관세청 경진대회 - AI_Mata 코드리뷰

0723 발표자료
- 이상윤, 송선영, 김태우

전처리

- 범주형 데이터, 수치형 데이터
- 속성의 도메인 값을 기준으로 우범률을 삽입하여 학습에 활용
- 원핫 인코딩 활용
- 이 외 중요도가 낮은 속성은 제외

```
train_set.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
```

전처리

- object 형 데이터 들의 값들은 어떻게 구성되어 있는가?

비교적 가짓수가 많은 특성 (3,4,5) 과 가짓수가 적은 특성 (그 외)

```
train_set['수입자부호'].value_counts()
```

```
GT9BIDW    221
TJLOZDO    204
XWSLOCI    197
8VEOX4Q    175
G85A40I    170
          ...
JRHOV2X      1
ESNYTOR      1
T8SIJTI      1
X4F1YX6      1
8YQEPGP      1
Name: 수입자부호, Length: 8429, dtype: int64
```

```
0  신고번호      89619 non-null  int64
1  신고일자      89619 non-null  object
2  통관지세관부호  89619 non-null  int64 39
3  신고인부호     89619 non-null  object 944
4  수입자부호     89619 non-null  object 8429
5  해외거래처부호  62966 non-null  object 4644  7자리, (원래는 13자리, 국가2+상호6+일련번호4+오류검증1)
6  특송업체부호   29787 non-null  object 90  6자리, (항공기2+일련번호4)-
7  수입통관계획코드  89619 non-null  object 7
8  수입신고구분코드  89619 non-null  object 4
9  수입거래구분코드  89619 non-null  int64  25  원핫
10 수입종류코드    89619 non-null  int64  10  원핫
11 징수형태코드    89619 non-null  int64  11  원핫
12 신고중량(KG)  89619 non-null  float64 Log변환
13 과세가격원화금액  89619 non-null  float64 Log변환
14 운송수단유형코드  89619 non-null  int64  6
15 반입보세구역부호  89619 non-null  int64  (연도 2자리 + 일련번호 6자리)
16 HS10단위부호  89619 non-null  int64  (앞 6자리 국제, 뒤 4자리 국내)
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
```

전처리

```
[ ]  # 12, 13 중량, 금액 log 화
     tdf_12 = np.log1p(newtrain_df['신고중량(KG)'])
     newtrain_df['신고중량(KG)'] = tdf_12
```

```
[ ]  tdf_13 = np.log1p(newtrain_df['과세가격원화금액'])
     newtrain_df['과세가격원화금액'] = tdf_13
```

- 중량과 가격 같은 단순한 수치형 데이터는 log처리
- 범주형 데이터는 원핫인코딩 or 우범률을 바탕으로 정리

```
tdf_7 = pd.DataFrame(newtrain_df[newtrain_df['우범여부'] != 0]['수입통관계획코드'].value_counts())
tdf_7['전체수입거래'] = newtrain_df['수입통관계획코드'].value_counts()
tdf_7 = tdf_7.reset_index()

tdf_7['per'] = ''
for i in range(len(tdf_7['index'])):
    tdf_7['per'][i] = tdf_7['수입통관계획코드'][i]/tdf_7['전체수입거래'][i]

tdf_7
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/index
  import sys
```

	index	수입통관계획코드	전체수입거래	per
0	D	6595	31522	0.209219
1	C	6497	32336	0.200922
2	E	2574	8181	0.314631
3	F	2519	8498	0.296423
4	Z	837	2145	0.39021
5	B	670	6773	0.0989222
6	H	91	164	0.554878

전처리

```
[ ]  tdf_7 = pd.DataFrame(newtrain_df[newtrain_df['우범여부'] != 0]['수입통관계획코드'].value_counts())
     tdf_7['전체수입거래'] = newtrain_df['수입통관계획코드'].value_counts()
     tdf_7 = tdf_7.reset_index()

     tdf_7['per'] = ''
     for i in range(len(tdf_7['index'])):
         tdf_7['per'][i] = tdf_7['수입통관계획코드'][i]/tdf_7['전체수입거래'][i]

     tdf_7
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
  import sys
```

	index	수입통관계획코드	전체수입거래	per
0	D	6595	31522	0.209219
1	C	6497	32336	0.200922
2	E	2574	8181	0.314631
3	F	2519	8498	0.296423
4	Z	837	2145	0.39021
5	B	670	6773	0.0989222
6	H	91	164	0.554878

우범률로 특성 값을 정리할 때,

- 각 column 별로 percent 를 구함.

