Spotify 데이터를 활용한 장르 예측

BITAmin 6기 복습 프로젝트

5조 김지윤 김현우 이수진 이현진

2021 / 01 / 30

발표 순서

01

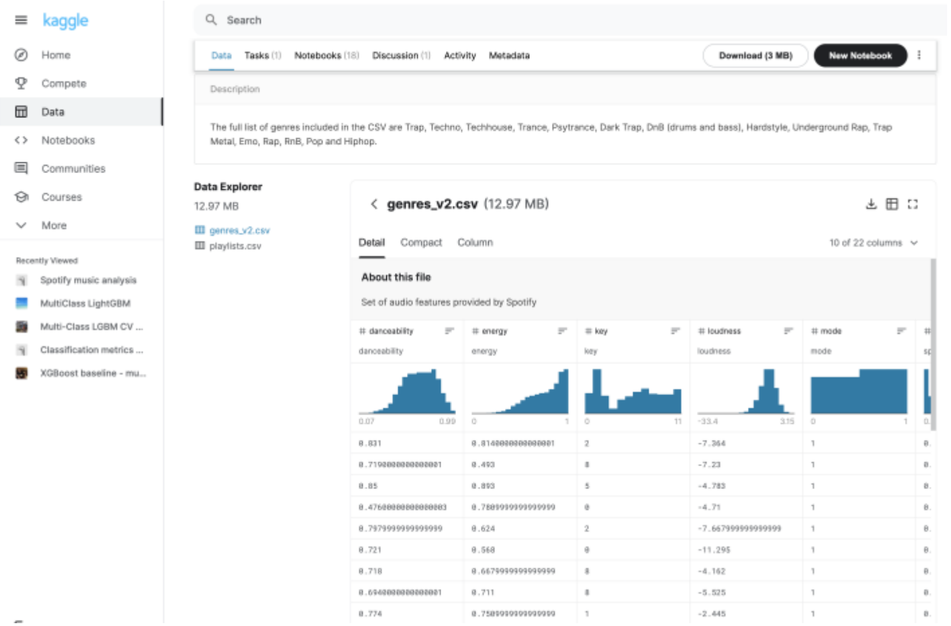
데이터
설명

02

EDA 및
전처리

03

모델
적용



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
danceabili	energy	key	loudness	mode	speechine	acousticne	instrumen	liveness	valence	tempo	type	id	url	track_href	analysis_ur	duration_r	time_signa	genre	song_nam	Unnamed:	title
0.831	0.814	2	-7.364	1	0.42	0.0598	0.0134	0.0556	0.389	156.985	audio_feat	2Vc6NJ9P\	spotify:trac	https://api	https://api	124539	4	Dark Trap	Mercury: Retrograde		
0.719	0.493	8	-7.23	1	0.0794	0.401	0	0.118	0.124	115.08	audio_feat	7pgJBLVz5	spotify:trac	https://api	https://api	224427	4	Dark Trap	Pathology		
0.85	0.893	5	-4.783	1	0.0623	0.0138	4.14E-06	0.372	0.0391	218.05	audio_feat	0vSWgAlfp	spotify:trac	https://api	https://api	98821	4	Dark Trap	Symbiote		
0.476	0.781	0	-4.71	1	0.103	0.0237	0	0.114	0.175	186.948	audio_feat	0VSXnJqQ	spotify:trac	https://api	https://api	123661	3	Dark Trap	ProductOfDrugs (Prod. The Viru		
0.798	0.624	2	-7.668	1	0.293	0.217	0	0.166	0.591	147.988	audio_feat	4jCeguq9r	spotify:trac	https://api	https://api	123298	4	Dark Trap	Venom		
0.721	0.568	0	-11.295	1	0.414	0.0452	0.212	0.128	0.109	144.915	audio_feat	6fsypiJHyV	spotify:trac	https://api	https://api	112511	4	Dark Trap	Gatteka		
0.718	0.668	8	-4.162	1	0.137	0.0254	0.0078	0.124	0.038	130.826	audio_feat	0XfQbq7D	spotify:trac	https://api	https://api	77584	4	Dark Trap	kamikaze (+ pulse)		
0.694	0.711	8	-5.525	1	0.221	0.0397	0	0.112	0.283	138.049	audio_feat	0LLeuNBW	spotify:trac	https://api	https://api	127524	3	Dark Trap	T.R.U. (Totally Rotten Undergrou		
0.774	0.751	1	-2.445	1	0.198	0.0614	0	0.0728	0.189	219.96	audio_feat	37gqBnUA	spotify:trac	https://api	https://api	140326	4	Dark Trap	I Put My Dick in Your Mental		
0.893	0.907	11	-10.406	1	0.367	0.152	0.0311	0.558	0.302	199.942	audio_feat	2ggqfj97q	spotify:trac	https://api	https://api	121979	4	Dark Trap	Andromeda		
0.864	0.365	8	-10.219	1	0.0655	0.187	0	0.116	0.0478	189.938	audio_feat	7EL7ifncKZ	spotify:trac	https://api	https://api	101172	4	Dark Trap	BRAINFOOD		
0.736	0.932	1	-3.726	1	0.271	0.146	0.0025	0.182	0.18	124.514	audio_feat	0QIF3I617I	spotify:trac	https://api	https://api	115775	4	Dark Trap	Troll Under the Bridge		
0.825	0.761	8	-5.389	1	0.104	0.0111	0.00359	0.334	0.161	149.97	audio_feat	5o7ZDvfO	spotify:trac	https://api	https://api	163371	4	Dark Trap	1000 Rounds		
0.767	0.576	10	-9.683	0	0.256	0.145	2.61E-06	0.0968	0.187	139.99	audio_feat	1umsRbM	spotify:trac	https://api	https://api	96062	4	Dark Trap	Sacrifice		
0.765	0.726	5	-5.58	1	0.191	0.0077	0	0.619	0.27	128.014	audio_feat	4SKqOHK\	spotify:trac	https://api	https://api	135079	4	Dark Trap	Backpack		
0.617	0.541	6	-4.113	1	0.78	0.125	0	0.369	0.43	159.996	audio_feat	4Ag89Y7q	spotify:trac	https://api	https://api	107999	4	Dark Trap	D(R)Own		
0.755	0.298	1	-15.032	1	0.0915	0.154	0.329	0.101	0.0372	199.958	audio_feat	28xkYPSP(	spotify:trac	https://api	https://api	123054	4	Dark Trap	Okay,ButThisIsTheLastTime		
0.814	0.575	11	-9.635	1	0.0635	0.172	0.000291	0.109	0.288	120.004	audio_feat	3uE1swbcf	spotify:trac	https://api	https://api	192833	4	Dark Trap	TakingOutTheTrash		
0.812	0.813	10	-5.583	0	0.0984	0.0987	0.00015	0.0758	0.348	128.066	audio_feat	3KJrwOuq	spotify:trac	https://api	https://api	180880	4	Dark Trap	Io sono qui		
0.849	0.648	7	-6.188	1	0.0832	0.0725	0.00592	0.0984	0.196	212.15	audio_feat	4irYeuAi87	spotify:trac	https://api	https://api	111020	4	Dark Trap	Paris		

