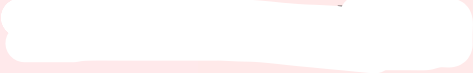
The background is a light pink gradient. It features several abstract shapes: a large blue shape in the top left, a white cloud-like shape in the center, a large red shape on the right, and a yellow shape at the bottom right. There are also some small grey dots and white wavy lines on the left side.

잡케어 추천 알고리즘 경진대회

잡케어 서비스에 적용 가능한 추천 알고리즘 개발





CONTENTS

01

분석 목표 및 데이터 소개

02

EDA (탐색적 자료 분석)

03

데이터 전처리

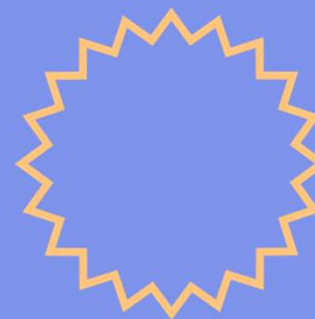
04

모델링 후 예측 및 평가



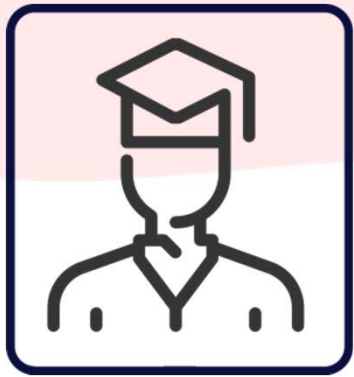


01



분석 목표 및 데이터 소개

잡케어 추천 알고리즘 경진대회

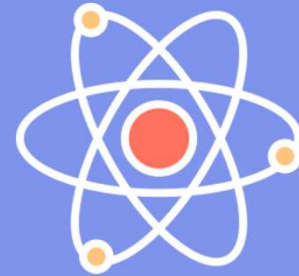


...



잡케어






일 자리를 탐색하는 구직자에게 구직자의 이력서를 통해 직무역량을
분석하여 훈련, 일자리 상담 등에 활용할 수 있도록 지원하는
시스템



회원 속성, 회원 선호 속성,
컨텐츠 속성 등을 이용하여
컨텐츠 이용 여부를 예측

활용 데이터

분석 활용 데이터 목록

-  test
-  train
-  속성_D_코드
-  속성_H_코드
-  속성_L_코드



train

훈련용 데이터. 칼럼 35개



test

테스트용 데이터. 칼럼 34개



속성 D, L, H 코드

train, test 데이터의 일부 칼럼 속성에 대한 데이터.
속성 D, L 코드 칼럼 5개. 속성 H 코드 칼럼 3개.

활용 데이터

데이터 칼럼 설명

train, test 데이터

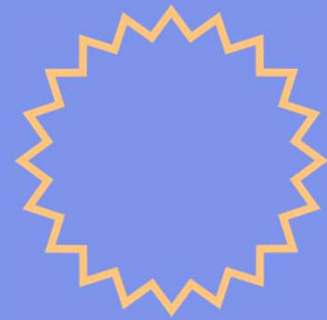
칼럼명	칼럼 설명	형태	비고
target	컨텐츠 사용 여부	명목형	예측할 변수, 0 또는 1
contents_attribute	컨텐츠 속성	명목형	I,A,J,J 하위,C,K,L,D,M,E,H
person_attribute	회원 속성	명목형	A, A 하위, B
person_prefer	회원 선호 속성	명목형	C, D1,D2,D3,E,F,G,H1,H2,H3
contents_open_dt	컨텐츠 열람 일시	날짜	
match_yn	속성 D,H 매칭 여부	논리형	회원선호속성과 컨텐츠속성 일치 여부
contents_rn	컨텐츠 번호	수치형	
person_rn	사용자 번호	수치형	

속성 D, L, H 코드 데이터

칼럼명	비고
속성 코드	D,L,H
대분류 코드	D,L,H
중분류 코드	D,L,H
소분류 코드	D,L
세분류 코드	D,L



02



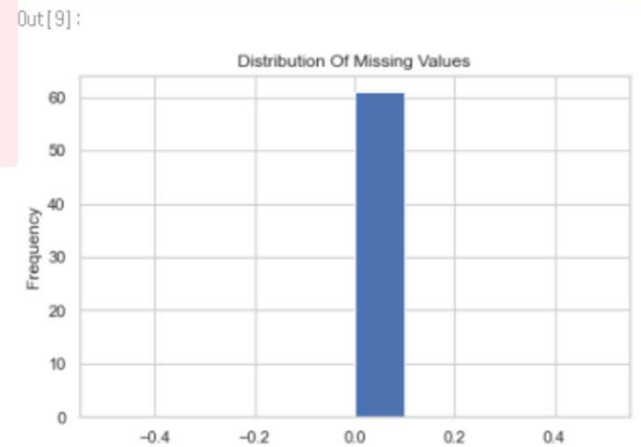
EDA(탐색적 자료 분석)

EDA

```
job 데이터 target 분포도:  
0    251106  
1    250845  
Name: target, dtype: int64
```

Target Columns
분포도 파악

```
job_data['target'].value_counts()
```



데이터에 포함된
결측치 파악

```
mis_val_df['mis_val_bool'].value_counts()
```


int형에 속하는 컬럼들에 특징 값 분포

```
job_data.select_dtypes('int64').apply(pd.Series.nunique, axis=0)
```

Out[12]:

id	501951	person_rn	300177
person_attribute_a	2	contents_rn	283359
person_attribute_a_1	8	person_prefer_h_1_u	19
person_attribute_b	6	person_prefer_h_2_u	19
person_prefer_c	5	person_prefer_h_3_u	19
person_prefer_d_1	1093	contents_attribute_h_u	17
person_prefer_d_2	1081	contents_attribute_l_n	736
person_prefer_d_3	1043	contents_attribute_l_s	305
person_prefer_e	12	contents_attribute_l_m	79
person_prefer_f	1	contents_attribute_l_l	21
person_prefer_g	1	Y	1
person_prefer_h_1	279	M	11
person_prefer_h_2	279	D	31
person_prefer_h_3	279	target	2
contents_attribute_l	3	dtype: int64	
contents_attribute_a	3		
contents_attribute_j_1	9		
contents_attribute_j	2		
contents_attribute_c	4		
contents_attribute_k	2		
contents_attribute_l	1752		
contents_attribute_d	1065		
contents_attribute_m	5		
contents_attribute_e	12		
contents_attribute_h	250		

