

# 우범화물 탐지 경진대회

송실대학교 송파고 

20172568 홍인혁 20192185 공유경  
20192193 김수빈 20192209 김호균

# Contents

01

## EDA & 데이터 전처리

- Library & Loading
- 데이터 확인
- 결측값, 이상값 처리
- Feature Engineering
- Feature Selection
- 데이터 분할

02

## 모델 구성

- 모델 선정
- 임계값 선정
- 파라미터 선정

03

## 결과 도출

- 데이터 전처리 결과
- F1 값 비교
- 결론

# 01

[EDA & 데이터 전처리]

# 1. EDA & 데이터 전처리

## 0) Library & Loading

```
import pandas as pd
import numpy as np
np.random.seed(1000)

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from scipy import stats

#데이터 불러오기
train_data=pd.read_csv('train.csv',encoding='UTF-8')
test_data=pd.read_csv('test.csv',encoding='UTF-8')
```

# 1. EDA & 데이터 전처리

## 1) 데이터 확인

Out [4]:

	신고번호	신고일자	통관지세관부호	신고인부호	수입자부호	해외거래처부호	특송업체부호	수입통관계획코드	수입신고구분코드	수입거래구분코드	수입종류코드	징수형태코드	신고중량(KG)	과세가격원화금액	운송수단유형코드	반입보세구역부호	HS10단위부호	적출국가코드	원산지국가코드	관세율구분코드	관세율
0	37453	2020-01-01	40	10UUA	435E04J	CFLCEFM	NaN	C	A	15	21	11	4181.6	1.207812e+04	40	4077180	4202999000	CN	CN	A	8.0
1	150339	2020-01-01	20	7E3BD	1NTJ7F6	NaN	NaN	C	B	15	21	11	7977.0	5.938688e+05	10	4002001	9503003919	CN	CN	A	8.0
2	55710	2020-01-01	10	QHZ00	DGHPNRA	ZEXKR7K	8W5SEL	F	B	94	21	11	246.0	6.238490e+04	40	15002001	8708290000	CN	CN	FCN1	3.2
3	413154	2020-01-01	20	SZIVC	G85A4OI	WD62ULK	0X6YQV	C	B	15	21	11	4435.3	4.321373e+06	10	4002001	2933699099	CN	CN	FCN1	0.0
4	223511	2020-01-01	40	BU7II	9I00BFP	Q22MMTW	NaN	F	B	15	21	11	4564.4	1.212105e+06	10	2002079	6205200000	CN	CN	A	13.0

5 rows × 23 columns

### ● 연속형 변수 (2개)

: 신고 중량(KG), 과세가격원화금액

### ● 범주형 변수 (19개)

: 신고중량(KG), 과세가격원화금액 제외한 나머지 변수

“ 총 21개 ”

# 1. EDA & 데이터 전처리

## 1) 데이터 확인

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 89619 entries, 0 to 89618  
Data columns (total 23 columns):
```

```
#   Column      Non-Null Count  Dtype
```

```
---  ---  
0   신고번호      89619 non-null    int64  
1   신고일자      89619 non-null    object  
2   통관지세관부호  89619 non-null    int64  
3   신고인부호     89619 non-null    object  
4   수입자부호     89619 non-null    object  
5   해외거래처부호  62966 non-null    object  
6   특송업체부호    29787 non-null    object  
7   수입통관계획코드 89619 non-null    object  
8   수입신고구분코드 89619 non-null    object  
9   수입거래구분코드 89619 non-null    int64
```

