



관세청 경진대회 재리뷰

0813 랩세미나 이상윤



계기

왜 실험값과 실제 예측값의 격차가 컸을까?

실험값 F1: 0.6238

실제 데이터 예측값 F1: 0.3968

경우의 수

1. 데이터를 잘못 대입 하였다.
2. 방법이 잘못 되었다.

per : train.csv 의 확률
t_per : test.csv 의 확률

```
[120] result3 = result3.fillna(0)
      result3[['우범', '전체']] = result3[['우범', '전체']].astype(int)
      result3[result3['per'] - result3['t_per'] > 0.15]
```

	index	신고인부호	전체부호	per	우범	전체	t_per
54	0Q91O	67	274	0.244526	3	34	0.088235
85	DQZRG	53	166	0.319277	2	19	0.105263
102	2HCPY	45	182	0.247253	2	21	0.095238
106	FWWE2	43	134	0.320896	2	18	0.111111
112	OC7YM	42	180	0.233333	1	16	0.062500
...
912	IQV4O	1	5	0.200000	0	0	0.000000
913	4M21C	1	6	0.166667	0	0	0.000000
915	3UAA4	1	2	0.500000	0	0	0.000000
919	U8FBG	1	6	0.166667	0	0	0.000000
920	6TYJM	1	3	0.333333	0	0	0.000000

316 rows × 7 columns

교수님의 조언



Kunwoo Park

나에게 ▾

2021. 7. 26. 오전 9:33



상윤 학생,

1. 테스트 셋에서의 변환된 피쳐값을 추론에 썼다는 뜻인가요? Train set 기준으로 뽑힌 값을 써야할 것입니다.
2. 해당 문제는 train.csv 에서 스플릿 실험하는 과정에서도 성능이 안 좋게 나왔을 텐데 split 을 어떻게 했나요.
3. 혹시 feature를 전체 train.csv 에서 뽑고 split만 했나요.
4. 각 변환값들이 지니는 의미를 생각해보고, 실제 관세업무에서 말이 되는 것들 위주로 포함해서 실험을 다시 해보세요.
상윤학생이 제안한 변환 방법이 맞은 되는데, 모든 카테고리 변수에 잘 적용될지는 의문이기도 합니다. 변환할 때도 다른 팀처럼 있는 카테고리를 다 쓰지 말고 몇개의 그룹을 지어서 비율을 계산할 수도 있을 것입니다.

1. test set을 기준으로 변환을 하였는가? train set을 기준으로 써야한다.
2. (1) test set을 기준으로 변환했다면 split 실험하는 과정에서도 성능이 좋지 않게 나왔을 것이다.
3. 혹시 feature를 전체 train.csv에서 뽑고 split만 했는가?



기존 방식

1. 범주형 데이터(ex. 남녀)와 수치형 데이터(ex. 키몸무게)의 구분

-> 무게와 가격은 log 취함. 나머지는 확률로 치환

2. train_test_split 으로 train set, test set 분리

3.1. train set을 개별 모델로 성능 측정

RandomForest F1 : 0.5681

DecisionTree F1 : 0.5209

Adaboost F1 : 0.5754

Xgboost F1 : 0.5588

LightGBM F1 : 0.5858

3.2. Stacking Ensemble 사용 (KFold 활용)



문제점



- Train Set과 Test Set을 나눈 이유?
=>전처리와 Split의 순서가 잘못되었다.

문제점

$n = 8$



Test



Train

Model 1



- KFold를 진행할 때 역시, train을 기반으로 전처리 하여 test도 전처리 해야한다.
-

실험 과정..

- train.csv 전체 전처리 : 3시간
- train set -> KFold -> train set 를 기준으로 전처리

[18] preprocessing(X_train)

df.shape[0] : 39
통관지세관부호
df.shape[0] : 942
신고인부호
df.shape[0] : 8275
수입자부호
df.shape[0] : 4588
해외거래처부호
df.shape[0] : 90
특송업체부호
df.shape[0] : 7
수입통관계획코드
df.shape[0] : 25
수입거래구분코드
df.shape[0] : 10
수입종류코드
df.shape[0] : 11
질수형태코드
df.shape[0] : 6
운송수단유형코드
df.shape[0] : 408
반입보세구역부호

✓ 2시간 34분 4초 오전 5:52에 완료됨

[64] preprocessing(X_test2)

df.shape[0] : 39
통관지세관부호
df.shape[0] : 942
신고인부호
df.shape[0] : 8275
수입자부호
df.shape[0] : 4588
해외거래처부호
df.shape[0] : 90
특송업체부호
df.shape[0] : 7
수입통관계획코드
df.shape[0] : 25
수입거래구분코드
df.shape[0] : 10
수입종류코드
df.shape[0] : 11
질수형태코드
df.shape[0] : 6
운송수단유형코드
df.shape[0] : 408
반입보세구역부호

✓ 40분 19초 오전 7:47에 완료됨

실험 결과

- train set의 feature로 전처리, 학습을 한 test set의 LGBM 결과..

