HOME CREDIT - CREDIT RISK MODEL STABILITY

채무 불이행 고객 예측

kaggle

Search

Create

Home

Competitions

Datasets

Models

Code

Discussions

Learn

More

Your Work

VIEWED

Home Credit - Credit ...

Home Credit 2024 Sta...

AI Mathematical Olym...

Model Validation

Your First Machine Le...

EDITED

Home Credit 2024 Sta...

Exercise: Syntax, Varia...

HOME CREDIT GROUP · FEATURED CODE COMPETITION · 23 DAYS TO GO

Submit Prediction

Home Credit - Credit Risk Model Stability

Create a model measured against feature stability over time

Overview Data Code Models Discussion Leaderboard Rules Team Submissions

Overview

The goal of this competition is to predict which clients are more likely to default on their loans. The evaluation will favor solutions that are stable over time.

Your participation may offer consumer finance providers a more reliable and longer-lasting way to assess a potential client's default risk.

Start
3 months ago

Close
23 days to go

Merger & Entry

Description

The absence of a credit history might mean a lot of things, including young age or a preference for cash. Without traditional data, someone with little to no credit history is likely to be denied. Consumer finance providers must accurately determine which clients can repay a loan and which cannot and data is key. If data science could help better predict one's repayment capabilities, loans might become more accessible to those who may benefit from them the most.

Currently, consumer finance providers use various statistical and machine learning methods to predict loan risk. These

Competition Host
Home Credit Group

Prizes & Awards
$105,000
Awards Points & Medals

Participation
25,178 Entrants
3,680 Participants
2,916 Teams
44,444 Submissions

Tags

Tabular Banking Custom Metric

Table of Contents

HOME CREDIT GROUP

다국적 금융 회사

신용도가 낮은 소비자에게 대출 제공

라이브러리 및 데이터 불러오기

```python
import polars as pl
import numpy as np
import pandas as pd
import lightgbm as lgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score

dataPath = "C://Users//noon9//Untitled Folder//home-credit-credit-risk-model-stability//"
```

대용량 데이터를 처리하는 라이브러리

TRAIN_STATIC

```
pl.read_csv(dataPath + "csv_files//train//train_static_0_0.csv")
```

shape: (1_003_757, 168)

case_id	actualdpdtolerance_344P	amtinstpaidbefduel24m_4187115A	annuity_780A	annuitynextmonth_57A	applicationcnt_361L	applications30d_658L	applicati
i64	f64	str	f64	f64	f64	f64	
0	null	null	1917.6	0.0	0.0	0.0	
1	null	null	3134.0	0.0	0.0	0.0	
2	null	null	4937.0	0.0	0.0	0.0	
3	null	null	4643.6	0.0	0.0	1.0	
4	null	null	3390.2	0.0	0.0	1.0	
...	...	...	...	...	...	...	
2651088	0.0	"117624.79"	3045.0	4488.0	0.0	0.0	
2651089	0.0	"134887.4"	1200.0	4382.8003	0.0	0.0	
2651090	0.0	"69186.62"	6000.0	0.0	0.0	0.0	
2651091	0.0	"117331.0"	11565.4	13781.2	0.0	0.0	
2651092	0.0	"44442.2"	3045.0	3000.0	0.0	1.0	

TEST_STATIC

```
pl.read_csv(dataPath + "csv_files//test//test_static_0_0.csv")
```

shape: (10, 168)

case_id	actualdpdtolerance_344P	amtinstpaidbefduel24m_4187115A	annuity_780A	annuitynextmonth_57A	applicationcnt_361L	applications30d_658L	applicati
i64	f64	f64	f64	f64	f64	f64	
57543	0.0	191767.36	3674.6	1218.2001	0.0	0.0	
57551	0.0	71036.4	2844.6	0.0	0.0	1.0	
57552	0.0	183992.0	6298.8003	12155.4	0.0	0.0	
57569	0.0	0.0	4682.6	0.0	0.0	1.0	
57630	0.0	0.0	8905.0	0.0	0.0	0.0	
57631	0.0	null	2540.6	0.0	0.0	0.0	
57632	0.0	63647.402	4732.0	0.0	0.0	0.0	
57635	0.0	null	1167.4	0.0	0.0	0.0	
57637	0.0	43677.184	4300.4	0.0	0.0	0.0	
57639	0.0	142333.14	12599.601	0.0	0.0	0.0	