```
10  수입종류코드   89619 non-null    int64  
11  징수형태코드   89619 non-null    int64  
12  신고중량(KG)   89619 non-null    float64  
13  과세가격원화금액 89619 non-null    float64  
14  운송수단유형코드 89619 non-null    int64  
15  반입보세구역부호 89619 non-null    int64  
16  HS10단위부호   89619 non-null    int64  
17  적출국가코드   89619 non-null    object  
18  원산지국가코드  89619 non-null    object  
19  관세율구분코드  89619 non-null    object  
20  관세율         89619 non-null    float64  
21  우범여부       89619 non-null    int64  
22  핵심적발       89619 non-null    int64  
dtypes: float64(3), int64(10), object(10)  
memory usage: 15.7+ MB
```

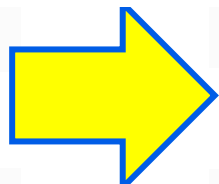
전체 데이터 개수 : 89619개

Feature 개수가 89619가 아닌 경우  
결측값 존재함을 의미

**NA(결측 값)을 포함한 변수 확인 !**

# 1. EDA & 데이터 전처리

## 2) 결측값, 이상값 처리

	통관지 세관부호	신고인 부호	수입자부호	해외거래처 부호	특송업체 부호	수입통 관 계 획 코드	수입신 고 구 분 코드	수입거 래 구 분 코드	수입종 류 코드	징수 형태 코드		통관지 세관부호	신고인 부호	수입자부호	해외거래처 부호	특송업체 부호	수입통 관 계 획 코드	수입신 고 구 분 코드	수입거 래 구 분 코드	수입종 류 코드	징수 형태 코드	
0	40	10UUA	435E04J	CFLCEFM	NaN	C	A	15	21	11	결측값 대체 	0	40	10UUA	435E04J	CFLCEFM	NO	C	A	15	21	11
1	20	7E3BD	1NTJ7F6	NaN	NaN	C	B	15	21	11		1	20	7E3BD	1NTJ7F6	NO	NO	C	B	15	21	11
2	10	QHZ00	DGHPNRA	ZEXKR7K	8W5SEL	F	B	94	21	11		2	10	QHZ00	DGHPNRA	ZEXKR7K	8W5SEL	F	B	94	21	11
3	20	SZIVC	G85A4OI	WD62ULK	0X6YQV	C	B	15	21	11		3	20	SZIVC	G85A4OI	WD62ULK	0X6YQV	C	B	15	21	11
4	40	BU7II	9I00BFP	Q22MMTW	NaN	F	B	15	21	11		4	40	BU7II	9I00BFP	Q22MMTW	NO	F	B	15	21	11
5	20	H1HB2	J0HY67G	HWI9WCT	2TNP07	C	B	11	21	21		5	20	H1HB2	J0HY67G	HWI9WCT	2TNP07	C	B	11	21	21

해외거래처부호/ 특송업체 NA(결측 값) 존재

신고 시, 입력되지 않은 범주형 데이터이므로  
제외하지 않고 **입력되지 않았**다는 의미로

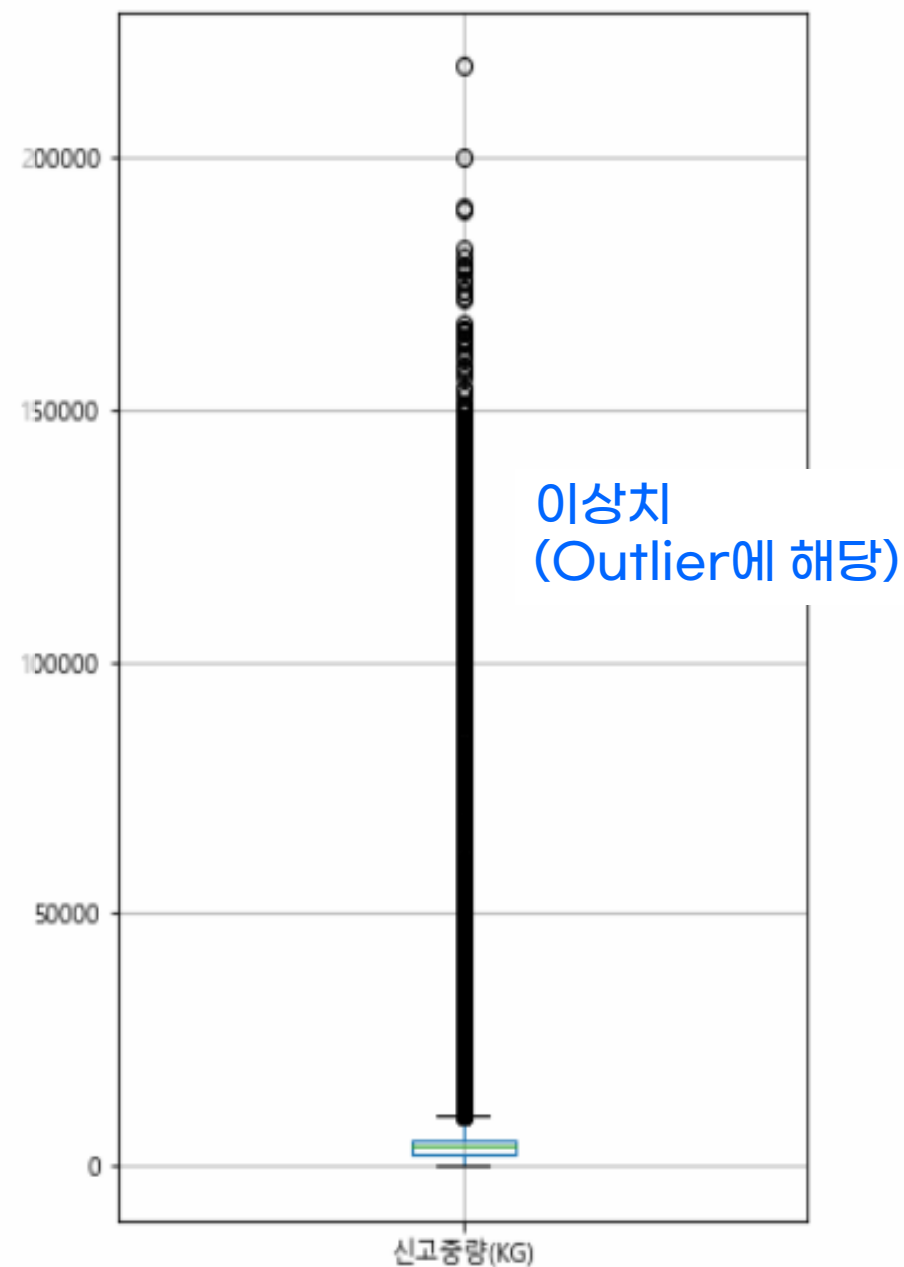
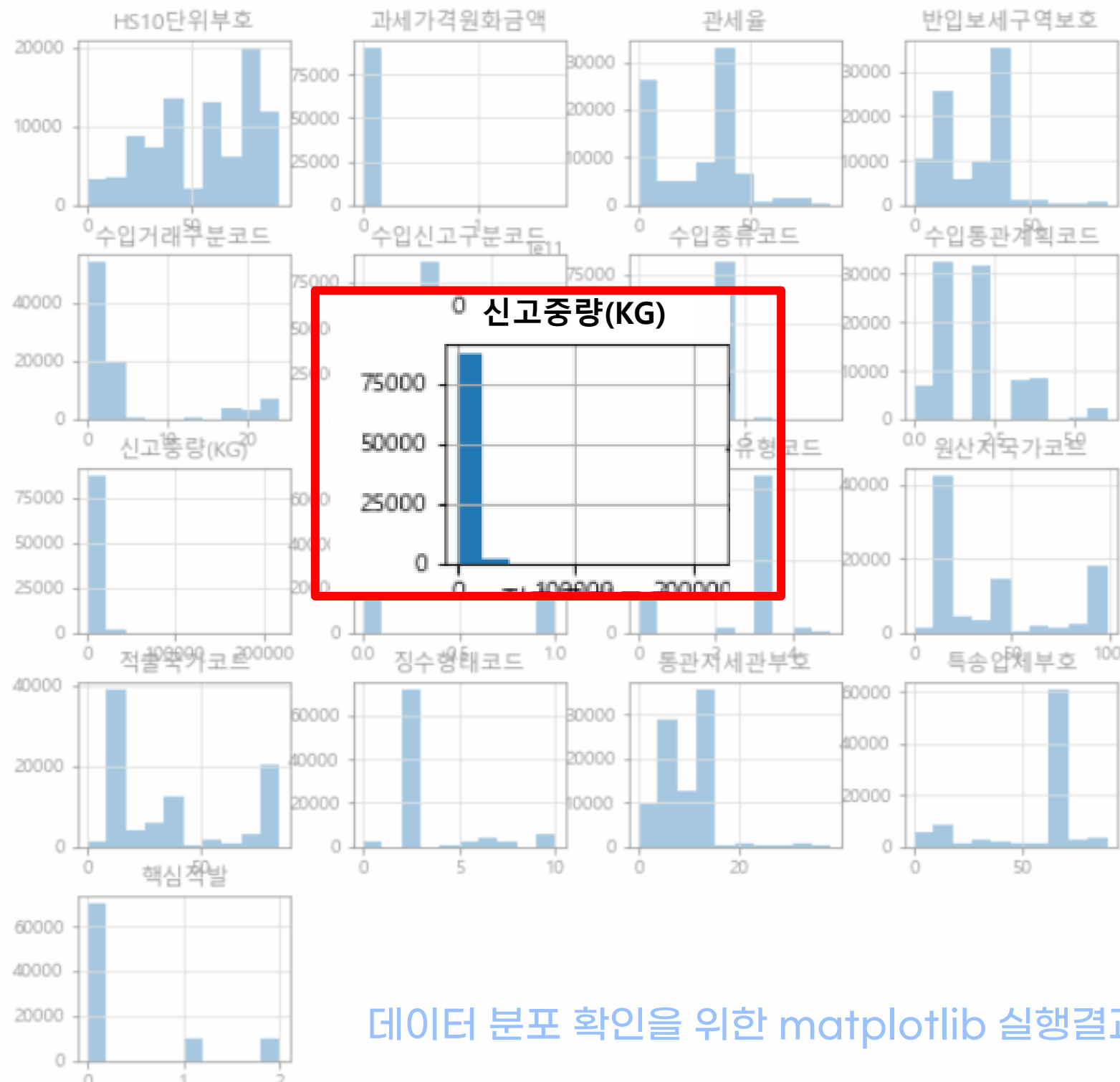
**‘NO’** 로 결측값 대체

# 1. EDA & 데이터 전처리

## 2) 결측값, 이상값 처리

```
In [11]: import matplotlib.pyplot as plt
plt.rc('font', family='Malgun Gothic')
data.hist(figsize=(10,10))
```

```
In [12]: data.boxplot(column='신고중량(KG)', figsize=(5,8))
```



matplotlib을 통해 데이터  
시각화한 결과,  
**연속형 데이터 중  
신고중량(KG)에서 이상치를  
확인하여 제거함.**

데이터 분포 확인을 위한 matplotlib 실행결과



# 1. EDA & 데이터 전처리

## 3) Feature Engineering

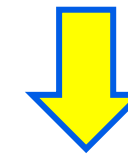
```
# 반입보세구역부호 전처리 #
area_code=df['반입보세구역부호']
Area_code=[]
for i in area_code:
    if(int(i/1000000)<10):
        string_number='0'+str(i)
        string_number=string_number[0:3]
        Area_code.append(string_number)
    elif(int(i/10000000)<10):
        co=str(i)
        a=co[0:3]
        Area_code.append(a)
df.drop('반입보세구역부호',axis=1,inplace=True)
df['반입보세구역부호'] = pd.Series(Area_code, index=df.index)