- 속성 값의 종류가 적은 경우 보기 쉽다.

- 다만, 우범여부가 한 개도 검출되지 않은 속성값은 이 df에 포함되지 않게된다.

전처리

우범률(per)을 데이터 프레임에 넣어주는 함수

- 2중 for문을 사용하게 된다.

1. 각 행의 값을 바꿔야하기 때문에 (약 9만)
2. 각 행의 값에 대응하는 per값을 찾아야한다.

```
def set_data(tdf, col):
#  F_df[col] = F_df[col].astype('float')
  for i in range(len(newtrain_df['신고번호'])):
#    if i > 100: break
    for j in range(len(tdf['index'])):
      if F_df.loc[i,col] == tdf.loc[j,'index']:
        F_df.loc[i,col] = tdf.loc[j,'per']
```

이 후 set_data(tdf_2, '통관지세관부호') 와 같은 식으로 전처리

전처리

- 반입보세구역부호(연도2 + 일련번호6)
- HS코드(국가번호 + 일련번호)

㉙검사(반입)장소 - 보세구역부호	AN.. - 보세구역명칭 AN..	18 30	M C	M C	공통 서류	○ 수입물품을 검사 또는 반입할 장소 -검사 또는 반입장소의 보세구역부호와 화물의 장치위치를 18자리 이내로 기재 *타소장치장인 경우 장치위치는 '년도(2)+일련번호(6)'기재 *수입신고전 물품반출신고후 수입신고하는경우 물품반출신고시의 화물반입 보세구역부호 기재 -보세구역 명칭(신고서 출력시)	- 13011013-가1-A-123456 - 13001000275

전처리 (원핫x)

	통관지세관부호	신고인부호	해외거래처부호	특송업체부호	수입통관계획코드	수입신고구분코드	수입거래구분코드	수입종류코드	징수형태코드	신고중량(KG)	과세가격원화금액	운송수단유형코드	반입보세구역부호
0	0.187514	0.171717	0.333333	0.134126	0.200922	0.187812	0.240956	0.222002	0.179888	8.338688	9.399234	0.229221	7180
1	0.205825	0.302632	0.059055	0.134126	0.200922	0.213355	0.240956	0.222002	0.179888	8.984443	13.294415	0.193455	2001
2	0.123937	0.124503	0.333333	0.529221	0.296423	0.213355	0.298248	0.222002	0.179888	5.509388	11.041095	0.229221	2001
3	0.205825	0.166667	0.133333	0.309156	0.200922	0.213355	0.240956	0.222002	0.179888	8.397576	15.279084	0.193455	2001
4	0.187514	0.169388	0.409091	0.134126	0.296423	0.213355	0.240956	0.222002	0.179888	8.426261	14.007870	0.193455	2079
...	...	...	...	...	...	...	...	...	...	...	...	...	...
89614	0.205825	0.411765	0.666667	0.355744	0.314631	0.213355	0.167830	0.222002	0.423480	8.317033	13.422571	0.193455	6033
89615	0.187514	0.293680	0.333333	0.355744	0.200922	0.213355	0.240956	0.222002	0.179888	8.536800	13.563502	0.193455	2079
89616	0.205825	0.124503	0.101852	0.360186	0.209219	0.213355	0.240956	0.222002	0.179888	8.461511	13.530412	0.229221	6006
89617	0.292453	0.163004	0.000000	0.360186	0.209219	0.213355	0.167830	0.222002	0.179888	8.563542	9.205722	0.193455	7009
89618	0.205825	0.211364	0.200000	0.134126	0.098922	0.213355	0.240956	0.222002	0.179888	8.586645	16.517456	0.193455	2001

89619 rows × 13 columns

모든 전처리를 마친 데이터 프레임의 모습

전처리 (원핫o)