캐글에서 찾은 Spotify 데이터
노래의 댄스 가능성, 에너지, 높낮이 등을 보여주는 column과 장르가 나타난 csv 데이터

	T	U	V	W	X
	This D.J.				
	H20 (feat. Pharoahe Monch, Rakaa Iriscience, & Indj				
	Daddy's Girl				
	'94				
	Born 2 Live				
	Can't Hold On				
	Still D.R.E.				
e		0	Dirtybird Players		
e		1	Tech House Movement		
e		2	tech house		
e		3	Tech House Bangerz		
e		4	tech house		
e		5	blanc | Tech House		
e		6	Toolroom Tech House		
e		7	Tech house		
e		8	Dirtybird//Techhouse by Mack¥		

데이터를 살펴본 결과, 21,526행과 21,527행을 경계로

T열(song_name)이 V열(title)로 이동

→ 21,526행 이전 V열은 NaN값

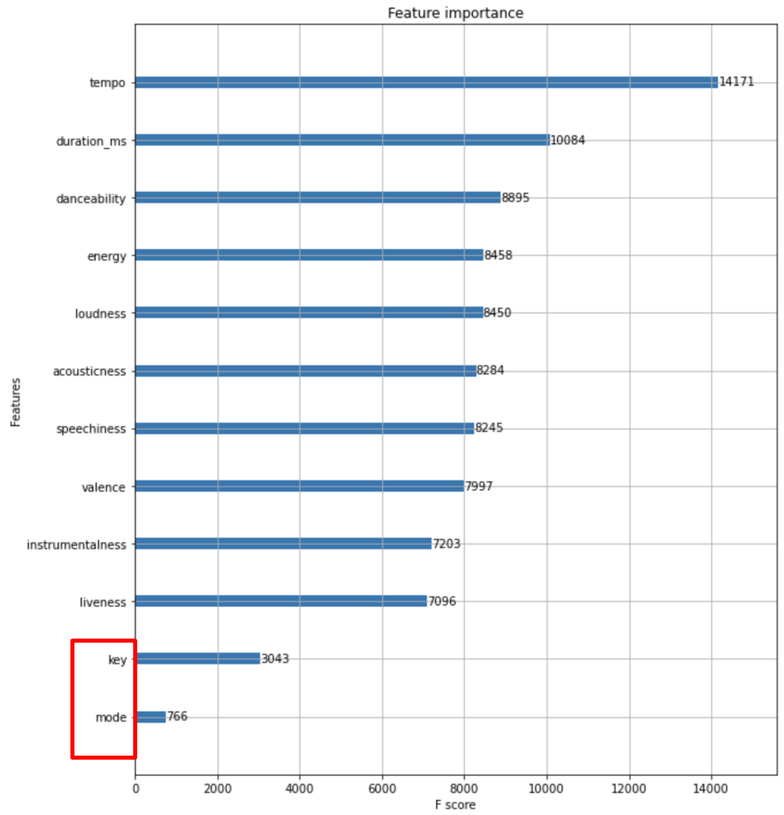
→ 21,526행 이후 T열은 NaN값

```
song_name=df['song_name']
song_name.dropna(inplace=True)
title=df['title']
title.dropna(inplace=True)
```

```
idx_df=pd.concat([song_name, title], axis=0)
idx_df=pd.DataFrame(idx_df)
df.drop(['song_name', 'title'], axis=1, inplace=True)
df['title']=idx_df
df=df.set_index(['title'])
```

song_name과 title 열을 합치기

1. song_name 열과 title 열을 concat을 통해 병합
2. 병합한 피처를 idx_df라는 변수에 담아두고,
3. 기존 데이터에서 song_name과 title 열을 제거한 뒤,
4. 기존 데이터에 title 열을 생성해 idx_df를 추가



피처 중요도 확인

피처 중요도를 확인해 본 결과, key와 mode의 중요도가 매우 낮게 나타남

→ key와 mode 피처 drop

```
fig, ax = plt.subplots(figsize = (10, 12))
plot_importance(xgb_model, ax = ax)
```

```
data.drop(['key', 'mode'], axis = 1, inplace = True)
```

범주형이며, 데이터 분석에 영향 미치지 않는 피처 제거

type	id	uri	track_href	analysis_u
audio_feat	2Vc6NJ9P	spotify:tra	https://ap	https://ap
audio_feat	7pgJBLVz	spotify:tra	https://ap	https://ap
audio_feat	0vSWgAlf	spotify:tra	https://ap	https://ap
audio_feat	0VSXnJqQ	spotify:tra	https://ap	https://ap

T	U	V
song_name	Unnamed: 0	title
Mercury: Retrc		

```
df=pd.read_csv('./genres_v2.csv')
df.drop(['Unnamed: 0', 'type', 'id', 'time_signature', 'track_href','analysis_url','uri'], axis=1, inplace=True)
nalist=df[(df['song_name'].isnull()==True)&(df['title'].isnull()==True)].index
df.drop(nalist, axis=0, inplace=True)
```

```
encoder=LabelEncoder()
encoder.fit(df['genre'])
labels=encoder.transform(df['genre'])
```

```
df['genre']=labels
```

```
df['genre'].value_counts()
```

```
7     5875
0     4578
2     3022
13    2999
14    2987
11    2975
8     2966
10    2961
12    2956
9     2936
5     2099
6     1956
4     1848
1     1680
3      461
Name: genre, dtype: int64
```