# HS10단위부호 전처리 #
HS_code=df['HS10단위부호']
HS_code=list(HS_code)
Hs_code=[]
for i in HS_code:
    Hs_code.append(int(i/1000000))

df.drop('HS10단위부호',axis=1,inplace=True)
df['HS10단위부호'] = pd.Series(Hs_code, index=df.index)
```

Feature Engineering 전체 코드

총 21개의 데이터 중,  
**반입보세구역부호**와 **HS10단위부호**가  
가공이 필요하다고 판단함.



데이터 별로 쓰임에 맞게  
가공하는 과정을 거침

# 1. EDA & 데이터 전처리

## 3) Feature Engineering

### 반입보세구역부호

(1) 세관 부호(301)

부 호	기 관 명	부 호	기 관 명
000	관 세 청	039	북 부 산 세 관
001	기 획 조 정 관 실	037	부 산 우 편 센 터
002	감 사 관 실	050	마 산 세 관
003	조 사 감 시 국	051	경 남 남 부 세 관
004	정 보 협 력 국	054	통 영 센 터
005	통 관 지 원 국	053	창 원 세 관
006	심 사 정 책 국	053	진 해 센 터
008	중 앙 관 세 분 석 소	056	경 남 서 부 세 관
009	관 세 국 경 관 리 연 수 원	052	사 천 센 터
007	관 세 평 가 분 류 원	140	김 해 공 항 세 관
010	서 울 세 관	020	인 천 세 관(인천항)
012	성 남 세 관	014	안 산 세 관
011	의 정 부 센 터	023	부 평 센 터

무역통계부호\_2020

No	보세구역부호	보세구역명	세관	보세구역소재지	보세구역전화번호	과태료 부과여부	가산세 징수여부	통관부서	하역장소 보세구역여부
11	01011096	삼덕보세창고	서울세관	서울특별시 성동구 아차산로 67	024980141	N	N	부서	N
12	01011182	(주)동양물류센타 보세창고	서울세관	서울특별시 성동구 독성로17가길 49	0246942258	N	N	부서	N
13	01011209	(주)선수물류 보세창고	서울세관	서울특별시 성동구 돌레19길 13	024640095	N	N	부서	N
2	02001002	제2컨테이너검사센터세관검사장	인천세관	인천광역시 중구 축항대로118번길 1 35 (항동7가, ICT인천남항컨테이너터 미널)	4523368	N	N	부서	N
3	02001003	제3컨테이너검사센터세관검사장	인천세관	인천광역시 중구 서해대로30번길 32 (신흥동3가, 인천남항컨테이너물류기 지)	03278403305	N	N	부서	N

국가관세종합정보망  
서비스\_보세구역

반입보세구역부호 앞 3자리는 세관별 분류를 의미

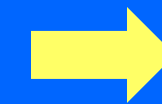
# 1. EDA & 데이터 전처리

## 3) Feature Engineering

### 반입보세구역부호

	신고번호	신고일자	통관지세관부호	신고인부호	수입자부호	해외거래처부호	특송업체부호	수입통관계획코드	수입신고구분코드	수입거래구분코드	...	과세가격원화금액	운송수단유형코드	반입보세구역부호	HS10단위부호	적출국가코드	원산지국가코드	관세율분코드	관세율	우범여부	핵심적발
0	37453	2020-01-01	40	10UUA	435E04J	CFLCEFM	NaN	C	A	15	...	1.207812e+04	40	4077180	4202999000	CN	CN	A	8.0	0	0
1	150339	2020-01-01	20	7E3BD	1NTJ7F6	NaN	NaN	C	B	15	...	5.938688e+05	10	4002001	9503003919	CN	CN	A	8.0	0	0

반입 보세 구역 부호



세관 별 분류: 앞 3자리

반입 보세 구역 부호 8자리 기재



반입보세구역부호 7자리인 경우, 첫번째 자리 '0' 추가

\*맨 앞이 0인 경우, 엑셀과 같은 데이터 파일 변환 과정에서 데이터 소실되는 경우 존재.

# 1. EDA & 데이터 전처리

## 3) Feature Engineering

### 반입보세구역부호

```
# 반입보세구역부호 전처리 #
area_code=df['반입보세구역부호']
Area_code=[]
for i in area_code:
    if(int(i/1000000)<10):
        string_number='0'+str(i)
        string_number=string_number[0:3]
        Area_code.append(string_number)
    elif(int(i/10000000)<10):
        co=str(i)
        a=co[0:3]
        Area_code.append(a)
df.drop('반입보세구역부호',axis=1,inplace=True)
df['반입보세구역부호'] = pd.Series(Area_code, index=df.index)

# HS10단위부호 전처리 #
HS_code=df['HS10단위부호']
HS_code=list(HS_code)
Hs_code=[]
for i in HS_code:
    Hs_code.append(int(i/1000000))

df.drop('HS10단위부호',axis=1,inplace=True)
df['HS10단위부호'] = pd.Series(Hs_code, index=df.index)
```

### 반입보세구역부호

- 반입보세구역부호 7자리 경우, 첫째 자리 '0' 추가
- 세관 별 분류 : 앞 3자리

# 1. EDA & 데이터 전처리

## 3) Feature Engineering

### 반입보세구역부호

```
# 반입보세구역부호 전처리 #
area_code=df['반입보세구역부호']
Area_code=[]
for i in area_code:
    if(int(i/1000000)<10):
        string_number='0'+str(i)
        string_number=string_number[0:3]
        Area_code.append(string_number)

    elif(int(i/10000000)<10):
        co=str(i)
        a=co[0:3]
        Area_code.append(a)
df.drop('반입보세구역부호',axis=1,inplace=True)
df['반입보세구역부호'] = pd.Series(Area_code, index=df.index)
```

```
# HS10단위부호 전처리 #
HS_code=df['HS10단위부호']
HS_code=list(HS_code)
Hs_code=[]
for i in HS_code:
    Hs_code.append(int(i/1000000))

df.drop('HS10단위부호',axis=1,inplace=True)
df['HS10단위부호'] = pd.Series(Hs_code, index=df.index)
```

### 반입보세구역부호

- 반입보세구역부호 7자리 경우, 첫째 자리 '0' 추가
- 세관 별 분류 : 앞 3자리

# 1. EDA & 데이터 전처리

## 3) Feature Engineering

### HS10단위부호

#### HS코드

**요약** 국제통일상품분류체계에 따라 대외 무역거래 상품을 총괄적으로 분류한 품목분류 코드

국제통일상품분류체계에 따라 대외 무역거래 상품을 총괄적으로 분류한 품목분류 코드. 관세나 무역통계, 운송, 보험 등 다양한 목적에 사용된다. 국제협약에 따라 HS코드는 10자리까지 사용할 수 있다. 6자리까지는 국제 공통으로 사용하는 코드로서 앞의 1~2자리는 상품의 군별 구분, 3~4자리는 소분류로 동일류 내 품목의 종류별, 5~6자리는 세분류 동일호 내 품목의 용도·기능 등에 따른 분류다. 7자리부터는 각 나라에서 세분화해 부여하며 우리나라는 10자리를 사용한다.

시사경제용어사전

HS코드 앞 6자리는 국제 공통 사용 코드

HSCODE	품목명
0100	제1류 산동물
0200	제2류 육과 식용설육
0300	제3류 어패류

No	HSCode	
1	0101.29-1000	<a href="#">경주말</a>
2	0106.14-9000	<a href="#">번식용이 아닌 살아있는 토끼</a>
3	0106.19-1000	<a href="#">살아있는 개</a>
4	0106.19-9000	<a href="#">살아있는 기타 포유동물</a>
5	0106.20-1000	<a href="#">살아있는 뱀</a>
6	0106.20-3000	<a href="#">살아있는 거북</a>
7	0106.20-9000	<a href="#">살아있는 기타 파충류</a>

[그림] 국가관세종합정보망 서비스

# 1. EDA & 데이터 전처리

## 3) Feature Engineering

### HS10단위부호

```
# 반입보세구역부호 전처리 #
area_code=df['반입 보세 구역 부호']
Area_code=[]
for i in area_code:
    if(int(i/1000000)<10):
        string_number='0'+str(i)
        string_number=string_number[0:3]
        Area_code.append(string_number)
    elif(int(i/10000000)<10):
        co=str(i)
        a=co[0:3]
        Area_code.append(a)
```

```
# HS10단위부호 전처리 #
HS_code=df['HS10단위 부호']
HS_code=list(HS_code)
Hs_code=[]
for i in HS_code:
    Hs_code.append(int(i/1000000))

df.drop('HS10단위 부호',axis=1,inplace=True)
df['HS10단위 부호'] = pd.Series(Hs_code, index=df.index)
```

### HS10단위부호

- HS10단위 부호에 대하여  
2자리 씩 나눠서 결과값 비교 결과,  
-> **6자리로 분류**했을 때,  
결과값이 가장 높았음.

# 1. EDA & 데이터 전처리

## 3) Feature Engineering

### 레이블 인코딩

##### 범주형 변수에 레이블 인코딩 적용 #####

