

Discussion 분석

amazon

01. 데이터 크기 줄이기

	원본데이터	Parquet	Feather
전체 데이터	50.31GB	10.41GB	5.4GB
Train	16.39GB	3.35GB	1.8GB
Test	33.82GB	6.96GB	3.6GB

참고 discussiong

Parquet : <https://www.kaggle.com/code/odins0n/load-parquet-files-with-low-memory/>

Parquet + Feather : <https://www.kaggle.com/datasets/ruchi798/parquet-files-amexdefault-prediction>

Parquet + Feather : <https://www.kaggle.com/datasets/raddar/amex-data-integer-dtypes-parquet-format>

02. EDA

train_data.csv의 모든 column 나열 (188개) (ID, S_2제외) :

- 범주형 : ['B_30', 'B_38', 'D_63', 'D_64', 'D_66', 'D_68', 'D_114', 'D_116', 'D_117', 'D_120', 'D_126'],
- 연속형 : ['P_2', 'D_39', 'B_1', 'B_2', 'R_1', 'S_3', 'D_41', 'B_3', 'D_42', 'D_43', 'D_44', 'B_4', 'D_45',
'B_5', 'R_2', 'D_46', 'D_47', 'D_48', 'D_49', 'B_6', 'B_7', 'B_8', 'D_50', 'D_51', 'B_9', 'R_3',
'D_52', 'P_3', 'B_10', 'D_53', 'S_5', 'B_11', 'S_6', 'D_54', 'R_4', 'S_7', 'B_12', 'S_8', 'D_55',
'D_56', 'B_13', 'R_5', 'D_58', 'S_9', 'B_14', 'D_59', 'D_60', 'D_61', 'B_15', 'S_11', 'D_62',
'D_65', 'B_16', 'B_17', 'B_18', 'B_19', 'B_20', 'S_12', 'R_6', 'S_13', 'B_21', 'D_69', 'B_22',
'D_70', 'D_71', 'D_72', 'S_15', 'B_23', 'D_73', 'P_4', 'D_74', 'D_75', 'D_76', 'B_24', 'R_7',
'D_77', 'B_25', 'B_26', 'D_78', 'D_79', 'R_8', 'R_9', 'S_16', 'D_80', 'R_10', 'R_11', 'B_27',
'D_81', 'D_82', 'S_17', 'R_12', 'B_28', 'R_13', 'D_83', 'R_14', 'R_15', 'D_84', 'R_16', 'B_29',
'S_18', 'D_86', 'D_87', 'R_17', 'R_18', 'D_88', 'B_31', 'S_19', 'R_19', 'B_32', 'S_20', 'R_20',
'R_21', 'B_33', 'D_89', 'R_22', 'R_23', 'D_91', 'D_92', 'D_93', 'D_94', 'R_24', 'R_25', 'D_96',
'S_22', 'S_23', 'S_24', 'S_25', 'S_26', 'D_102', 'D_103', 'D_104', 'D_105', 'D_106', 'D_107',
'B_36', 'B_37', 'R_26', 'R_27', 'D_108', 'D_109', 'D_110', 'D_111', 'B_39', 'D_112', 'B_40',
'S_27', 'D_113', 'D_115', 'D_118', 'D_119', 'D_121', 'D_122', 'D_123', 'D_124', 'D_125',
'D_127', 'D_128', 'D_129', 'B_41', 'B_42', 'D_130', 'D_131', 'D_132', 'D_133', 'R_28', 'D_134',
'D_135', 'D_136', 'D_137', 'D_138', 'D_139', 'D_140', 'D_141', 'D_142', 'D_143', 'D_144', 'D_145']