가장 많은 특징 값을 갖는 것이 'person_rn'
동일한 특징 값을 갖는 것은
['person_prefer_f', 'person_prefer_g']
→ 추후 drop

Float형에 속하는 컬럼들에 특징 값 분포

```
job_data.select_dtypes('float64').apply(pd.Series.nunique, axis=0)
```

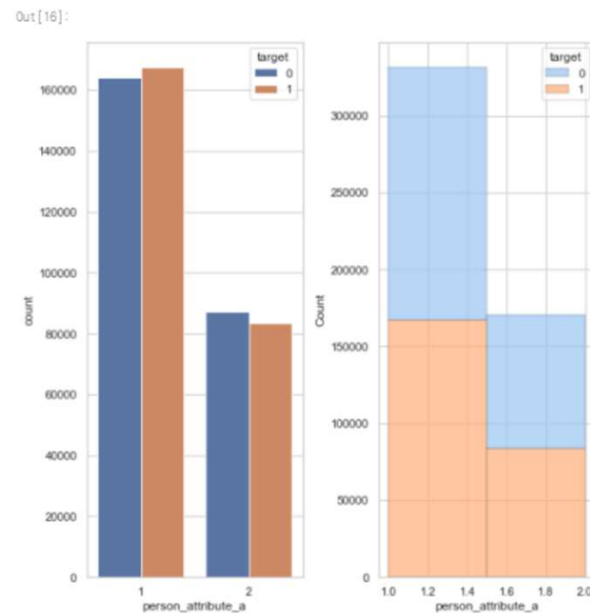
Out[13]:

person_prefer_d_1_n	443
person_prefer_d_1_s	36
person_prefer_d_1_m	137
person_prefer_d_1_l	11
person_prefer_d_2_n	435
person_prefer_d_2_s	36
person_prefer_d_2_m	137
person_prefer_d_2_l	11
person_prefer_d_3_n	420
person_prefer_d_3_s	36
person_prefer_d_3_m	136
person_prefer_d_3_l	11
contents_attribute_d_n	431
contents_attribute_d_s	36
contents_attribute_d_m	137
contents_attribute_d_l	11
dtype: int64	

- Float형인 것에 비해서 생각보다
특징 값의 분포도가 크지 않음.

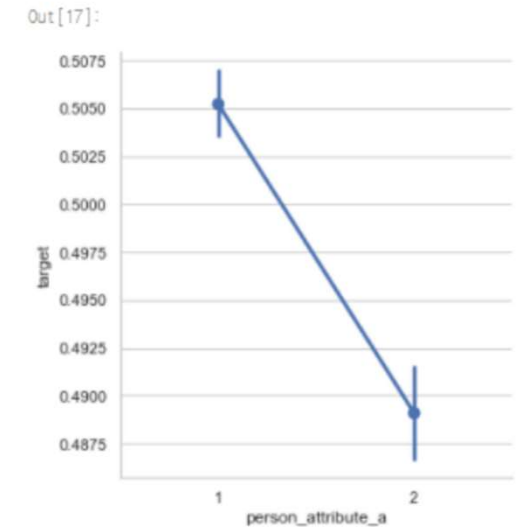
x = 'person_attribute_a'
y = 'target'

```
f, ax = plt.subplots(1, 2, figsize = (8, 8))  
  
x = 'person_attribute_a'  
y = 'target'  
  
sns.countplot('person_attribute_a', hue = 'target', data = job_data, ax = ax[0])  
  
sns.histplot(job_data,  
              x = "person_attribute_a",  
              hue="target",  
              multiple="stack",  
              palette="pastel",  
              edgecolor=".10",  
              linewidth=".20",  
              bins = 2,  
              ax = ax[1])  
  
plt.tight_layout(h_pad = 2.5)
```



```
sns.factorplot(x, y, data = job_data)
```

Out[17]:
<seaborn.axisgrid.FacetGrid at 0x28904d1fb88>

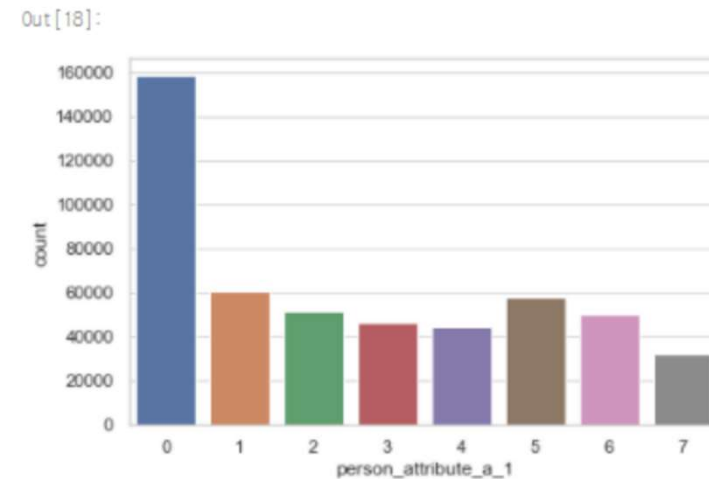


1의 특징 값을 값을 갖는 사람들이 target 1일 확률이 더욱 높음



person_attribute_a_1 회원 속성 A 하위 속성 1 특징 값

```
print("person_attribute_a_1: ")  
print(job_data['person_attribute_a_1'].value_counts())  
  
sns.countplot('person_attribute_a_1', data =job_data)
```



