Fake / Authentic user Instagram

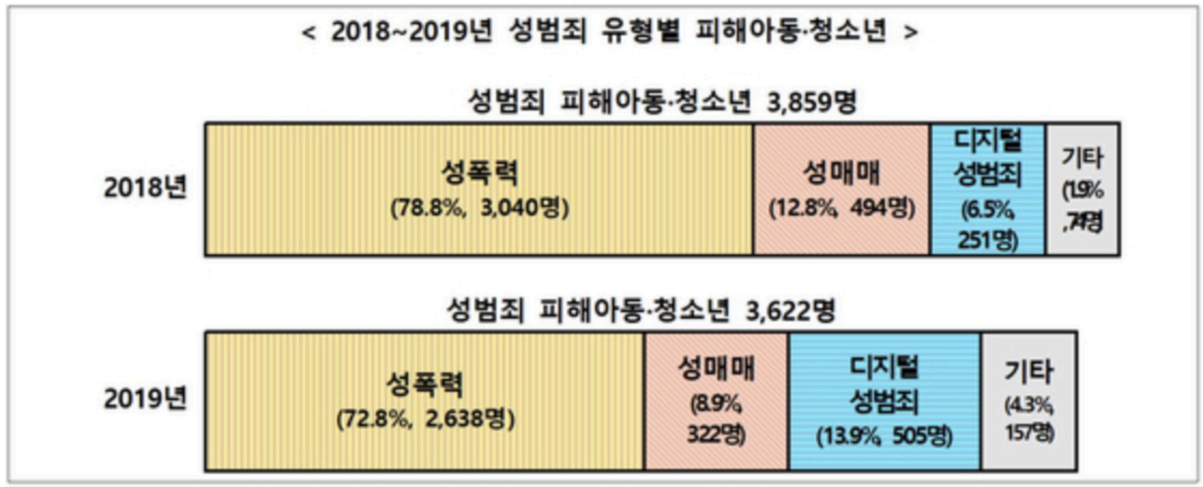
송나현

CONTENTS

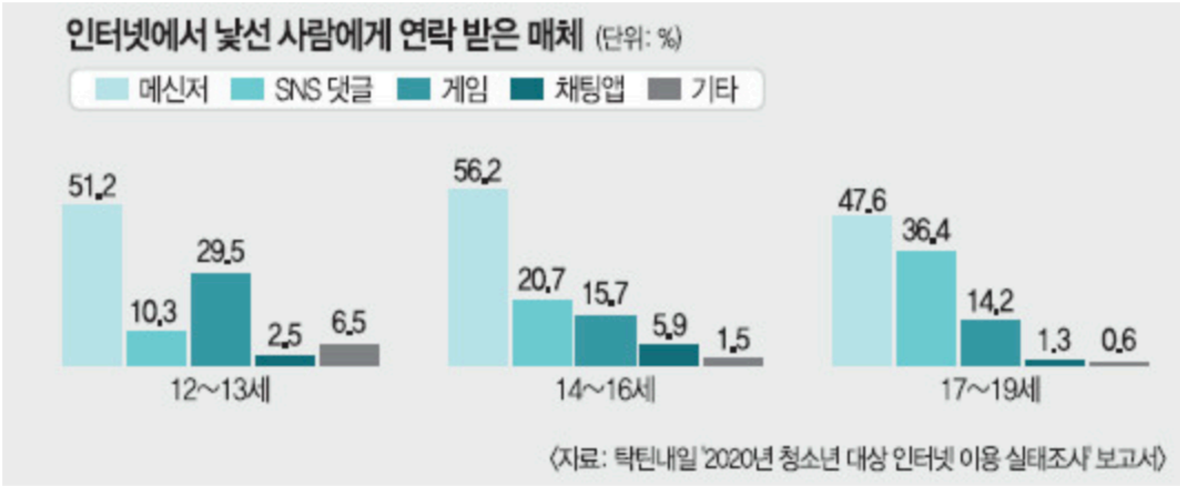
01

서론

▪ 서론



2018~2019년 성범죄 유형별 피해아동·청소년 〈여성가족부〉



02

EDA

▪ EDA

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
post	43307	NaN	NaN	NaN	152.552	701.72	0	4	22	102	76200
follower	43307	NaN	NaN	NaN	827.477	12503.9	0	107	289	712.5	1.9e+06
following	43307	NaN	NaN	NaN	2817.93	2806.1	0	483	1400	6100	7800
biography_lengh	43307	NaN	NaN	NaN	46.2633	60.6034	0	0	13	85	555
picture	43307	NaN	NaN	NaN	0.932644	0.250641	0	1	1	1	1
link	43307	NaN	NaN	NaN	0.142448	0.349513	0	0	0	0	1
caption_length	43307	NaN	NaN	NaN	120.868	212.129	-1	2	30	133	3274
caption_zero	43307	NaN	NaN	NaN	0.291397	0.356736	0	0	0.111111	0.529412	1
non_image_percentage	43307	NaN	NaN	NaN	0.184412	0.258915	0	0	0.056	0.278	1
like_rate	43307	NaN	NaN	NaN	20.4049	146.742	0	1.95	7.85	17.99	26650
comment_rate	43307	NaN	NaN	NaN	1.11732	6.80205	0	0.05	0.32	0.95	1009.09
location_tag	43307	NaN	NaN	NaN	0.16909	0.280428	0	0	0	0.231	1
hashtag	43307	NaN	NaN	NaN	0.449444	1.201	0	0	0	0.444	30
promotional_keywords	43307	NaN	NaN	NaN	0.0440439	0.266913	0	0	0	0	20
followers_keywords	43307	NaN	NaN	NaN	0.0644748	0.624448	0	0	0	0	58
cosine_similarity	43307	NaN	NaN	NaN	0.348074	0.37686	0	0.0391645	0.166252	0.654545	1
post_interval	43307	NaN	NaN	NaN	442.006	875.626	0	9.99069	146.031	517.563	26786.1
class	43307	4	a	12054	NaN	NaN	NaN	NaN	NaN	NaN	NaN

학습 데이터

- 모두 수치형 데이터

- picture, link의 경우 인코딩 필요

▪ EDA

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
post	43307	NaN	NaN	NaN	152.552	701.72	0	4	22	102	76200
follower	43307	NaN	NaN	NaN	827.477	12503.9	0	107	289	712.5	1.9e+06
following	43307	NaN	NaN	NaN	2817.93	2806.1	0	483	1400	6100	7800
biography_lengh	43307	NaN	NaN	NaN	46.2633	60.6034	0	0	13	85	555
picture	43307	NaN	NaN	NaN	0.932644	0.250641	0	1	1	1	1
link	43307	NaN	NaN	NaN	0.142448	0.349513	0	0	0	0	1
caption_length	43307	NaN	NaN	NaN	120.868	212.129	-1	2	30	133	3274
caption_zero	43307	NaN	NaN	NaN	0.291397	0.356736	0	0	0.111111	0.529412	1
non_image_percentage	43307	NaN	NaN	NaN	0.184412	0.258915	0	0	0.056	0.278	1
like_rate	43307	NaN	NaN	NaN	20.4049	146.742	0	1.95	7.85	17.99	26650
comment_rate	43307	NaN	NaN	NaN	1.11732	6.80205	0	0.05	0.32	0.95	1009.09
location_tag	43307	NaN	NaN	NaN	0.16909	0.280428	0	0	0	0.231	1
hashtag	43307	NaN	NaN	NaN	0.449444	1.201	0	0	0	0.444	30
promotional_keywords	43307	NaN	NaN	NaN	0.0440439	0.266913	0	0	0	0	20
followers_keywords	43307	NaN	NaN	NaN	0.0644748	0.624448	0	0	0	0	58
cosine_similarity	43307	NaN	NaN	NaN	0.348074	0.37686	0	0.0391645	0.166252	0.654545	1
post_interval	43307	NaN	NaN	NaN	442.006	875.626	0	9.99069	146.031	517.563	26786.1
class	43307	4	a	12054	NaN	NaN	NaN	NaN	NaN	NaN	NaN

