Use Machine Learning to detect Age-Related conditions

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
%matplotlib inline
import seaborn as sns
import gc
import re as re
from collections import Counter

from tqdm.auto import tqdm
import math
from sklearn.model_selection import KFold, StratifiedKFold, train_test_split, GridSearchCV
from sklearn.metrics import roc_auc_score, accuracy_score, confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay
from sklearn.preprocessing import LabelEncoder,StandardScaler

import warnings
warnings.filterwarnings('ignore')

from glob import glob
from pathlib import Path
import joblib
import pickle
import os
import random

import time
from xgboost import XGBClassifier
from sklearn.metrics import log_loss
%matplotlib inline
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this ver
sion of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Load Dataset

In [2]:
```
train            = pd.read_csv('/input/icr-identify-age-related-conditions/train.csv')
test             = pd.read_csv('/input/icr-identify-age-related-conditions/test.csv')
```

In [3]:
```
train.head()
```

Out[3]:

	Id	AB	AF	AH	AM	AR	AX	AY	AZ	BC	...	FL	FR	FS	GB	
0	000ff2bfdfe9	0.209377	3109.03329	85.200147	22.394407	8.138688	0.699861	0.025578	9.812214	5.555634	...	7.298162	1.73855	0.094822	11.339138	72
1	007255e47698	0.145282	978.76416	85.200147	36.968889	8.138688	3.632190	0.025578	13.517790	1.229900	...	0.173229	0.49706	0.568932	9.292698	72
2	013f2bd269f5	0.470030	2635.10654	85.200147	32.360553	8.138688	6.732840	0.025578	12.824570	1.229900	...	7.709560	0.97556	1.198821	37.077772	88
3	043ac50845d5	0.252107	3819.65177	120.201618	77.112203	8.138688	3.685344	0.025578	11.053708	1.229900	...	6.122162	0.49706	0.284466	18.529584	82
4	044fb8a146ec	0.380297	3733.04844	85.200147	14.103738	8.138688	3.942255	0.054810	3.396778	102.151980	...	8.153058	48.50134	0.121914	16.408728	146

5 rows × 58 columns

EDA

In [4]:
```
# summary table function
def summary(df):
    print(f'data shape: {df.shape}')
    summ = pd.DataFrame(df.dtypes, columns=['data type'])
    summ['#missing'] = df.isnull().sum().values
    summ['%missing'] = df.isnull().sum().values / len(df)* 100
    summ['#unique'] = df.nunique().values
    desc = pd.DataFrame(df.describe(include='all').transpose())
    summ['min'] = desc['min'].values
    summ['max'] = desc['max'].values
    summ['first quartile'] = desc.loc[:, '25%'].values
    summ['second quartile'] = desc.loc[:, '50%'].values
    summ['third quartile'] = desc.loc[:, '75%'].values

    return summ
```

In [5]:
```
summary(train)
```

```
data shape: (617, 58)
```

Out[5]:

	data type	#missing	%missing	#unique	min	max	first quartile	second quartile	third quartile
Id	object	0	0.000000	617	NaN	NaN	NaN	NaN	NaN
AB	float64	0	0.000000	217	0.081187	6.161666	0.252107	0.354659	0.559763
AF	float64	0	0.000000	599	192.59328	28688.18766	2197.34548	3120.31896	4361.63739
AH	float64	0	0.000000	227	85.200147	1910.123198	85.200147	85.200147	113.73954
AM	float64	0	0.000000	605	3.177522	630.51823	12.270314	20.53311	39.139886
AR	float64	0	0.000000	130	8.138688	178.943634	8.138688	8.138688	8.138688
AX	float64	0	0.000000	427	0.699861	38.27088	4.128294	5.031912	6.431634
AY	float64	0	0.000000	148	0.025578	10.315851	0.025578	0.025578	0.036845
AZ	float64	0	0.000000	484	3.396778	38.971568	8.12958	10.46132	12.969516
BC	float64	0	0.000000	259	1.2299	1463.693448	1.2299	1.2299	5.081244
BD	float64	0	0.000000	617	1693.62432	53060.59924	4155.70287	4997.96073	6035.8857
BN	float64	0	0.000000	53	9.8868	29.3073	19.4205	21.186	23.6577
BP	float64	0	0.000000	612	72.948951	2447.81055	156.847239	193.908816	247.803462
BQ	float64	60	9.724473	515	1.331155	344.644105	27.834425	61.642115	134.009015
BR	float64	0	0.000000	566	51.216883	179250.2529	424.990642	627.417402	975.649259
BZ	float64	0	0.000000	115	257.432377	50092.4593	257.432377	257.432377	257.432377
CB	float64	2	0.324149	553	12.49976	2271.436167	23.317567	42.55433	77.310097
CC	float64	3	0.486224	602	0.176874	4.103032	0.563688	0.658715	0.772206
CD	float64	0	0.000000	584	23.3876	633.534408	64.724192	79.819104	99.81352
CF	float64	0	0.000000	586	0.510888	200.967526	5.066306	9.123	13.565901
CH	float64	0	0.000000	135	0.003184	0.224074	0.023482	0.02786	0.034427
CL	float64	0	0.000000	123	1.050225	31.688153	1.050225	1.050225	1.228445
CR	float64	0	0.000000	595	0.069225	3.039675	0.589575	0.7308	0.85935
CS	float64	0	0.000000	576	13.784111	267.942823	29.782467	34.83513	40.529401
CU	float64	0	0.000000	307	0.137925	4.951507	1.070298	1.351665	1.660617
CW	float64	0	0.000000	426	7.03064	64.521624	7.03064	36.019104	37.935832
DA	float64	0	0.000000	611	6.9064	210.33092	37.94252	49.18094	61.40876
DE	float64	0	0.000000	616	35.998895	2103.40519	188.81569	307.509595	507.8962
DF	float64	0	0.000000	137	0.23868	37.895013	0.23868	0.23868	0.23868
DH	float64	0	0.000000	191	0.040995	1.060404	0.295164	0.358023	0.426348
DI	float64	0	0.000000	571	60.23247	1049.168078	102.703553	130.05063	165.836955
DL	float64	0	0.000000	604	10.3456	326.2362	78.23224	96.26496	110.64068
DN	float64	0	0.000000	576	6.339496	62.808096	20.888264	25.2488	30.544224
DU	float64	1	0.162075	253	0.005518	161.355315	0.005518	0.251741	1.05869
DV	float64	0	0.000000	39	1.74307	25.19293	1.74307	1.74307	1.74307
DY	float64	0	0.000000	590	0.804068	152.355164	14.715792	21.642456	34.058344
EB	float64	0	0.000000	439	4.926396	94.95858	5.965392	8.149404	10.503048
EE	float64	0	0.000000	513	0.286201	18.324926	1.648679	2.616119	3.91007
EG	float64	0	0.000000	610	185.5941	30243.75878	1111.160625	1493.817413	1905.701475
EH	float64	0	0.000000	127	0.003042	42.569748	0.003042	0.085176	0.237276
EJ	object	0	0.000000	2	NaN	NaN	NaN	NaN	NaN
EL	float64	60	9.724473	311	5.394675	109.125159	30.927468	71.949306	109.125159
EP	float64	0	0.000000	275	78.526968	1063.594578	78.526968	78.526968	112.766654
EU	float64	0	0.000000	455	3.828384	6501.26448	4.324656	22.641144	49.085352
FC	float64	1	0.162075	600	7.534128	3030.655824	25.815384	36.394008	56.714448
FD	float64	0	0.000000	337	0.29685	1578.654237	0.29685	1.870155	4.880214
FE	float64	0	0.000000	615	1563.136688	143224.6823	5164.66626	7345.143424	10647.95165
FI	float64	0	0.000000	498	3.58345	35.851039	8.523098	9.945452	11.516657
FL	float64	1	0.162075	388	0.173229	137.932739	0.173229	3.028141	6.238814
FR	float64	0	0.000000	435	0.49706	1244.22702	0.49706	1.131	1.51206
FS	float64	2	0.324149	161	0.06773	31.365763	0.06773	0.250601	0.535067