대부분의 사람들이 **0의 값을** 취하고 있는 것을 볼 수 있음

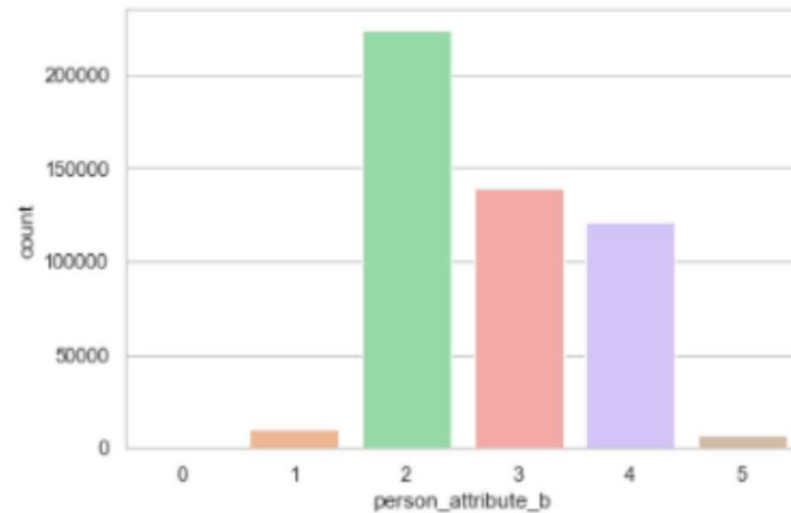


person_attribute_b 회원 속성 B 특징 값

```
print("person_attribute_b: ")  
print(job_data['person_attribute_b'].value_counts())  
  
sns.countplot('person_attribute_b', data =job_data, palette='pastel')
```



Out[26]:



다른 값들에 비해서 0의 값이 매우 적은 것을 볼 수 있음

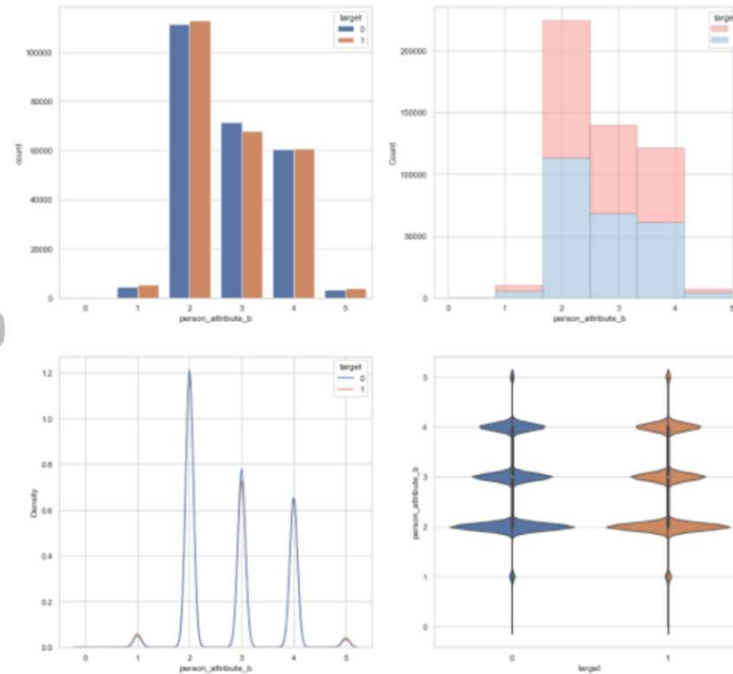


x = 'person_attribute_b'
y = 'target'

```
x = 'person_attribute_b'  
y = 'target'  
sns.countplot(x, hue = y, data =job_data, ax=ax[0][0])  
sns.histplot(job_data,  
             x = x,  
             hue=y,  
             multiple="stack",  
             palette="Pastel1",  
             edgecolor=".10",  
             linewidth=".20",  
             bins = 6,  
             ax = ax[0][1])  
sns.kdeplot(data=job_data, x=x, hue=y, ax=ax[1][0])  
sns.violinplot(data=job_data, x=y, y=x, ax=ax[1][1])  
plt.show()
```



Out[28]:

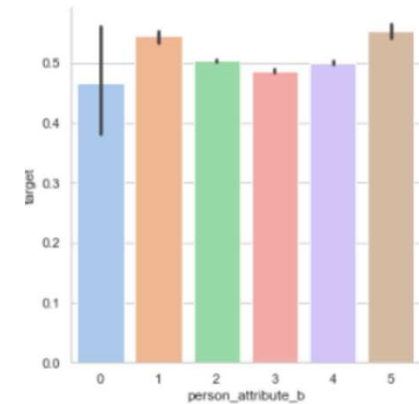


```
sns.catplot(x=x, y=y, kind="bar", data=job_data, palette='pastel')
```

ut[29]:

<seaborn.axisgrid.FacetGrid at 0x23904f16a48>

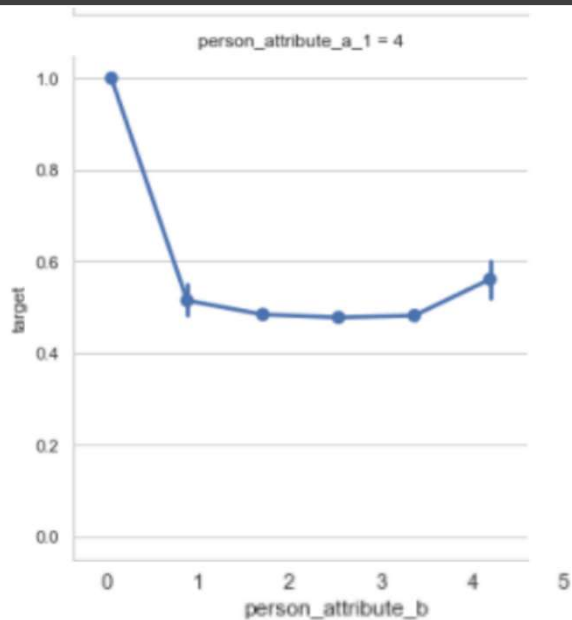
ut[29]:



💬 1의 특징과 5의 특징을 제외하고는 target 값이 1일 확률이 절반 이하 수준임 💬

x = 'person_attribute_b'
y = 'target'

```
sns.catplot(x=x, y=y, kind="point", row = "person_attribute_a_1", data=job_data)
```



```
job_data.loc[(job_data['person_attribute_a_1'] == 4) & (job_data['person_attribute_b'] == 0)]
```

Out[32]:

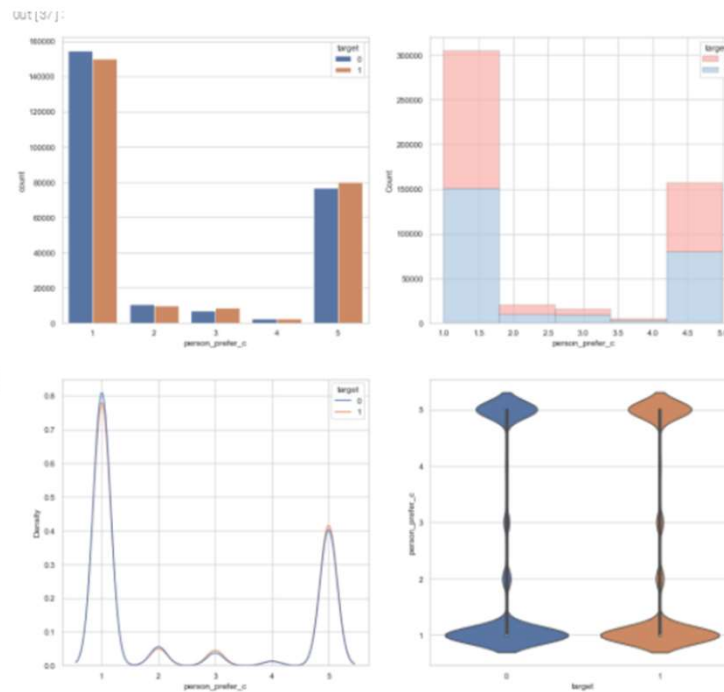
	id	d_l_match_yn	d_m_match_yn	d_s_match_yn	h_l_match_yn	h_m_match_yn	h_s_match_yn
427988	427988	False	False	False	True	True	True
429248	429248	False	False	False	True	True	True

이러한 특징 값을 갖는 사람은 2명뿐이다.
매우 희귀한 경우라고 할 수 있다.

💬 person_attribute_a_1 = 4 이고 person_attribute_b = 0이면 무조건 target = 1이 된다. 💬

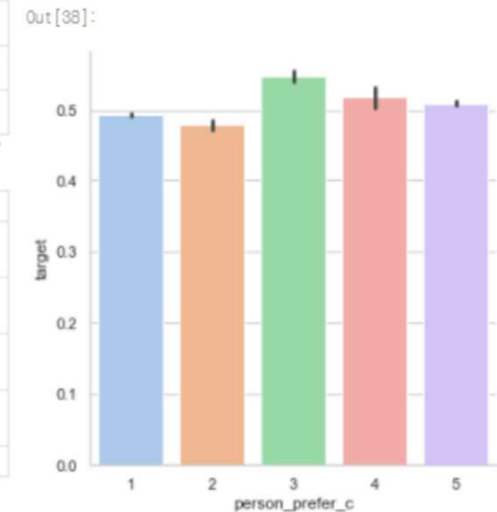
x = 'person_prefer_c'
y = 'target'

```
x = 'person_prefer_c'  
y = 'target'  
sns.countplot(x, hue = y, data = job_data, ax=ax[0][0])  
sns.histplot(job_data,  
             x = x,  
             hue=y,  
             multiple="stack",  
             palette="Pastel1",  
             edgecolor=".10",  
             linewidth=".20",  
             bins = 5,  
             ax = ax[0][1])  
sns.kdeplot(data=job_data, x=x, hue=y, ax=ax[1][0])  
sns.violinplot(data=job_data, x=y, y=x, ax=ax[1][1])  
plt.show()
```



```
sns.catplot(x=x, y=y, kind="bar", data=job_data, palette='pastel')
```

Out[38]:
<seaborn.axisgrid.FacetGrid at 0x23908982f08>

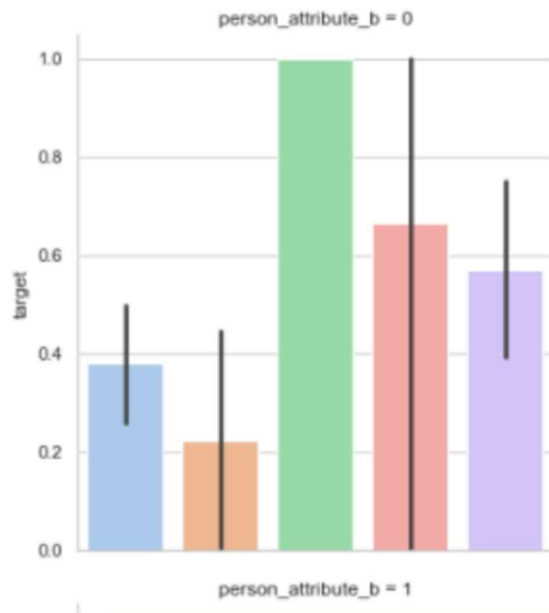


target 값에 대한 특별한 특징은 보이지 않음



x = 'person_prefer_c'
y = 'target'

```
sns.catplot(x=x, y=y, kind="bar", data=job_data, palette='pastel')
```



```
job_data.loc[(job_data['person_prefer_c'] == 3) & (job_data['person_attribute_b'] == 0)]
```

Out[40]:

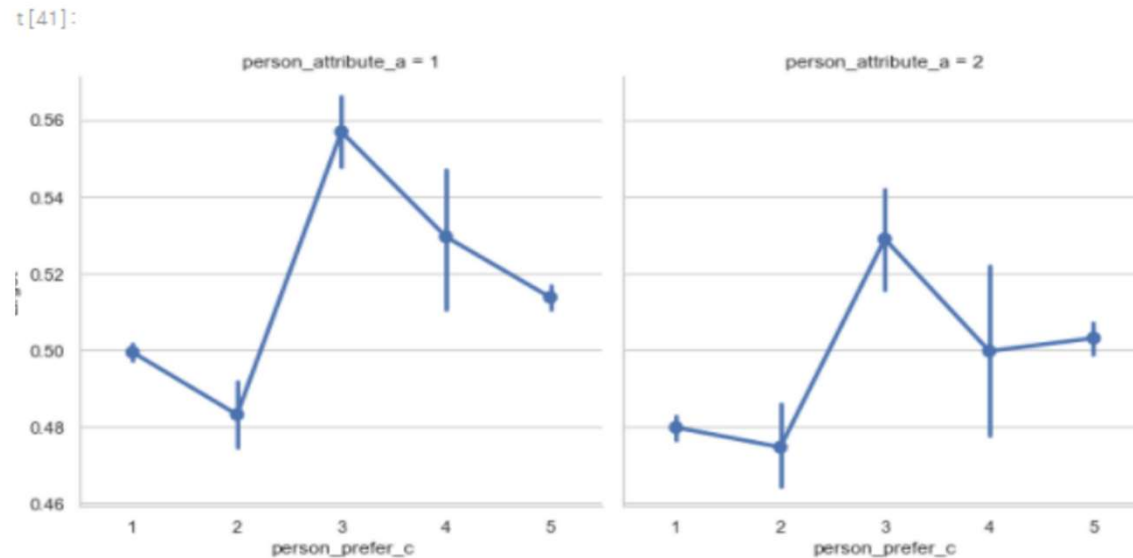
	id	d_l_match_yn	d_m_match_yn	d_s_match_yn	h_l_match_yn	h_m_match_yn	h_s_match_yn
144520	144520	False	False	False	False	False	False
157233	157233	True	True	True	True	False	False
220839	220839	True	False	False	True	True	True
309103	309103	False	False	False	False	False	False
314503	314503	False	False	False	False	False	False
378324	378324	False	False	False	True	False	False
437122	437122	True	True	True	False	False	False

위와 같은 특징 값을 갖는 사람들은 모두 target = 1 이 된다.
이러한 값을 갖을 확률은 0.13945584329944556%로 매우 희귀한 경우

person_prefer_c 3이고 person_attribute_b = 0일 경우에
무조건 적으로 target = 1인 것을 볼 수 있음.

x = 'person_prefer_c'
y = 'target'

```
sns.catplot(x=x, y=y, kind="point", col = "person_attribute_a", data=job_data)
```



person_attribute_a의 값에 상관 없이 그래프의 모양은 비슷하게 나오는 것을 볼 수 있음

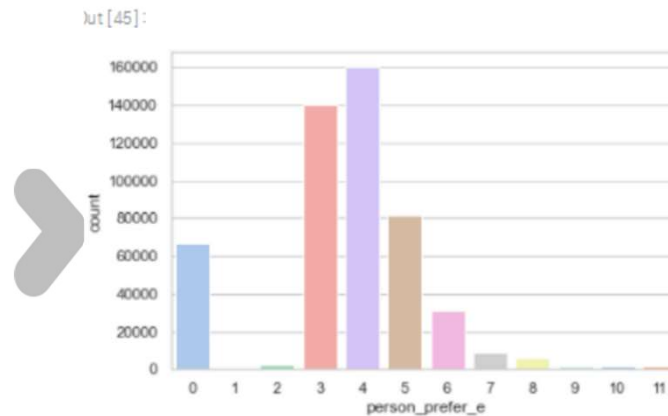
person_attribute_a 값이 1일 때 가 전체적으로 target 값이 1일 확률이 높음

person_attribute_a 값이 2가 될 때 person_prefer_c에서 4의 값을 갖는 것이 다소 target 값일 확률이 떨어짐

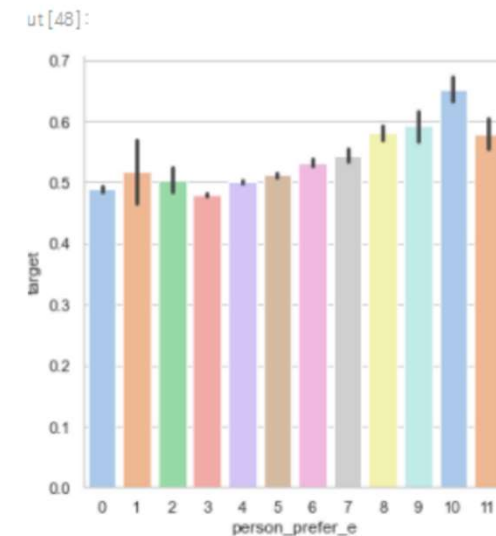
x = 'person_prefer_e'
y = 'target'

```
sns.catplot(x=x, y=y, kind="bar", data=job_data, palette='pastel')
```

```
print("person_prefer_e: ")  
print(job_data[person_prefer_e].value_counts())  
  
x = 'person_prefer_e'  
y = 'target'  
  
sns.countplot('person_prefer_e', data =job_data, palette='pastel')
```