타겟 데이터

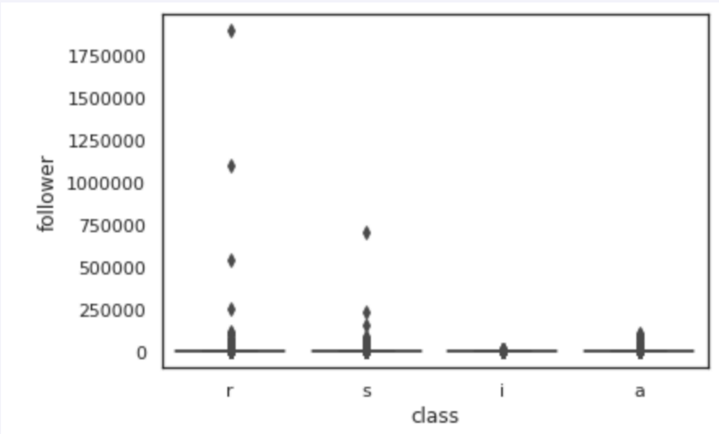
- A : 활성화 되어 있는 가짜 계정

- I : 비활성화 되어 있는 가짜 계정

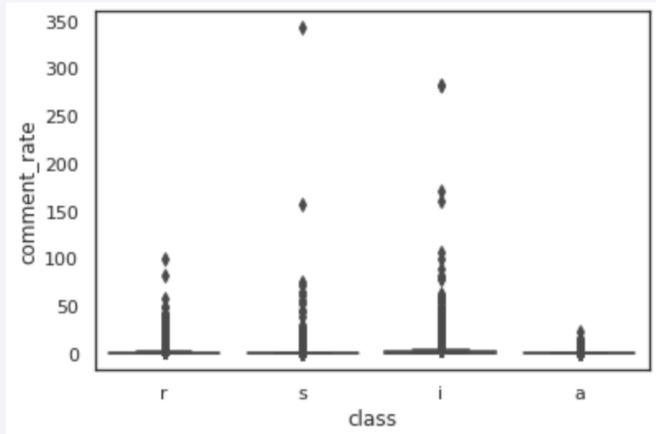
- R : 진짜 계정

- S : 스팸 계정

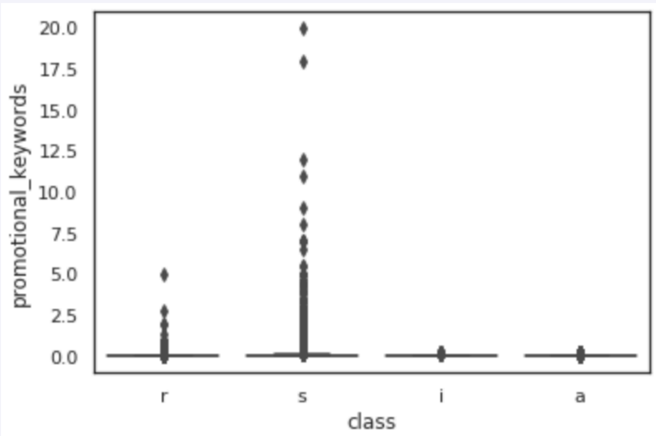
▪ EDA



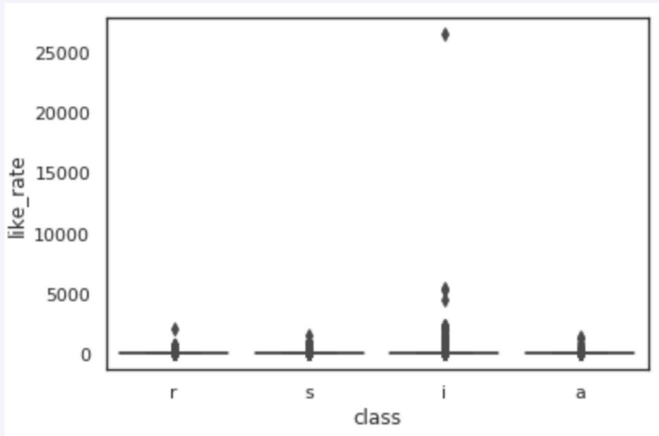
follower
1750000
1500000
1250000
1000000
750000
500000
250000
0
r
s
i
a
class