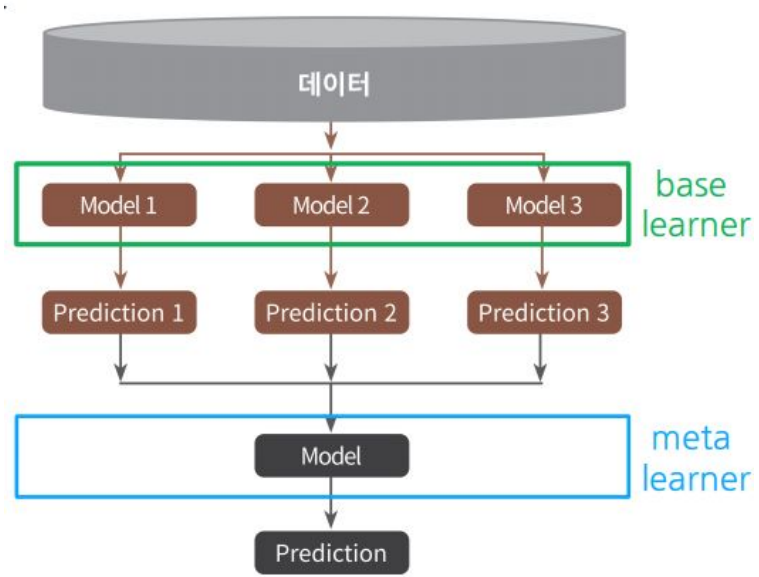
	통관지세관부호	신고인부호	수입자부호	해외거래처부호	특송업체부호	신고중량(KG)	과세가격원화금액	반입보세구역부호	수입통관계획코드_B	수입통관계획코드_C	수입통관계획코드_D	수입통관계획코드_E	수입통관계획코드_F	수입통관계획코드_H	수입통관계획코드_Z	수입신고구분코드_A	수입신고구분코드_B	수입신고구분코드_D	수입신고구분코드_E	수입거래구분코드_11	수입거래구분코드_12	수입거래구분코드_13	수입거래구분코드_15	수입거래구분코드_21	수입거래구분코드_22	수입거래구분코드_29	수입거래구분코드_51	수입거래구분코드_53	수입거래구분코드_55	수입거래구분코드_59	수입거래구분코드_71	수입거래구분코드_80	수입거래구분코드_83
0	0.187514	0.171717	0.208333	0.333333	0.134126	8.338688	9.399234	0.310345	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0.205825	0.302632	0.157025	0.059055	0.134126	8.984443	13.294415	0.101934	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	0.123937	0.124503	0.500000	0.333333	0.529221	5.509388	11.041095	0.101934	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0.205825	0.166667	0.170588	0.133333	0.309156	8.397576	15.279084	0.101934	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4	0.187514	0.169388	0.222222	0.409091	0.134126	8.426261	14.007870	0.119360	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89614	0.205825	0.411765	0.250000	0.666667	0.355744	8.317033	13.422571	0.278552	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
89615	0.187514	0.293680	0.176471	0.333333	0.355744	8.536800	13.563502	0.119360	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
89616	0.205825	0.124503	0.150000	0.101852	0.360186	8.461511	13.530412	0.303030	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
89617	0.292453	0.163004	0.136364	0.000000	0.360186	8.563542	9.205722	0.154528	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
89618	0.205825	0.211364	0.209302	0.200000	0.134126	8.586645	16.517456	0.101934	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

- 가짓수가 비교적 적은 특성을 원-핫 인코딩으로 처리한 모습

모델

- Stacking Ensemble

- 베이스 모델의 예측 값을 새로운 X값으로 받아와서 메타 모델이 최종 예측 값을 내놓는 모델



모델

베이스 모델

- Random Forest
- Decision Tree
- Ada boost
- Xgboost

```
rf_clf.fit(X_train, y_train)
dt_clf.fit(X_train, y_train)
ada_clf.fit(X_train, y_train)
xgb_clf.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set= evals, verbose=True)
```

메타 모델

- LGBM

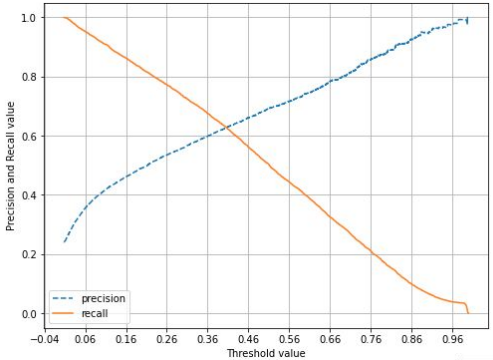
```
final= LGBMClassifier(learning_rate=0.05, max_depth=9, num_iterations=200, random_state=0)
evals = [(X_test, y_test)]

final.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="logloss", eval_set= evals, verbose=True)
stack_final = final.predict(X_test)
```