```
category_df = df[범주형].apply(encoding_label)
```

category\_df

	통관지세 관부호	수입통관 계획코드	해외거래 처부호	특송업 체부호	수입신고 구분코드	수입거래 구분코드	수입종 류코드	징수형 태코드	운송수단 유형코드	반입보세 구역부호	HS10단위 부호	적출국 가코드	원산지국 가코드	관세율구 분코드	관세 율
0	13	1	1591	63	0	3	4	2	3	14	287	15	19	0	37
1	6	1	3033	63	1	3	4	2	0	14	737	15	19	0	37
2	0	4	4589	28	1	23	4	2	3	35	674	15	19	13	15
3	6	1	4211	2	1	3	4	2	0	14	180	15	19	13	0
4	13	4	3386	63	1	3	4	2	0	6	402	15	19	0	46
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10268	16	3	2603	85	2	6	6	7	3	9	737	15	19	13	15
10269	6	2	1920	80	1	16	7	2	0	6	180	15	19	13	13
10270	13	6	2290	63	1	0	4	5	3	13	492	81	95	1	0
10271	9	1	68	63	1	0	4	10	0	14	731	15	19	1	0
10272	13	1	665	2	1	0	4	7	3	32	172	37	16	17	0

- 데이터 전처리 과정을 거친 후, 범주형 데이터를 **레이블 인코딩** 을 통해 데이터를 변환함.

- **레이블 인코딩 (Label Encoding)**  
: 문자열로 구성되어있는 데이터를 카테고리화하는 방법

Q. 원핫 인코딩을 사용하지 않은 이유는?

원-핫 인코딩의 경우 차원을 늘어나 데이터가 클 경우, 벡터 저장을 위해 매우 큰 공간이 필요함.

실제 우범화물 예측 모델 적용 시, 저장 공간 측면에서 매우 비효율적인 표현 방법이라고 판단함.



# 1. EDA & 데이터 전처리

## 4) Feature Selection

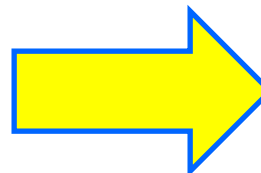
### 1.0.1 treemodel 기반 feature 선택 결과

```
from sklearn.ensemble import ExtraTreesClassifier

etc_model = ExtraTreesClassifier()
etc_model.fit(X, y)

print(etc_model.feature_importances_)
feature_list = pd.concat([pd.Series(X.columns), pd.Series(etc_model.feature_importances_)])
feature_list.columns = ['features_name', 'importance']
feature_list.sort_values("importance", ascending=False)[:17]
```

	features_name	importance
1	특송업체부호	0.098113
16	신고중량(KG)	0.096090
15	신고인부호	0.072639
14	수입자부호	0.072194
17	과세가격원화금액	0.068738
8	반입보세구역부호	0.067147
0	통관지세관부호	0.065619
9	HS10단위부호	0.065317
3	수입거래구분코드	0.051314
6	징수형태코드	0.048092
10	적출국가코드	0.047243
2	수입통관계획코드	0.046787



Tree 기반 모델  
Feature 중요성 측정 결과

연속형 및 **고유값이 가장 많은 값** 들이  
중요도 상위에 위치함

train\_data.nunique()

신고번호	89619	징수형태코드	11
신고일자	322	신고중량(KG)	50573
통관지세관부호	39	과세가격원화금액	88187
신고인부호	944	운송수단유형코드	6
수입자부호	8429	반입보세구역부호	571
해외거래처부호	4643	HS10단위부호	2396
특송업체부호	89	적출국가코드	86
수입통관계획코드	7	원산지국가코드	100
수입신고구분코드	4	관세율구분코드	36
수입거래구분코드	25	관세율	87
수입종류코드	10		

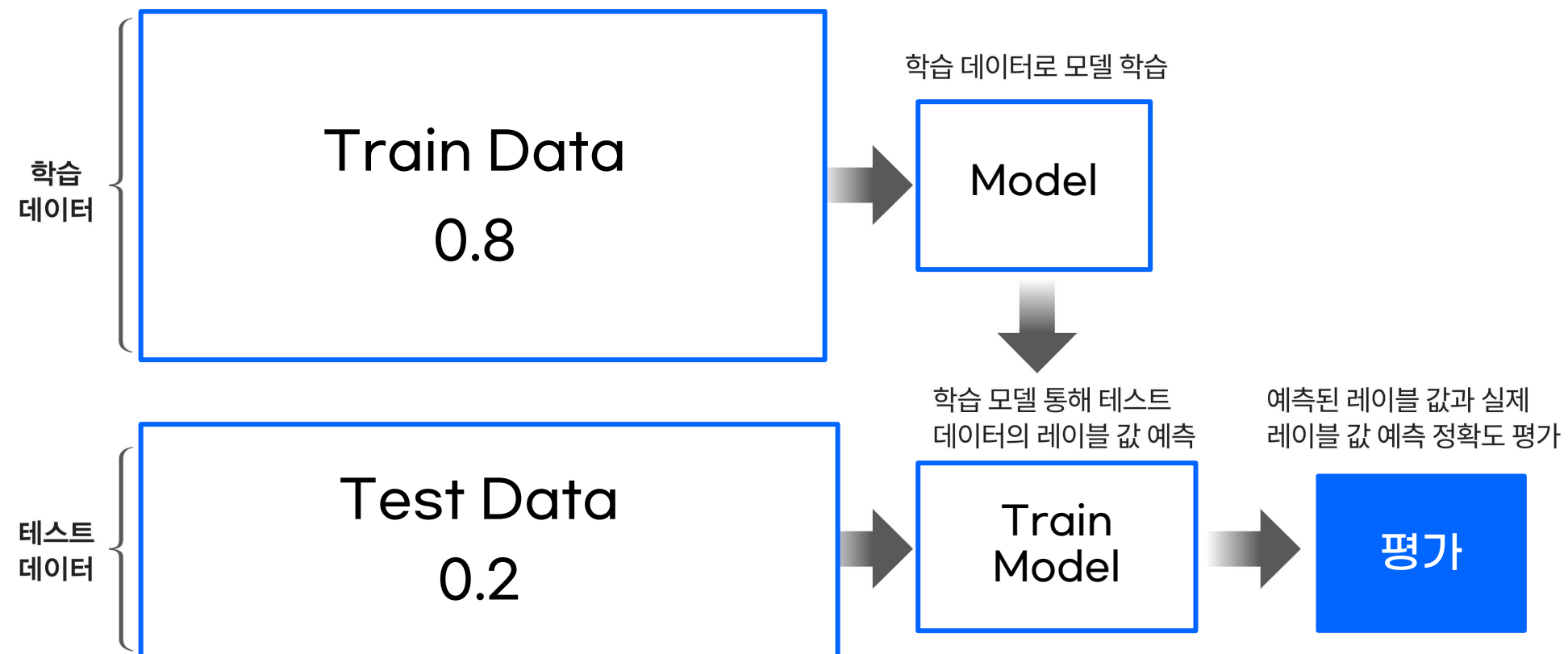
따라서, 제대로된 중요도 평가를 위해  
feature를 하나씩 제외시키며  
실험을 통해 중요도를 고려함.

**신고번호, 신고일자, 신고인 부호, 수입자 부호 제외!**

# 1. EDA & 데이터 전처리

## 5) 데이터 분할