comment_rate
350
300
250
200
150
100
50
0
r
s
i
a
class

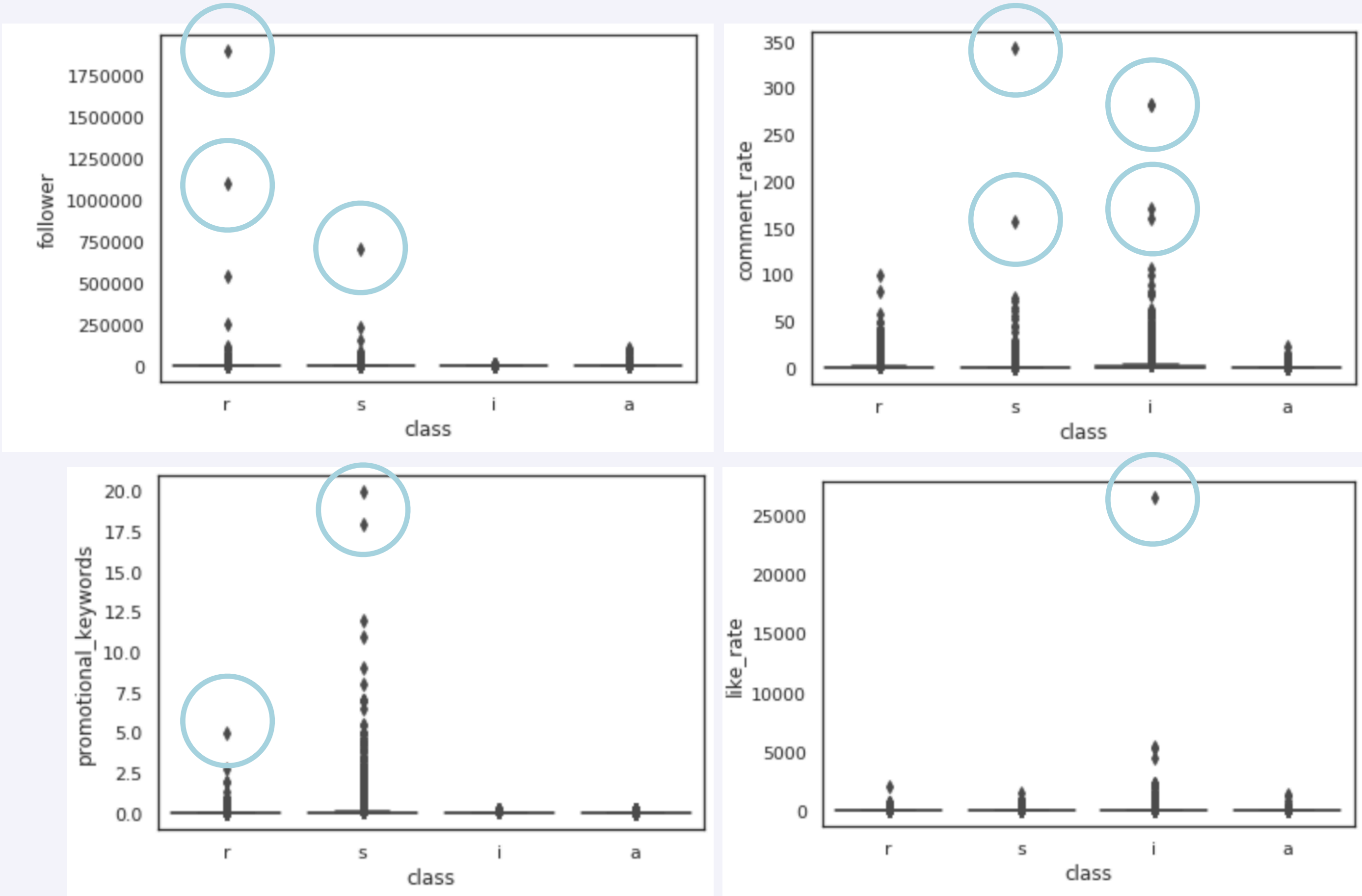


promotional_keywords
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
r
s
i
a
class



like_rate
25000
20000
15000
10000
5000
0
r
s
i
a
class

▪ EDA



follower
1750000
1500000
1250000
1000000
750000
500000
250000
0
r
s
i
a
class
comment_rate
350
300
250
200
150
100
50
0
r
s
i
a
class
promotional_keywords
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
r
s
i
a
class
like_rate
25000
20000
15000
10000
5000
0
r
s
i
a
class

03

특성 공학

▪ 특성 공학

class	following>follower	like/post	comment/post
0	0	5474.920000	270.500000
0	1	15519.060000	1558.170000
0	0	13848.000000	912.000000
0	0	22484.001100	1771.000000
0	1	2099.880000	399.000000
...	...	...	...
2	1	5527.730046	4.210000
1	1	533.260006	33.279999
1	1	1298.459994	48.450002
1	1	4399.960035	199.709998
3	1	11286.000108	133.000001

특성 추가

-following>follower :
팔로잉이 더 많으면 1, 아니면 0

- like/post :
게시물당 좋아요 개수

- comment/post :
게시물당 댓글 개수

타겟 데이터 매핑

- r : 0
- l : 1
- a : 2
- s : 3

04

모델링

▪ 모델링 - Baseline

```
#타겟 데이터 확인 -> 거의 valanced
y_train.value_counts(normalize=True)
```

```
2      0.286207
0      0.245188
3      0.244706
1      0.223899
Name: class, dtype: float64
```

```
#baseline model
major = y_train.mode()[0]
y_pred = [major] * len(y_train)

from sklearn.metrics import accuracy_score
accuracy_score(y_train, y_pred)
```

```
0.28620702444090005
```

▪ 모델링 - Baseline

```
#타겟 데이터 확인 -> 거의 valanced
y_train.value_counts(normalize=True)
```

```
2     0.286207
0     0.245188
3     0.244706
1     0.223899
Name: class, dtype: float64
```

```
#baseline model
major = y_train.mode()[0]
y_pred = [major] * len(y_train)

from sklearn.metrics import accuracy_score
accuracy_score(y_train, y_pred)
```

```
0.286207024409005
```

평가지표로 accuracy 사용

▪ 모델링 - RandomForestClassifier

```
from sklearn.pipeline import make_pipeline
from category_encoders import TargetEncoder
from sklearn.ensemble import RandomForestClassifier

pipe = make_pipeline(
    TargetEncoder(),
    RandomForestClassifier(random_state=2)
)


pipe.fit(X_train, y_train)
```

```
              precision    recall  f1-score   support

           0       0.84      0.78      0.81      1702
           1       0.93      0.94      0.94      1479
           2       0.92      0.93      0.93      1970
           3       0.93      0.97      0.95      1590

    accuracy                           0.91      6741
   macro avg       0.90      0.91      0.90      6741
weighted avg       0.90      0.91      0.90      6741
```

▪ 모델링 - RandomForestClassifier

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint, uniform

pipe_tg = make_pipeline(
    TargetEncoder(),
    RandomForestClassifier(random_state=2)
)


dists = {
    'targetencoder__min_samples_leaf': randint(1, 10),
    'targetencoder__smoothing': [1.,10.],
    'randomforestclassifier__n_estimators': randint(100, 500),
    'randomforestclassifier__max_depth': randint(0, 20),
    'randomforestclassifier__max_features': uniform(0, 1),
    'randomforestclassifier__min_samples_leaf': randint(1, 10)
}

clf_tg = RandomizedSearchCV(
    pipe_tg,
    param_distributions=dists,
    n_iter=3,
    cv=3,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)