D_* (96개) Delinquency 연체 변수	범주형 63, 64, 66, 68, 114, 116, 117, 120, 126, 연속형 39, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 61, 62, 65, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 86, 87, 88, 89, 91, 92, 93, 94, 96, 102, 103, 104, 105, //106, 107, 108, 109, 110, 111, 113, 115, 118, 119, 121, 122, 123, 124, 125, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145 정수형 112
S_* (21개) Spend 지출 변수	시계열 2, 연속형 3, 5, 6, 7, 9, 12, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27 정수형 8, 11, 13 ([S_11]은 15, 16, 17 값이 존재한다.)
P_* (3개) Payment 지불 변수	연속형 2, 3, 4
B_* (38개) Balance 균형 변수	범주형 30, 38, 연속형 1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 32, 33, 36, 37, 39, 40, 41, 42 정수형 8, 18, 19
R_* (28개) Risk 위험 변수	연속형 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28 정수형 26 ([R_26]은 0.2 이하의 유일한 값이 548160이 있다. 그러나 실제로 확인한 결과 노이즈가 추가된 6개의 구분된 값이었다. 그래서 int8로 변환이 가능하다.)

→ 익명화되고, 부호화 된 데이터이지만 시계형, 범주형, 연속형, 정수형 데이터를 구분했다.

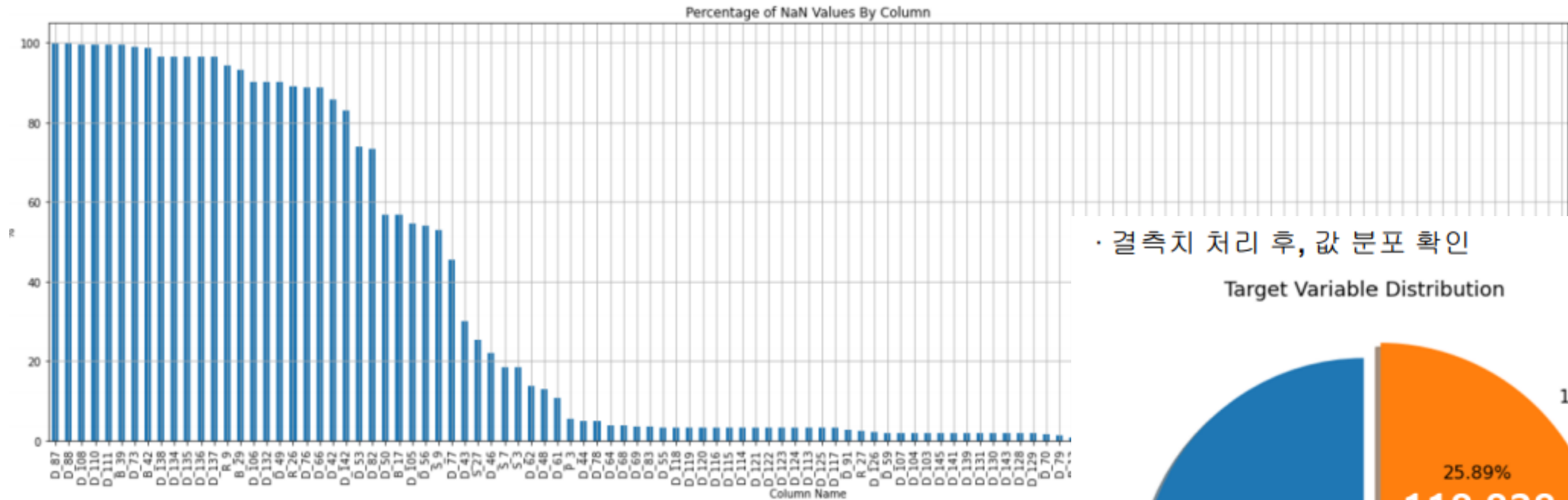
정수 유형을 찾는 discussion

- <https://www.kaggle.com/code/raddar/the-data-has-random-uniform-noise-added>
- <https://www.kaggle.com/code/cdeotte/xgboost-starter-0-793>
- <https://www.kaggle.com/competitions/amex-default-prediction/discussion/328514>

