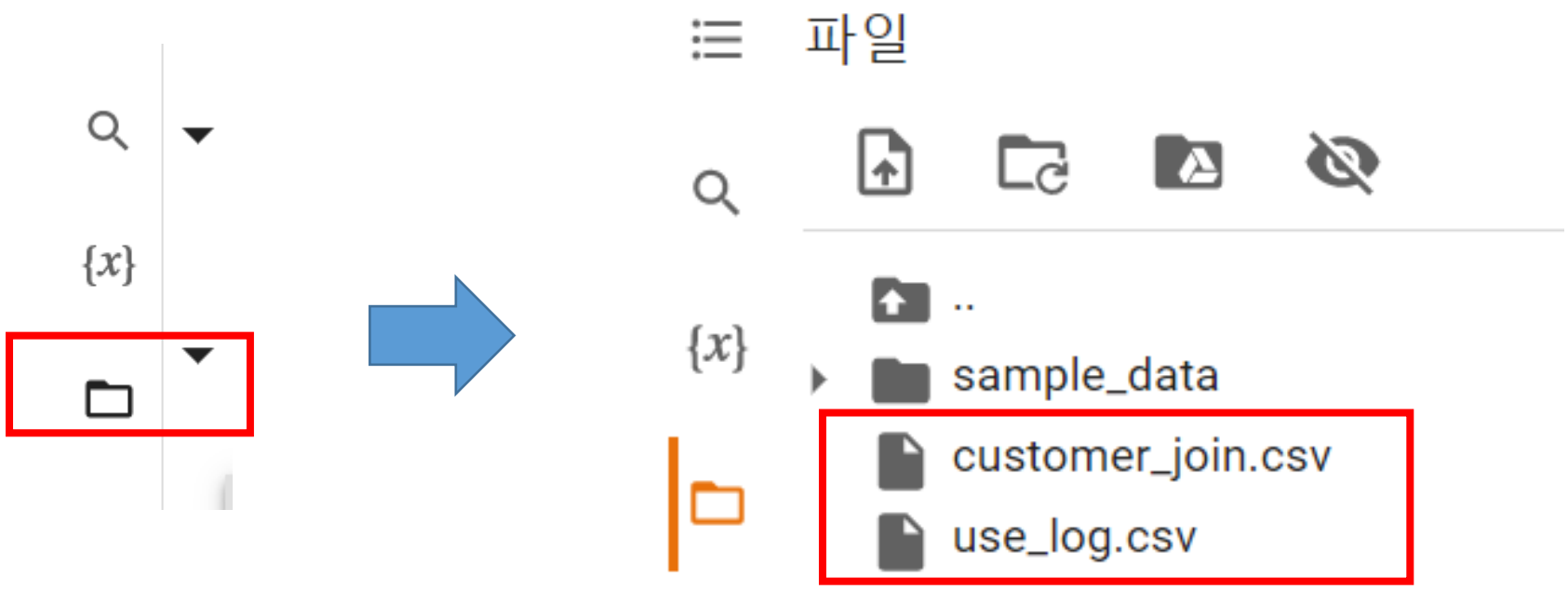
- 센터를 언제든 사용할 수 있는 종일 회원, 낮에만 사용할 수 있는 주간회원, 밤에만 사용할 수 있는 야간회원, 이렇게 3가지 종류의 회원 구분이 있다.
- 일반적으로 입회비가 들지만, 비정기적으로 입회비 반액 할인이나 무료 행사를 한다.
- 월말까지 탈퇴 신청을 하면 그 다음 달 말에 탈퇴가 된다.

	파일 이름	개요
1	use_log.csv	센터의 이용 이력 데이터, 기간은 2018년 4월~2019년 3월
	customer_master.csv	2019년 3월 말 시점의 회원 데이터
	class_master.csv	회원 구분 데이터(종일, 주간, 야간)
	campaign_master.csv	행사 구분 데이터(입회비 유무 등)
2	customer_join.csv	앞에서 생성한 이용이력을 포함한 고객 데이터