clf_tg.fit(X_train, y_train)
```

▪ 모델링 - RandomForestClassifier

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint, uniform

pipe_tg = make_pipeline(
    TargetEncoder(),
    RandomForestClassifier(random_state=2)
)


dists = {
    'targetencoder__min_samples_leaf': randint(1, 10),
    'targetencoder__smoothing': [1.,10.],
    'randomforestclassifier__n_estimators': randint(100, 500),
    'randomforestclassifier__max_depth': randint(0, 20),
    'randomforestclassifier__max_features': uniform(0, 1),
    'randomforestclassifier__min_samples_leaf': randint(1, 10)
}

clf_tg = RandomizedSearchCV(
    pipe_tg,
    param_distributions=dists,
    n_iter=3,
    cv=3,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)

clf_tg.fit(X_train, y_train)
```

	precision	recall	f1-score	support
0	0.83	0.79	0.81	1702
1	0.94	0.95	0.94	1479
2	0.92	0.93	0.93	1970
3	0.93	0.96	0.94	1590
accuracy			0.91	6741
macro avg	0.90	0.91	0.91	6741
weighted avg	0.90	0.91	0.90	6741

▪ 모델링 - LGBMClassifier

```
from lightgbm import LGBMClassifier

pipe_lg = make_pipeline(
    TargetEncoder(),
    LGBMClassifier()
)


dists = {
    'targetencoder__min_samples_leaf': randint(1, 10),
    'targetencoder__smoothing': [1.,10.],
    'lgbmclassifier__learning_rate': [0.1, 0.01],
    'lgbmclassifier__num_leaves': randint(1, 50),
    'lgbmclassifier__max_depth': randint(1, 20),
    'lgbmclassifier__n_estimators': randint(100, 500)
}

clf_lg = RandomizedSearchCV(
    pipe_lg,
    param_distributions=dists,
    n_iter=5,
    cv=3,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)

clf_lg.fit(X_train, y_train)
```

▪ 모델링 - LGBMClassifier

```python
from lightgbm import LGBMClassifier

pipe_lg = make_pipeline(
    TargetEncoder(),
    LGBMClassifier()
)


dists = {
    'targetencoder__min_samples_leaf': randint(1, 10),
    'targetencoder__smoothing': [1.,10.],
    'lgbmclassifier__learning_rate': [0.1, 0.01],
    'lgbmclassifier__num_leaves': randint(1, 50),
    'lgbmclassifier__max_depth': randint(1, 20),
    'lgbmclassifier__n_estimators': randint(100, 500)
}

clf_lg = RandomizedSearchCV(
    pipe_lg,
    param_distributions=dists,
    n_iter=5,
    cv=3,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)

clf_lg.fit(X_train, y_train)
```

검증 데이터 사용 결과

	precision	recall	f1-score	support
0	0.92	0.90	0.91	1702
1	0.95	0.95	0.95	1479
2	0.98	0.98	0.98	1970
3	0.97	0.98	0.98	1590
accuracy			0.96	6741
macro avg	0.95	0.95	0.95	6741
weighted avg	0.95	0.96	0.95	6741