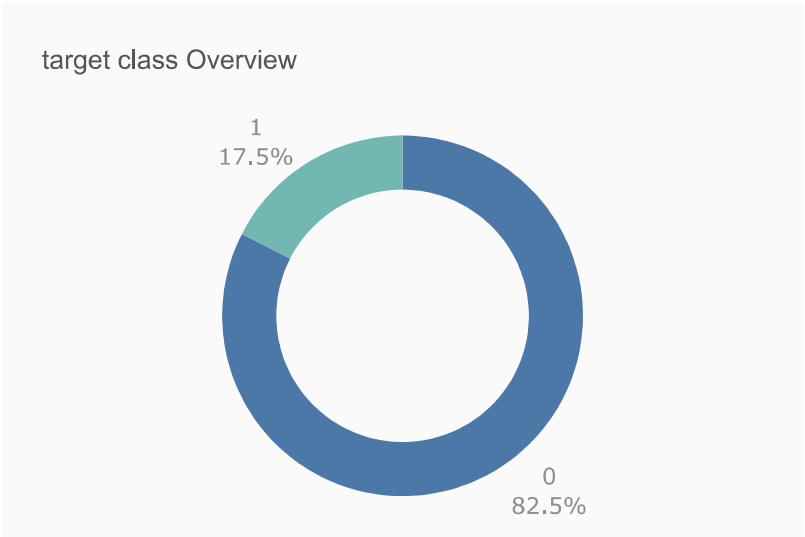
	data type	#missing	%missing	#unique	min	max	first quartile	second quartile	third quartile
GB	float64	0	0.000000	560	4.102182	135.781294	14.036718	18.771436	25.608406
GE	float64	0	0.000000	264	72.611063	1497.351958	72.611063	72.611063	127.591671
GF	float64	0	0.000000	611	13.038894	143790.0712	2798.992584	7838.27361	19035.70924
GH	float64	0	0.000000	596	9.432735	81.210825	25.034888	30.608946	36.863947
GI	float64	0	0.000000	615	0.897628	191.194764	23.011684	41.007968	67.931664
GL	float64	1	0.162075	355	0.001129	21.978	0.124392	0.337827	21.978
Class	int64	0	0.000000	2	0.0	1.0	0.0	0.0	0.0

- There are 56 X variables and 1 target(y) variable, while 1 variable(id) is not relevant for modelling
- All variable types are float64, except for the EJ variable which is of type object.
- 9 variables have missing value. The missing value ratio is not very high.

In [6]:
```
# select numerical and categorical variables respectively.
num_cols = test.select_dtypes(include=['float64']).columns.tolist()
cat_cols = test.select_dtypes(include=['object']).columns.tolist()
cat_cols.remove('Id')
```

Inspecting the class imbalance

In [7]:
```
fig2 = px.pie(train, names='Class',
              height=400, width=600,
              hole=0.7,
              title='target class Overview',
                    color_discrete_sequence=['#4c78a8', '#72b7b2'])
fig2.update_traces(hovertemplate=None, textposition='outside', textinfo='percent+label', rotation=0)
fig2.update_layout(margin=dict(t=100, b=30, l=0, r=0), showlegend=False,
                         plot_bgcolor='#fafafa', paper_bgcolor='#fafafa',
                         title_font=dict(size=20, color='#555', family="Lato, sans-serif"),
                         font=dict(size=17, color='#8a8d93'),
                         hoverlabel=dict(bgcolor="#444", font_size=13, font_family="Lato, sans-serif"))
fig2.show()
```



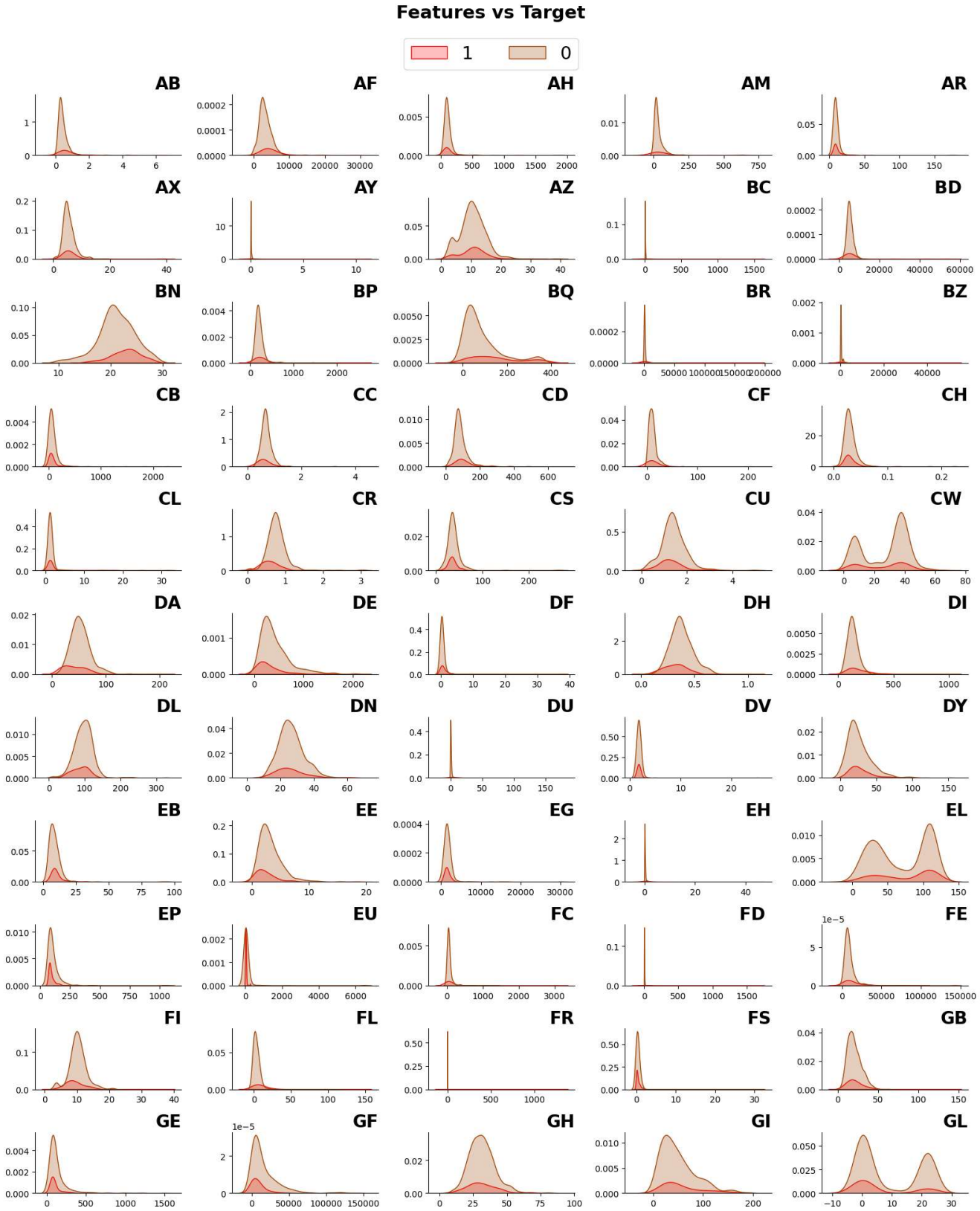
The dataset is quite imbalanced. Maybe we need to try a few over-sampling methods like SMOTE in future versions.