데이터 불러오기

데이터 업로드

- 데이터 파일을 구글 드라이브 임시 공간에 업로드



데이터 불러오기

데이터 불러오기

- pandas를 이용하여 use_log.csv 파일을 데이터 프레임으로 불러온다.
- 결측치를 확인해 본다. usedate에 결측치가 1개 있다.

```
import pandas as pd
uselog = pd.read_csv('use_log.csv')
uselog.isnull().sum()
```

```
log_id          0
customer_id     0
usedate         1
dtype: int64
```

데이터 불러오기

데이터 불러오기

- pandas를 이용하여 customer_join.csv 파일을 데이터 프레임으로 불러온다.
- 결측치를 확인한다. end_date 외에는 결측치가 없다. (end_date가 결측인 고객은 탈퇴 안 한 고객)

```
customer = pd.read_csv('customer_join.csv')

customer.isnull().sum()
```

```
customer_id          0
name                 0
class                0
gender               0
start_date           0
end_date          2842
campaign_id          0
is_deleted           0
class_name           0
price                0
campaign_name        0
mean                 0
median               0
max                  0
min                  0
routine_flg          0
calc_date            0
membership_period    0
dtype: int64
```

회원 군집화

군집화를 하기 위해 데이터 준비

- 비지도학습으로 군집화를 한다.
- [[mean, median, max, min, membership_period만 사용하기 위해 필요한 값 추출

```
customer_clustering = customer[["mean", "median",“ max", "min",
                                "membership_period"]]

customer_clustering.head()
```

	mean	median	max	min	membership_period
0	4.833333	5.0	8	2	47
1	5.083333	5.0	7	3	47
2	4.583333	5.0	6	3	47
3	4.833333	4.5	7	2	47
4	3.916667	4.0	6	1	47

회원 군집화

K-means 클러스터링으로 회원 군집화

- K-means 클러스터링을 사용한다. (K=4)
- 각 데이터의 크기가 크게 다르기 때문에 표준화를 한다. StandardScaler()

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
customer_clustering_sc = sc.fit_transform(customer_clustering)

kmeans = KMeans(n_clusters=4, random_state=0)
clusters = kmeans.fit(customer_clustering_sc)
customer_clustering["cluster"] = clusters.labels_
print(customer_clustering["cluster"].unique())
customer_clustering.head()
```

	mean	median	max	min	membership_period	cluster
0	4.833333	5.0	8	2	47	1
1	5.083333	5.0	7	3	47	1
2	4.583333	5.0	6	3	47	1
3	4.833333	4.5	7	2	47	1
4	3.916667	4.0	6	1	47	1

클러스터링 결과 분석

데이터의 수와 평균값 집계

- mean→월평균값, median→월중앙값, max→월최댓값, min→월최솟값, membership_period→회원기간
- 4개의 군집별로 집계

```
customer_clustering.columns = ["월평균값","월중앙값", "월최댓값", "월최솟값",
                               "회원기간", "cluster"]

customer_clustering.groupby("cluster").count()
```

cluster	월평균값	월중앙값	월최댓값	월최솟값	회원기간
0	840	840	840	840	840
1	1249	1249	1249	1249	1249
2	771	771	771	771	771
3	1332	1332	1332	1332	1332

군집화 결과 분석

데이터의 수와 평균값 집계

- 각 군집의 특징을 파악하기 위해 군집마다 평균값을 계산한다.
- 군집 0은 회원 기간은 짧지만 이용율이 높다.
- 군집 1은 회원 기간은 길지만 이용율이 낮다.