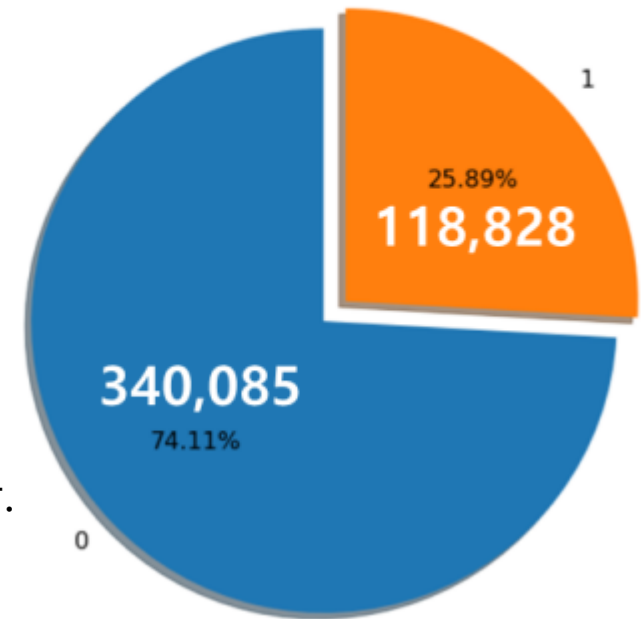
데이터 노이즈를 제거한 데이터셋

- <https://www.kaggle.com/datasets/raddar/amex-data-integer-dtypes-parquet-format>

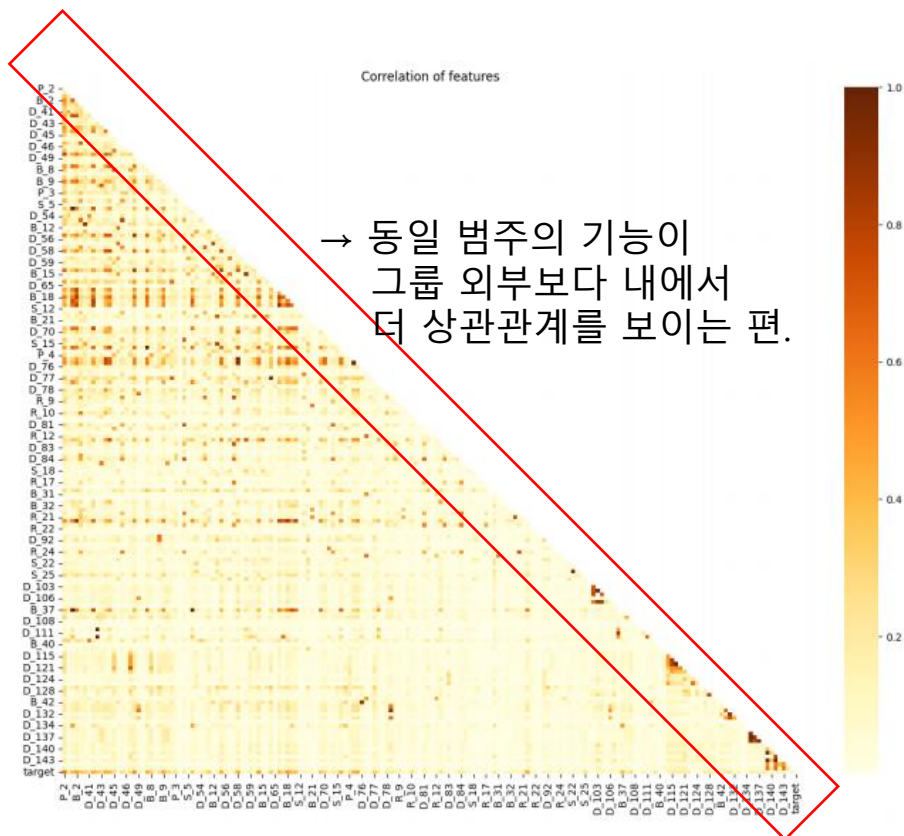
Missig data



- 결측치 수가 전체의 50% 이상인 feature 들이 꽤 많다.
- 결측치 비율이 80%가 넘는 23개의 columns를 제거해준다.
- 결측치 처리 이후 target 값은 1(25%), 0(75%) 분포로 나타났다.