```
X=data  
y=train_data['우범여부']  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
                                                    random_state=1000)
```



train\_test\_split() 을 이용하여

**Test Data : Train Data = 2 : 8**

로 데이터를 분할함.

# 1. EDA & 데이터 전처리 요약

## 데이터 전처리 전과정

1	data_test																
	통관지 세관부 호	수입통 관계획 코드	해외거 래처부 호	특송 업체 부호	수입신 고구분 코드	수입거 래구분 코드	수입 종류 코드	징수 형태 코드	운송수단 유형코드	반입보세 구역부호	HS10 단위부 호	적출국 가코드	원산지 국가코 드	관세율 구분코 드	관 세 율	신고중 량(KG)	과세가격원화 금액
0	6	4	100	63	1	3	4	2	3	9	34	81	95	0	64	3731.9	2.755287e+06
1	13	3	3033	63	1	23	4	2	3	14	760	81	95	0	0	6406.5	8.517864e+05
2	13	1	3033	63	1	0	4	2	3	14	99	81	95	0	37	5824.9	6.532926e+04
3	4	1	3033	63	1	0	4	2	3	6	58	81	95	0	14	3798.3	1.028587e+06
4	0	1	3033	63	1	0	4	2	3	0	676	83	97	29	0	3795.7	1.677178e+06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10268	16	3	2603	85	2	6	6	7	3	9	737	15	19	13	15	3063.2	2.843916e+04
10269	6	2	1920	80	1	16	7	2	0	6	180	15	19	13	13	1595.7	3.654290e+06
10270	13	6	2290	63	1	0	4	5	3	13	492	81	95	1	0	2432.3	2.926935e+06
10271	9	1	68	63	1	0	4	10	0	14	731	15	19	1	0	8484.3	1.534544e+05
10272	13	1	665	2	1	0	4	7	3	32	172	37	16	17	0	2200.4	3.277707e+05

- 결측값 제거  
: 해외거래처부호, 특송업체부호
- 이상치 제거  
: 신고 중량(KG)
- Feature Engineering  
반입보세구역부호: 세관별 분류 (앞 3자리)  
HS10단위부호: 국제공통코드 분류(앞 6자리)
- Label Encoding 범주형 데이터
- 신고번호, 신고일자, 신고인 부호, 수입자 부호 제외

# 02

## 모델 구성

- 모델 선정
- 임계값 선정
- 파라미터 선정

## 2.모델구성 1) 모델 선정

### 수치형 변수와 우범여부의 상관계수

	신고중량(KG)	과세가격원화금액	관세율	우범여부
신고중량(KG)	1.000000	0.021801	0.006555	0.040808
과세가격원화금액	0.021801	1.000000	-0.004266	0.008332
관세율	0.006555	-0.004266	1.000000	-0.008179
우범여부	0.040808	0.008332	-0.008179	1.000000

#### 상관관계 판별지수

상관계수	강도	상관계수	강도
+,- 0.9이상	매우높은 상관관계	+0.2에서+- 0.4	낮은 상관관계
+0.7에서+- 0.9	높은 상관관계	+0.2미만	거의 관계없음
+0.4에서+- 0.7	다소높은 상관관계		

대부분의 변수가 범주형  
수치형 범주의 상관계수 매우 크지 않음

주로 수치형 데이터에  
사용하는 회귀 모형보다  
**트리 모형**이 더  
적합하다고 판단

## 2. 모델구성 1) 모델 선정

### 결정 트리(Decision Tree)

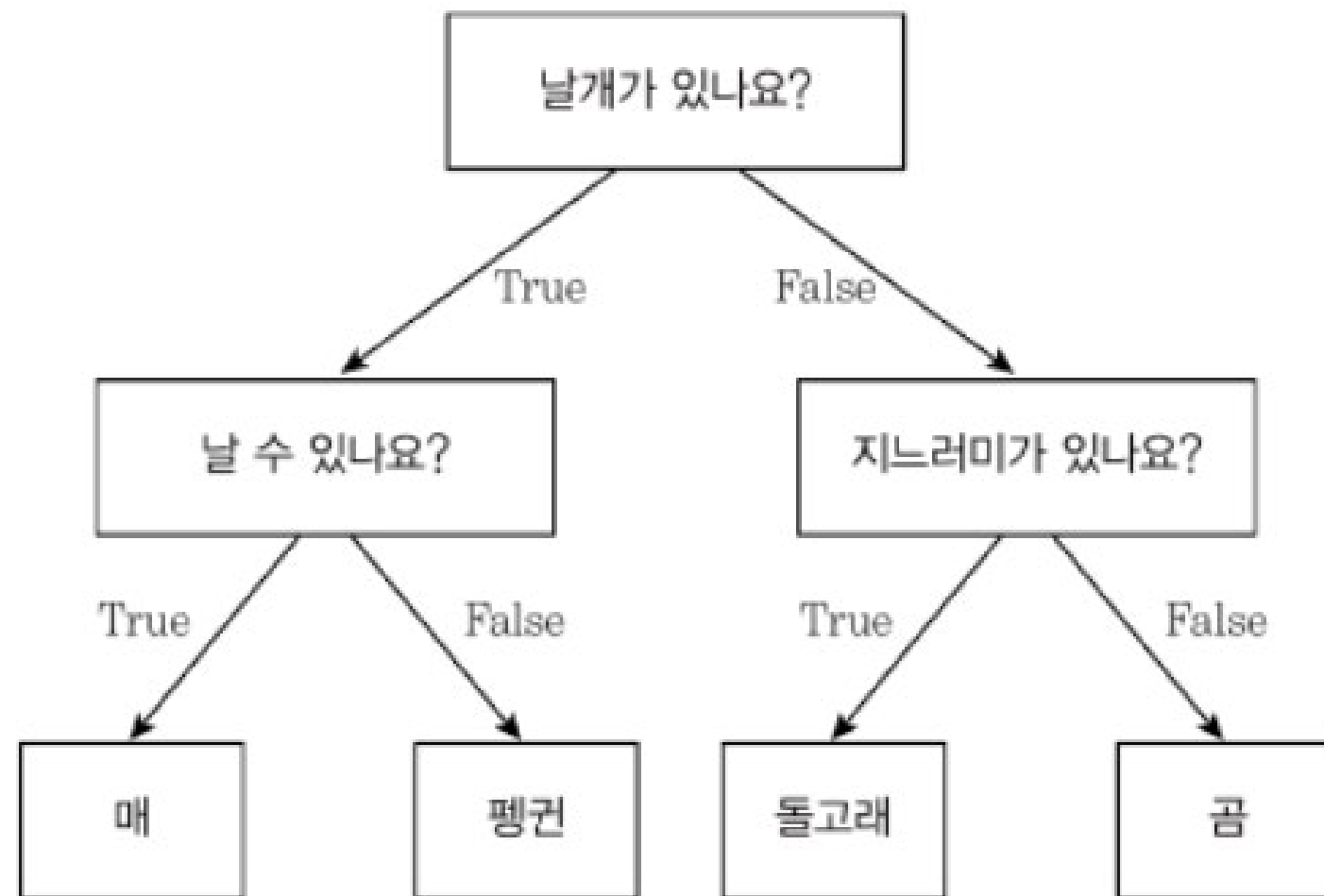


그림 2-22 몇 가지 동물들을 구분하기 위한 결정 트리

## Decision Tree

1. 분류 문제에 적합
2. 수치형+범주형 특성 가능

그러나, train data에 민감

> 결정 트리를 결합해 성능 향상

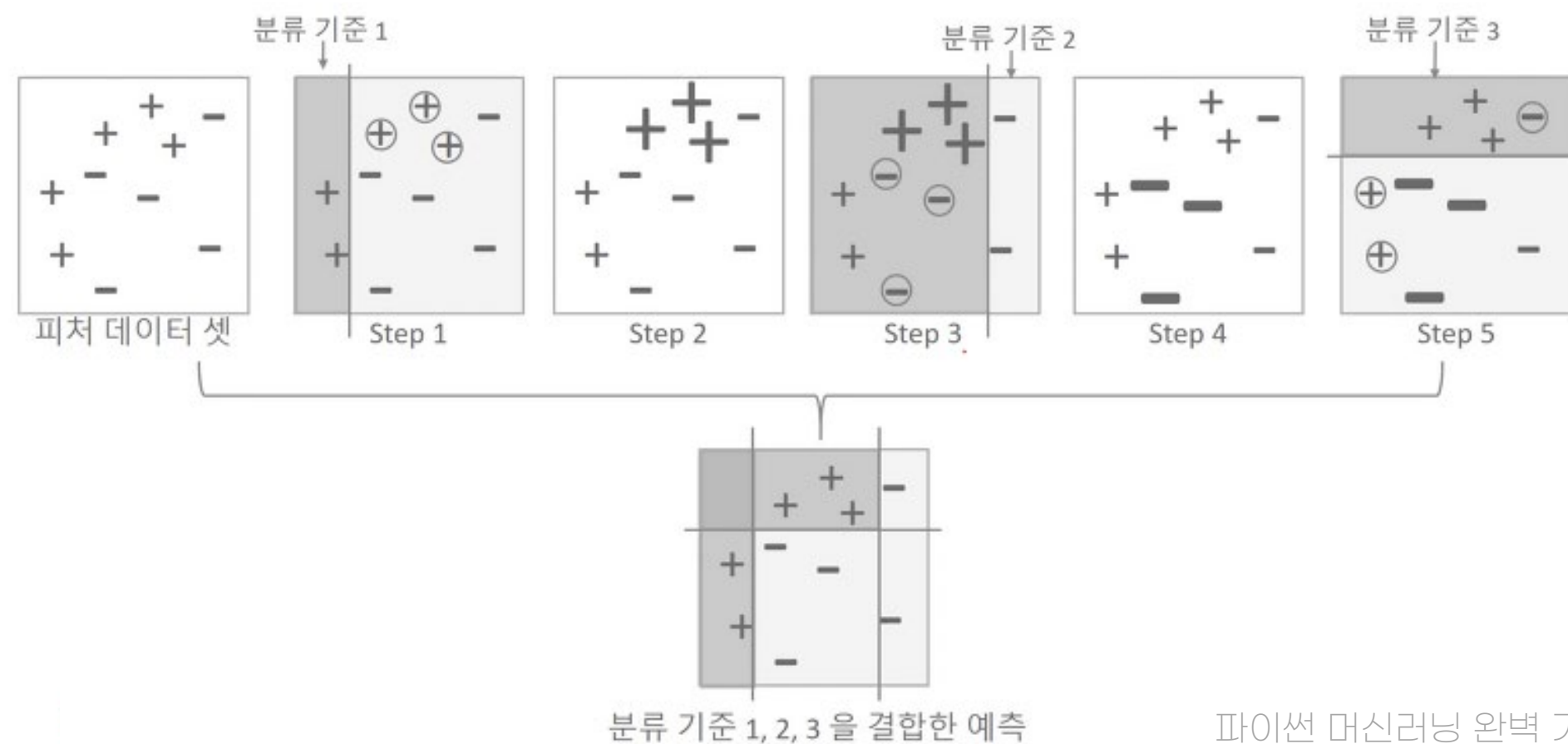
“ 앙상블 ”

## 2. 모델구성 1) 모델 선정

앙상블 학습 중에 Boosting 알고리즘 선택

# Boosting

여러 개의 약한 학습기를  
순차적으로 학습-예측하면서  
잘못 예측한 데이터에 가중치  
부여를 통해 오류를 개선해  
나가면서 학습하는 방식



## 2.모델구성 1) 모델 선정

Boosting 방식 중 XG Boost, Light GBM 사용

### XGBoost



핵심적발 적용

### Light GBM



우범여부 적용



## 2. 모델구성 2) 임계값 선정

### 모델 디폴트 값으로 학습 및 평가