case_id - 신용 건별 고유 식별자

date_decision - 대출승인이 결정됐던 날

Target - 고객이 특정 신용건(대출)에 대한 채무불이행 여부

WEEK_NUM - 주 단위로 데이터를 집계하는 데 사용되는 주 번호

MONTH - 월 단위 집계

num_group - 과거 기록을 위한 인덱싱

- **P** - Transform DPD (Days past due)
- **M** - Masking categories
- **A** - Transform amount
- **D** - Transform date
- **T** - Unspecified Transform
- **L** - Unspecified Transform

. pmts_month_158T : 활성계약에 대한 지불 내역

pmts_month_706T : 종료계약에 대한 지불 내역

dateofcredstart_181D : 신용계약의 시작날짜

데이터 타입 설정 함수 (POLARS)

```python
def set_table_dtypes(df: pl.DataFrame) -> pl.DataFrame:

    for col in df.columns:
        if col[-1] in ("P", "A"):
            df = df.with_columns(pl.col(col).cast(pl.Float64).alias(col))
    return df
```

컬럼 이름의 마지막 문자가 'P' 또는 'A'인 경우, Float64로 변환

변환 함수 (PANDAS)

```python
def convert_strings(df: pd.DataFrame) -> pd.DataFrame:
    for col in df.columns:
        if df[col].dtype.name in ['object', 'string']:

            df[col] = df[col].astype("string").astype('category')
            current_categories = df[col].cat.categories
            new_categories = current_categories.to_list() + ["Unknown"]
            new_dtype = pd.CategoricalDtype(categories=new_categories, ordered=True)
            df[col] = df[col].astype(new_dtype)
    return df
```

문자열 컬럼을 범주형(카테고리)데이터로 변환

"Unknown" 카테고리 추가

트레인 데이터 로드 및 전처리

```
train_basetable = pl.read_csv(dataPath + "csv_files//train//train_base.csv")
train_static = pl.concat(
    [
        pl.read_csv(dataPath + "csv_files//train//train_static_0_0.csv").pipe(set_table_dtypes),
        pl.read_csv(dataPath + "csv_files//train//train_static_0_1.csv").pipe(set_table_dtypes),
    ],
    how="vertical_relaxed",
)
train_static_cb = pl.read_csv(dataPath + "csv_files//train//train_static_cb_0.csv").pipe(set_table_dtypes)
train_person_1 = pl.read_csv(dataPath + "csv_files//train//train_person_1.csv").pipe(set_table_dtypes)
train_credit_bureau_b_2 = pl.read_csv(dataPath + "csv_files//train//train_credit_bureau_b_2.csv").pipe(set_table_dtypes)
```

TRAIN_PERSON_1

train_person_1

shape: (2_973_991, 37)

case_id	birth_259D	birthdate_87D	childnum_185L	contaddr_district_15M	contaddr_matchlist_1032L	contaddr_smempladdr_334L	contaddr_zipcode_807M	ed
i64	str	str	f64	str	bool	bool	str	
0	"1986-07-01"	null	null	"P88_18_84"	false	false	"P167_100_165"	
0	null	null	null	"a55475b1"	null	null	"a55475b1"	
0	null	null	null	"a55475b1"	null	null	"a55475b1"	
0	null	null	null	"a55475b1"	null	null	"a55475b1"	
1	"1957-08-01"	null	null	"P103_93_94"	false	false	"P176_37_166"	
...	...	...	...	...	...	...	...	
2703451	null	null	null	"a55475b1"	null	null	"a55475b1"	
2703452	"1977-08-01"	null	null	"P133_44_167"	false	false	"P59_150_74"	
2703453	"1950-02-01"	null	null	"P123_6_84"	false	false	"P46_103_143"	
2703453	null	null	null	"a55475b1"	null	null	"a55475b1"	
2703454	"1948-04-01"	null	null	"P48_127_19"	false	false	"P78_144_175"	