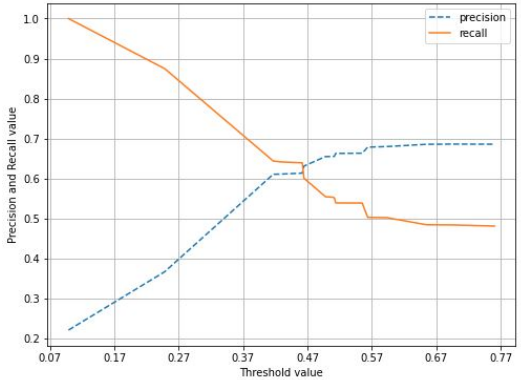
모델

- LGBM(좌) 과 스태킹 앙상블(우)

- F1 값은 비슷하게 나타나지만

  스태킹 앙상블이 좀 더 안정적인 그래프를 보여준다.





모델

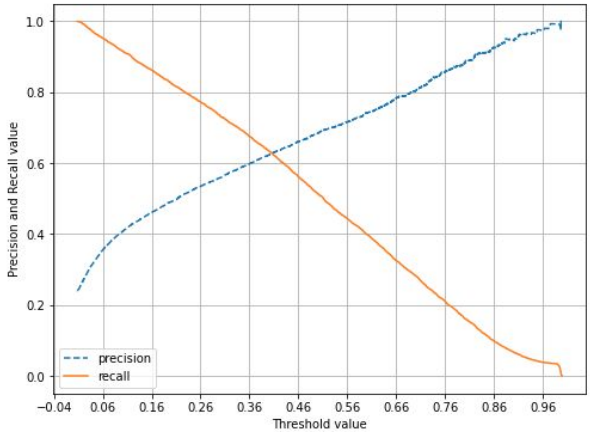
- 메타 모델 (LGBM) 으로만 예측 했을 경우의 정밀도, 재현율 그래프
- F1 값은 ...

```
binarizer = Binarizer(threshold = 0.33)
lgbm_pred_th = binarizer.fit_transform(pred_proba_)
```

```
print('최종 메타 모델의 F1: {0:.4f}'.format(f1_score(y_test, stack_final)))
print('최종 메타 모델의 임계값 조정 F1: {0:.4f}'.format(f1_score(y_test, lgbm_pred_th)))
```

```
최종 메타 모델의 F1: 0.5870
최종 메타 모델의 임계값 조정 F1: 0.6366
```

```
# 정밀도, 재현율 그래프

lgbm_pred_proba = final.predict_proba(X_test)
lgbm_pred_proba_c1 = final.predict_proba(X_test)[:,1]
precision_recall_curve_plot(y_test, lgbm_pred_proba_c1)
```



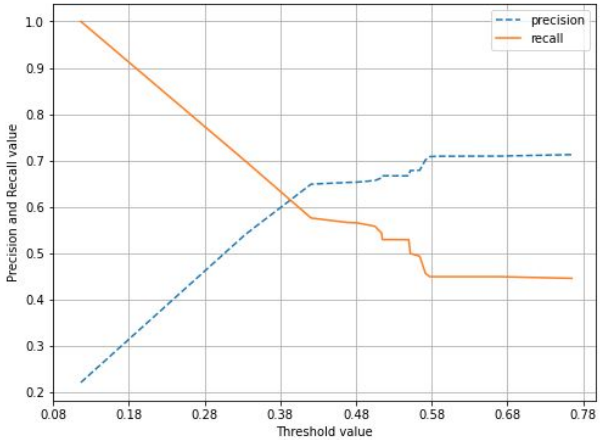
모델

```
Stack_final_X_train = np.concatenate((xgb_train, ada_train, dt_train, rf_train), axis=1)
Stack_final_X_test = np.concatenate((xgb_test, ada_test, dt_test, rf_test), axis=1)
print('원본 학습 피처 데이터 Shape : ', X_train.shape, '원본 테스트 피처 Shape : ', X_test.shape)
print('Stacking 학습 피처 데이터 Shape : ', Stack_final_X_train.shape, 'Stacking 테스트 피처 Shape : ', Stack_final_X_test.shape)
```

```
원본 학습 피처 데이터 Shape :  (71695, 14) 원본 테스트 피처 Shape :  (17924, 14)
Stacking 학습 피처 데이터 Shape :  (71695, 4) Stacking 테스트 피처 Shape :  (17924, 4)
```