```
X=data  
y=train_data['우범여부']  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
                                                    random_state=1000)
```

```
lgbm_wrapper = LGBMClassifier()  
  
evals = [(X_test, y_test)]  
lgbm_wrapper.fit(X_train, y_train, eval_metric="logloss", eval_set=evals, verbose=False)  
  
preds = lgbm_wrapper.predict(X_test)  
lgbm_pred_proba = lgbm_wrapper.predict_proba(X_test)[:, 1]
```

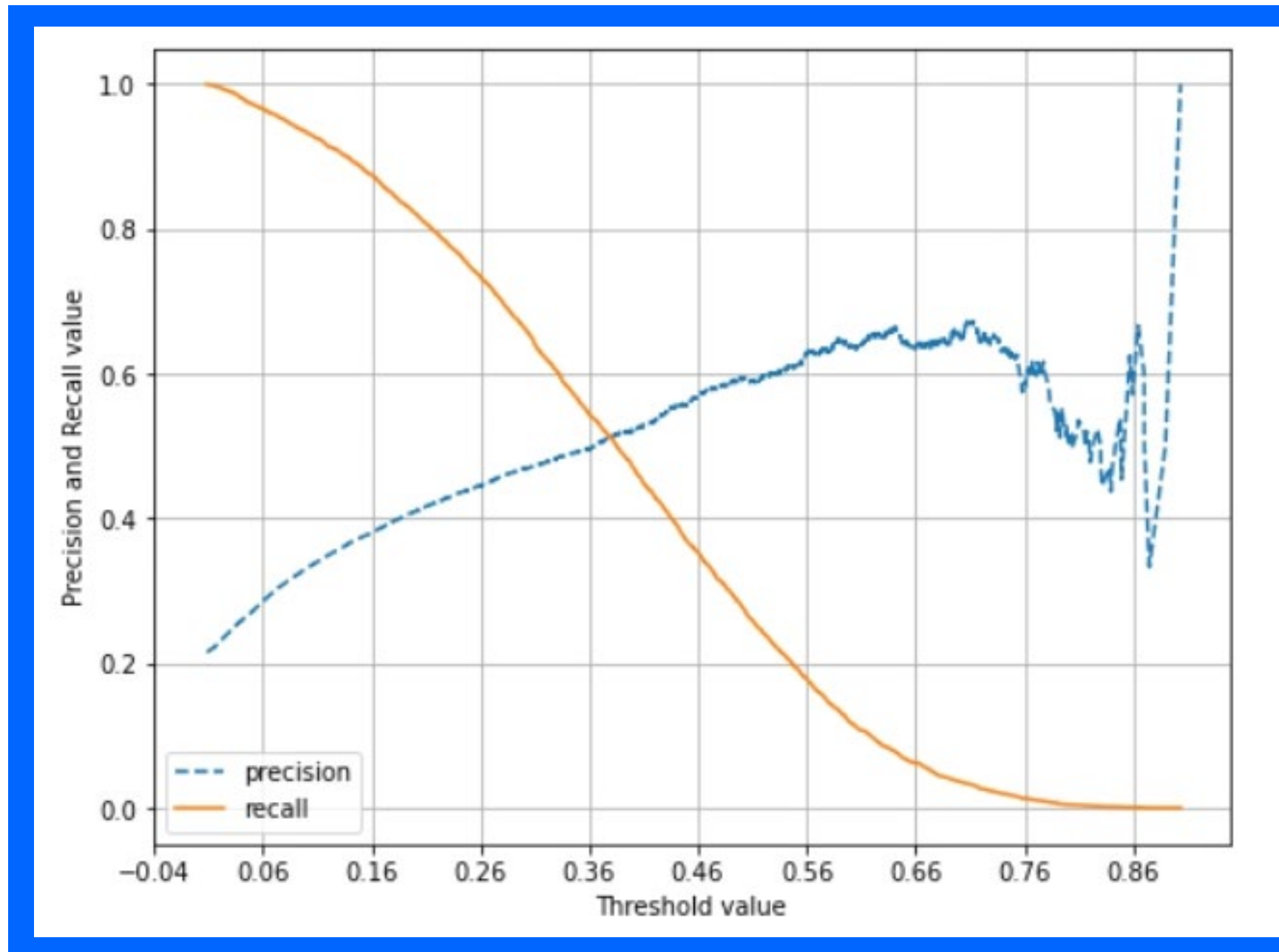
```
[[12992    687]  
 [ 2760    993]]  
정확도: 0.8023, 정밀도: 0.5911  
재현율: 0.2646, F1: 0.3655
```

**재현율** = 실제 우범 건수 중 우범이라고 예측한 건수의 비율

**정밀도** = 우범이라고 예측한 건수 중 실제 우범 건수의 비율

## 2.모델구성 2) 임계값 선정

정밀도, 재현율 그래프 시각화



정밀도가 높아질수록  
재현율이 낮아짐을 확인

## 2. 모델구성 2) 임계값 선정

F1-score = 정밀도와 재현율의 조화 평균

$$(F1-score) = 2 \times \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

### 모델 성능 측정 지표