TRAIN_PERSON_1 테이블 그룹화 및 집계

```
train_person_1_feats_1 = train_person_1.group_by("case_id").agg(
    pl.col("mainoccupationinc_384A").max().alias("mainoccupationinc_384A_max"),
    (pl.col("incometype_1044T") == "SELFEMPLOYED").max().alias("mainoccupationinc_384A_any_selfemployed")
)
train_person_1_feats_1
```

shape: (1_526_659, 3)

case_id	mainoccupationinc_384A_max	mainoccupationinc_384A_any_selfemployed
i64	f64	bool
2535791	72000.0	false
1005921	22000.0	false
1301441	40000.0	false
1357092	80000.0	false
785388	77000.0	false
…	…	…
1768097	34000.0	false
909602	30000.0	false
1412642	60000.0	false
775017	40000.0	false
1798099	40000.0	false

TRAIN_PERSON_1 테이블 필터링 및 열 선택

```
train_person_1_feats_2 = train_person_1.select(["case_id", "num_group1", "housetype_905L"]).filter(
    pl.col("num_group1") == 0
).drop("num_group1").rename({"housetype_905L": "person_housetype"})
train_person_1_feats_2
```

shape: (1_526_659, 2)

case_id	person_housetype
i64	str
0	null
1	null
2	null
3	null
4	null
…	…
2703450	"OWNED"
2703451	null
2703452	null
2703453	null
2703454	null

TRAIN_CREDIT_BUREAU_B_2

train_credit_bureau_b_2

shape: (1_286_755, 6)

case_id	num_group1	num_group2	pmts_date_1107D	pmts_dpdvalue_108P	pmts_pmtsoverdue_635A
i64	i64	i64	str	f64	f64
467	0	0	"2018-11-15"	null	null
467	0	1	"2018-12-15"	null	null
467	1	0	"2018-12-15"	null	null
467	2	0	"2016-10-15"	0.0	0.0
467	2	1	"2016-11-15"	0.0	0.0
…	…	…	…	…	…
2703436	1	31	"2020-05-15"	0.0	0.0
2703436	1	32	"2020-06-15"	0.0	0.0
2703436	1	33	"2020-07-15"	0.0	0.0
2703436	1	34	"2020-08-15"	0.0	0.0
2703436	1	35	"2020-09-15"	0.0	0.0

TRAIN_CREDIT_BUREAU_B_2 테이블 그룹화 및 집계

```
train_credit_bureau_b_2_feats = train_credit_bureau_b_2.group_by("case_id").agg(
    pl.col("pmts_pmtsoverdue_635A").max().alias("pmts_pmtsoverdue_635A_max"),
    (pl.col("pmts_dpdvalue_108P") > 31).max().alias("pmts_dpdvalue_108P_over31")
)
train_credit_bureau_b_2_feats
```

shape: (36_447, 3)

case_id	pmts_pmtsoverdue_635A_max	pmts_dpdvalue_108P_over31
i64	f64	bool
53621	0.0	false
223684	0.0	false
1713554	0.0	false
1708984	0.0	false
744685	0.0	false
…	…	…
882429	0.0	false
1904001	0.8	true
1938255	0.4	true
1447979	0.0	false
914271	0.0	false