Distribution of numerical variables

In [8]:
```
figsize = (4*4, 20)
fig = plt.figure(figsize=figsize)
for idx, col in enumerate(num_cols):
    ax = plt.subplot(11,5, idx + 1)
    sns.kdeplot(
        data=train, hue='Class', fill=True,
        x=col, palette=['#9E3F00', 'red'], legend=False
    )

    ax.set_ylabel(''); ax.spines['top'].set_visible(False),
    ax.set_xlabel(''); ax.spines['right'].set_visible(False)
    ax.set_title(f'{col}', loc='right',
                 weight='bold', fontsize=20)
```

```
fig.suptitle(f'Features vs Target\n\n\n', ha='center',  fontweight='bold', fontsize=21)
fig.legend([1, 0], loc='upper center', bbox_to_anchor=(0.5, 0.96), fontsize=21, ncol=3)
plt.tight_layout()
plt.show()
```



"The distribution of most numerical variables differs, indicating that these variables possess predictive power to some extent."

"However, scaling might be necessary as they correspond to different scales."

Correlation heatmap for numerical variables

In [9]:
```
def plot_top_correlations(df: pd.core.frame.DataFrame, n: int, title_name: str='Top Correlations') -> None:
    # Calculate correlation between all variables
```

```
    corr = df.corr()

    # Select variables having highest absolute correlation
    top_corr_cols = corr.abs().nlargest(n, columns=corr.columns).index
    top_corr = corr.loc[top_corr_cols, top_corr_cols]

    fig, axes = plt.subplots(figsize=(10, 5))
    mask = np.zeros_like(top_corr)
    mask[np.triu_indices_from(mask)] = True
    sns.heatmap(top_corr, mask=mask, linewidths=.5, cmap='YlOrRd', annot=True)
    plt.title(title_name)
    plt.show()

# Plot heatmap of top 12 correlations in training data
plot_top_correlations(train[num_cols], 12, 'Top 12 Correlations in Train Dataset')
```

"Among the top 12 highly correlated numerical variables, while some exhibit strong correlations, the overall correlation among these health-related data variables is not very high."

Model training using XGB, LightGBM, CatBoost

In [10]:
```
import lightgbm as lgb
import xgboost as xgb
from catboost import Pool, CatBoostRegressor, CatBoostClassifier
```

In [11]:
```
class CFG:
    VER = 1
    AUTHOR = 'maverick'
    COMPETITION = 'icr-identify-age-related-conditions'
    DATA_PATH = Path('/input/icr-identify-age-related-conditions')
    OOF_DATA_PATH = Path('./oof')
    MODEL_DATA_PATH = Path('./models')
    METHOD_LIST = ['lightgbm', 'xgboost', 'catboost']
    seed = 3407 #52
    n_folds = 10 #replaced 20
    target_col = 'Class'
    metric = 'balanced_log_loss'
    metric_maximize_flag = False
    num_boost_round = 50500
    early_stopping_round = 500
    verbose = 2000
    boosting_type = 'dart'
    lgb_params = {
        'objective': 'binary', # 'binary', 'multiclass'
        'metric': None, # 'auc', 'multi_logloss'
        # 'num_class': None,
        'boosting': boosting_type,
        'device_type':'gpu',
        'learning_rate': 0.005,
        'num_leaves': 5,
        'feature_fraction': 0.50,
        'bagging_fraction': 0.80,
        'lambda_l1': 2,
        'lambda_l2': 4,
        'n_jobs': -1,
        'is_unbalance':True, #added balancing
        'verbose': -1, #added silence
        # 'min_data_in_leaf': 40,
        # 'bagging_freq': 10,
        'seed': seed,
```

```
    }
    xgb_params = {
        'objective': 'binary:logistic',
        'eval_metric': 'logloss',
        'learning_rate': 0.005,
        'max_depth': 4,
        'colsample_bytree': 0.50,
        'subsample': 0.80,
        'eta': 0.03,
        'gamma': 1.5,
        # 'lambda': 70,
        # 'min_child_weight': 8,
        # 'eval_metric':'logloss',
        # 'tree_method': 'gpu_hist',
        # 'predictor':'gpu_predictor',
        'random_state': seed,
    }

    cat_params = {
        'learning_rate': 0.005,
        'iterations': num_boost_round,
        'depth': 4, #
        'colsample_bylevel': 0.50,
        'subsample': 0.80,
        'l2_leaf_reg': 3, # 3, 30
        'random_seed': seed,
        'auto_class_weights': 'Balanced'
        # 'task_type':'GPU',
    }
```

In [12]:
```
!mkdir oof
!mkdir models
```

In [13]:
```
test[CFG.target_col] = -1
all_df = pd.concat([train, test])
```

Setting Seed

In [14]:
```
def seed_everything(seed):
    random.seed(seed)
    np.random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
```

Metric Calculation

In [15]:
```
def balanced_log_loss(y_true, y_pred):
    y_pred = np.clip(y_pred, 1e-15, 1-1e-15)
    nc = np.bincount(y_true)
    w0, w1 = 1/(nc[0]/y_true.shape[0]), 1/(nc[1]/y_true.shape[0])
    balanced_log_loss_score = (-w0/nc[0]*(np.sum(np.where(y_true==0,1,0) * np.log(1-y_pred))) - w1/nc[1]*(np.sum(np.where(y_true!=0,1,0) *
    return balanced_log_loss_score

# ====================================================
# LightGBM Metric
# ====================================================
def lgb_metric(y_pred, y_true):
    y_true = y_true.get_label()
    return 'balanced_log_loss', balanced_log_loss(y_true, y_pred), CFG.metric_maximize_flag

def xgb_metric(y_pred, y_true):
    y_true = y_true.get_label()
    return 'balanced_log_loss', balanced_log_loss(y_true, y_pred)

# ====================================================
# Catboost Metric
# ====================================================
class CatboostMetric(object):
    def get_final_error(self, error, weight): return error
    def is_max_optimal(self): return CFG.metric_maximize_flag
    def evaluate(self, approxes, target, weight):
        error = balanced_log_loss(np.array(target), approxes)
        return error, 0
```