```
[[12992  687]
 [ 2760  993]]
```

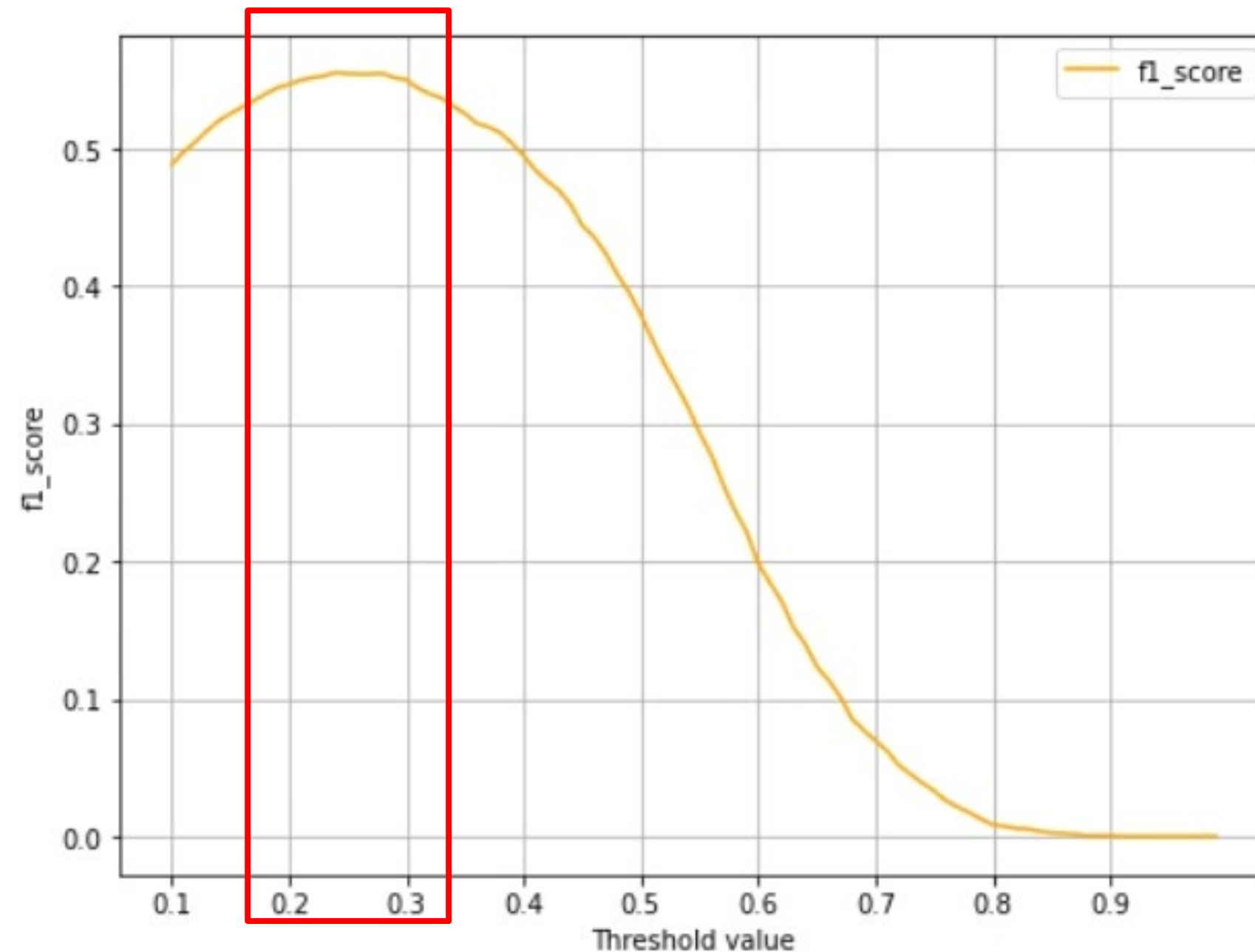
정확도: 0.8023, 정밀도: 0.5911  
재현율: 0.2646, F1: 0.3655

정밀도는 높지만 재현율이 상대적으로 낮아 F1-score 값이 낮음

재현율을 높이기 위해 임계값 조절

## 2. 모델구성 2) 임계값 선정

임계값 변화에 따른 f1-score 변화 그래프 시각화



**0.2~0.3**부근에서  
가장 높은 F1값

우범여부 0.26

핵심적발 0.28

## 2. 모델구성 2) 임계값 선정

```
In [55]: 1 for i in range(10,30,1):
2         i=i/100
3         pred=[]
4         for x in pred_probs:
5             if (x[1]+x[2]<0.28):
6                 pred.append(0)
7             elif (x[2]>i):
8                 pred.append(2)
9             else:
10                pred.append(1)
11
12 pred
13 pred=pd.DataFrame(pred)
14 print(i,"의 F1값 = ",f1_score(y_test,pred,average='macro'))
```

```
0.1 의 F1값 = 0.40449188311990464
0.11 의 F1값 = 0.40562891040896454
0.12 의 F1값 = 0.40856496461645514
0.13 의 F1값 = 0.4122321936189199
0.14 의 F1값 = 0.42089274057363757
0.15 의 F1값 = 0.43014916431474975
0.16 의 F1값 = 0.44396139673243623
0.17 의 F1값 = 0.45233393045485526
0.18 0.25 의 F1값 = 0.4733843923004462
0.19
0.2 의 F1값 = 0.4682406093221985
0.21 의 F1값 = 0.47134178588063386
0.22 의 F1값 = 0.47227218646399644
0.23 의 F1값 = 0.47326911832917734
0.24 의 F1값 = 0.4720459008284861
0.25 의 F1값 = 0.4733843923004462
0.26 의 F1값 = 0.47214146944320246
0.27 의 F1값 = 0.4721934435027118
0.28 의 F1값 = 0.4680704614906648
0.29 의 F1값 = 0.46678980622060284
```

### 핵심적발 1,2

예측 임계값이  
일정값 이상일 때 2로 구분

F1-score 결과값이  
가장 높은 값

**0.25** 로 선정

## 2. 모델구성 3) 파라미터 선정 - 우범여부

```
lgbm_wrapper = LGBMClassifier(random_state=1000, n_estimators=800, num_leaves=10, learning_rate=0.05,  
                               max_depth=5, min_child_samples=40, boost_from_average=False)
```

01

$n\_estimators = 800$

- 반복 수행 트리개수 지정
- 너무 크면 과적합 발생
- 반복 수행을 크게 잡고,  
조기 종단을 설정해 과적합을 방지함  
(early\_stopping\_rounds=100)

02

$num\_leaves = 10$

- 숫자를 늘리면 정확도가 높아짐
- 너무 크면 트리가 깊어지고,  
모델이 복잡해서 과적합 발생

03

$learning\_rate = 0.05$

- 반복횟수(800)에 적당한 학습률 선정

## 2. 모델구성 3) 파라미터 선정 - 우범여부

```
lgbm_wrapper = LGBMClassifier(random_state=1000, n_estimators=800, num_leaves=10, learning_rate=0.05,  
                               max_depth=5, min_child_samples=40, boost_from_average=False)
```

04

`max_depth = 5`

- 트리의 깊이를 5로 제한

05

`min_child_samples= 40`

- 최종 결정 클래스인 leaf node가  
되기 위한 최소한의 데이터 개체 수

06

`boost_from_average  
= False`

- 레이블이 불균형한 경우 True 설정은  
성능을 떨어뜨림  
- 디폴트 값이 True

## 2.모델구성 3) 파라미터 선정 - 핵심적발

```
params = {'max_depth' : 3, 'eta' : 0.1, 'objective' : 'multi:softprob', 'num_class':3,  
          'eval_metric' : 'mlogloss'}
```

01

**max\_depth = 3**

- 트리의 깊이를 3으로 제한

02

**Eta (learning rate)  
= 0.1**

- 적당한 학습률 선정

03

**Objective  
= multi:softprob**

- Multi 는 다중 분류
- 다중 분류 중 softprob는  
각 class에 속할 확률을 반환함



## 2.모델구성 3) 파라미터 선정 - 핵심적발