=> F1 스코어가 실제 측정 결과와 비슷하게 나온다.

```
lgbm_pre = lgbm_clf.predict(X_test2.iloc[:, :-2])

print('F1: {:.4f}'.format(f1_score(X_test2.iloc[:, :-2], lgbm_pre)))

F1: 0.3741
```

추론

- train set을 학습한 lgbm으로 test set을 예측하려하자 에러가 난 모습.

```
In [62]: F_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10273 entries, 0 to 10272
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  -
 0   통관지세관부호      10273 non-null float64
 1   신고인부호          10273 non-null object
 2   수입자부호          10273 non-null object
 3   해외거래처부호      10273 non-null object
 4   특송업체부호        10273 non-null object
 5   수입통관계획코드    10273 non-null object
 6   수입거래구분코드    10273 non-null float64
 7   수입종류코드        10273 non-null float64
 8   짐수형태코드        10273 non-null float64
 9   운송수단유형코드    10273 non-null float64
10   반입보세구역부호    10273 non-null float64
11   신고총량(KG)        10273 non-null float64
12   과세가격원화금액    10273 non-null float64
13   우벌여부            0 non-null     float64
14   핵심적발            0 non-null     float64
dtypes: float64(10), object(5)
memory usage: 1.2+ MB
```

```
In [45]: lgbm_pre = lgbm_clf.predict(F_df.iloc[:, :-2])
```

```
print('F1: {:.4f}'.format(f1_score(golden_df['우벌여부'], lgbm_pre)))
```

```
D:\anaconda3\lib\site-packages\lightgbm\basic.py in predict(self, data, start_iteration, num_iteration, raw_score, pred_leaf, pred_contrib, data_has_header, is_reshape)
    554     if isinstance(data, Dataset):
    555         raise TypeError("Cannot use Dataset instance for prediction, please use raw data instead")
--> 556     data = _data_from_pandas(data, None, None, self.pandas_categorical)[0]
    557     predict_type = C_API_PREDICT_NORMAL
    558     if raw_score:

D:\anaconda3\lib\site-packages\lightgbm\basic.py in _data_from_pandas(data, feature_name, categorical_feature, pandas_categorical)
    393     bad_indices = _get_bad_pandas_dtypes(data.dtypes)
    394     if bad_indices:
--> 395         raise ValueError("DataFrame.dtypes for data must be int, float or bool. %n"
    396                           "Did not expect the data types in the following fields: "
    397                           + ', '.join(data.columns[bad_indices]))
```

```
ValueError: DataFrame.dtypes for data must be int, float or bool.
Did not expect the data types in the following fields: 신고인부호, 수입자부호, 해외거래처부호, 특송업체부호, 수입통관계획코드
```

전처리가 끝났음에도 object 가 남아있다.

추론

- 위의 속성에서 **train set**에서 볼 수 없었던 값이 발견됨

이를 임의로 모두 0으로 처리하고 학습시키자,
결과는 그림과 같이 나오게 된다.

```
[92] for i in range(X_test2.shape[0]):  
      # if type(X_test2.iloc[i, 1]) == 'numpy.float64':  
      if  
          print('float')
```

```
[115] X_test2['신고인부호'] = pd.to_numeric(X_test2['신고인부호'], errors='coerce')  
X_test2['수입자부호'] = pd.to_numeric(X_test2['수입자부호'], errors='coerce')  
X_test2['해외거래처부호'] = pd.to_numeric(X_test2['해외거래처부호'], errors='coerce')  
X_test2['특송업체부호'] = pd.to_numeric(X_test2['특송업체부호'], errors='coerce')  
X_test2['수입통관계획코드'] = pd.to_numeric(X_test2['수입통관계획코드'], errors='coerce')
```

```
[117] X_test2 = X_test2.fillna(0)
```

```
[118] lgbm_pre = lgbm_clf.predict(X_test2.iloc[:, :-2])  
  
print('F1: {0:.4f}'.format(f1_score(X_test2.iloc[:, -2], lgbm_pre)))  
  
F1: 0.3741
```

test.csv 의 데이터도, golden 과 비교해보면 비슷한 결과가 나온다.



결론

- 시도했던 방법이 좋은 성능을 내지 못함을 확인했다.

하지만, 이는 어디까지나 모든 범주형 데이터를 확률 값으로 치환했을 때의 이야기이며, 다른 조합으로 전처리를 진행하면 결과가 나아질 수도 있으리라 생각한다.