In [16]:
```
def calc_log_loss_weight(y_true):
    nc = np.bincount(y_true)
    w0, w1 = 1/(nc[0]/y_true.shape[0]), 1/(nc[1]/y_true.shape[0])
    return w0, w1

def lightgbm_training(x_train: pd.DataFrame, y_train: pd.DataFrame, x_valid: pd.DataFrame, y_valid: pd.DataFrame, features: list, categori
    train_w0, train_w1 = calc_log_loss_weight(y_train)
    valid_w0, valid_w1 = calc_log_loss_weight(y_valid)
    lgb_train = lgb.Dataset(x_train, y_train, weight=y_train.map({0: train_w0, 1: train_w1}), categorical_feature=categorical_features)
    lgb_valid = lgb.Dataset(x_valid, y_valid, weight=y_valid.map({0: valid_w0, 1: valid_w1}), categorical_feature=categorical_features)
    model = lgb.train(
                params = CFG.lgb_params,
                train_set = lgb_train,
                num_boost_round = CFG.num_boost_round,
```

```
                valid_sets = [lgb_train, lgb_valid],
                early_stopping_rounds = CFG.early_stopping_round,
                verbose_eval = CFG.verbose,
                # feval = lgb_metric,
            )
    # Predict validation
    valid_pred = model.predict(x_valid)
    return model, valid_pred
def xgboost_training(x_train: pd.DataFrame, y_train: pd.DataFrame, x_valid: pd.DataFrame, y_valid: pd.DataFrame, features: list, categoric
    train_w0, train_w1 = calc_log_loss_weight(y_train)
    valid_w0, valid_w1 = calc_log_loss_weight(y_valid)
    xgb_train = xgb.DMatrix(data=x_train, label=y_train, weight=y_train.map({0: train_w0, 1: train_w1}))
    xgb_valid = xgb.DMatrix(data=x_valid, label=y_valid, weight=y_valid.map({0: valid_w0, 1: valid_w1}))
    model = xgb.train(
                CFG.xgb_params,
                dtrain = xgb_train,
                num_boost_round = CFG.num_boost_round,
                evals = [(xgb_train, 'train'), (xgb_valid, 'eval')],
                early_stopping_rounds = CFG.early_stopping_round,
                verbose_eval = CFG.verbose,
                # feval = xgb_metric,
                # maximize = CFG.metric_maximize_flag,
            )
    # Predict validation
    valid_pred = model.predict(xgb.DMatrix(x_valid), iteration_range=(0, model.best_ntree_limit))
    return model, valid_pred

def catboost_training(x_train: pd.DataFrame, y_train: pd.DataFrame, x_valid: pd.DataFrame, y_valid: pd.DataFrame, features: list, categori
    train_w0, train_w1 = calc_log_loss_weight(y_train)
    valid_w0, valid_w1 = calc_log_loss_weight(y_valid)
    cat_train = Pool(data=x_train, label=y_train, weight=y_train.map({0: train_w0, 1: train_w1}), cat_features=categorical_features)
    cat_valid = Pool(data=x_valid, label=y_valid, weight=y_valid.map({0: valid_w0, 1: valid_w1}), cat_features=categorical_features)
    model = CatBoostClassifier(**CFG.cat_params) # , eval_metric = CatboostMetric
    model.fit(cat_train,
              eval_set=[cat_valid],
              early_stopping_rounds=CFG.early_stopping_round,
              verbose=CFG.verbose,
              use_best_model=True)
    # Predict validation
    valid_pred = model.predict_proba(x_valid)[:, 1]
    return model, valid_pred

def gradient_boosting_model_cv_training(method: str, train_df: pd.DataFrame, features: list, categorical_features: list):
    # Create a numpy array to store out of folds predictions
    oof_predictions = np.zeros(len(train_df))
    oof_fold = np.zeros(len(train_df))
    kfold = StratifiedKFold(n_splits = CFG.n_folds, shuffle = True, random_state = CFG.seed)
    for fold, (train_index, valid_index) in enumerate(kfold.split(train_df, train_df[CFG.target_col])):
        print('-'*50)
        print(f'{method} training fold {fold + 1}')

        x_train = train_df[features].iloc[train_index]
        y_train = train_df[CFG.target_col].iloc[train_index]
        x_valid = train_df[features].iloc[valid_index]
        y_valid = train_df[CFG.target_col].iloc[valid_index]
        if method == 'lightgbm':
            model, valid_pred = lightgbm_training(x_train, y_train, x_valid, y_valid, features, categorical_features)
        if method == 'xgboost':
            model, valid_pred = xgboost_training(x_train, y_train, x_valid, y_valid, features, categorical_features)
        if method == 'catboost':
            model, valid_pred = catboost_training(x_train, y_train, x_valid, y_valid, features, categorical_features)

        # Save best model
        pickle.dump(model, open(CFG.MODEL_DATA_PATH / f'{method}_fold{fold + 1}_seed{CFG.seed}_ver{CFG.VER}.pkl', 'wb'))
        # Add to out of folds array
        oof_predictions[valid_index] = valid_pred
        oof_fold[valid_index] = fold + 1
        del x_train, x_valid, y_train, y_valid, model, valid_pred
        gc.collect()

    # Compute out of folds metric
    score = balanced_log_loss(train_df[CFG.target_col], oof_predictions)
    print(f'{method} our out of folds CV score is {score}')
    # Create a dataframe to store out of folds predictions
    oof_df = pd.DataFrame({'Id': train_df['Id'], CFG.target_col: train_df[CFG.target_col], f'{method}_prediction': oof_predictions, 'fold
    oof_df.to_csv(CFG.MODEL_DATA_PATH / f'oof_{method}_seed{CFG.seed}_ver{CFG.VER}.csv', index = False)
```

Feature Selection

In [17]:
```
numerical_features = ['AB', 'AF', 'AH', 'AM', 'AR', 'AX', 'AY', 'AZ', #'BC',
                      'BD', 'BN', 'BP', 'BQ', 'BR', 'BZ',
                      'CB', 'CC', 'CD', 'CF', 'CH', #'CL',
                      'CR', 'CS', 'CU', 'CW',
                      'DA', 'DE', 'DF', 'DH', 'DI', 'DL', 'DN', 'DU', 'DV', 'DY',
                      'EB', 'EE', 'EG', 'EH', 'EL', 'EP', 'EU',
                      'FC', 'FD', 'FE', 'FI', 'FL', 'FR', 'FS',
                      'GB', 'GE', 'GF', 'GH', 'GI', 'GL']
categorical_features = ['EJ']
features = numerical_features + categorical_features
```

```
str2int_dict = {}
str2int_dict['EJ'] = {'A': 1, 'B': 0}
seed_everything(CFG.seed)
```

Preprocessing

In [18]:
```
def Preprocessing(input_df: pd.DataFrame)->pd.DataFrame:
    input_df = input_df.rename(columns={'BD ': 'BD', 'CD ': 'CD', 'CW ': 'CW', 'FD ': 'FD'}) #?
    output_df = input_df.copy()
    sc = StandardScaler()
    output_df[numerical_features] = sc.fit_transform(output_df[numerical_features]) #added scaling
    for col in categorical_features:
        output_df[col] = input_df[col].map(str2int_dict[col])
    return output_df
all_df = Preprocessing(all_df)
```

In [19]:
```
train_df = all_df[all_df[CFG.target_col] != -1].copy()
test_df = all_df[all_df[CFG.target_col] == -1].copy()
```