```
customer_clustering.groupby("cluster").mean()
```

cluster	월평균값	월중앙값	월최댓값	월최솟값	회원기간
0	8.061942	8.047024	10.014286	6.175000	7.019048
1	4.677561	4.670937	7.233787	2.153723	36.915933
2	3.065504	2.900130	4.783398	1.649805	9.276265
3	5.539535	5.391141	8.756006	2.702703	14.867868

클러스터링 결과 시각화

시각화를 위한 차원 축소

- 5개의 변수를 2차원으로 그리기 위해 차원을 축소한다. 정보를 잃지 않게 하면서 새로운 축을 만든다.
- 앞에서 표준화한 데이터에 주성분 분석(PCA)을 실행한다.

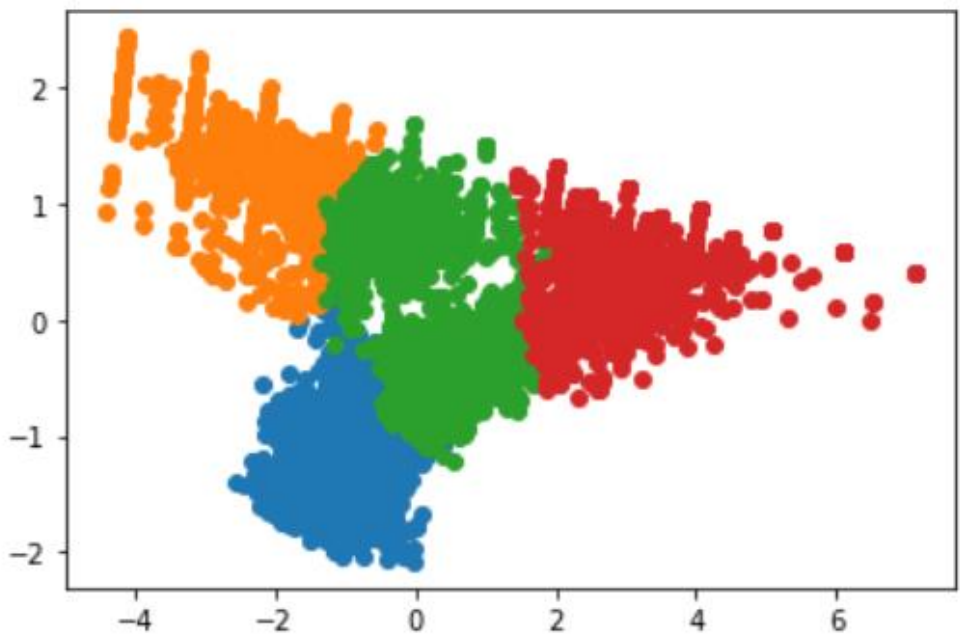
```
from sklearn.decomposition import PCA
X = customer_clustering_sc
pca = PCA(n_components=2)
pca.fit(X)
x_pca = pca.transform(X)
pca_df = pd.DataFrame(x_pca)
pca_df["cluster"] = customer_clustering["cluster"]
```

클러스터링 결과 시각화

시각화

- matplotlib을 사용하여 시각화한다.
- 색상이 섞이지 않고 깔끔하게 나뉜 것을 보니 PCA가 잘 되었다.

```
import matplotlib.pyplot as plt
%matplotlib inline
for i in customer_clustering["cluster"].unique():
    tmp = pca_df.loc[pca_df["cluster"]==i]
    plt.scatter(tmp[0], tmp[1])
```



탈퇴 회원의 경향 파악

탈퇴회원 집계

- is_deleted 컬럼을 추가해서 군집별, is_deleted별로 집계한다.
- 군집 0,1은 지속회원이 뚜렷하게 많고 군집 2는 모두 탈퇴회원이다. 군집 3은 지속회원이 더 많다.

```
customer_clustering = pd.concat([customer_clustering, customer], axis=1)

customer_clustering.groupby(["cluster","is_deleted"],as_index=False).count()[["cluster","is_deleted","customer_id"]]
```

	cluster	is_deleted	customer_id
0	0	0	821
1	0	1	19
2	1	0	1231
3	1	1	18
4	2	1	771
5	3	0	790
6	3	1	542

군집 0은 회원 기간이 짧지만 초기에 의욕적이어서 전체적으로 이용률이 높고, 군집 1은 회원 기간이 길고 이용률이 군집 0보다는 낮지만 지속 회원이 많은 것을 보면 이용이 안정적이다.

탈퇴 회원의 경향 파악

정기적/비정기적 이용 여부 파악

- 군집 0, 1, 3은 정기적으로 이용한 회원이 많고, 군집 2는 비정기적으로 이용한 회원이 많습니다.