인코딩

이후 코딩의 용이함을 위해

LabelEncoder를 통해 genre를 인코딩

```
print('인코딩 클래스 : ',encoder.classes_)
```

```
인코딩 클래스 :  ['Dark Trap' 'Emo' 'Hiphop' 'Pop' 'Rap' 'RnB' 'Trap Metal'
 'Underground Rap' 'dnb' 'hardstyle' 'psytrance' 'techhouse' 'techno'
 'trance' 'trap']
```

```
get_clf_eval(y_test, preds, pred_proba)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-28-9bb98b630742> in <module>
----> 1 get_clf_eval(y_test, preds, pred_proba)

~₩Anaconda3₩lib₩site-packages₩sklearn₩metrics₩_classification.py in _check_
set_wise_labels(y_true, y_pred, average, labels, pos_label)
   1261             if y_type == 'multiclass':
   1262                 average_options.remove('samples')
-> 1263             raise ValueError("Target is %s but average='binary'. P
lease "
   1264                              "choose another average setting, one of %
r."
   1265                              % (y_type, average_options))

ValueError: Target is multiclass but average='binary'. Please choose another averag
e setting, one of [None, 'micro', 'macro', 'weighted'].
```

랜덤 포레스트 정확도: 0.6585
Voting 문듀기 정확도: 0.3076
Decision Tree 예즉 정확도: 0.4493

문제점 발생

문제점 1)

y값은 총 14개의 값(multiclass)이 존재

모델은 binary로 target 인식

문제점 2)

레이블 인코딩 후 모델 돌렸을 때 너무 낮은 정확도

Label Encoding 말고 One-hot Encoding으로 인코딩 해야겠다고 판단

Softmax 형태의 One-hot Encoding 결과를
Y값에 넣는 방식으로는
objective = 'multiclass'로 설정했음에도
계속 오류

다른 방법

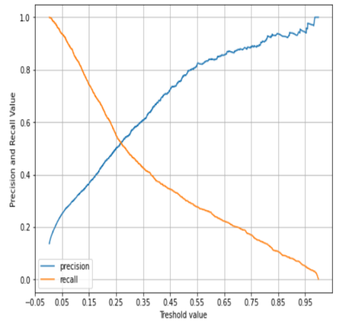
y 값을 binary로
입력 받는 모델

y matrix의 각 열 별로 해당 음악이
Dark Trap인지의 여부, Emo인지의 여부,
Hiphop인지의 여부, 등등 …
이렇게 순서대로 예측하는 방안을 선택

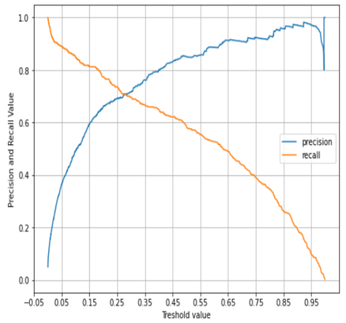
for문을 통해 반복

```
for i in range(15):
    xgb_wrapper=XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3, eta=0.1,objective='binary:logistic',
                              eval_metric='logloss')
    xgb_wrapper.fit(X_train, y_train[i])
    w_preds=xgb_wrapper.predict(X_test)
    w_pred_proba=xgb_wrapper.predict_proba(X_test)[:,1]
    get_clf_eval(y_test[i], w_preds, w_pred_proba)
    print(precision_recall_curve_plot(y_test[i], w_pred_proba))
```

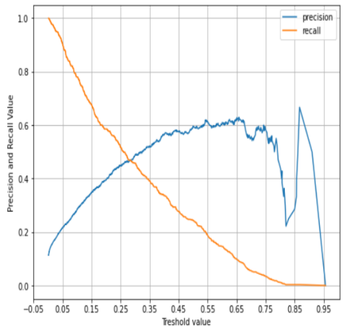
정확도:0.9100, 정밀도 0.7772, 재현율 :0.3077, F1:0.4409, AUC:0.889506



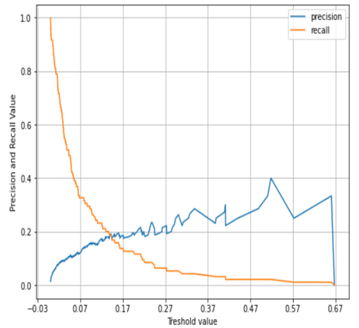
정확도:0.9790, 정밀도 0.8498, 재현율 :0.5806, F1:0.6899, AUC:0.967270



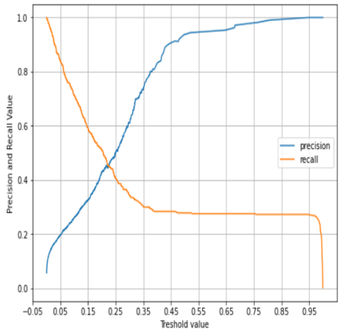
정확도:0.9336, 정밀도 0.5960, 재현율 :0.2443, F1:0.3465, AUC:0.906729



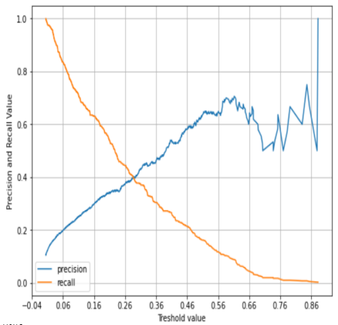
정확도:0.9885, 정밀도 0.3333, 재현율 :0.0211, F1:0.0396, AUC:0.90349



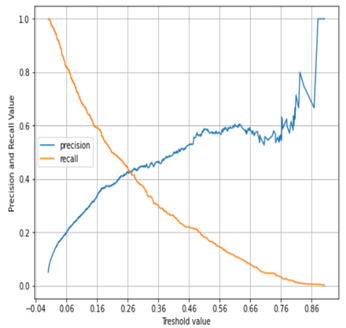
정확도:0.9677, 정밀도 0.9450, 재현율 :0.2784, F1:0.4301, AUC:0.928412