In [20]:
```
for method in CFG.METHOD_LIST:
    gradient_boosting_model_cv_training(method, train_df, features, categorical_features)
```

```
-------------------------------------------------
lightgbm training fold 1
[2000]  training's binary_logloss: 0.240573     valid_1's binary_logloss: 0.380602
[4000]  training's binary_logloss: 0.133936     valid_1's binary_logloss: 0.335777
[6000]  training's binary_logloss: 0.083121     valid_1's binary_logloss: 0.313393
[8000]  training's binary_logloss: 0.0549198    valid_1's binary_logloss: 0.311155
[10000] training's binary_logloss: 0.0411957    valid_1's binary_logloss: 0.323622
[12000] training's binary_logloss: 0.0319705    valid_1's binary_logloss: 0.333705
[14000] training's binary_logloss: 0.0269473    valid_1's binary_logloss: 0.343834
[16000] training's binary_logloss: 0.0230827    valid_1's binary_logloss: 0.352815
[18000] training's binary_logloss: 0.020925     valid_1's binary_logloss: 0.358794
[20000] training's binary_logloss: 0.0193932    valid_1's binary_logloss: 0.36387
[22000] training's binary_logloss: 0.0181028    valid_1's binary_logloss: 0.365714
[24000] training's binary_logloss: 0.0172531    valid_1's binary_logloss: 0.36771
[26000] training's binary_logloss: 0.0165144    valid_1's binary_logloss: 0.36882
[28000] training's binary_logloss: 0.016043     valid_1's binary_logloss: 0.370757
[30000] training's binary_logloss: 0.0158354    valid_1's binary_logloss: 0.372224
[32000] training's binary_logloss: 0.0156357    valid_1's binary_logloss: 0.375262
[34000] training's binary_logloss: 0.0154769    valid_1's binary_logloss: 0.377949
[36000] training's binary_logloss: 0.0153588    valid_1's binary_logloss: 0.379529
[38000] training's binary_logloss: 0.0152481    valid_1's binary_logloss: 0.381004
[40000] training's binary_logloss: 0.015148     valid_1's binary_logloss: 0.382658
[42000] training's binary_logloss: 0.015066     valid_1's binary_logloss: 0.383445
[44000] training's binary_logloss: 0.0149789    valid_1's binary_logloss: 0.384562
[46000] training's binary_logloss: 0.0149165    valid_1's binary_logloss: 0.384312
[48000] training's binary_logloss: 0.0148306    valid_1's binary_logloss: 0.384465
[50000] training's binary_logloss: 0.0147753    valid_1's binary_logloss: 0.384568
-------------------------------------------------
lightgbm training fold 2
[2000]  training's binary_logloss: 0.255251     valid_1's binary_logloss: 0.305696
[4000]  training's binary_logloss: 0.144125     valid_1's binary_logloss: 0.203399
[6000]  training's binary_logloss: 0.091037     valid_1's binary_logloss: 0.159946
[8000]  training's binary_logloss: 0.0593351    valid_1's binary_logloss: 0.137471
[10000] training's binary_logloss: 0.0440655    valid_1's binary_logloss: 0.127918
[12000] training's binary_logloss: 0.0340528    valid_1's binary_logloss: 0.124641
[14000] training's binary_logloss: 0.0284739    valid_1's binary_logloss: 0.123799
[16000] training's binary_logloss: 0.0242686    valid_1's binary_logloss: 0.123534
[18000] training's binary_logloss: 0.021996     valid_1's binary_logloss: 0.123676
[20000] training's binary_logloss: 0.0203448    valid_1's binary_logloss: 0.123332
[22000] training's binary_logloss: 0.0190207    valid_1's binary_logloss: 0.117218
[24000] training's binary_logloss: 0.0181458    valid_1's binary_logloss: 0.110904
[26000] training's binary_logloss: 0.0173272    valid_1's binary_logloss: 0.105216
[28000] training's binary_logloss: 0.016914     valid_1's binary_logloss: 0.101382
[30000] training's binary_logloss: 0.0167751    valid_1's binary_logloss: 0.0991921
[32000] training's binary_logloss: 0.0166785    valid_1's binary_logloss: 0.0981959
[34000] training's binary_logloss: 0.0164938    valid_1's binary_logloss: 0.0963396
[36000] training's binary_logloss: 0.0163961    valid_1's binary_logloss: 0.0945048
[38000] training's binary_logloss: 0.0163068    valid_1's binary_logloss: 0.0935821
[40000] training's binary_logloss: 0.0162859    valid_1's binary_logloss: 0.092862
[42000] training's binary_logloss: 0.0162149    valid_1's binary_logloss: 0.0918785
[44000] training's binary_logloss: 0.0160704    valid_1's binary_logloss: 0.0916051
[46000] training's binary_logloss: 0.016084     valid_1's binary_logloss: 0.0909328
[48000] training's binary_logloss: 0.0160163    valid_1's binary_logloss: 0.0905719
[50000] training's binary_logloss: 0.0159573    valid_1's binary_logloss: 0.0902551
-------------------------------------------------
lightgbm training fold 3
[2000]  training's binary_logloss: 0.245919     valid_1's binary_logloss: 0.359147
[4000]  training's binary_logloss: 0.135415     valid_1's binary_logloss: 0.291105
[6000]  training's binary_logloss: 0.084923     valid_1's binary_logloss: 0.246192
[8000]  training's binary_logloss: 0.0563978    valid_1's binary_logloss: 0.207595
[10000] training's binary_logloss: 0.0425637    valid_1's binary_logloss: 0.186845
[12000] training's binary_logloss: 0.0330554    valid_1's binary_logloss: 0.168603
[14000] training's binary_logloss: 0.0277481    valid_1's binary_logloss: 0.15705
[16000] training's binary_logloss: 0.0237372    valid_1's binary_logloss: 0.149003
[18000] training's binary_logloss: 0.0214382    valid_1's binary_logloss: 0.146831
[20000] training's binary_logloss: 0.0198107    valid_1's binary_logloss: 0.146182
[22000] training's binary_logloss: 0.0185385    valid_1's binary_logloss: 0.146802
[24000] training's binary_logloss: 0.0176257    valid_1's binary_logloss: 0.149218
[26000] training's binary_logloss: 0.016798     valid_1's binary_logloss: 0.150429
[28000] training's binary_logloss: 0.0163229    valid_1's binary_logloss: 0.152989
[30000] training's binary_logloss: 0.0161796    valid_1's binary_logloss: 0.155893
[32000] training's binary_logloss: 0.0161365    valid_1's binary_logloss: 0.159083
[34000] training's binary_logloss: 0.0160855    valid_1's binary_logloss: 0.163047
[36000] training's binary_logloss: 0.0160168    valid_1's binary_logloss: 0.166732
[38000] training's binary_logloss: 0.0159444    valid_1's binary_logloss: 0.169782
[40000] training's binary_logloss: 0.0158308    valid_1's binary_logloss: 0.17239
[42000] training's binary_logloss: 0.0157862    valid_1's binary_logloss: 0.174705
[44000] training's binary_logloss: 0.0157505    valid_1's binary_logloss: 0.177534
[46000] training's binary_logloss: 0.0157089    valid_1's binary_logloss: 0.180396
[48000] training's binary_logloss: 0.0156311    valid_1's binary_logloss: 0.182606
[50000] training's binary_logloss: 0.0156035    valid_1's binary_logloss: 0.18511
-------------------------------------------------
lightgbm training fold 4
[2000]  training's binary_logloss: 0.241712     valid_1's binary_logloss: 0.354468
[4000]  training's binary_logloss: 0.133762     valid_1's binary_logloss: 0.296722
[6000]  training's binary_logloss: 0.0822441    valid_1's binary_logloss: 0.285202
[8000]  training's binary_logloss: 0.0533863    valid_1's binary_logloss: 0.285733
[10000] training's binary_logloss: 0.0396003    valid_1's binary_logloss: 0.287819
[12000] training's binary_logloss: 0.0306938    valid_1's binary_logloss: 0.289484
[14000] training's binary_logloss: 0.0257471    valid_1's binary_logloss: 0.294007
```

```
[16000] training's binary_logloss: 0.0220611    valid_1's binary_logloss: 0.297164
[18000] training's binary_logloss: 0.0200204    valid_1's binary_logloss: 0.3027
[20000] training's binary_logloss: 0.0186709    valid_1's binary_logloss: 0.310359
[22000] training's binary_logloss: 0.0175881    valid_1's binary_logloss: 0.313839
[24000] training's binary_logloss: 0.0168835    valid_1's binary_logloss: 0.318571
[26000] training's binary_logloss: 0.0161774    valid_1's binary_logloss: 0.324982
[28000] training's binary_logloss: 0.015865     valid_1's binary_logloss: 0.330759
[30000] training's binary_logloss: 0.0157936    valid_1's binary_logloss: 0.334225
[32000] training's binary_logloss: 0.0156188    valid_1's binary_logloss: 0.336284
[34000] training's binary_logloss: 0.0155631    valid_1's binary_logloss: 0.339592
[36000] training's binary_logloss: 0.0154838    valid_1's binary_logloss: 0.341127
[38000] training's binary_logloss: 0.0154582    valid_1's binary_logloss: 0.344447
[40000] training's binary_logloss: 0.0153956    valid_1's binary_logloss: 0.348196
[42000] training's binary_logloss: 0.0153609    valid_1's binary_logloss: 0.352446
[44000] training's binary_logloss: 0.0153071    valid_1's binary_logloss: 0.35542
[46000] training's binary_logloss: 0.0152281    valid_1's binary_logloss: 0.357728
[48000] training's binary_logloss: 0.0151642    valid_1's binary_logloss: 0.360626
[50000] training's binary_logloss: 0.0151072    valid_1's binary_logloss: 0.364738
--------------------------------------------------
lightgbm training fold 5
[2000]  training's binary_logloss: 0.247366     valid_1's binary_logloss: 0.341665
[4000]  training's binary_logloss: 0.136308     valid_1's binary_logloss: 0.243956
[6000]  training's binary_logloss: 0.083049     valid_1's binary_logloss: 0.215872
[8000]  training's binary_logloss: 0.0541907    valid_1's binary_logloss: 0.206349
[10000] training's binary_logloss: 0.0405135    valid_1's binary_logloss: 0.200333
[12000] training's binary_logloss: 0.0314357    valid_1's binary_logloss: 0.196361
[14000] training's binary_logloss: 0.0266671    valid_1's binary_logloss: 0.19684
[16000] training's binary_logloss: 0.0229593    valid_1's binary_logloss: 0.196581
[18000] training's binary_logloss: 0.020887     valid_1's binary_logloss: 0.196581