→ unstack



→ 동일 범주의 기능이
그룹 외부보다 내에서
더 상관관계를 보이는 편.

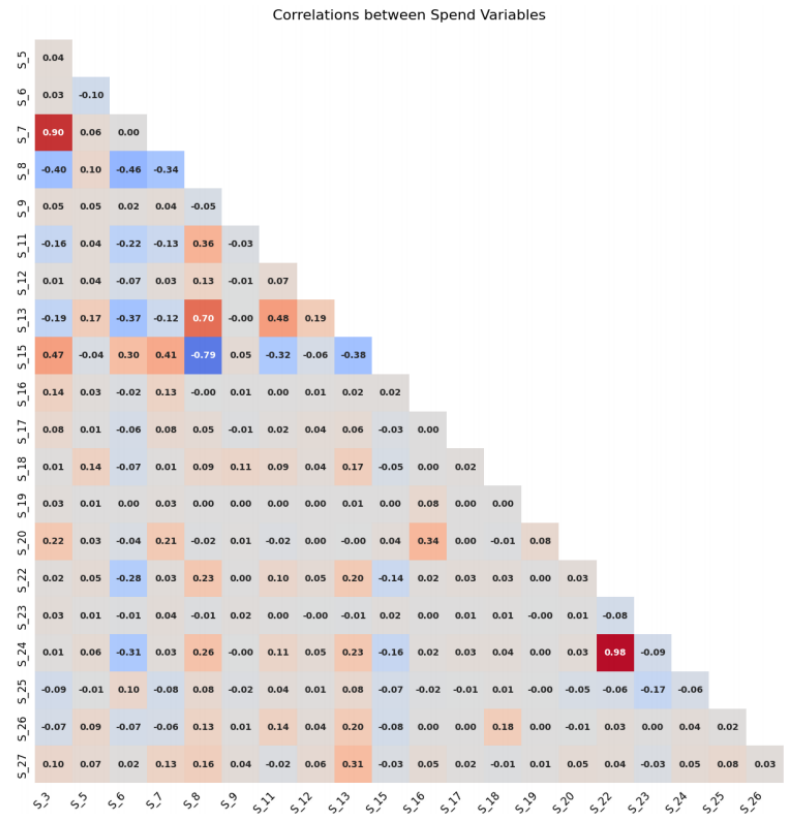
```
In [38]: unstacked = correlations.unstack()
unstacked = unstacked.sort_values(ascending=False, kind="quicksort").drop_duplicates().head(25)
unstacked
```

- 상관관계를 찾아보기 위해 heatmap으로 시각화해봤으나 상관관계가 뚜렷하지 않음.
- 일부 색이 진한 점들이 있는데, 이런 다시 결과값을 unstack 해줘야 함.