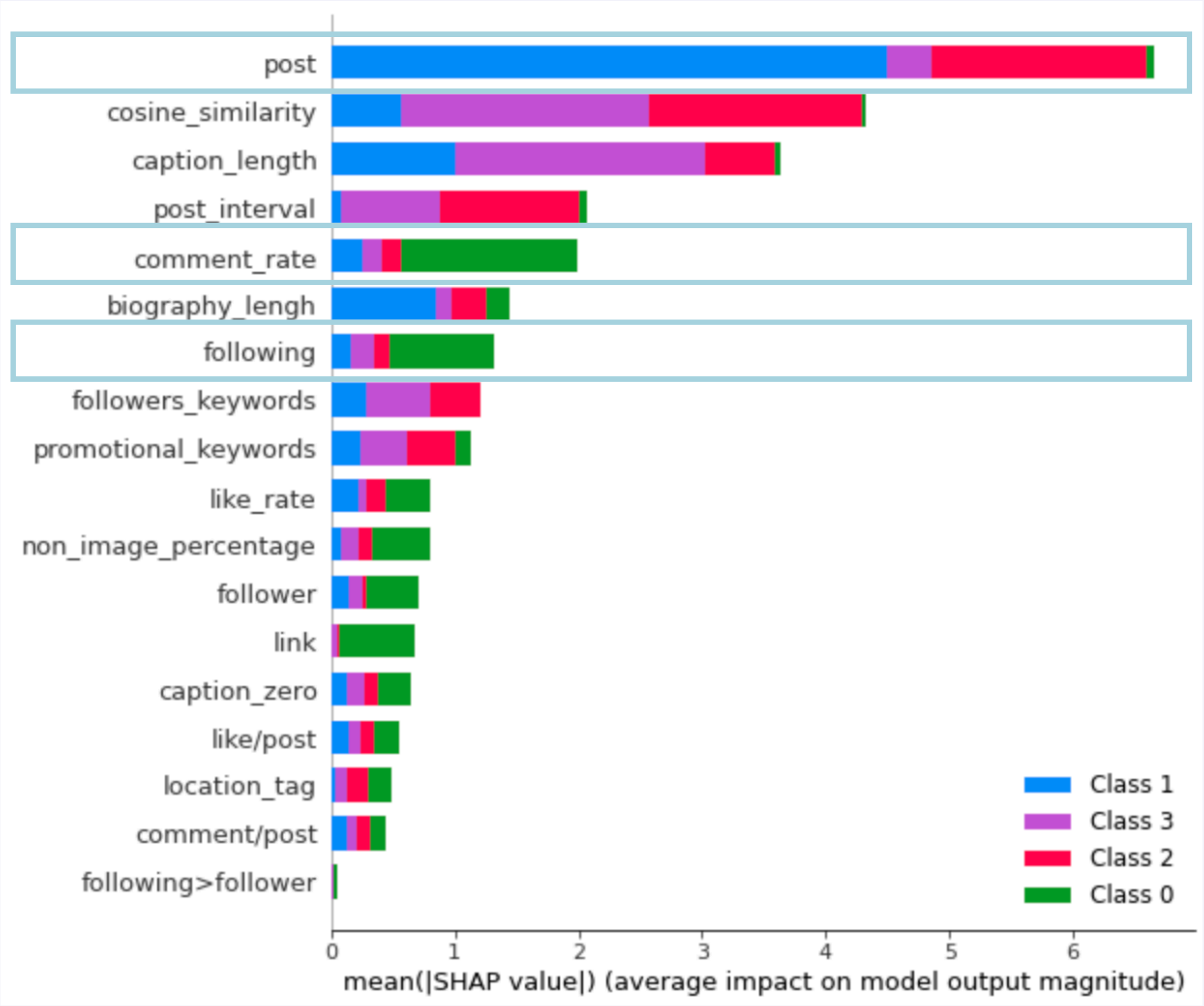
테스트 데이터 사용 결과

	precision	recall	f1-score	support
0	0.90	0.89	0.90	2100
1	0.96	0.95	0.95	1928
2	0.97	0.97	0.97	2358
3	0.96	0.98	0.97	2040
accuracy			0.95	8426
macro avg	0.95	0.95	0.95	8426
weighted avg	0.95	0.95	0.95	8426