- 학습 데이터를 각 베이스 모델의 예측값으로 바꿔주는 코드

```
# 정밀도, 재현율 그래프

lgbm_pred_proba = lgbm_final.predict_proba(Stack_final_X_test)
lgbm_pred_proba_c1 = lgbm_final.predict_proba(Stack_final_X_test)[:,1]
precision_recall_curve_plot(y_test, lgbm_pred_proba_c1)
```



- 메타 모델에서의 최종 결과

```
print('최종 메타 모델의 F1: {0:.4f}'.format(f1_score(y_test, stack_final)))
print('최종 메타 모델의 임계값 조정 F1: {0:.4f}'.format(f1_score(y_test, lgbm_pred_th)))
```

```
최종 메타 모델의 F1: 0.5870
최종 메타 모델의 임계값 조정 F1: 0.6238
```

핵심적발

	통관지세관부호	신고인부호	수입자부호	해외거래처부호	특송업체부호	수입통관계획코드	수입신고구분코드	수입거래구분코드	수입종류코드	징수형태코드	신고중량(KG)	과세가격원화금액	운송수단유형코드	반입보세구역부호	우범여부
0	0.187514	0.171717	0.208333	0.333333	0.134126	0.200922	0.187812	0.240956	0.222002	0.179888	8.338688	9.399234	0.229221	0.310345	0
1	0.205825	0.302632	0.157025	0.059055	0.134126	0.200922	0.213355	0.240956	0.222002	0.179888	8.984443	13.294415	0.193455	0.101934	0
2	0.123937	0.124503	0.500000	0.333333	0.529221	0.296423	0.213355	0.298248	0.222002	0.179888	5.509388	11.041095	0.229221	0.101934	1
3	0.205825	0.166667	0.170588	0.133333	0.309156	0.200922	0.213355	0.240956	0.222002	0.179888	8.397576	15.279084	0.193455	0.101934	0
4	0.187514	0.169388	0.222222	0.409091	0.134126	0.296423	0.213355	0.240956	0.222002	0.179888	8.426261	14.007870	0.193455	0.119360	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89614	0.205825	0.411765	0.250000	0.666667	0.355744	0.314631	0.213355	0.167830	0.222002	0.423480	8.317033	13.422571	0.193455	0.278552	1
89615	0.187514	0.293680	0.176471	0.333333	0.355744	0.200922	0.213355	0.240956	0.222002	0.179888	8.536800	13.563502	0.193455	0.119360	0
89616	0.205825	0.124503	0.150000	0.101852	0.360186	0.209219	0.213355	0.240956	0.222002	0.179888	8.461511	13.530412	0.229221	0.303030	0
89617	0.292453	0.163004	0.136364	0.000000	0.360186	0.209219	0.213355	0.167830	0.222002	0.179888	8.563542	9.205722	0.193455	0.154528	0
89618	0.205825	0.211364	0.209302	0.200000	0.134126	0.0989222	0.213355	0.240956	0.222002	0.179888	8.586645	16.517456	0.193455	0.101934	0

89619 rows × 15 columns

- 우범여부까지 추가한 모습

핵심적발

- 우범여부(Y1)와 핵심적발(Y2)을 다른 모델로 예측해보자는 목적으로 실시함.
- 우범여부가 1일 경우 핵심적발이 1 또는 2의 값을 갖는 만큼, Y2를 예측함에 있어서 가장 연관이 큰 Feature는 다름아닌 Y1이 된다.

```
print('최종 메타 모델의 F1: {0:.4f}'.format(f1_score(y_test2, stack_final2, average='macro')))
```

```
최종 메타 모델의 F1: 0.6736
```

핵심적발

- 0,1,2 총 3개의 값 중 하나를 예측해야함
- 이진 분류 모델로는 예측 불가능

-> 0,1,2 중 1 또는 2의 편향성이 짙은 이유라고 생각됨

```
test_result.to_csv('AI_Mata_4차제출_1.csv', index=False)
```

```
test_result['핵심적발'].value_counts()
```

```
0    8720
2    1286
1     267
Name: 핵심적발, dtype: int64
```

끝