ut[48]:
<seaborn.axisgrid.FacetGrid at 0x2390d799848>



값이 증가 할 수록 target일 확률이 높아짐
10일 경우에는 target 값이 될 확률이 60%가 넘음

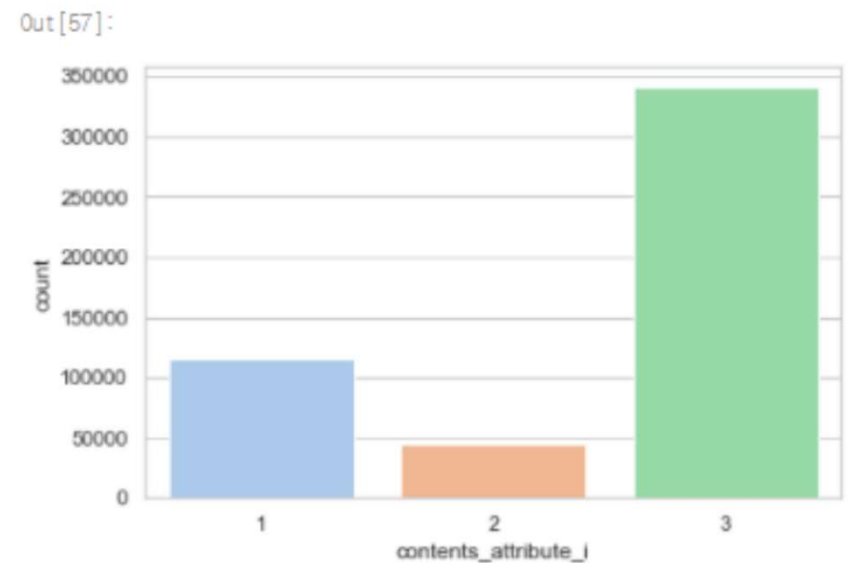


contents_attribute_i

```
# contents_attribute_i
print("contents_attribute_i: ")
print(job_data['contents_attribute_i'].value_counts())

x = 'contents_attribute_i'
y = 'target'

sns.countplot('contents_attribute_i', data =job_data, palette='pastel')
```



contents_attribute_i의 값에서는 3의 값이 분포가 가장 큼

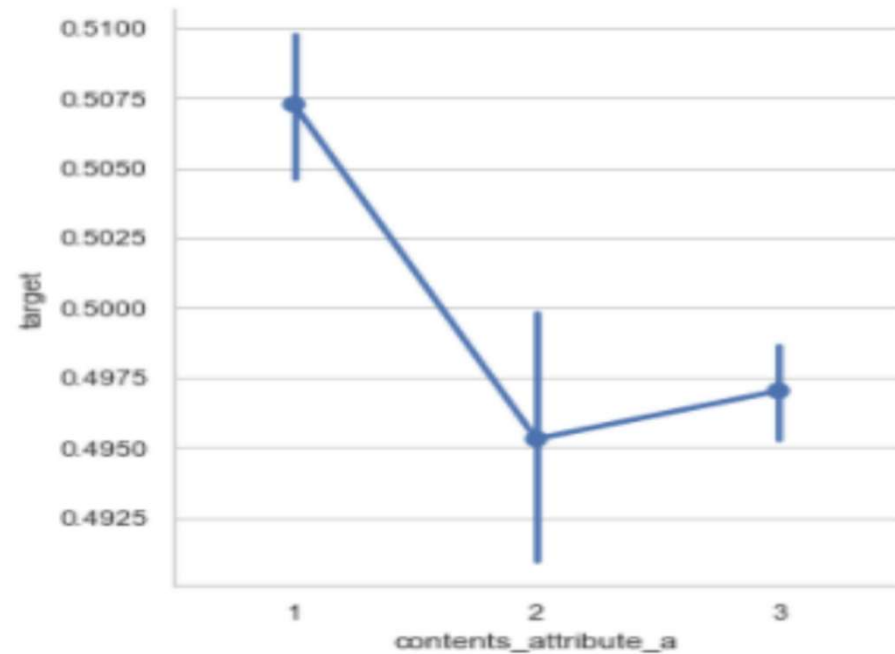


x = 'contents_attribute_a'
y = 'target'

```
sns.catplot(x=x, y=y, kind="point", data=job_data)
```



Out [75]:



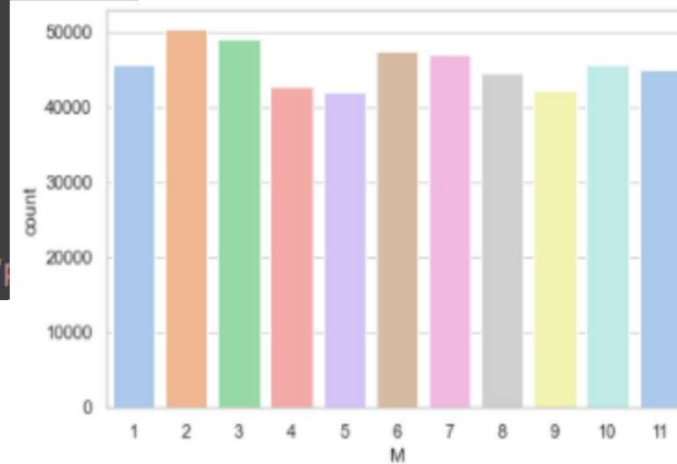
contents_attribute_a = 1일 때를 제외하고는 target 값의 확률이 50% 미만임

월 데이터(M)

```
# contents_attribute_j
print("M: ")
print(job_data["M"].value_counts())

x = 'M'
y = 'target'

sns.countplot("M", data =job_data, palette='')
```



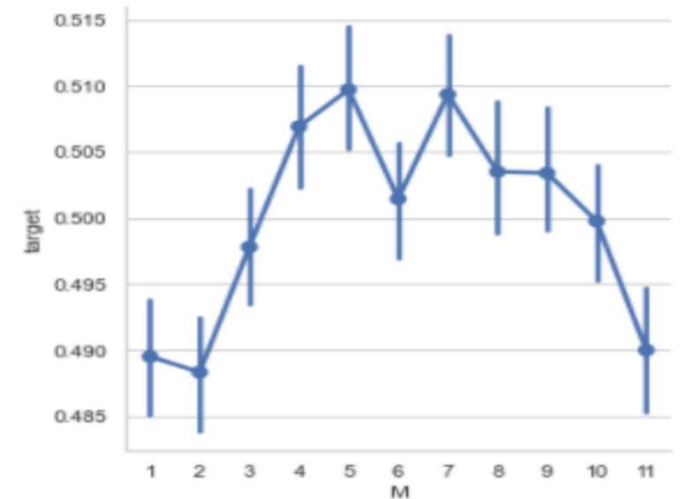
월 데이터에 대해서 전체적으로 균등하게 분포하고 있음
12월 데이터가 별도로 존재하지 않음

```
sns.catplot(x=x, y=y, kind="point", data=job_data)
```