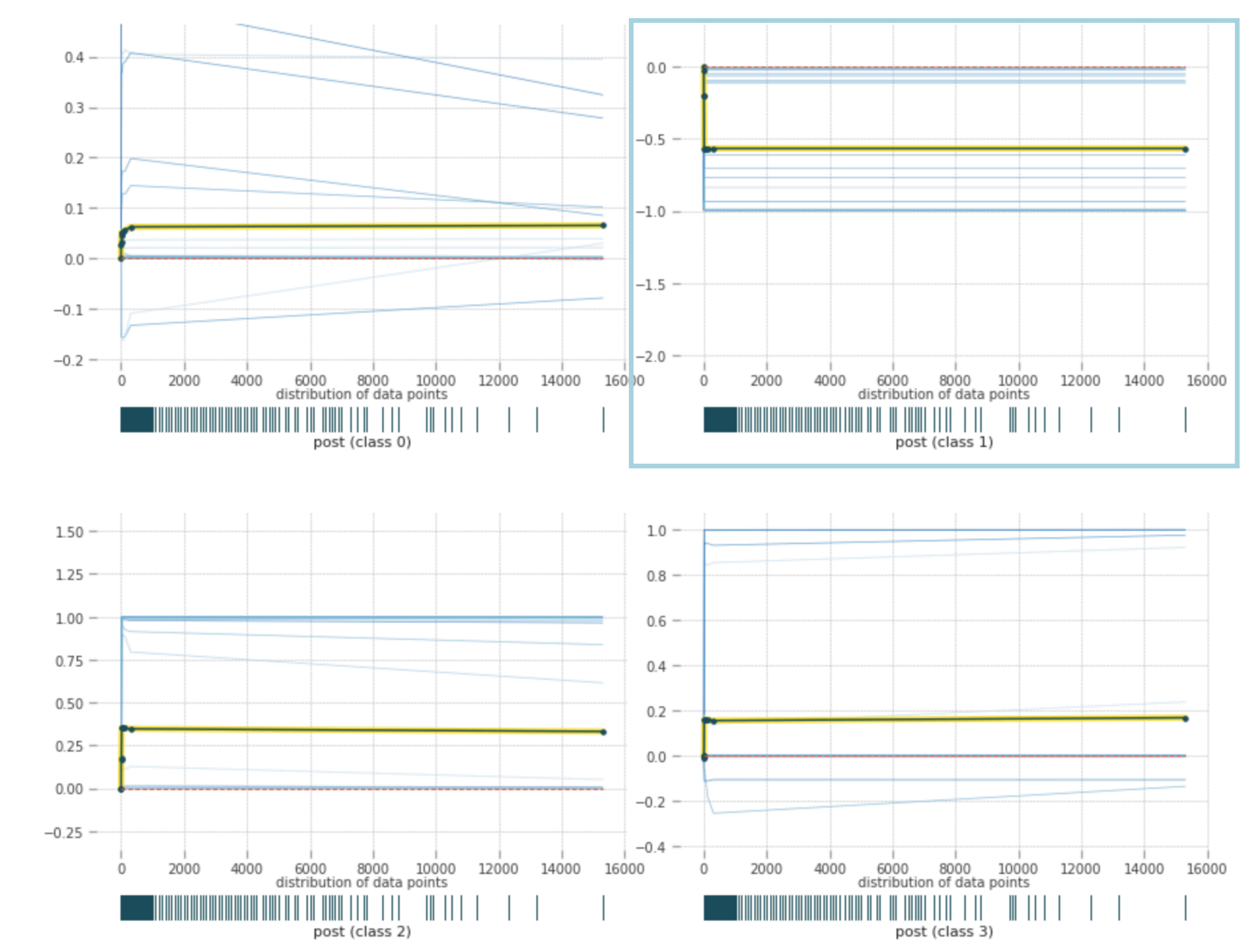
05

모델 해석

▪ 모델 해석

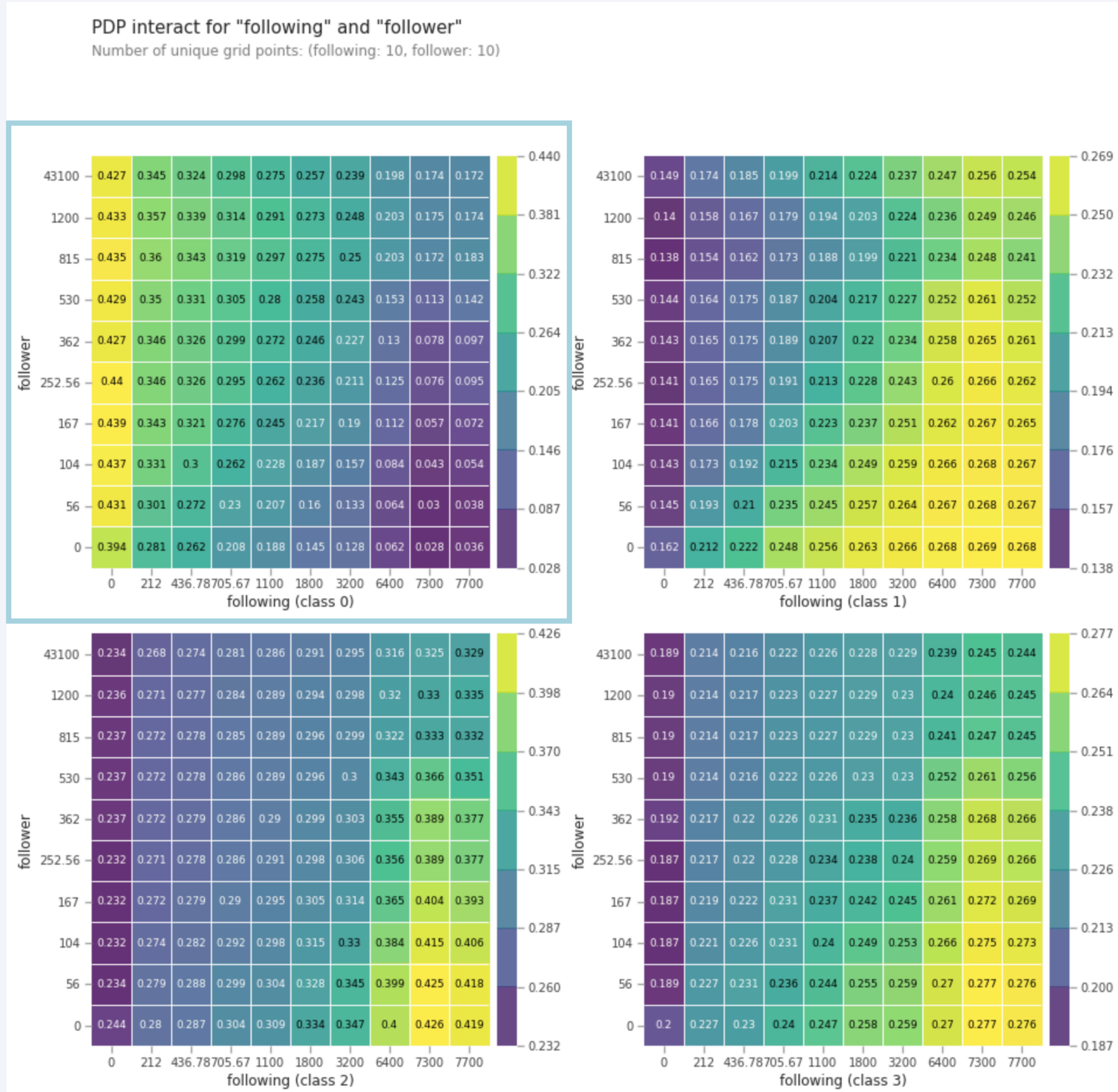

post
cosine_similarity
caption_length
post_interval
comment_rate
biography_lengh
following
followers_keywords
promotional_keywords
like_rate
non_image_percentage
follower
link
caption_zero
like/post
location_tag
comment/post
following>follower
0 1 2 3 4 5 6
mean(|SHAP value|) (average impact on model output magnitude)
Class 1
Class 3
Class 2
Class 0