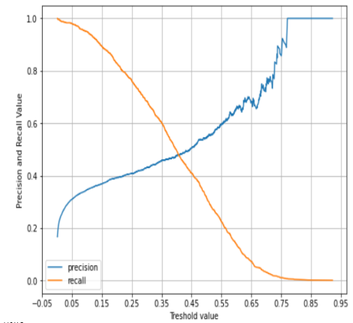
정확도:0.9548, 정밀도 0.6182, 재현율 :0.1667, F1:0.2625, AUC:0.918819



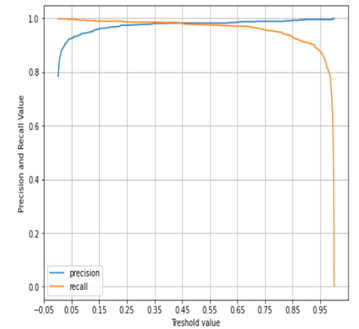
정확도:0.9564, 정밀도 0.5659, 재현율 :0.1891, F1:0.2835, AUC:0.918954



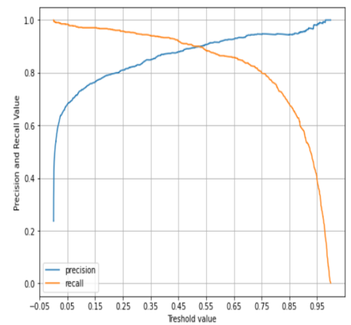
정확도:0.8739, 정밀도 0.5474, 재현율 :0.3171, F1:0.4016, AUC:0.891912



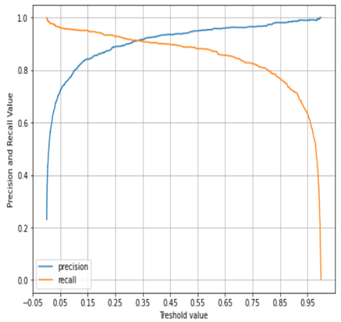
정확도:0.9972, 정밀도 0.9828, 재현율 :0.9761, F1:0.9795, AUC:0.999788



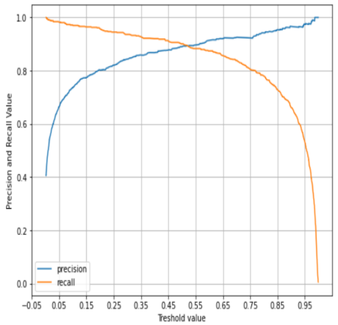
정확도:0.9853, 정밀도 0.8950, 재현율 :0.9038, F1:0.8994, AUC:0.995737



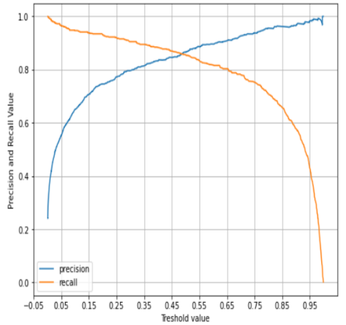
정확도:0.9878, 정밀도 0.9417, 재현율 :0.8803, F1:0.9142, AUC:0.995348



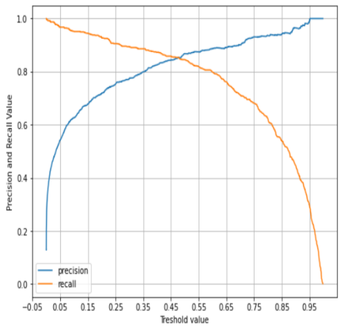
정확도:0.9861, 정밀도 0.8912, 재현율 :0.9007, F1:0.8959, AUC:0.996272



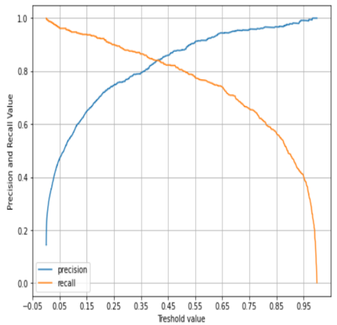
정확도:0.9810, 정밀도 0.8656, 재현율 :0.8552, F1:0.8604, AUC:0.992732