선택된 A 및 M 타입 열 추출

```
selected_static_cols = []
for col in train_static.columns:
    if col[-1] in ("A", "M"):
        selected_static_cols.append(col)
print(selected_static_cols)

selected_static_cb_cols = []
for col in train_static_cb.columns:
    if col[-1] in ("A", "M"):
        selected_static_cb_cols.append(col)
print(selected_static_cb_cols)
```

```
['amtinstpaidbefduel24m_4187115A', 'annuity_780A', 'annuitynextmonth_57A', 'avginstallast24m_3658937A', 'avglnamtstart24m_4525187A', 'avgoutstandbalancel6m_4187114A', 'avgpmtlast12m_4525200A', 'credamount_770A', 'currdebt_22A', 'currdebtcredtyperange_828A', 'disbursedcredamount_1113A', 'downpmt_116A', 'inittransactionamount_650A', 'lastapprcommoditycat_1041M', 'lastapprcommoditytypec_5251766M', 'lastapprcredamount_781A', 'lastcancelreason_561M', 'lastotherinc_902A', 'lastotherlnsexpense_631A', 'lastrejectcommoditycat_161M', 'lastrejectcommodtypec_5251769M', 'lastrejectcredamount_222A', 'lastrejectreason_759M', 'lastrejectreasonclient_4145040M', 'maininc_215A', 'maxannuity_159A', 'maxannuity_4075009A', 'maxdebt4_972A', 'maxinstallast24m_3658928A', 'maxlnamtstart6m_4525199A', 'maxoutstandbalancel12m_4187113A', 'maxpmtlast3m_4525190A', 'previouscontdistrict_112M', 'price_1097A', 'sumoutstandtotal_3546847A', 'sumoutstandtotalest_4493215A', 'totaldebt_9A', 'totalsettled_863A', 'totinstallast1m_4525188A']
['description_5085714M', 'education_1103M', 'education_88M', 'maritalst_385M', 'maritalst_893M', 'pmtaverage_3A', 'pmtaverage_4527227A', 'pmtaverage_4955615A', 'pmtssum_45A']
```

모든 테이블 결합

```
data = train_basetable.join(
    train_static.select(["case_id"]+selected_static_cols), how="left", on="case_id"
).join(
    train_static_cb.select(["case_id"]+selected_static_cb_cols), how="left", on="case_id"
).join(
    train_person_1_feats_1, how="left", on="case_id"
).join(
    train_person_1_feats_2, how="left", on="case_id"
).join(
    train_credit_bureau_b_2_feats, how="left", on="case_id"
)
data.head()
```

shape: (5, 58)

case_id	date_decision	MONTH	WEEK_NUM	target	amtinstpaidbefduel24m_4187115A	annuity_780A	annuitynextmonth_57A	avginstallast24m_3658937A	avgl
i64	str	i64	i64	i64	f64	f64	f64	f64	
0	"2019-01-03"	201901	0	0	null	1917.6	0.0	null	
1	"2019-01-03"	201901	0	0	null	3134.0	0.0	null	
2	"2019-01-04"	201901	0	0	null	4937.0	0.0	null	
3	"2019-01-03"	201901	0	0	null	4643.6	0.0	null	
4	"2019-01-04"	201901	0	1	null	3390.2	0.0	null	

테스트 데이터 로드 및 전처리

트레인 데이터와 동일한 처리 방식

```
test_basetable = pl.read_csv(dataPath + "csv_files//test//test_base.csv")
test_static = pl.concat(
    [
        pl.read_csv(dataPath + "csv_files//test//test_static_0_0.csv").pipe(set_table_dtypes),
        pl.read_csv(dataPath + "csv_files//test//test_static_0_1.csv").pipe(set_table_dtypes),
        pl.read_csv(dataPath + "csv_files//test//test_static_0_2.csv").pipe(set_table_dtypes),
    ],
    how="vertical_relaxed",
)
test_static_cb = pl.read_csv(dataPath + "csv_files//test//test_static_cb_0.csv").pipe(set_table_dtypes)
test_person_1 = pl.read_csv(dataPath + "csv_files//test//test_person_1.csv").pipe(set_table_dtypes)
test_credit_bureau_b_2 = pl.read_csv(dataPath + "csv_files//test//test_credit_bureau_b_2.csv").pipe(set_table_dtypes)
```