[20000] training's binary_logloss: 0.0193373    valid_1's binary_logloss: 0.19775
[22000] training's binary_logloss: 0.0181064    valid_1's binary_logloss: 0.19707
[24000] training's binary_logloss: 0.0173531    valid_1's binary_logloss: 0.196517
[26000] training's binary_logloss: 0.0166049    valid_1's binary_logloss: 0.196913
[28000] training's binary_logloss: 0.0161905    valid_1's binary_logloss: 0.195339
[30000] training's binary_logloss: 0.0160866    valid_1's binary_logloss: 0.195122
[32000] training's binary_logloss: 0.0159147    valid_1's binary_logloss: 0.196548
[34000] training's binary_logloss: 0.0157572    valid_1's binary_logloss: 0.19736
[36000] training's binary_logloss: 0.0156425    valid_1's binary_logloss: 0.197567
[38000] training's binary_logloss: 0.0155601    valid_1's binary_logloss: 0.198249
[40000] training's binary_logloss: 0.0155023    valid_1's binary_logloss: 0.199145
[42000] training's binary_logloss: 0.0155052    valid_1's binary_logloss: 0.200517
[44000] training's binary_logloss: 0.0154495    valid_1's binary_logloss: 0.201886
[46000] training's binary_logloss: 0.015375     valid_1's binary_logloss: 0.202715
[48000] training's binary_logloss: 0.0153219    valid_1's binary_logloss: 0.203269
[50000] training's binary_logloss: 0.0152853    valid_1's binary_logloss: 0.203606
--------------------------------------------------
lightgbm training fold 6
[2000]  training's binary_logloss: 0.251822     valid_1's binary_logloss: 0.305254
[4000]  training's binary_logloss: 0.137752     valid_1's binary_logloss: 0.2149
[6000]  training's binary_logloss: 0.0841859    valid_1's binary_logloss: 0.178178
[8000]  training's binary_logloss: 0.0550538    valid_1's binary_logloss: 0.164767
[10000] training's binary_logloss: 0.0411488    valid_1's binary_logloss: 0.157213
[12000] training's binary_logloss: 0.0319465    valid_1's binary_logloss: 0.151383
[14000] training's binary_logloss: 0.0267807    valid_1's binary_logloss: 0.152271
[16000] training's binary_logloss: 0.0229214    valid_1's binary_logloss: 0.151375
[18000] training's binary_logloss: 0.0208031    valid_1's binary_logloss: 0.15206
[20000] training's binary_logloss: 0.01932      valid_1's binary_logloss: 0.152681
[22000] training's binary_logloss: 0.0180624    valid_1's binary_logloss: 0.154805
[24000] training's binary_logloss: 0.0172254    valid_1's binary_logloss: 0.154982
[26000] training's binary_logloss: 0.0164263    valid_1's binary_logloss: 0.154331
[28000] training's binary_logloss: 0.0159945    valid_1's binary_logloss: 0.152144
[30000] training's binary_logloss: 0.0158889    valid_1's binary_logloss: 0.148446
[32000] training's binary_logloss: 0.0157587    valid_1's binary_logloss: 0.145524
[34000] training's binary_logloss: 0.015568     valid_1's binary_logloss: 0.14315
[36000] training's binary_logloss: 0.0153968    valid_1's binary_logloss: 0.142553
[38000] training's binary_logloss: 0.0152982    valid_1's binary_logloss: 0.141984
[40000] training's binary_logloss: 0.0152153    valid_1's binary_logloss: 0.141163
[42000] training's binary_logloss: 0.0151323    valid_1's binary_logloss: 0.140455
[44000] training's binary_logloss: 0.0151358    valid_1's binary_logloss: 0.139606
[46000] training's binary_logloss: 0.0150408    valid_1's binary_logloss: 0.139614
[48000] training's binary_logloss: 0.0150066    valid_1's binary_logloss: 0.140056
[50000] training's binary_logloss: 0.0149563    valid_1's binary_logloss: 0.140212
--------------------------------------------------
lightgbm training fold 7
[2000]  training's binary_logloss: 0.256157     valid_1's binary_logloss: 0.282579
[4000]  training's binary_logloss: 0.14371      valid_1's binary_logloss: 0.179044
[6000]  training's binary_logloss: 0.0899296    valid_1's binary_logloss: 0.135825
[8000]  training's binary_logloss: 0.0587899    valid_1's binary_logloss: 0.11667
[10000] training's binary_logloss: 0.0439796    valid_1's binary_logloss: 0.106559
[12000] training's binary_logloss: 0.0339917    valid_1's binary_logloss: 0.0994576
[14000] training's binary_logloss: 0.028343     valid_1's binary_logloss: 0.0936961
[16000] training's binary_logloss: 0.0241691    valid_1's binary_logloss: 0.0905641
[18000] training's binary_logloss: 0.0218713    valid_1's binary_logloss: 0.0906256
[20000] training's binary_logloss: 0.0201041    valid_1's binary_logloss: 0.0903999
[22000] training's binary_logloss: 0.018607     valid_1's binary_logloss: 0.0903996
[24000] training's binary_logloss: 0.0176802    valid_1's binary_logloss: 0.0922518
[26000] training's binary_logloss: 0.0169163    valid_1's binary_logloss: 0.0939674
[28000] training's binary_logloss: 0.016554     valid_1's binary_logloss: 0.0944728
[30000] training's binary_logloss: 0.0163403    valid_1's binary_logloss: 0.0946632
[32000] training's binary_logloss: 0.0160182    valid_1's binary_logloss: 0.0959939
```

```
[34000] training's binary_logloss: 0.015841    valid_1's binary_logloss: 0.0979622
[36000] training's binary_logloss: 0.0158555   valid_1's binary_logloss: 0.0995334
[38000] training's binary_logloss: 0.0158898   valid_1's binary_logloss: 0.101227
[40000] training's binary_logloss: 0.0158912   valid_1's binary_logloss: 0.103071
[42000] training's binary_logloss: 0.0159894   valid_1's binary_logloss: 0.104938
[44000] training's binary_logloss: 0.0159996   valid_1's binary_logloss: 0.106075
[46000] training's binary_logloss: 0.0159974   valid_1's binary_logloss: 0.107834
[48000] training's binary_logloss: 0.0160343   valid_1's binary_logloss: 0.109342
[50000] training's binary_logloss: 0.0160203   valid_1's binary_logloss: 0.11061
--------------------------------------------------
lightgbm training fold 8
[2000]  training's binary_logloss: 0.249766    valid_1's binary_logloss: 0.317574
[4000]  training's binary_logloss: 0.139762    valid_1's binary_logloss: 0.231413
[6000]  training's binary_logloss: 0.087177    valid_1's binary_logloss: 0.193275
[8000]  training's binary_logloss: 0.0570707   valid_1's binary_logloss: 0.169029
[10000] training's binary_logloss: 0.0423716   valid_1's binary_logloss: 0.155111
[12000] training's binary_logloss: 0.0325945   valid_1's binary_logloss: 0.146866
[14000] training's binary_logloss: 0.0273896   valid_1's binary_logloss: 0.141895
[16000] training's binary_logloss: 0.0234877   valid_1's binary_logloss: 0.139626
[18000] training's binary_logloss: 0.0213089   valid_1's binary_logloss: 0.138133
[20000] training's binary_logloss: 0.0197246   valid_1's binary_logloss: 0.136997
[22000] training's binary_logloss: 0.0183928   valid_1's binary_logloss: 0.135534
[24000] training's binary_logloss: 0.0175528   valid_1's binary_logloss: 0.133181
[26000] training's binary_logloss: 0.0168514   valid_1's binary_logloss: 0.132824
[28000] training's binary_logloss: 0.0164903   valid_1's binary_logloss: 0.132744
[30000] training's binary_logloss: 0.0163053   valid_1's binary_logloss: 0.132812
[32000] training's binary_logloss: 0.016056    valid_1's binary_logloss: 0.132853
[34000] training's binary_logloss: 0.0159538   valid_1's binary_logloss: 0.132945
[36000] training's binary_logloss: 0.0158875   valid_1's binary_logloss: 0.133574
[38000] training's binary_logloss: 0.0158548   valid_1's binary_logloss: 0.134527
[40000] training's binary_logloss: 0.0157682   valid_1's binary_logloss: 0.13497
[42000] training's binary_logloss: 0.0157762   valid_1's binary_logloss: 0.135267
[44000] training's binary_logloss: 0.0157078   valid_1's binary_logloss: 0.135402
[46000] training's binary_logloss: 0.0156055   valid_1's binary_logloss: 0.134848
[48000] training's binary_logloss: 0.0155719   valid_1's binary_logloss: 0.134877
[50000] training's binary_logloss: 0.0154665   valid_1's binary_logloss: 0.134386
--------------------------------------------------
lightgbm training fold 9
[2000]  training's binary_logloss: 0.249132    valid_1's binary_logloss: 0.330581
[4000]  training's binary_logloss: 0.13719     valid_1's binary_logloss: 0.259868
[6000]  training's binary_logloss: 0.0856526   valid_1's binary_logloss: 0.247146
[8000]  training's binary_logloss: 0.0564939   valid_1's binary_logloss: 0.242483
[10000] training's binary_logloss: 0.0421256   valid_1's binary_logloss: 0.243425
[12000] training's binary_logloss: 0.032474    valid_1's binary_logloss: 0.247867
[14000] training's binary_logloss: 0.0270399   valid_1's binary_logloss: 0.254305
[16000] training's binary_logloss: 0.0230006   valid_1's binary_logloss: 0.258114
[18000] training's binary_logloss: 0.020673    valid_1's binary_logloss: 0.260256
[20000] training's binary_logloss: 0.0190398   valid_1's binary_logloss: 0.255699
[22000] training's binary_logloss: 0.017727    valid_1's binary_logloss: 0.257821
[24000] training's binary_logloss: 0.0168632   valid_1's binary_logloss: 0.26109
[26000] training's binary_logloss: 0.0160323   valid_1's binary_logloss: 0.264075
[28000] training's binary_logloss: 0.0155176   valid_1's binary_logloss: 0.266002
[30000] training's binary_logloss: 0.0152811   valid_1's binary_logloss: 0.267597
[32000] training's binary_logloss: 0.0150165   valid_1's binary_logloss: 0.270536
[34000] training's binary_logloss: 0.014719    valid_1's binary_logloss: 0.274964
[36000] training's binary_logloss: 0.0144499   valid_1's binary_logloss: 0.280472
[38000] training's binary_logloss: 0.0143374   valid_1's binary_logloss: 0.281893
[40000] training's binary_logloss: 0.0142739   valid_1's binary_logloss: 0.283633
[42000] training's binary_logloss: 0.0142475   valid_1's binary_logloss: 0.285381
[44000] training's binary_logloss: 0.0142173   valid_1's binary_logloss: 0.286783