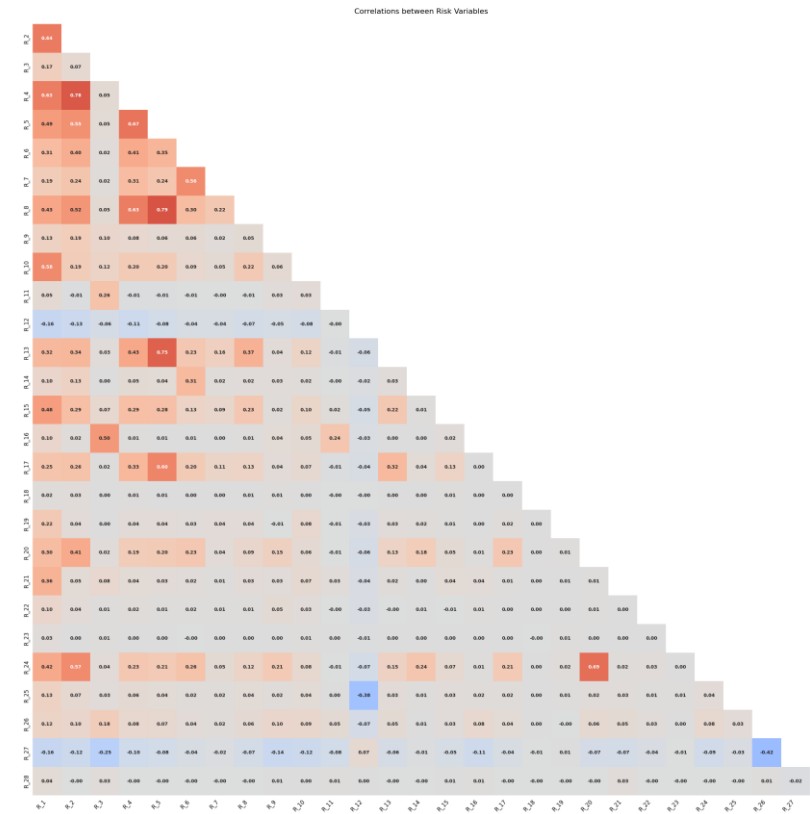
discussion - EDA : 변수 간 상관관계 확인

<https://www.kaggle.com/competitions/amex-default-prediction/discussion/338569>

<https://www.kaggle.com/code/tawejsash/american-express-default-prediction>



S_* 상관관계




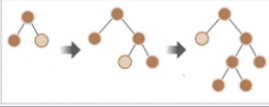
R_* 상관관계

다루지 못한 중간 discussion 페이지

- Graphical explanation of the competition metric
평가지표 라인
- 유용한 노트북 링크 (팀 내용 / discussion 내용과 겹치는 부분 多)
<https://www.kaggle.com/competitions/amex-default-prediction/discussion/328565>
- DART 알고리즘은 LGBM 향상에 유용합니다.
<https://www.kaggle.com/competitions/amex-default-prediction/discussion/334670>

CatBoost

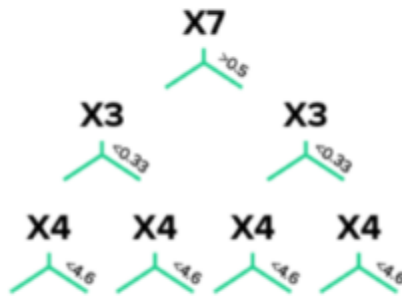
Part 3 >> Generate Machine Learning Model

			
1 Random Forest	2 XGBoost	3 LightGBM	4 CatBoost
ensemble ⊃ bagging ⊃ RF	Ensemble ⊃ boosting ⊃ GBM ⊃ XGBoost, LightGBM		
<ul style="list-style-type: none">subsample에 대해 다수의 결정 트리 classifier를 최적화여러 추론 결과 평균 → 예측 정확도를 개선 → 과적합방지hyperparameter xgbm보다 적음	<ul style="list-style-type: none">트리기반 앙상블 균형트리분할방식병렬학습이 가능 → GBM의 단점인 오랜 수행시간을 극복예측 성능 우수가지치기 기능 존재 조기중단 기능 존재	<ul style="list-style-type: none">트리기반 앙상블 리프중심트리분할방식XGBoost보다 더 빠르고 메모리 사용량 적음예측 성능 우수10000건 이하의 dataset 처리시 과적합 가능성 ↑	<ul style="list-style-type: none">Categorical Boosting 범주형 feature 처리중점Gradient Boosting 기반 XGB, LGBM보다 우수Feature 자동 타깃인코딩망각 결정 트리 사용 (oblivious decision tree)

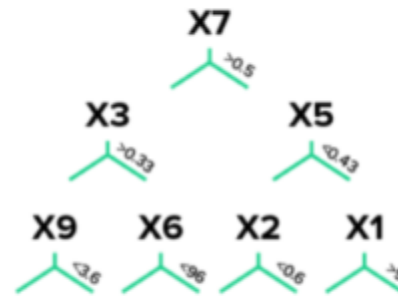
→ CatBoost만 떼어 제가 말았기에 조금 보완을 해봤습니다.