정확도:0.9793, 정밀도 0.8677, 재현율 :0.8436, F1:0.8555, AUC:0.990981

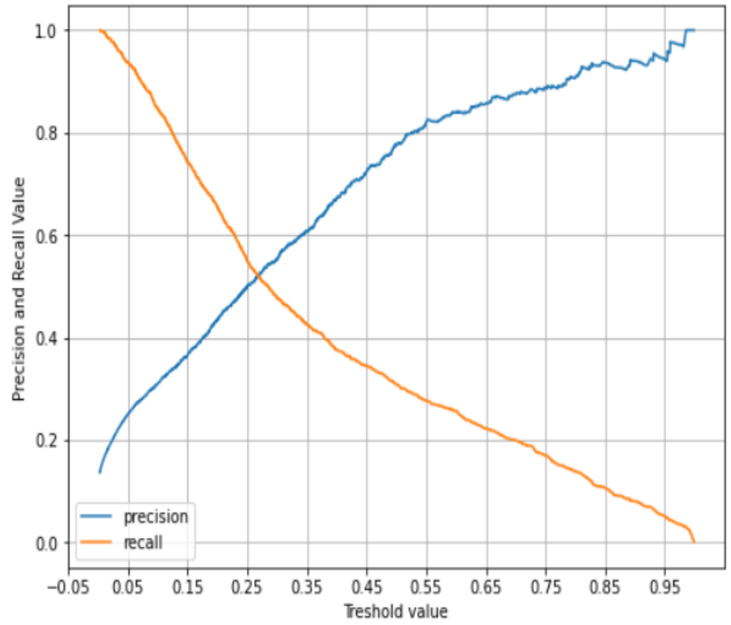


정확도:0.9794, 정밀도 0.8795, 재현율 :0.8056, F1:0.8410, AUC:0.989942



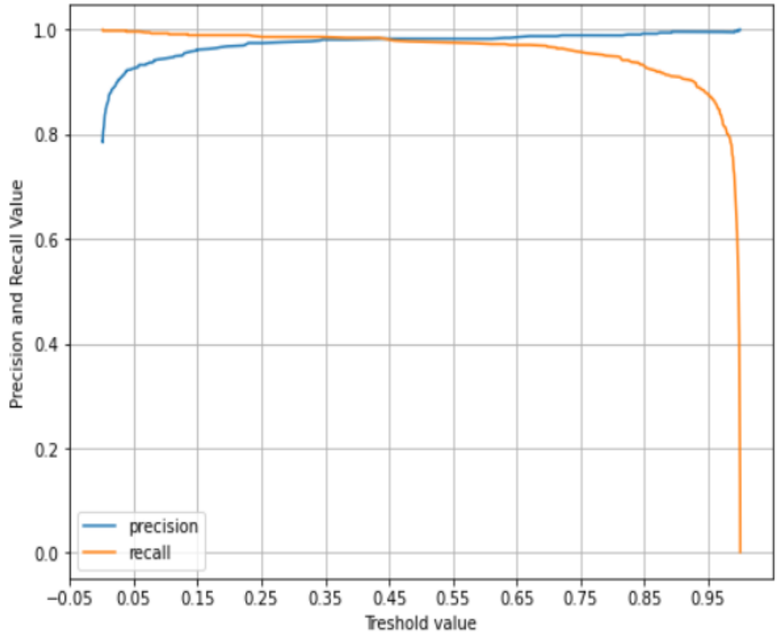
공통점 매우 높은 정확도

정확도:0.9100, 정밀도 0.7772, 재현율 :0.3077, F1:0.4409, AUC:0.889506

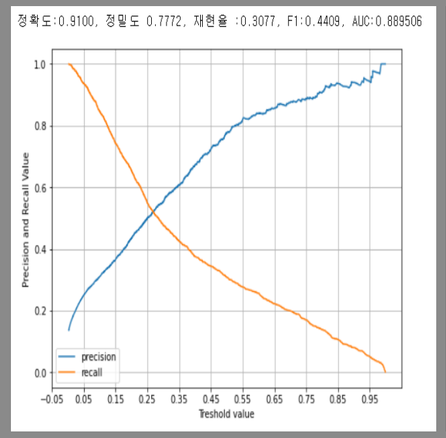


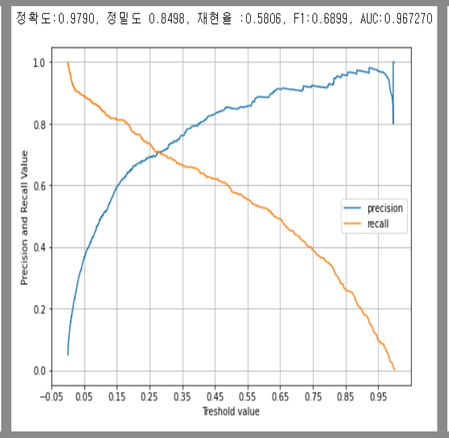
0번 장르 재현율이 현저히 낮은 장르

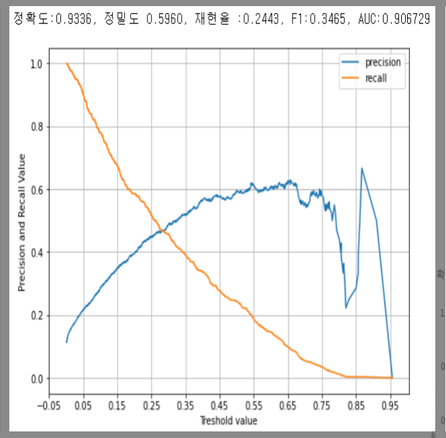
정확도:0.9972, 정밀도 0.9828, 재현율 :0.9761, F1:0.9795, AUC:0.999788