[46000] training's binary_logloss: 0.0141486   valid_1's binary_logloss: 0.288023
[48000] training's binary_logloss: 0.0141513   valid_1's binary_logloss: 0.290197
[50000] training's binary_logloss: 0.0140934   valid_1's binary_logloss: 0.292563
--------------------------------------------------
lightgbm training fold 10
[2000]  training's binary_logloss: 0.257802    valid_1's binary_logloss: 0.304896
[4000]  training's binary_logloss: 0.143402    valid_1's binary_logloss: 0.217758
[6000]  training's binary_logloss: 0.0877063   valid_1's binary_logloss: 0.179941
[8000]  training's binary_logloss: 0.0564943   valid_1's binary_logloss: 0.15912
[10000] training's binary_logloss: 0.0415212   valid_1's binary_logloss: 0.148131
[12000] training's binary_logloss: 0.0317468   valid_1's binary_logloss: 0.1419
[14000] training's binary_logloss: 0.0265053   valid_1's binary_logloss: 0.137396
[16000] training's binary_logloss: 0.0226465   valid_1's binary_logloss: 0.134378
[18000] training's binary_logloss: 0.0204782   valid_1's binary_logloss: 0.132486
[20000] training's binary_logloss: 0.0189869   valid_1's binary_logloss: 0.1304
[22000] training's binary_logloss: 0.0178852   valid_1's binary_logloss: 0.129018
[24000] training's binary_logloss: 0.0171111   valid_1's binary_logloss: 0.128068
[26000] training's binary_logloss: 0.0163095   valid_1's binary_logloss: 0.128016
[28000] training's binary_logloss: 0.0158448   valid_1's binary_logloss: 0.127429
[30000] training's binary_logloss: 0.0157027   valid_1's binary_logloss: 0.126766
[32000] training's binary_logloss: 0.0154712   valid_1's binary_logloss: 0.126664
[34000] training's binary_logloss: 0.015415    valid_1's binary_logloss: 0.126957
[36000] training's binary_logloss: 0.0152814   valid_1's binary_logloss: 0.127199
[38000] training's binary_logloss: 0.0152584   valid_1's binary_logloss: 0.127962
[40000] training's binary_logloss: 0.0152167   valid_1's binary_logloss: 0.128843
[42000] training's binary_logloss: 0.015232    valid_1's binary_logloss: 0.129459
[44000] training's binary_logloss: 0.0152039   valid_1's binary_logloss: 0.129409
[46000] training's binary_logloss: 0.0151354   valid_1's binary_logloss: 0.12968
[48000] training's binary_logloss: 0.0150907   valid_1's binary_logloss: 0.130332
[50000] training's binary_logloss: 0.0150543   valid_1's binary_logloss: 0.130257
```

```
lightgbm our out of folds CV score is 0.2433478814875392
--------------------------------------------------
xgboost training fold 1
[0]     train-logloss:0.68955   eval-logloss:0.69139
[1049]  train-logloss:0.04785   eval-logloss:0.42206
--------------------------------------------------
xgboost training fold 2
[0]     train-logloss:0.68980   eval-logloss:0.69074
[2000]  train-logloss:0.02634   eval-logloss:0.16347
[4000]  train-logloss:0.02197   eval-logloss:0.15946
[4021]  train-logloss:0.02195   eval-logloss:0.15945
--------------------------------------------------
xgboost training fold 3
[0]     train-logloss:0.68966   eval-logloss:0.69015
[2000]  train-logloss:0.02572   eval-logloss:0.28721
[2641]  train-logloss:0.02333   eval-logloss:0.28584
--------------------------------------------------
xgboost training fold 4
[0]     train-logloss:0.68948   eval-logloss:0.68999
[1056]  train-logloss:0.04640   eval-logloss:0.42808
--------------------------------------------------
xgboost training fold 5
[0]     train-logloss:0.68972   eval-logloss:0.69013
[2000]  train-logloss:0.02536   eval-logloss:0.25311
[3652]  train-logloss:0.02184   eval-logloss:0.24964
--------------------------------------------------
xgboost training fold 6
[0]     train-logloss:0.68961   eval-logloss:0.69091
[2000]  train-logloss:0.02610   eval-logloss:0.31762
[2141]  train-logloss:0.02534   eval-logloss:0.31758
--------------------------------------------------
xgboost training fold 7
[0]     train-logloss:0.68969   eval-logloss:0.68963
[2000]  train-logloss:0.02639   eval-logloss:0.18459
[4000]  train-logloss:0.02192   eval-logloss:0.17804
[4525]  train-logloss:0.02150   eval-logloss:0.17832
--------------------------------------------------
xgboost training fold 8
[0]     train-logloss:0.68965   eval-logloss:0.69105
[2000]  train-logloss:0.02540   eval-logloss:0.16245
[4000]  train-logloss:0.02140   eval-logloss:0.15699
[4242]  train-logloss:0.02125   eval-logloss:0.15714
--------------------------------------------------
xgboost training fold 9
[0]     train-logloss:0.68962   eval-logloss:0.69116
[1516]  train-logloss:0.03121   eval-logloss:0.31140
--------------------------------------------------
xgboost training fold 10
[0]     train-logloss:0.68953   eval-logloss:0.68989
[2000]  train-logloss:0.02554   eval-logloss:0.13727
[4000]  train-logloss:0.02116   eval-logloss:0.13203
[6000]  train-logloss:0.02009   eval-logloss:0.12967
[8000]  train-logloss:0.01945   eval-logloss:0.12838
[10000] train-logloss:0.01907   eval-logloss:0.12795
[10012] train-logloss:0.01905   eval-logloss:0.12795
xgboost our out of folds CV score is 0.36168974268250453
--------------------------------------------------
catboost training fold 1
0:      learn: 0.6899332        test: 0.6907982 best: 0.6907982 (0)     total: 58.1ms   remaining: 48m 54s
2000:   learn: 0.0414667        test: 0.3914124 best: 0.3802861 (1519)  total: 3.41s    remaining: 1m 22s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.3802861273
bestIteration = 1519

Shrink model to first 1520 iterations.
--------------------------------------------------
catboost training fold 2
0:      learn: 0.6902880        test: 0.6897226 best: 0.6897226 (0)     total: 1.88ms   remaining: 1m 35s
2000:   learn: 0.0455178        test: 0.1893871 best: 0.1890963 (1983)  total: 3.3s     remaining: 1m 20s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.174910968
bestIteration = 2391

Shrink model to first 2392 iterations.
--------------------------------------------------
catboost training fold 3
0:      learn: 0.6899973        test: 0.6903132 best: 0.6903132 (0)     total: 1.7ms    remaining: 1m 26s
2000:   learn: 0.0454009        test: 0.2314102 best: 0.2310818 (1996)  total: 3.31s    remaining: 1m 20s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.2227033843
bestIteration = 2594

Shrink model to first 2595 iterations.
--------------------------------------------------
catboost training fold 4
0:      learn: 0.6902377        test: 0.6903190 best: 0.6903190 (0)     total: 1.8ms    remaining: 1m 30s
Stopped by overfitting detector  (500 iterations wait)
```

```
bestTest = 0.3824828654
bestIteration = 1023

Shrink model to first 1024 iterations.
--------------------------------------------------
catboost training fold 5
0:      learn: 0.6902422        test: 0.6899385 best: 0.6899385 (0)     total: 1.97ms   remaining: 1m 39s
2000:   learn: 0.0419670        test: 0.2187739 best: 0.2187590 (1995)  total: 3.28s    remaining: 1m 19s
4000:   learn: 0.0081315        test: 0.1913288 best: 0.1913245 (3989)  total: 6.66s    remaining: 1m 17s
6000:   learn: 0.0037223        test: 0.1836002 best: 0.1834333 (5990)  total: 9.91s    remaining: 1m 13s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.1828987159
bestIteration = 6258

Shrink model to first 6259 iterations.
--------------------------------------------------
catboost training fold 6
0:      learn: 0.6902818        test: 0.6906277 best: 0.6906277 (0)     total: 1.71ms   remaining: 1m 26s
2000:   learn: 0.0465258        test: 0.2728627 best: 0.2702758 (1706)  total: 3.38s    remaining: 1m 21s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.2702757782
bestIteration = 1706

Shrink model to first 1707 iterations.
--------------------------------------------------
catboost training fold 7
0:      learn: 0.6901248        test: 0.6907622 best: 0.6907622 (0)     total: 1.95ms   remaining: 1m 38s
2000:   learn: 0.0448237        test: 0.1739162 best: 0.1738803 (1995)  total: 3.33s    remaining: 1m 20s
4000:   learn: 0.0091614        test: 0.1518411 best: 0.1518221 (3999)  total: 6.63s    remaining: 1m 17s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.1474822276
bestIteration = 4828

Shrink model to first 4829 iterations.
--------------------------------------------------
catboost training fold 8
0:      learn: 0.6906348        test: 0.6923224 best: 0.6923224 (0)     total: 2.02ms   remaining: 1m 41s
2000:   learn: 0.0459817        test: 0.1827642 best: 0.1827270 (1999)  total: 3.59s    remaining: 1m 27s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.1636855918
bestIteration = 2864

Shrink model to first 2865 iterations.
--------------------------------------------------
catboost training fold 9
0:      learn: 0.6892691        test: 0.6898655 best: 0.6898655 (0)     total: 1.79ms   remaining: 1m 30s
2000:   learn: 0.0461002        test: 0.3204117 best: 0.3190725 (1908)  total: 3.31s    remaining: 1m 20s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.3190725037
bestIteration = 1908

Shrink model to first 1909 iterations.
--------------------------------------------------
catboost training fold 10
0:      learn: 0.6908658        test: 0.6907668 best: 0.6907668 (0)     total: 1.91ms   remaining: 1m 36s
2000:   learn: 0.0460972        test: 0.1584097 best: 0.1583195 (1999)  total: 3.29s    remaining: 1m 19s
4000:   learn: 0.0091052        test: 0.1137035 best: 0.1136661 (3957)  total: 6.56s    remaining: 1m 16s
Stopped by overfitting detector  (500 iterations wait)

bestTest = 0.1096450309
bestIteration = 5000

Shrink model to first 5001 iterations.
catboost our out of folds CV score is 0.3166778761418612
```