CatBoost, XGBoost, LGBM 비교

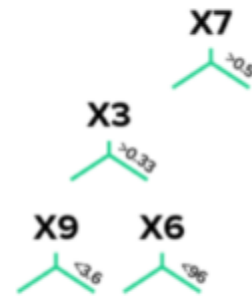
Tree growth examples:



CatBoost



XGBoost



LightGBM

Catboost vs XGBoost vs LightGBM

LGBM은 DFS(깊이 우선 탐색)으로 트리를 깊게 형성
반면 XGBoost는 BFS(너비 우선 탐색)으로 넓게 트리를 형성
XGBoost와 CatBoost의 차이는 트리 구조의 대칭 여부에 따라 차이
Feature를 모두 동일하게 대칭 구조로 형성하면 비합리적으로 보일 수 있으나
예측 시간을 감소하여 Boosting 계열 알고리즘의 느린 학습 속도를 보완.

CatBoost 특징, Parameter

- XGBoost처럼 트리를 병렬적으로 학습을 수행해나가기 때문에 기본적으로 학습 속도는 느린 편.
- 데이터 대부분이 수치형인 경우 LGBM보다 학습 속도가 느리다.
- AMEX에서는 LGBM이 가장 잘 나온다는 discussion이 많았음.

Parameter	
cat_features	범주형 피처를 직접 인코딩하고 열 인덱스를 cat_features로 전달하지 않는 경우 범주형 피처의 catboost 전처리를 활용하기 위해 필요
one_hot_max_size	
learning_rate & n_estimators	Learning_rate가 작을수록 모델을 활용하는데 n_estimator가 더 많이 필요.
max_depth	기본 트리 깊이로 매개 변수 train 시간에 영향
subsample	
colsample_bylevel, colsample_bytree, colsample_bynode	열의 표본 비율
l2_leaf_reg	L2 정규화 계수
random_strength	random+_strength는 점수에 임의성을 추가해 과적합을 줄일 수 있음

→ 파라미터는 많이 변경을 안 해봐서 크게 자신은 없습니다.

출처: <https://blog.naver.com/dlekdms7931/222719354333>

CatBoost CPU & GPU

No.	내용요약	소요시간	Parameter	score
1	Base Catboost	1128.0s - CPU	Iter = 1000 Verbose=100	0.784
2	Catboost_GPU (V1)	258.2s-GPU	Bagging_temperature = 0.2 Iter = 3000 Vervose=True	0.786
3	Catboost GPU(V2)	280.6s-GPU	Bagging_temperature = 0.2 Random_state=22 Iter = 5000 Vervose=True	0.786
4	Ensemble 가중평균 Catboost + BaseXGBoost + BaseLGBM	41.2s		0.787

Speed Up XGB, CatBoost, and LGBM by 20x

<https://www.kaggle.com/competitions/amex-default-prediction/discussion/328606>

► CatBoost GPU

Kaggle jupyter 노트북에서 가속기를 GPU로 선택하고, 다음으로 분류기를 생성할 때, task_type='GPU'를 추가.

```
clf = CatBoostClassifier(iterations=5000, random_state=22,
                        task_type = 'GPU')
```

8시간 아닌 25분으로 속도 향상.

메모리 줄이기

```
9]:  
#cleanup  
del X_train, X_val, y_train, y_val  
gc.collect()
```

```
9]:  
42
```

수업 때 다룬 것처럼 gc.collect()를 통해 RAM을 비워주면서 진행했기에 Test data를 불러와서 최종 제출하는 부분까지의 뒷쪽이 실행되었음.