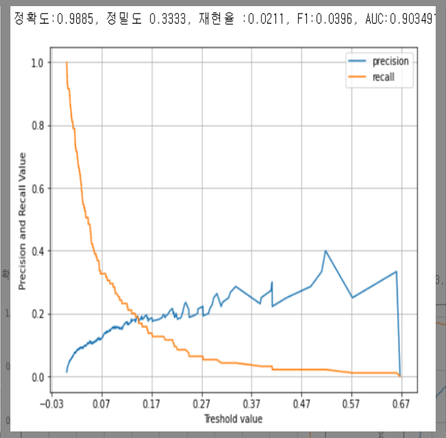


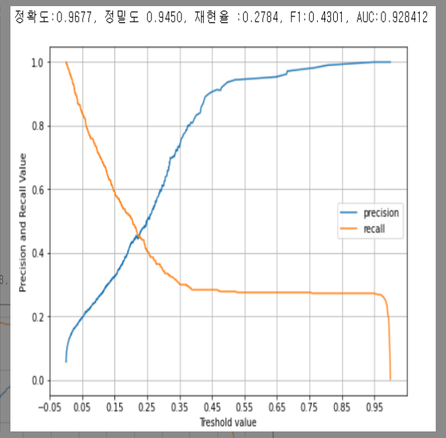
8번 장르 예측 성능이 매우 좋은 장르

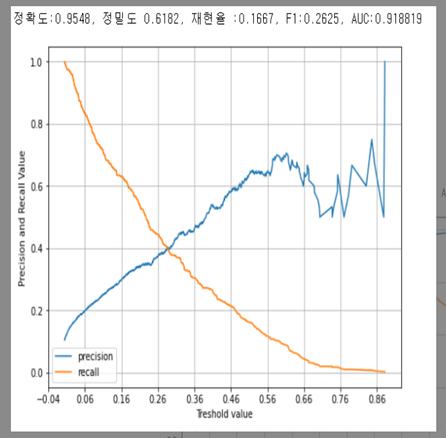


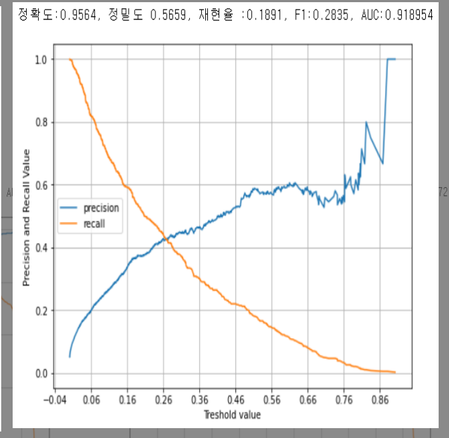


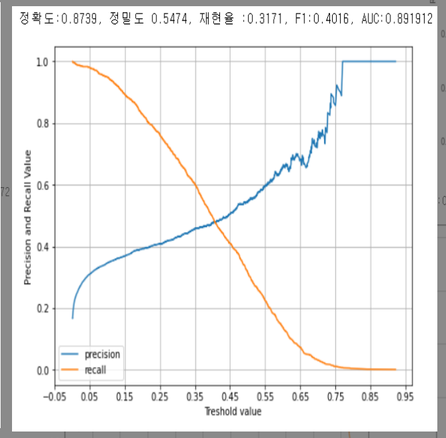






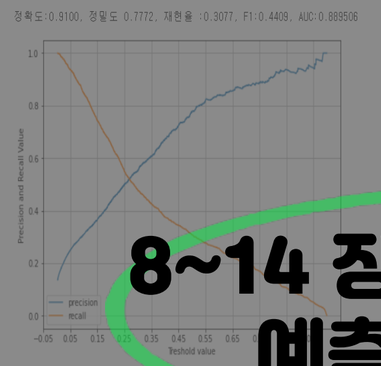


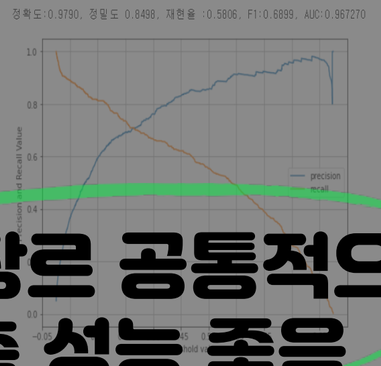


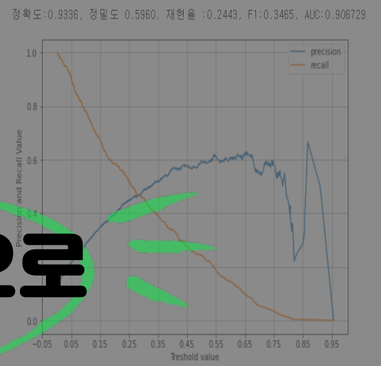


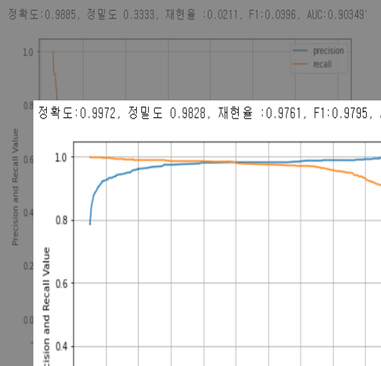
0~7 장르 공통적으로
재현율이 너무 낮음

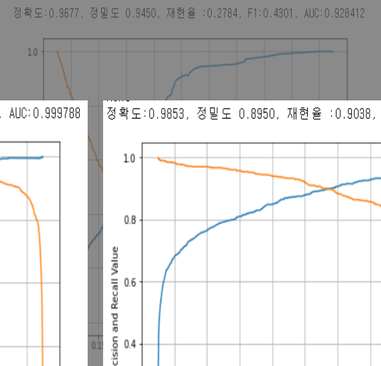
8~14 장르 공통적으로 예측 성능 좋음

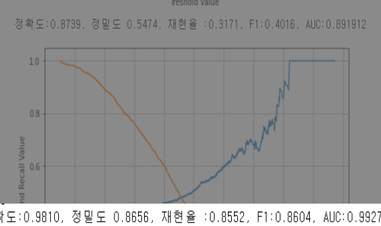


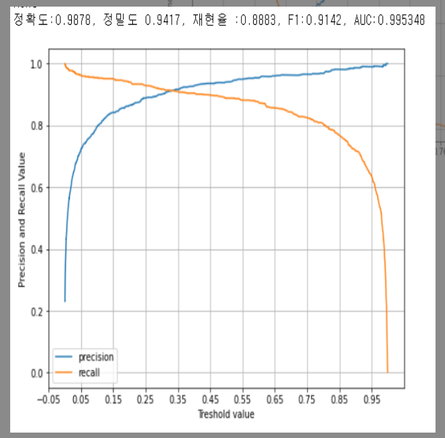


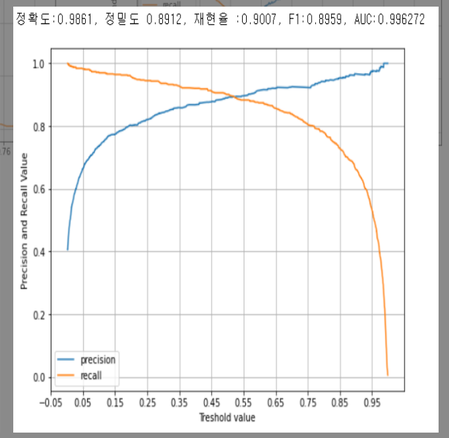


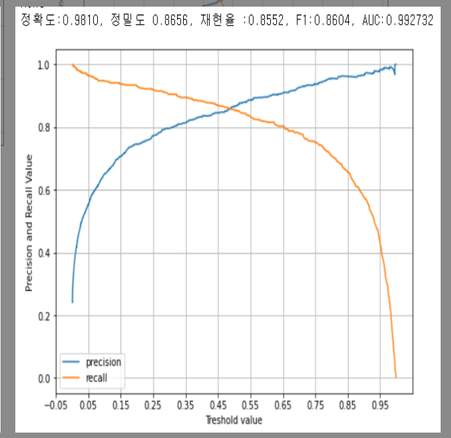


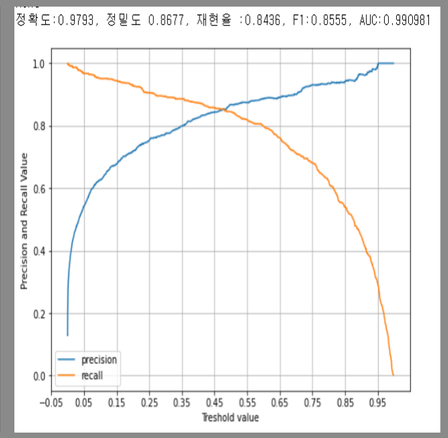


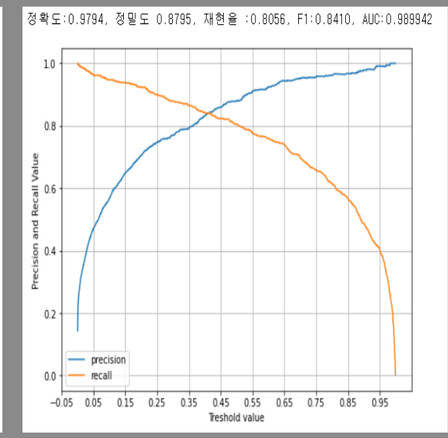












```
df[df['genre']==0]=7
df[df['genre']==1]=7
df[df['genre']==2]=7
df[df['genre']==3]=7
df[df['genre']==4]=7
df[df['genre']==5]=7
df[df['genre']==6]=7
```

예측 성능이 좋지 않은 0~7번 장르를 '기타 장르'로 통일

→ 'Dark Trap', 'Emo', 'Hiphop', 'Pop', 'Rap', 'RnB', 'Trap Metal', 'Underground Rap' 장르가 모두 'Other items'가 된 것

