PART I
의사결정나무

Requirement 1-1

```python
# TODO: Requirement 1-1. MAKE vip column

cursor.execute('''
    ALTER TABLE users ADD vip TINYINT(1) default 0;''')



filepath = './dataset/vip_list.csv'
with open(filepath, 'r', encoding='utf-8') as csv_data:
    next(csv_data, None)  # skip the headers
    row_count = 0
    for row in csv_data:
        # Change the null data
        row = row.strip().split(',')
        for i in range(len(row)):
            row[i] = row[i].replace('"', '')
        for idx, data in enumerate(row):
            if data == '':
                row[idx] = None
        row = tuple(row)
        cursor.execute(f'UPDATE users SET vip = 1 WHERE user_id = {row[0]};')
```

1. users 테이블에 vip 컬럼 추가

2. vip_list.csv 파일을 읽어 해당 csv 파일 속 존재하는 user_id를 찾고 이에 대응되는 users 테이블 속의 vip 컬럼에 0과 1을 통해 vip 여부를 구분

Requirement 1-2

```
# TODO: Requirement 1-2. WRITE MYSQL QUERY AND EXECUTE. SAVE to .csv file

cursor.execute('''
SELECT users.user_id,
(SELECT users.vip) AS vip_list,
(SELECT users.user_yelping_since_year) AS user_yelping_since_year,
(SELECT COUNT(*) FROM reviews AS r WHERE users.user_id = r.user_id) AS user_review_counts,
(SELECT users.user_fans) AS user_fans,
(SELECT users.user_votes_funny) AS user_votes_funny,
(SELECT users.user_votes_useful) AS user_votes_useful,
(SELECT users.user_votes_cool) AS user_votes_cool,
(SELECT users.user_average_stars) AS user_average_stars,
(SELECT SUM(likes) FROM tips AS t WHERE users.user_id = t.user_id) AS user_tip_counts
 FROM users;''')


df1 = pd.DataFrame(cursor.fetchall())
df1.columns = cursor.column_names
df1.set_index('user_id', inplace = True)
df1.fillna(0, inplace= True)
df1.to_csv('DMA_project2_team%02d_part1.csv' % team)
```

1. nested query를 이용하여 requirement에 해당하는 내용을 SELECT 문으로 받아온다.

2. dataframe을 이용해 저장하고 인덱스 및 컬럼명을 재지정하며 결측치 (특히 _counts)를 0으로 바꿔준다.

Requirement 1-3

```
# TODO: Requirement 1-3. MAKE AND SAVE DECISION TREE
# gini file name: DMA_project2_team##_part1_gini.pdf
# entropy file name: DMA_project2_team##_part1_entropy.pdf

features_col = [i for i in df1.columns if i != 'vip_list']
features = df1[features_col]

#Decision Tree(gini)
DT_gini = tree.DecisionTreeClassifier(criterion='gini', min_samples_leaf=8, max_depth=4)
X = df1[features_col]
Y = df1['vip_list']
DT_gini.fit(X,Y)

#Visualization Decision Tree(gini)
graph = tree.export_graphviz(DT_gini, out_file=None, feature_names=features_col, class_names=['normal', 'BEST'])
graph = graphviz.Source(graph)
graph.render('DMA_project2_team03_part1_gini', view=False)
```