테스트 데이터 로드 및 전처리

트레인 데이터와 동일한 처리 방식

```
# test_person_1 테이블 그룹화 및 집계
test_person_1_feats_1 = test_person_1.group_by("case_id").agg(
    pl.col("mainoccupationinc_384A").max().alias("mainoccupationinc_384A_max"),
    (pl.col("incometype_1044T") == "SELFEMPLOYED").max().alias("mainoccupationinc_384A_any_selfemployed")
)

# test_person_1 테이블 필터링 및 열 선택
test_person_1_feats_2 = test_person_1.select(["case_id", "num_group1", "housetype_905L"]).filter(
    pl.col("num_group1") == 0
).drop("num_group1").rename({"housetype_905L": "person_housetype"})

# test_credit_bureau_b_2 테이블 그룹화 및 집계
test_credit_bureau_b_2_feats = test_credit_bureau_b_2.group_by("case_id").agg(
    pl.col("pmts_pmtsoverdue_635A").max().alias("pmts_pmtsoverdue_635A_max"),
    (pl.col("pmts_dpdvalue_108P") > 31).max().alias("pmts_dpdvalue_108P_over31")
)

# 모든 테이블 결합
data_submission = test_basetable.join(
    test_static.select(["case_id"] + selected_static_cols), how="left", on="case_id"
).join(
    test_static_cb.select(["case_id"] + selected_static_cb_cols), how="left", on="case_id"
).join(
    test_person_1_feats_1, how="left", on="case_id"
).join(
    test_person_1_feats_2, how="left", on="case_id"
).join(
    test_credit_bureau_b_2_feats, how="left", on="case_id"
)
```

data_submission

shape: (10, 57)

case_id	date_decision	MONTH	WEEK_NUM	amtinstpaidbefduel24m_4187115A	annuity_780A	annuitynextmonth_57A	avginstallast24m_3658937A	avglnamtsta
i64	str	i64	i64	f64	f64	f64	f64	
57543	"2021-05-14"	202201	100	191767.36	3674.6	1218.2001	16049.4	
57549	"2022-01-17"	202201	100	129704.4	5742.6	3546.6	32426.201	
57551	"2020-11-27"	202201	100	71036.4	2844.6	0.0	8357.2	
57552	"2020-11-27"	202201	100	183992.0	6298.8003	12155.4	7440.4	
57569	"2021-12-20"	202201	100	0.0	4682.6	0.0	null	
57630	"2021-03-16"	202201	100	0.0	8905.0	0.0	null	
57631	"2022-06-04"	202201	100	null	2540.6	0.0	null	
57632	"2022-02-05"	202201	100	63647.402	4732.0	0.0	3536.0	
57633	"2022-01-25"	202201	100	null	8273.0	0.0	null	
57634	"2021-01-27"	202201	100	39948.8	1165.8	0.0	3994.8	

데이터 세트를 훈련, 검증 및 테스트 세트로 분할

```
case_ids = data["case_id"].unique().shuffle(seed=1)
case_ids_train, case_ids_test = train_test_split(case_ids, train_size=0.6, random_state=1)
case_ids_valid, case_ids_test = train_test_split(case_ids_test, train_size=0.5, random_state=1)

cols_pred = []
for col in data.columns:
    if col[-1].isupper() and col[:-1].islower():
        cols_pred.append(col)
```

POLARS 데이터를 PANDAS 데이터로 변환

```
def from_polars_to_pandas(case_ids: pl.DataFrame) -> pl.DataFrame:
    return (
        data.filter(pl.col("case_id").is_in(case_ids))[["case_id", "WEEK_NUM", "target"]].to_pandas(),
        data.filter(pl.col("case_id").is_in(case_ids))[cols_pred].to_pandas(),
        data.filter(pl.col("case_id").is_in(case_ids))["target"].to_pandas()
    )
```

```
base_train, X_train, y_train = from_polars_to_pandas(case_ids_train)
base_valid, X_valid, y_valid = from_polars_to_pandas(case_ids_valid)
base_test, X_test, y_test = from_polars_to_pandas(case_ids_test)
```

```
for df in [X_train, X_valid, X_test]:
    df = convert_strings(df)
```

훈련, 검증 및 테스트 세트의 크기

```
print(f"Train: {X_train.shape}")
print(f"Valid: {X_valid.shape}")
print(f"Test: {X_test.shape}")
```

```
Train: (915995, 48)
Valid: (305332, 48)
Test: (305332, 48)
```