Inference

In [21]:

```python
def lightgbm_inference(x_test: pd.DataFrame):
    test_pred = np.zeros(len(x_test))
    for fold in range(CFG.n_folds):
        model = pickle.load(open(CFG.MODEL_DATA_PATH / f'lightgbm_fold{fold + 1}_seed{CFG.seed}_ver{CFG.VER}.pkl', 'rb'))
        # Predict
        test_pred += model.predict(x_test)
    return test_pred / CFG.n_folds
def xgboost_inference(x_test: pd.DataFrame):
    test_pred = np.zeros(len(x_test))
    for fold in range(CFG.n_folds):
        model = pickle.load(open(CFG.MODEL_DATA_PATH / f'xgboost_fold{fold + 1}_seed{CFG.seed}_ver{CFG.VER}.pkl', 'rb'))
        # Predict
        test_pred += model.predict(xgb.DMatrix(x_test), iteration_range=(0, model.best_ntree_limit))
    return test_pred / CFG.n_folds

def catboost_inference(x_test: pd.DataFrame):
    test_pred = np.zeros(len(x_test))
```

```
    for fold in range(CFG.n_folds):
        model = pickle.load(open(CFG.MODEL_DATA_PATH / f'catboost_fold{fold + 1}_seed{CFG.seed}_ver{CFG.VER}.pkl', 'rb'))
        # Predict
        test_pred += model.predict_proba(x_test)[:, 1]
    return test_pred / CFG.n_folds

def gradient_boosting_model_inference(method: str, test_df: pd.DataFrame, features: list, categorical_features: list):
    x_test = test_df[features]
    if method == 'lightgbm':
        test_pred = lightgbm_inference(x_test)
    if method == 'xgboost':
        test_pred = xgboost_inference(x_test)
    if method == 'catboost':
        test_pred = catboost_inference(x_test)
    return test_pred
```

In [22]:
```
for method in CFG.METHOD_LIST:
    test_df[f'{method}_pred_prob'] = gradient_boosting_model_inference(method, test_df, features, categorical_features)
```

The result

In [23]:
```
test_df['class_1'] = 0.9 * test_df['lightgbm_pred_prob'] + 0.05 * test_df['xgboost_pred_prob'] + 0.05 * test_df['catboost_pred_prob']
test_df['class_0'] = 1 - test_df['class_1']
test_df[list(result_df)].to_csv('result.csv', index=False)
```

In [24]:
```
test_df[list(result_df)]
```

Out[24]:

	Id	class_0	class_1
0	00eed32682bb	0.641967	0.358033
1	010ebe33f668	0.641967	0.358033
2	02fa521e1838	0.641967	0.358033
3	040e15f562a2	0.641967	0.358033
4	046e85c7cc7f	0.641967	0.358033