```
df['genre'].value_counts()
```

```
7     21519
13     2999
14     2987
11     2975
8      2966
10     2961
12     2956
9      2936
Name: genre, dtype: int64
```

새롭게 지정한 장르의 IR을 살펴본 결과

기타 장르로 합친 7번 장르로 구분된 음악이 매우 많음

→ 데이터 불균형 처리 필요

SMOTE를 통한 데이터 불균형 처리

```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=23)
smote=SMOTE(random_state=0)
X_train_over, y_train_over=smote.fit_sample(X_train, y_train)
```

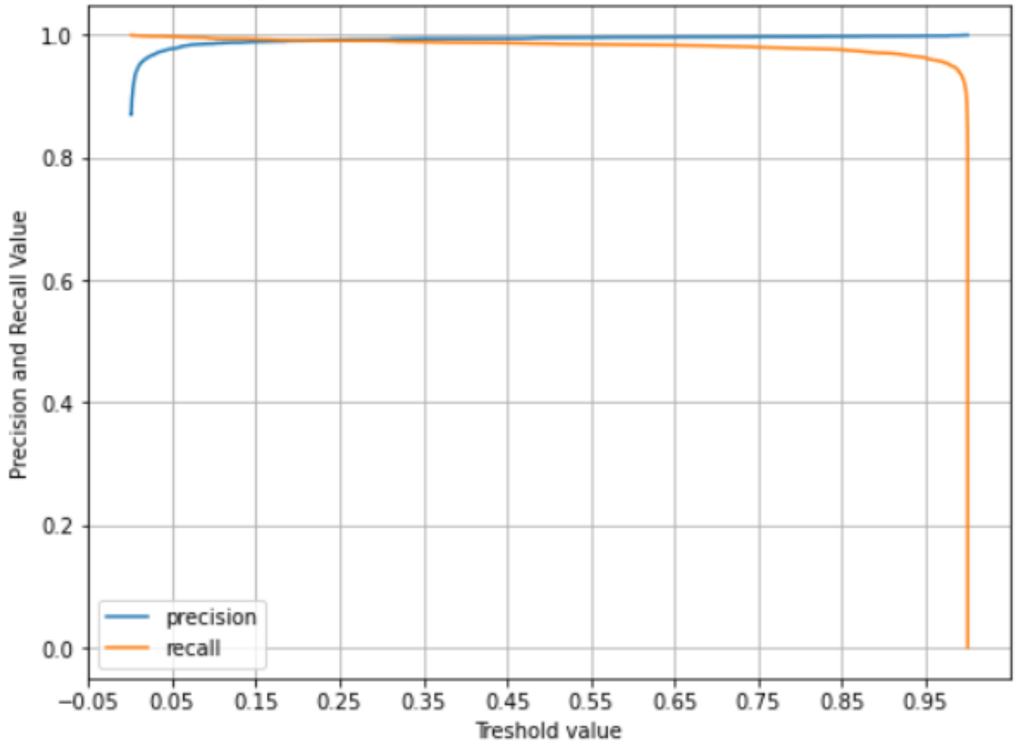
```
y_train_over=pd.get_dummies(y_train_over)
y_test=pd.get_dummies(y_test)
```

모델 재실행

```
for i in range(15):
    xgb_wrapper=XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3, eta=0.1,objective='binary:logistic',
                              eval_metric='logloss')
    xgb_wrapper.fit(X_train, y_train[i])
    w_preds=xgb_wrapper.predict(X_test)
    w_pred_proba=xgb_wrapper.predict_proba(X_test)[:,1]
    get_clf_eval(y_test[i], w_preds, w_pred_proba)
    print(precision_recall_curve_plot(y_test[i], w_pred_proba))
```