```
customer_clustering.groupby(["cluster","routine_flg"],as_index=False).count()[["cluster","routine_flg","customer_id"]]
```

	cluster	routine_flg	customer_id
0	0	0	52
1	0	1	788
2	1	0	2
3	1	1	1247
4	2	0	499
5	2	1	272
6	3	0	226
7	3	1	1106

다음 달 이용횟수 예측을 위한 데이터 준비

회귀분석을 위한 준비

- 과거 6개월의 이용 데이터를 사용해 다음 달의 이용 횟수를 예측하는 지도학습의 회귀분석을 합니다.
- 2018년 10월~2019년 3월까지의 6개월의 고객별 이용횟수를 목적변수 데이터로 사용합니다.

```
uselog["usedate"] = pd.to_datetime(uselog["usedate"])
uselog["연월"] = uselog["usedate"].dt.strftime("%Y%m")
uselog_months = uselog.groupby(["연월","customer_id"],as_index=False).count()
uselog_months.rename(columns={"log_id":"count"}, inplace=True)
del uselog_months["usedate"]
uselog_months.head()
```

	연월	customer_id	count
0	201804	AS002855	4
1	201804	AS009013	2
2	201804	AS009373	3
3	201804	AS015315	6
4	201804	AS015739	7

다음 달 이용횟수 예측을 위한 데이터 준비

회귀분석을 위한 준비

```
year_months = list(uselog_months["연월"].unique())
predict_data = pd.DataFrame()
for i in range(6, len(year_months)):
    tmp = uselog_months.loc[uselog_months["연월"]==year_months[i]]
    tmp.rename(columns={"count":"count_pred"}, inplace=True)
    for j in range(1, 7):
        tmp_before = uselog_months.loc[uselog_months["연월"]==year_months[i-j]]
        del tmp_before["연월"]
        tmp_before.rename(columns={"count":"count_{}".format(j-1)}, inplace=True)
        tmp = pd.merge(tmp, tmp_before, on="customer_id", how="left")
    predict_data = pd.concat([predict_data, tmp], ignore_index=True)
predict_data.head()
```

다음 달 이용횟수 예측을 위한 데이터 준비

회귀분석을 위한 준비

- count_pred는 학습을 위해 사용하려는 데이터
- count_0은 1개월 전의 이용 데이터, count_1은 2개월 전의 이용 데이터…
- NaN은 이용기간이 짧아 데이터가 존재하지 않는 경우

	연월	customer_id	count_pred	count_0	count_1	count_2	count_3	count_4	count_5
0	201810	AS002855	3	7.0	3.0	5.0	5.0	5.0	4.0
1	201810	AS008805	2	2.0	5.0	7.0	8.0	NaN	NaN
2	201810	AS009373	5	6.0	6.0	7.0	4.0	4.0	3.0
3	201810	AS015233	7	9.0	11.0	5.0	7.0	7.0	NaN
4	201810	AS015315	4	7.0	3.0	6.0	3.0	3.0	6.0

다음 달 이용횟수 예측을 위한 데이터 준비

회귀분석을 위한 준비

- dropna()로 결측치를 포함하는 데이터를 삭제하고 index를 초기화 합니다.
- 6개월 이상 재적 중인 회원만 대상 회원이 되었습니다.

```
predict_data = predict_data.dropna()
predict_data = predict_data.reset_index(drop=True)
predict_data.head()
```

	연월	customer_id	count_pred	count_0	count_1	count_2	count_3	count_4	count_5
0	201810	AS002855	3	7.0	3.0	5.0	5.0	5.0	4.0
1	201810	AS009373	5	6.0	6.0	7.0	4.0	4.0	3.0
2	201810	AS015315	4	7.0	3.0	6.0	3.0	3.0	6.0
3	201810	AS015739	5	6.0	5.0	8.0	6.0	5.0	7.0
4	201810	AS019860	7	5.0	7.0	4.0	6.0	8.0	6.0

특성 추가

회원 기간 추가

- 회원 기간은 시계열 변화를 볼 수 있기 때문에 지금과 같은 시계열 데이터에 유효할 가능성이 있습니다.
- 먼저 고객 데이터인 customer의 start_date 컬럼을 앞에서 작성한 predict_data에 결합합니다.