Out[175]:

<seaborn.axisgrid.FacetGrid at 0x23a7bb42308>

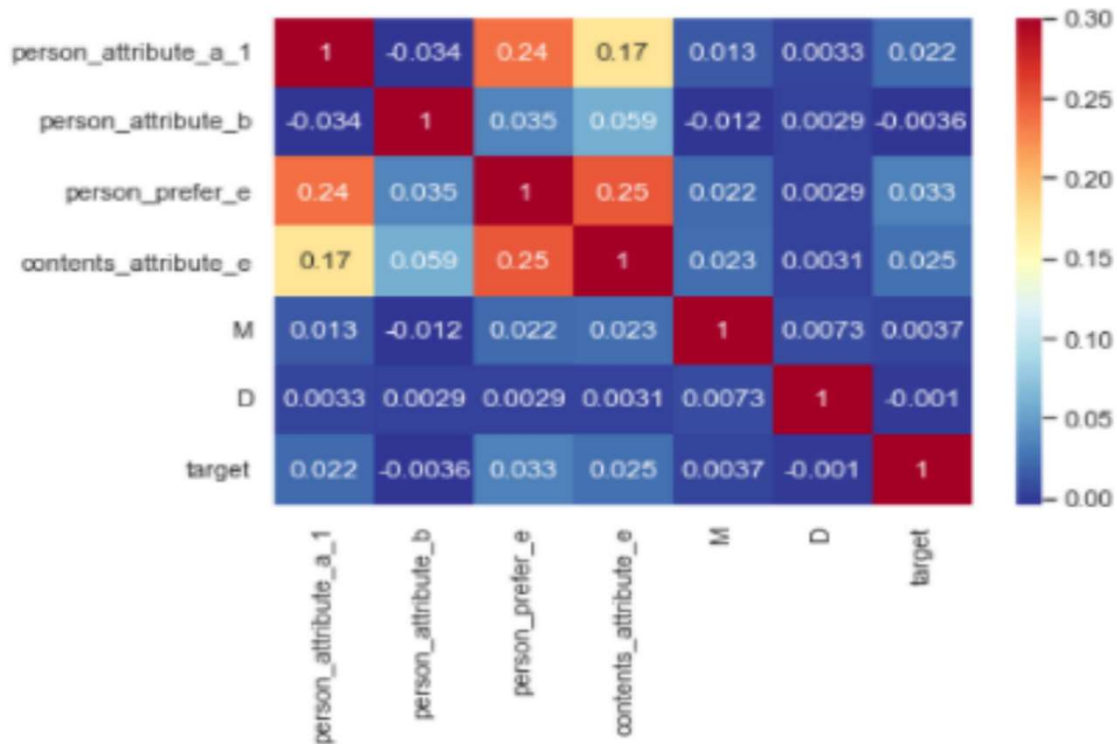
Out[175]:



1월, 2월, 11월이 전체 에서 가장 target과 관련이
없는 달인 것을 볼 수 있음

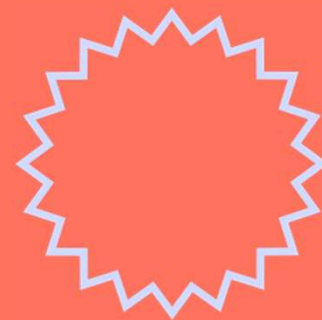
상관계수

['person_attribute_a_1', 'person_attribute_b',
'person_prefer_e', 'contents_attribute_e', 'M', 'D', 'target']

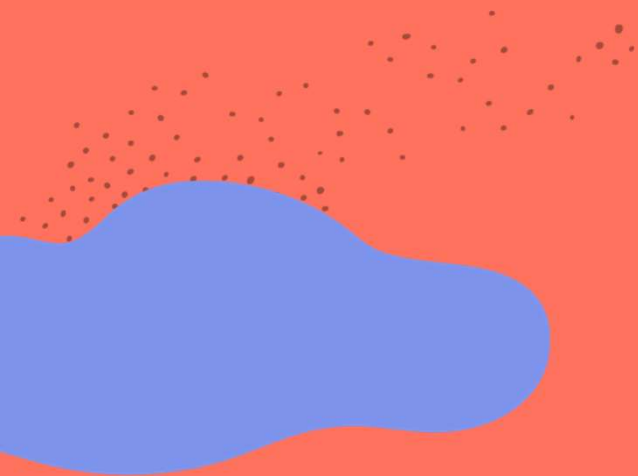


'contents_attribute_e' 와 'person_prefer_e'가
0.25로 가장 높은 상관계수를 보임

03



데이터 전처리



데이터 전처리

변수 drop

```
["id","person_prefer_f","person_prefer_g" ,"contents_open_dt"]
```



학습에 필요없는 컬럼 리스트 제거

데이터 전처리



```
cols_equi = [  
    ("contents_attribute_c", "person_prefer_c"),  
    ("contents_attribute_e", "person_prefer_e"),  
    ("person_prefer_d_1_s", "contents_attribute_d_s"),  
    ("person_prefer_d_2", "contents_attribute_d"),  
    ("person_prefer_d_2_n", "contents_attribute_d_n"),  
    ("person_prefer_d_2_s", "contents_attribute_d_s"),  
    ("person_prefer_d_2_m", "contents_attribute_d_m"),  
    ("person_prefer_d_2_l", "contents_attribute_d_l"),  
    ("person_prefer_d_3", "contents_attribute_d"),  
    ("person_prefer_d_3_n", "contents_attribute_d_n"),  
    ("person_prefer_d_3_s", "contents_attribute_d_s"),  
    ("person_prefer_d_3_m", "contents_attribute_d_m"),  
    ("person_prefer_d_3_l", "contents_attribute_d_l"),  
    ("person_prefer_h_2", "contents_attribute_h"),  
    ("person_prefer_h_2_l", "contents_attribute_h_l"),  
    ("person_prefer_h_2_m", "contents_attribute_h_m"),  
    ("person_prefer_h_3", "contents_attribute_h"),  
    ("person_prefer_h_3_l", "contents_attribute_h_l"),  
    ("person_prefer_h_3_m", "contents_attribute_h_m")]  
  
for col1, col2 in cols_equi:  
    train2[f"{col1}_{col2}"] = (train2[col1] == train2[col2]).astype(int)  
  
for col1, col2 in cols_equi:
```