▪ 모델 해석

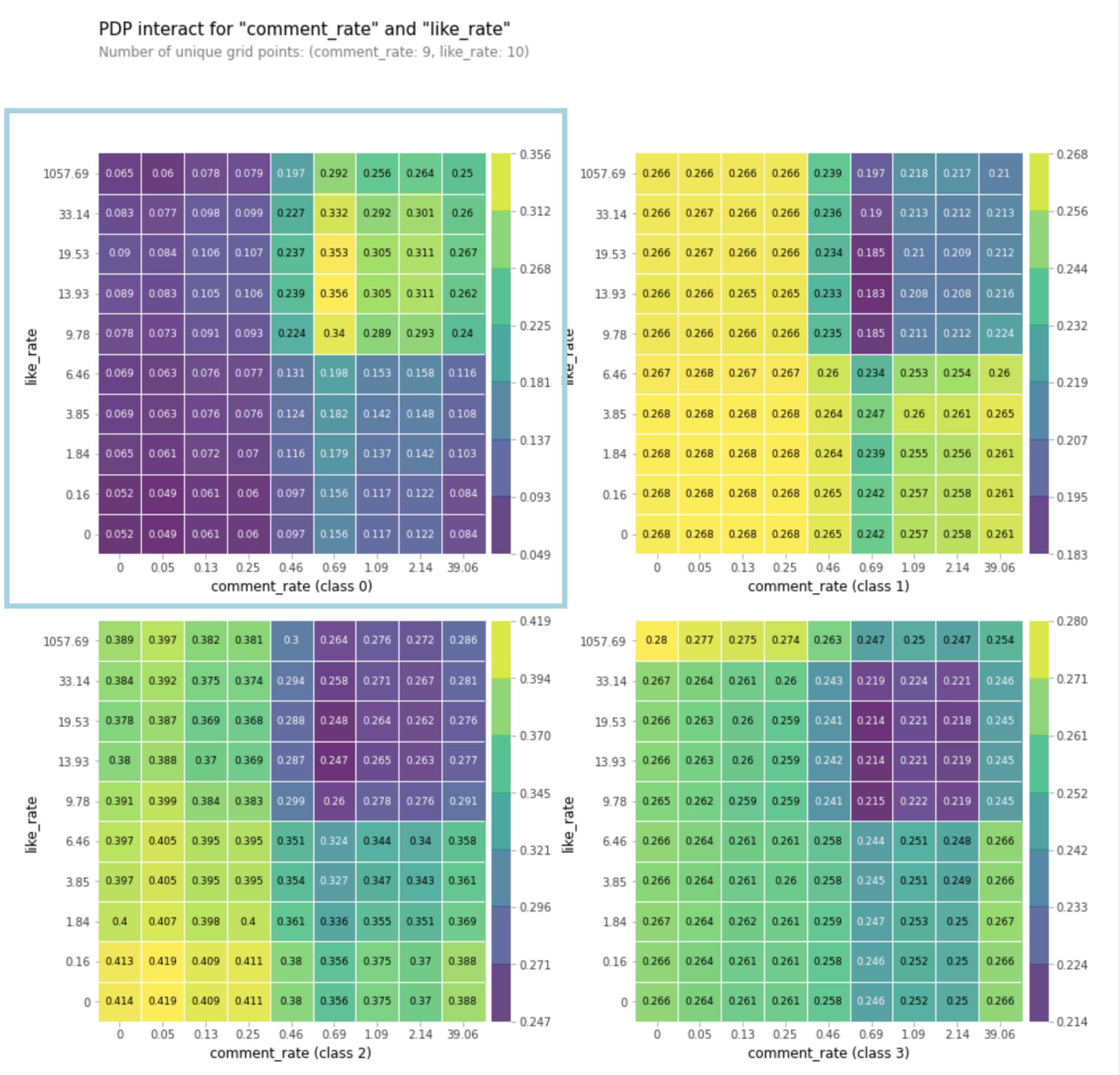


distribution of data points
post (class 0)
distribution of data points
post (class 1)
distribution of data points
post (class 2)
distribution of data points
post (class 3)

▪ 모델 해석


PDP interact for "following" and "follower"
Number of unique grid points: (following: 10, follower: 10)
follower
following (class 0)
following (class 1)
following (class 2)
following (class 3)

▪ 모델 해석


PDP interact for "comment_rate" and "like_rate"
Number of unique grid points: (comment_rate: 9, like_rate: 10)
comment_rate (class 0)
comment_rate (class 1)
comment_rate (class 2)
comment_rate (class 3)
like_rate

06

결론

- 결론

- 타겟값은 네가지로 분류가 되나 진짜 계정을 제외한 나머지 계정들은 비슷한 분포를 보인다.

  〉추후 연구에서는 진짜 계정/가짜 계정 두가지 클래스로 타겟값을 선정해야 할 것

- '낯선 사람'과 '랜선친구'를 구분하는데 도움이 될 것

감사합니다.