```
params = {'max_depth' : 3, 'eta' : 0.1, 'objective' : 'multi:softprob', 'num_class':3,  
          'eval_metric' : 'mlogloss'}
```

04

num\_class = 3

- 다중 분류 클래스의 수 (0,1,2)

05

eval\_metric  
= mlogloss

- 검증 데이터에 적용되는 모델 선택  
- 다중 분류일때 음수 로그 가능성

# 03

## 결과 도출

- 데이터 전처리 결과
- F1 값 비교
- 결론

### 3. 결과도출 1) 데이터 전처리 결과

1	### 확인용 ###
2	data

	통관지 세관부 호	수입통 관계획 코드	해외거 래처부 호	특송 업체 부호	수입신 고구분 코드	수입거 래구분 코드	수입 종류 코드	징수 형태 코드	운송수단 유형코드	반입보세 구역부호	HS10 단위부 호	적출국 가코드	원산지 국가코 드	관세율 구분코 드	관 세 율	신고중 량(KG)	과세가격원화 금액
0	13	1	1591	63	0	3	4	2	3	14	287	15	19	0	37	4181.6	1.207812e+04
1	6	1	3033	63	1	3	4	2	0	14	737	15	19	0	37	7977.0	5.938688e+05
2	0	4	4589	28	1	23	4	2	3	35	674	15	19	13	15	246.0	6.238490e+04
3	6	1	4211	2	1	3	4	2	0	14	180	15	19	13	0	4435.3	4.321373e+06
4	13	4	3386	63	1	3	4	2	0	6	402	15	19	0	46	4564.4	1.212105e+06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89614	6	3	680	16	1	0	4	5	0	6	216	81	95	1	31	4092.0	6.750686e+05
89615	13	1	127	16	1	3	4	2	0	6	556	15	19	13	0	5098.0	7.772372e+05
89616	6	2	2270	76	1	3	4	2	3	15	211	81	95	1	31	4728.2	7.519390e+05
89617	4	2	4158	76	1	0	4	2	0	14	270	31	19	1	31	5236.2	9.952920e+03
89618	6	0	1740	63	1	3	4	2	0	14	410	15	19	13	15	5358.6	1.490871e+07

87156 rows × 17 columns

1	data_test
---	-----------

	통관지 세관부 호	수입통 관계획 코드	해외거 래처부 호	특송 업체 부호	수입신 고구분 코드	수입거 래구분 코드	수입 종류 코드	징수 형태 코드	운송수단 유형코드	반입보세 구역부호	HS10 단위부 호	적출국 가코드	원산지 국가코 드	관세율 구분코 드	관 세 율	신고중 량(KG)	과세가격원화 금액
0	6	4	100	63	1	3	4	2	3	9	34	81	95	0	64	3731.9	2.755287e+06
1	13	3	3033	63	1	23	4	2	3	14	760	81	95	0	0	6406.5	8.517864e+05
2	13	1	3033	63	1	0	4	2	3	14	99	81	95	0	37	5824.9	6.532926e+04
3	4	1	3033	63	1	0	4	2	3	6	58	81	95	0	14	3798.3	1.028587e+06
4	0	1	3033	63	1	0	4	2	3	0	676	83	97	29	0	3795.7	1.677178e+06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10268	16	3	2603	85	2	6	6	7	3	9	737	15	19	13	15	3063.2	2.843916e+04
10269	6	2	1920	80	1	16	7	2	0	6	180	15	19	13	13	1595.7	3.654290e+06
10270	13	6	2290	63	1	0	4	5	3	13	492	81	95	1	0	2432.3	2.926935e+06
10271	9	1	68	63	1	0	4	10	0	14	731	15	19	1	0	8484.3	1.534544e+05
10272	13	1	665	2	1	0	4	7	3	32	172	37	16	17	0	2200.4	3.277707e+05

10273 rows × 17 columns

전처리된  
Train data

전처리된  
Test data

### 3. 결과도출 2) F1 score 비교 - 파라미터 임계값 조정

#### 우범여부

```
1 X=data
2 y=train_data['우범여부']
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
4                                                    random_state=1000)
```

```
1 lgbm_wrapper = LGBMClassifier(random_state=1000)
2
3 evals = [(X_test, y_test)]
4 lgbm_wrapper.fit(X_train, y_train)
5
6 preds2 = lgbm_wrapper.predict(X_test)
7 lgbm_pred_proba = lgbm_wrapper.predict_proba(X_test)[: , 1]
```

```
1 f1 = f1_score(y_test, preds2)
2 f1
```

Out [36]: 0.36554389839867474

```
1 X=data
2 y=train_data['우범여부']
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
4                                                    random_state=1000)
```

```
1 lgbm_wrapper = LGBMClassifier(random_state=1000, n_estimators=800, num_leaves=10,
2                               learning_rate=0.05, max_depth=5, min_child_samples=40, boost_from_average=False)
3
4 evals = [(X_test, y_test)]
5 lgbm_wrapper.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set=evals, verbose=True)
```

```
6
7 preds2 = lgbm_wrapper.predict(X_test)
8 lgbm_pred_proba = lgbm_wrapper.predict_proba(X_test)[: , 1]
```

```
1 preds = [ 1 if x > 0.26 else 0 for x in lgbm_pred_proba]
2 f1 = f1_score(y_test, preds)
3 f1
```

Out [39]: 0.5541901692183724

0.3655 → 0.5542로 크게 증가

### 3. 결과도출 2) F1 score 비교 - 파라미터 임계값 조정

#### 핵심적발

```
1 y=train_data['핵심적발']
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
3                                                    random_state=1000)
4
5 params = {'num_class':3}
6
7 dtrain = xgb.DMatrix(data=X_train, label=y_train)
8 dtest = xgb.DMatrix(data=X_test, label=y_test)
9
10 wlist = [(dtrain, 'train'), (dtest, 'eval')]
11 xgb_model = xgb.train(params=params, dtrain=dtrain)
```

```
1 pred_probs = xgb_model.predict(dtest)
```

```
1 f1=f1_score(y_test, pred_probs, average='macro')
2 f1
```

Out [123]: 0.3089043787334179

0.3089 → 0.4734 로 크게 증가

```
1 y=train_data['핵심적발']
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
3                                                    random_state=1000)
4
5
6 dtrain = xgb.DMatrix(data=X_train, label=y_train)
7 dtest = xgb.DMatrix(data=X_test, label=y_test)
8
9 params = {'max_depth': 3, 'eta': 0.1, 'objective': 'multi:softprob', 'num_class':3,
10          'eval_metric': 'mlogloss'}
11
12 num_rounds=400
13
14 wlist = [(dtrain, 'train'), (dtest, 'eval')]
15 xgb_model = xgb.train(params=params, dtrain=dtrain, num_boost_round=num_rounds,
16                       early_stopping_rounds=100, evals=wlist)
```

```
1 pred_probs = xgb_model.predict(dtest)
```

```
1 pred=[]
2 for x in pred_probs:
3     if (x[1]+x[2]<0.28):
4         pred.append(0)
5     elif (x[2]>0.25):
6         pred.append(2)
7     else:
8         pred.append(1)
```

```
1 pred = pd.DataFrame(pred)
2 f1=f1_score(y_test, pred, average='macro')
```

Out [42]: 0.4733843923004462

### 3. 결과도출 3) 결론

#### 핵심적발 1, 2 구분 기준

- 핵심적발 f1-score 값이 우범여부에 비해 조금 낮았음.
- 본 경진대회에서 활용한 수입신고 기본 항목 21개 외 핵심적발 1,2를 구분할 수 있는 기준을 적용한다면 실제 핵심적발에 대한 결과값(f1-score)은 높아질 것으로 기대됨.

#### Application

- 초기에는 업무 환경에 따라 recall 값 향상에 집중하여 위험성을 낮추고, 이후, 모델의 학습 능력 향상됨에 따라 이를 조절하면 성능이 향상될 것으로 기대됨.

**감사합니다.**