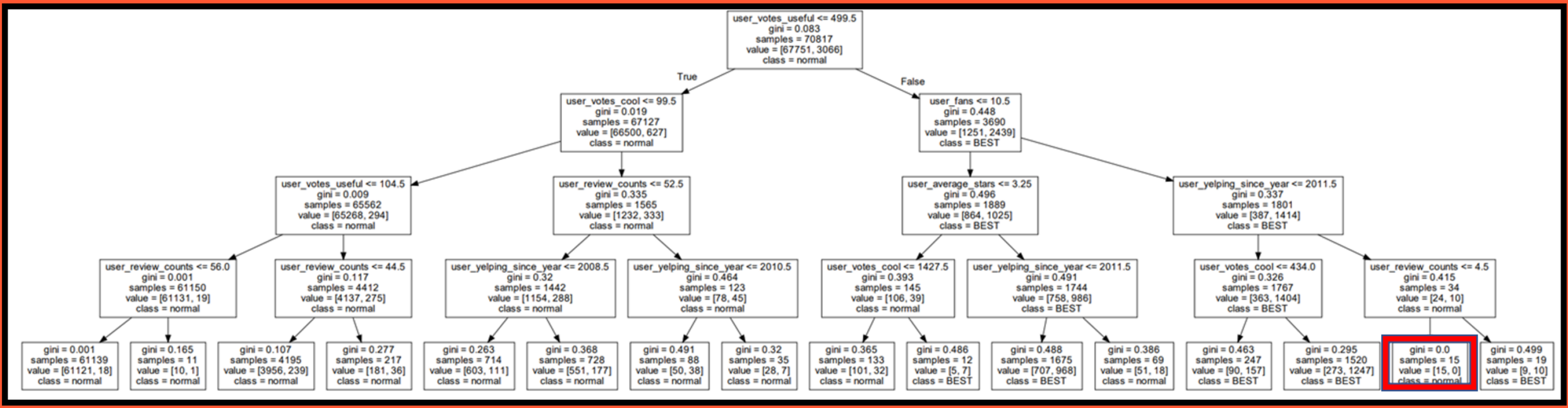
1. 데이터로부터 입력변수와 출력변수를 나눈다.
2. 조건에 맞게 Decision Tree 생성하고 입력변수와 출력변수를 통해 모델을 학습시킨다.
3. Entropy Decision Tree의 경우도 같은 방식으로 만들 수 있다.

의사결정나무 생성 - Gini

- 사용 라이브러리: sklearn.tree.DecisionTreeClassifier
- Node impurity criterion: gini / entropy
- 결과 파일명: node impurity criterion에 따라 DMA_project2_team##_part1_gini.pdf, DMA_project2_team##_part1_entropy.pdf 로 구분함.
- 분석 목표: VIP 선정 기준
- min_samples_leaf: 8
- max_depth: 4
- feature names: user_yelping_since_year, user_review_count, user_fans, user_votes_funny, user_votes_useful, user_votes_cool, user_average_stars, user_tip_counts
- class names: normal, BEST

1. 분류의 처음을 user_votes_useful을 이용해 시작한다.

2. 4번의 분류 끝에 완전히 분류되어 혼잡도가 0인 경우는 한 가지만 존재한다.