```
predict_data = pd.merge(predict_data,
                        customer[["customer_id","start_date"]],
                        on="customer_id", how="left")

predict_data.head()
```

	연월	customer_id	count_pred	count_0	count_1	count_2	count_3	count_4	count_5	start_date
0	201810	AS002855	3	7.0	3.0	5.0	5.0	5.0	4.0	2016-11-01
1	201810	AS009373	5	6.0	6.0	7.0	4.0	4.0	3.0	2015-11-01
2	201810	AS015315	4	7.0	3.0	6.0	3.0	3.0	6.0	2015-07-01
3	201810	AS015739	5	6.0	5.0	8.0	6.0	5.0	7.0	2017-06-01
4	201810	AS019860	7	5.0	7.0	4.0	6.0	8.0	6.0	2017-10-01

특성 추가

회원 기간을 월 단위로 작성

```
predict_data["now_date"] = pd.to_datetime(predict_data["연월"], format="%Y%m")
predict_data["start_date"] = pd.to_datetime(predict_data["start_date"])
from dateutil.relativedelta import relativedelta
predict_data["period"] = None
for i in range(len(predict_data)):
    delta = relativedelta(predict_data["now_date"][i], predict_data["start_date"][i])
    predict_data["period"][i] = delta.years*12 + delta.months
predict_data.head()
```

	연월	customer_id	count_pred	count_0	count_1	count_2	count_3	count_4	count_5	start_date	now_date	period
0	201810	AS002855	3	7.0	3.0	5.0	5.0	5.0	4.0	2016-11-01	2018-10-01	23
1	201810	AS009373	5	6.0	6.0	7.0	4.0	4.0	3.0	2015-11-01	2018-10-01	35
2	201810	AS015315	4	7.0	3.0	6.0	3.0	3.0	6.0	2015-07-01	2018-10-01	39
3	201810	AS015739	5	6.0	5.0	8.0	6.0	5.0	7.0	2017-06-01	2018-10-01	16
4	201810	AS019860	7	5.0	7.0	4.0	6.0	8.0	6.0	2017-10-01	2018-10-01	12

다음 달 이용횟수 예측 모델 구축

학습데이터와 평가데이터를 준비하고 학습

- 2018년 4월 이후에 새로 가입한 회원 데이터만 이용해서 모델을 작성한다.
- scikit-learn의 LinearRegression() 사용한다.
- 학습데이터(75%)와 평가데이터(25%)로 나누고 학습을 한다.

```
predict_data = predict_data.loc[predict_data["start_date"]>=pd.to_datetime("20180401")]
from sklearn import linear_model
import sklearn.model_selection
model = linear_model.LinearRegression()
X = predict_data[["count_0","count_1","count_2","count_3","count_4","count_5","period"]]
y = predict_data["count_pred"]
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X,y)
model.fit(X_train, y_train)
```

```
LinearRegression()
```

다음 달 이용횟수 예측 모델 구축

성능 평가

- 0.6 정도의 정확도가 나왔다.
- 정확도를 높이기 위해 여러 가지 방법을 시도해보세요.

```
print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

```
0.6132496248552759
0.5894013013679484
```

모델에 기여하는 특성 확인

계수 출력

- 설명 변수마다 기여하는 계수를 출력해 봅시다.
- count_0이 가장 크고 과거로 거슬러 올라갈수록 기여도가 작아지는 경향이 있음을 알 수 있습니다.

```
coef=pd.DataFrame({"feature_names":X.columns, "coefficient":model.coef_})
coef
```

	feature_names	coefficient
0	count_0	0.315385
1	count_1	0.186596
2	count_2	0.175341
3	count_3	0.193020
4	count_4	0.077035
5	count_5	0.101716
6	period	0.066524

다음 달의 이용횟수 예측

예측

- x1, x2에 회원 두 명의 지난 6개월 간의 이용 횟수 데이터를 저장하고 다음 달 방문 횟수를 예측합니다.
- 회원 1(x1)은 3.8회, 회원 2(x2)는 1.9회로 예측되었습니다.

```
x1 = [3, 4, 4, 6, 8, 7, 8]
x2 = [2, 2, 3, 3, 4, 6, 8]
x_pred = [x1, x2]


model.predict(x_pred)
```

```
array([3.82492042, 1.97208737])
```

데이터 덤프

데이터 덤프

- 데이터를 use_log_months.csv 파일로 덤프합니다.

```
uselog_months.to_csv("use_log_months.csv",index=False)
```