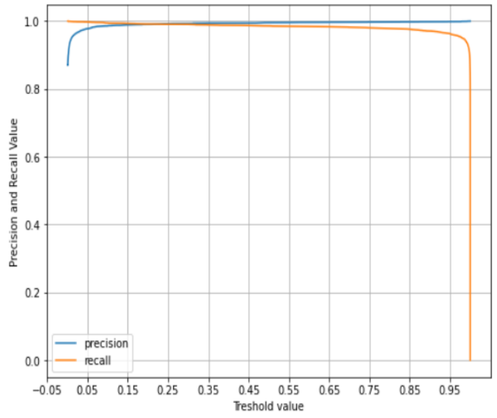
하나로 통일한 장르의 예측 성능 매우 우수

정확도:0.9905, 정밀도 0.9956, 재현율 :0.9859, F1:0.9907, AUC:0.999636

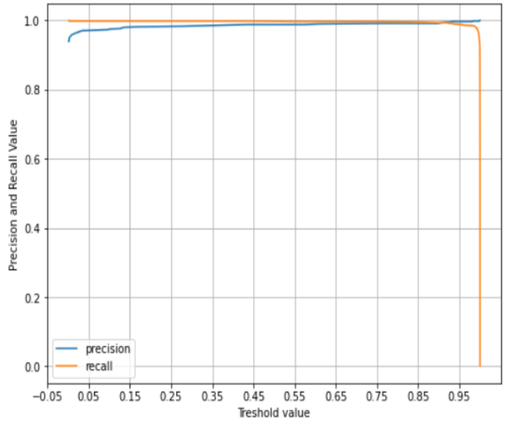


전반적으로 성능 향상

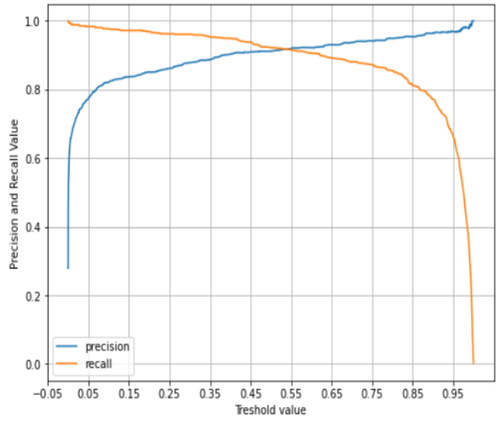
정확도:0.9905, 정밀도 0.9956, 재현율 :0.9859, F1:0.9907, AUC:0.999636



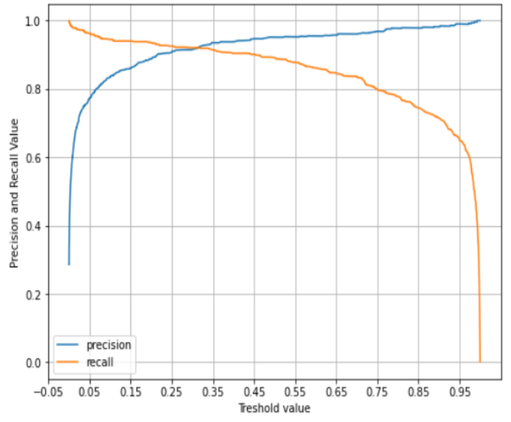
정확도:0.9989, 정밀도 0.9882, 재현율 :0.9966, F1:0.9924, AUC:0.999982



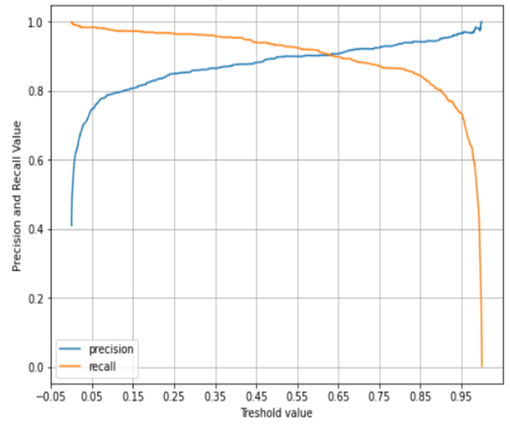
정확도:0.9879, 정밀도 0.9128, 재현율 :0.9217, F1:0.9172, AUC:0.997231



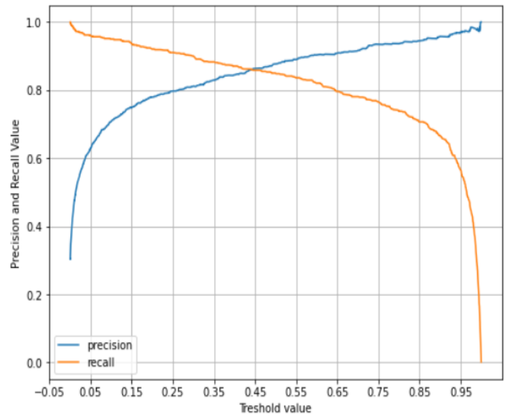
정확도:0.9888, 정밀도 0.9516, 재현율 :0.8916, F1:0.9206, AUC:0.996303



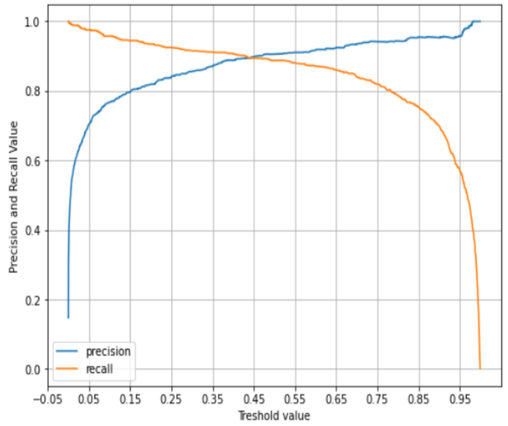
정확도:0.9885, 정밀도 0.8991, 재현율 :0.9326, F1:0.9156, AUC:0.997226



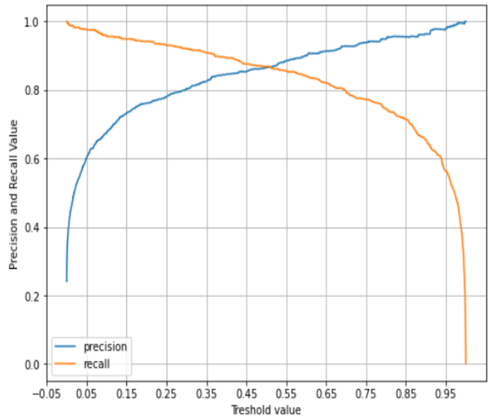
정확도:0.9814, 정밀도 0.8770, 재현율 :0.8483, F1:0.8624, AUC:0.993167



정확도:0.9852, 정밀도 0.9068, 재현율 :0.8876, F1:0.8971, AUC:0.995220



정확도:0.9820, 정밀도 0.8656, 재현율 :0.8687, F1:0.8671, AUC:0.994513



지금까지 상큼한
5조의 발표였습니다!

발표에 대한 질의응답은 언제나 환영입니다 :)

발표 끝!