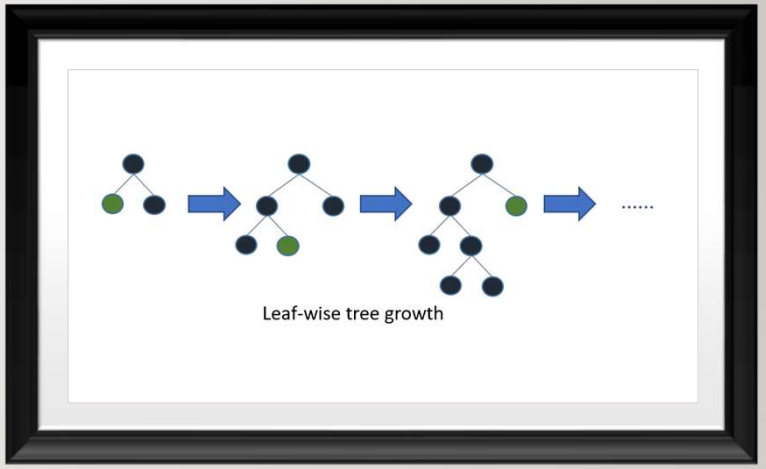
LIGHTGBM 모델 훈련

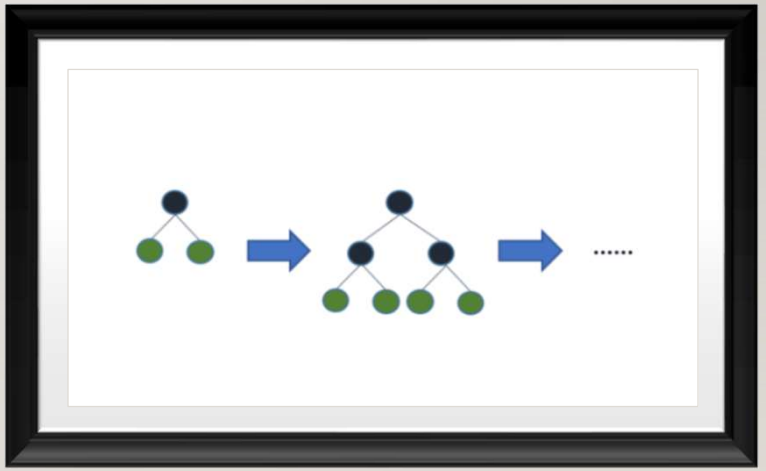
```
lgb_train = lgb.Dataset(X_train, label=y_train)
lgb_valid = lgb.Dataset(X_valid, label=y_valid, reference=lgb_train)

params = {
    "boosting_type": "gbdt",
    "objective": "binary",
    "metric": "auc",
    "max_depth": 3,
    "num_leaves": 31,
    "learning_rate": 0.05,
    "feature_fraction": 0.9,
    "bagging_fraction": 0.8,
    "bagging_freq": 5,
    "n_estimators": 1000,
    "verbose": -1,
}

gbm = lgb.train(
    params,
    lgb_train,
    valid_sets=lgb_valid,
    callbacks=[lgb.log_evaluation(50), lgb.early_stopping(10)]
)
```

LIGHTGBM

리프 기준 분할

LEAF-WISE





LIGHTGBM 파라미터

파라미터	default	설명
num_iterations	100	반복 수행하려는 트리의 개수 (너무 크면 오버피팅 발생)
objective	regression	수치예측이면 regression, 이진분류이면 binary
learning_rate	0.1	부스팅 스탭 반복할 때 학습률, 0~1 사이의 값
max_depth	1	트리의 깊이
min_data_in_leaf	20	한 리프의 최소 데이터 수 (decision tree의 min_sample_leaf와 동일, 오버피팅 제어)
num_leaves	31	하나의 트리가 가질 수 있는 최대 리프 개수
boosting	gbdt	부스팅 방법 (gbdt: Gradient Boosting DecisionTree / rf: RandomForest)
bagging_fraction	1.0	데이터 샘플링 비율, 오버피팅 제어
feature_fraction	1.0	개별 트리 학습 시 무작위로 선택하는 feature의 비율
lambda_l1	0.0	L1 regulation 제어
lambda_l2	0.0	L2 regulation 제어
metric	""	성능평가를 어떤 것으로 할 것인지 (auc, l1, l2 등)