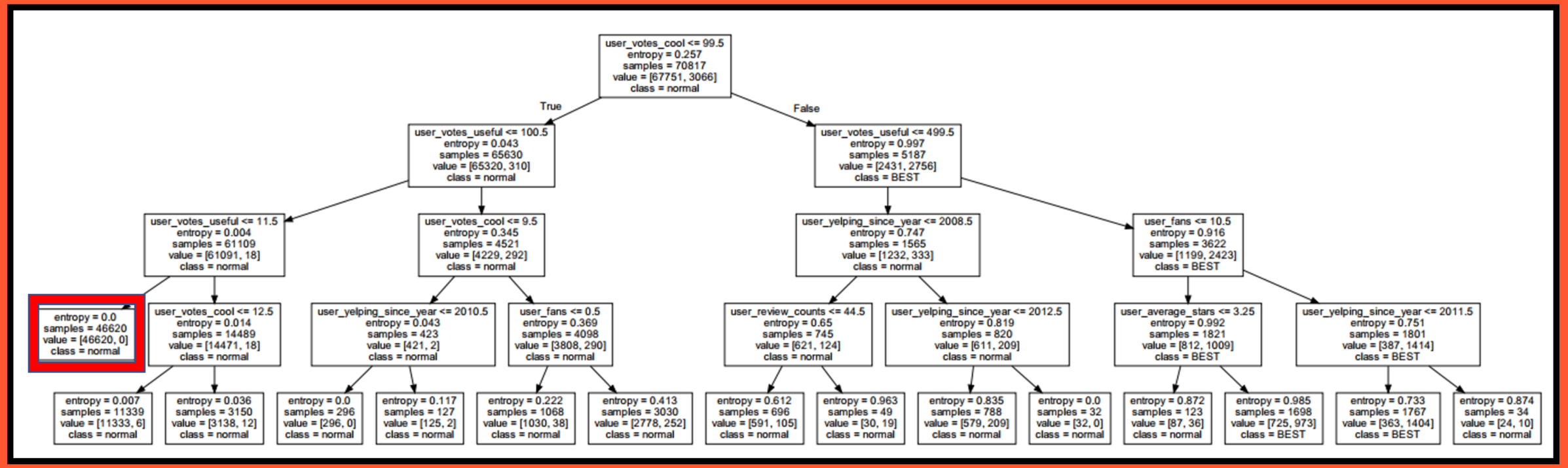


의사결정나무 생성 - Entropy

- 사용 라이브러리: sklearn.tree.DecisionTreeClassifier
- Node impurity criterion: gini / entropy
- 결과 파일명: node impurity criterion에 따라 DMA_project2_team##_part1_gini.pdf, DMA_project2_team##_part1_entropy.pdf 로 구분함.
- 분석 목표: VIP 선정 기준
- min_samples_leaf: 8
- max_depth: 4
- feature names: user_yelping_since_year, user_review_count, user_fans, user_votes_funny, user_votes_useful, user_votes_cool, user_average_stars, user_tip_counts
- class names: normal, BEST

1. 분류의 처음을 user_votes_cool을 이용해 시작한다.

2. Gini와는 달리 3번의 분류만으로 모두 정확하게 분류되어 혼잡도가 0인 부분이 존재한다.



Model 1

feature_importances?	DT_gini	DT_entropy
user_yelping_since_yea	0.0114628842766411	0.00475044179147732
user_review_counts	0.00478617993106071	0.000873058501065395
user_fans	0.0324858644639734	0.0148462976790971
user_votes_funny	0.0	0.0
user_votes_useful	0.899363606865762	0.143706181908005
user_votes_cool	0.0448548640225474	0.833664078843923
user_average_stars	0.00704660044001562	0.00215994127643265
user_tip_counts	0.0	0.0

user_votes_funny,
user_review_counts,
user_tip_counts
제거

Input feature(5) :
user_yelping_since_year, user_fans,
user_votes_useful, user_votes_cool,
user_average_stars
Node impurity criterion : gini
Min_samples_leaf : 8
Max_depth : 4

Model 2

```
param = {'criterion' : ['gini', 'entropy'], 'min_samples_leaf' : [6,8,10,12], 'max_depth' : [3,4,5,6]}
model = tree.DecisionTreeClassifier()
grid_dtree = GridSearchCV(model, param_grid=param, cv=5, refit=True, return_train_score=True)
grid_dtree.fit(X, Y)
print("\nnew model2에서 최고로 선정된 parameters : {0}\n".format(grid_dtree.best_params_))
```

```
new model2에서 최고로 선정된 parameters : {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 10}
```

Input feature(8) :
user_yelping_since_year,
user_review_counts, user_fans,
user_votes_funny, user_votes_useful,
user_votes_cool, user_average_stars,
user_tip_counts
Node impurity criterion : entropy
Min_samples_leaf : 10
Max_depth : 5

교차검증을 이용한 평가

```python
# origin
score = cross_val_score(DT_gini, X, Y, scoring='accuracy', cv=5)
print(f"origin gini model 교차검증 평균 점수 : {np.mean(score)}")

# new 1
score = cross_val_score(DT_new_gini, X_new, Y_new, scoring='accuracy', cv=5)
print(f"new model 1 (feature 3개 제거) 교차검증 평균 점수 : {np.mean(score)}")

# new 2
new_DT = grid_dtree.best_estimator_
score = cross_val_score(new_DT, X, Y,scoring='accuracy', cv=5)
print(f"new model 2 (samples_leaf와 depth 수정) 교차검증 평균 점수 : {np.mean(score)}")
```

```
origin gini model 교차검증 평균 점수 : 0.9743423091335652
new model 1 (feature 3개 제거) 교차검증 평균 점수 : 0.9745541166995004
new model 2 (samples_leaf와 depth 수정) 교차검증 평균 점수 : 0.974709449027354
```

- **GridSearchCV를 이용한 최적 parameter 선정**
- **Feature 제거를 통해 의미 없는 입력 제거 및 과적합↓**

Input feature(8) :
user_yelping_since_year,
user_review_counts, user_fans,
user_votes_funny, user_votes_useful,
user_votes_cool, user_average_stars,
user_tip_counts
Node impurity criterion : gini
Min_samples_leaf : 8
Max_depth : 4

Input feature(5) :
user_yelping_since_year,
user_fans, user_votes_useful,
user_votes_cool,
user_average_stars
Node impurity criterion : gini
Min_samples_leaf : 8
Max_depth : 4

Input feature(8) :
user_yelping_since_year,
user_review_counts, user_fans,
user_votes_funny, user_votes_useful,
user_votes_cool, user_average_stars,
user_tip_counts
Node impurity criterion : entropy
Min_samples_leaf : 10
Max_depth : 5