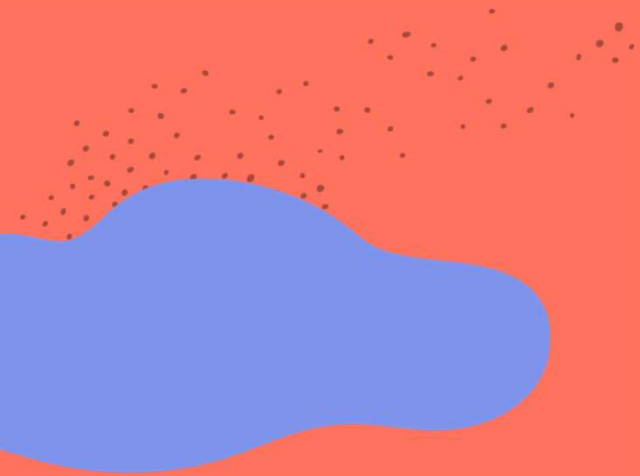
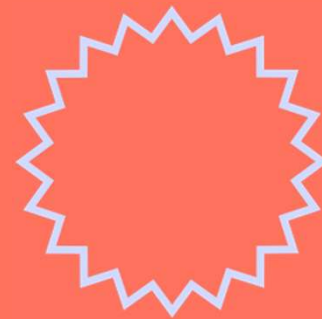
**# 회원 속성과 콘텐츠 속성의 동일한 코드 여부에
대한 컬럼명 리스트**

회원번호속성과 콘텐츠 속성 코드 매칭 여부 추가

04



모델링 후 예측 및 평가



다른 모델과의 f1 점수 비교

```
1 model = lgbm.LGBMClassifier(objective="binary", **study.best_params)
2 model.fit(
3     X_train,
4     y_train
5 )
6
7 pred = model.predict(X_test)
8 f1_score(y_test, pred)
```

0.6236020053991516

Light GBM의 f1 score :0.623

```
1 # sample 100개, tree depth = 100(max)
2
3 clf = RandomForestClassifier(n_estimators=100, max_depth=100, random_state=0)
4 clf.fit(train_x, train_y)
5
6 predict2 = clf.predict(test_x)
7 print(accuracy_score(test_y, predict2))
```

0.621001882638882

Random Forest의 f1 score :0.621

Catboost 특징

Level-wise Tree

Ordered Boosting

Random Permutation

Ordered Target Encoding

학습 파라미터 설정

```
is_holdout = False  
n_splits = 5  
iterations = 3000  
patience = 50
```

```
cv = KFold(n_splits=n_splits, shuffle=True, random_state=SEED)
```

- 데이터 분할수 : n_splits
- epoch를 나누어서 실행하는 횟수를 나타냄 : iteration
- 성능이 증가하지 않는 epoch을 몇 번이나 허용 할 지 정함 : patience

모델링 - 학습 시작

```
models = []
for tri, vai in cv.split(x_train):
    print("=" * 50)
    preds = []
    model = CatBoostClassifier(iterations=iterations, random_state=SEED, task_type="GPU", eval_metric="F1", cat_features=cat_features, one_hot_max_size=4)
    model.fit(x_train.iloc[tri], y_train[tri],
              eval_set=[(x_train.iloc[vai], y_train[vai])],
              early_stopping_rounds=patience,
              verbose=100,
              use_best_model=True
              )

    models.append(model)
    scores.append(model.get_best_score()["validation"]["F1"])
    if is_holdout:
        break
```

- catbootclassifier로 모델을 만듦
- model_fit으로 model_compile에서 지정한 방식으로 학습 진행
- eval_set으로 검증 세트 지정
- Append로 기존 리스트에 요소 추가 후 break로 반복문을 끝냄

Cv 결과

```
1 print(scores)
2 print(np.mean(scores))
```

```
[0.6862737699969816, 0.6856276933276506, 0.6823716897117412, 0.6802265066223585, 0.6767257638626933]
0.6822450847042851
```

약 0.68의 결과

Threshold값 0.4로 설정에 따른 검정점수 확인 및 추론

```
1 #threshold값 변경에 따른 검증점수 확인 및 추론
2 pred_list = []
3 scores = []
4 for i,(tri, vai) in enumerate( cv.split(x_train) ):
5     pred = models[i].predict_proba(x_train.iloc[vai])[:, 1]
6     pred = np.where(pred >= threshold , 1, 0)
7     score = f1_score(y_train[vai],pred)
8     scores.append(score)
9     pred = models[i].predict_proba(x_test)[:, 1]
10    pred_list.append(pred)
11 print(scores)
12 print(np.mean(scores))
```

```
[0.7126660503138533, 0.7125042678069163, 0.7084975715489905, 0.7115424019145815, 0.7060092921387218]
0.7102439167446126
```

Threshold는 분류기준을 뜻함

Threshold=0.4로 설정했더니 f1값이 0.71로 증가

결과

● WINNER ● 1% ● 4% ● 10%						전체 랭킹 >
#	팀	팀 멤버	최종점수	제출수	등록일	
201	djv		0.69883	14	13일 전	

Private 리더보드 201위

마무리 및 소감

잘한 점

- 모델링에 대해 체계적으로 배운 적이 없었기에 이번 기회를 통해 이론을 공부하여 적용하고자 노력했다.
- 여러가지 방법으로 모델링한 후 최적의 방법을 도출할 수 있었다.

아쉬운 점

- 머신러닝에 대해 더 잘 알았다면 더 좋은 결과가 있었을 것이라는 아쉬움이 남는다.



THANK YOU

발표를 들어주셔서 감사합니다 :)