LIGHTGBM 모델 훈련

```
Training until validation scores don't improve for 10 rounds
[50]    valid_0's auc: 0.705963
[100]   valid_0's auc: 0.724362
[150]   valid_0's auc: 0.731423
[200]   valid_0's auc: 0.735874
[250]   valid_0's auc: 0.739009
[300]   valid_0's auc: 0.740965
[350]   valid_0's auc: 0.742924
[400]   valid_0's auc: 0.744582
[450]   valid_0's auc: 0.745977
[500]   valid_0's auc: 0.747033
[550]   valid_0's auc: 0.747877
[600]   valid_0's auc: 0.749039
[650]   valid_0's auc: 0.750087
[700]   valid_0's auc: 0.750863
Early stopping, best iteration is:
[739]   valid_0's auc: 0.751216
```

AUC 점수

```
for base, X in [(base_train, X_train), (base_valid, X_valid), (base_test, X_test)]:
    y_pred = gbm.predict(X, num_iteration=gbm.best_iteration)
    base["score"] = y_pred

print(f'The AUC score on the train set is: {roc_auc_score(base_train["target"], base_train["score"])}')
print(f'The AUC score on the valid set is: {roc_auc_score(base_valid["target"], base_valid["score"])}')
print(f'The AUC score on the test set is: {roc_auc_score(base_test["target"], base_test["score"])}')
```

```
The AUC score on the train set is: 0.7641229176605 93
The AUC score on the valid set is: 0.7512157223309048
The AUC score on the test set is: 0.7483072129459662
```

GINI 계수의 안정성 점수 (통계적 분산 정도)

```
def gini_stability(base, w_fallingrate=88.0, w_resstd=-0.5):
    gini_in_time = base.loc[:, ["WEEK_NUM", "target", "score"]]#
        .sort_values("WEEK_NUM")#
        .groupby("WEEK_NUM")[["target", "score"]]#
        .apply(lambda x: 2*roc_auc_score(x["target"], x["score"])-1).tolist()

    x = np.arange(len(gini_in_time))
    y = gini_in_time
    a, b = np.polyfit(x, y, 1)
    y_hat = a*x + b
    residuals = y - y_hat
    res_std = np.std(residuals)
    avg_gini = np.mean(gini_in_time)
    return avg_gini + w_fallingrate * min(0, a) + w_resstd * res_std

stability_score_train = gini_stability(base_train)
stability_score_valid = gini_stability(base_valid)
stability_score_test = gini_stability(base_test)

print(f'The stability score on the train set is: {stability_score_train}')
print(f'The stability score on the valid set is: {stability_score_valid}')
print(f'The stability score on the test set is: {stability_score_test}')
```

```
The stability score on the train set is: 0.4976648127691175
The stability score on the valid set is: 0.4726726686264489
The stability score on the test set is: 0.4583643686935092
```

예측

```
X_submission = data_submission[cols_pred].to_pandas()
X_submission = convert_strings(X_submission)
categorical_cols = X_train.select_dtypes(include=['category']).columns

for col in categorical_cols:
    train_categories = set(X_train[col].cat.categories)
    submission_categories = set(X_submission[col].cat.categories)
    new_categories = submission_categories - train_categories
    X_submission.loc[X_submission[col].isin(new_categories), col] = "Unknown"
    new_dtype = pd.CategoricalDtype(categories=train_categories, ordered=True)
    X_train[col] = X_train[col].astype(new_dtype)
    X_submission[col] = X_submission[col].astype(new_dtype)

y_submission_pred = gbm.predict(X_submission, num_iteration=gbm.best_iteration)

y_submission_pred
```

```
array([0.0108652 , 0.0546489 , 0.00741195, 0.00928448, 0.06616796,
       0.01272284, 0.03471827, 0.00326202, 0.06028441, 0.00793056])
```

예측값

```
submission = pd.DataFrame({
    "case_id": data_submission["case_id"].to_numpy(),
    "score": y_submission_pred
}).set_index('case_id')
submission.to_csv("./submission.csv")
```

	A	B	C	D	E	F
1	case_id	score				
2	57543	0.010865				
3	57549	0.054649				
4	57551	0.007412				
5	57552	0.009284				
6	57569	0.066168				
7	57630	0.012723				
8	57631	0.034718				
9	57632	0.003262				
10	57633	0.060284				
11	57634	0.007931				