In [1]:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import os
import sys
import json
import logging

sys.path.append('/home/moegwjawe1/atec_project/train/lmf/lib/python3.6/site-pack

import numpy as np
import pandas as pd

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV

import joblib

input_data_path = '/home/admin/workspace/job/input/train.jsonl'
output_model_path = '/home/admin/workspace/job/output/train.model'
result_path = '/home/admin/workspace/job/output/result.json'

input_data_path = '/mnt/atec/train.jsonl'

logging.warning("Begin Train.")
# 1. 读取训练数据进行训练
import time
time_start = time.time()
# pd_raw_data = pd.read_json(input_data_path, encoding='utf-8', lines=True, nrow
pd_raw_data = pd.read_json(input_data_path, encoding='utf-8', lines=True)
time_end=time.time()
print('time cost', time_end - time_start, 's')
logging.warning("Finish Read.")
```

```
WARNING:root:Begin Train.
WARNING:root:Finish Read.

time cost 20.96392798423767 s
```

In [2]:

```
# 2. 数据预处理
pd_data = pd_raw_data.copy(deep=True)
print(pd_data.shape)
# pd_data = pd_data.drop(pd_data.columns[np.where((pd_data.dtypes!='int64').valu
# 2.1 去除 label = -1
pd_data = pd_data.drop(pd_data[(pd_data['label'] == -1)].index)
# 2.2 去除文字列
pd_data = pd_data.drop(columns=['memo_polish'],axis=1)
print(pd_data.shape)
# 2.3 去除离群点
min_max_map={}
columns = pd_data.columns.values.tolist()[1:-1] # 去除 id 和 label 的所有列
for col in columns:
    if pd_data.dtypes[col]=='float64':
        d_min, d_max = pd_data[col].quantile([0.0001, 0.9999])
    elif pd_data.dtypes[col]=='int64':
        d_min, d_max = pd_data[col].quantile([0.001, 0.999])
    else:
        print("ERROR: ========================")
    print(col,d_min,d_max)
    min_max_map[col]=(d_min,d_max)
    pd_data = pd_data.drop(pd_data[(pd_data[col]>d_max)].index)
    pd_data = pd_data.drop(pd_data[(pd_data[col]<d_min)].index)
#     pd_data[pd_data[col]>d_max]=np.nan
#     pd_data[pd_data[col]<d_min]=np.nan
print(pd_data.shape)
# 2.4 去除取值只有一个值的列
columns = pd_data.columns.values.tolist()[1:-1] # 去除 id 和 label 的所有列
drop_list = []
for col in columns:
    num = len(pd_data[col].unique())
    if num == 1:
        drop_list.append(col)
pd_data = pd_data.drop(columns=drop_list)
print(pd_data.shape)
```

```
x353 -1.001 1.0
x354 0.0 116.53200000000652
x355 1.0 51.31267096773692
x356 -100.0 91.0
x357 -1.001 1.0
x358 -100.0 1082.7211013887252
x359 -100.0 392.0
x360 -1.001 1.0
x361 -100.0 1.0
x362 -1.003 0.0
x363 -100.0 1.0
x364 -100.0 50.0
x365 -100.0 0.0
x366 -100.0 36.0
x367 -100.0 -100.0
x368 -1.0 1.0
x369 0.0 161.0
x370 -1.001 1.0
x371 -100.0 -100.0
x372 0.0 1.0
```

In [4]:

```
print(len(drop_list))
print(drop_list)
print(min_max_map)
file = open('min_max_map.txt', 'w')
for k,v in min_max_map.items():
    file.write(str(k)+' '+str(v[0]) + ' ' +str(v[1])+'\n')
file.close()
```

```
82
['x2', 'x8', 'x13', 'x18', 'x25', 'x32', 'x33', 'x42', 'x48', 'x55',
'x68', 'x70', 'x71', 'x72', 'x77', 'x79', 'x91', 'x96', 'x97', 'x100',
'x107', 'x110', 'x111', 'x113', 'x121', 'x130', 'x135', 'x141', 'x14
5', 'x152', 'x155', 'x178', 'x184', 'x198', 'x202', 'x207', 'x209', 'x
215', 'x224', 'x225', 'x228', 'x232', 'x239', 'x248', 'x249', 'x258',
'x261', 'x264', 'x265', 'x277', 'x279', 'x280', 'x281', 'x289', 'x29
2', 'x300', 'x319', 'x325', 'x341', 'x343', 'x347', 'x351', 'x367', 'x
371', 'x373', 'x384', 'x385', 'x389', 'x394', 'x400', 'x419', 'x427',
'x431', 'x436', 'x441', 'x442', 'x451', 'x452', 'x456', 'x461', 'x46
2', 'x475']
{'x0': (0.0, 1.0), 'x1': (-2.0, 179.0), 'x2': (-100.0, -100.0), 'x3':
(-1.001, 2623.970399996324), 'x4': (-1.001, 300.0), 'x5': (-1.001, 30
0.0), 'x6': (-100.0, 0.0), 'x7': (-100.0, 100000.0), 'x8': (-1.0, 1.
0), 'x9': (-100.0, 1.0), 'x10': (0.0, 23.0), 'x11': (-1.001, 2.4516447
24085354), 'x12': (0.0, 136.15200000003097), 'x13': (0.0, 2.0), 'x14':
(2.0, 8.0), 'x15': (-1.0, 112.63400000000547), 'x16': (-1.0, 25.0), 'x
17': (0.0, 2030.7250000000058), 'x18': (-1.0, 5113.980449986075), 'x1
9': (-100.0, 465.0), 'x20': (-1.0, 1526.0), 'x21': (-100.0, 1.0), 'x2
```

In [16]:

```
# 3. 数据转 numpy
np_data = pd_data.values
# 去除索引列
np_data = np_data[:, 1:]
# 分成 input 和 output，output 降维到一维
np_data_input = np_data[:, :np_data.shape[1]-1]
np_data_input = np_data_input.astype(np.float64)

print(np_data_input.shape)
print(np_data_input)

np_data_output = np_data[:, np_data.shape[1]-1:]
np_data_output = np_data_output.astype(np.int64)
np_data_output = np.squeeze(np_data_output, axis=1)

print(np_data_output.shape)
print(np_data_output)

test_size=0.2
X_train, X_test, Y_train, Y_test = train_test_split(np_data_input, np_data_outpu
logging.warning("Finish Process.")
```

```
(43031, 398)
[[   1.      -2.       0.    ... -100.       0.       1.    ]
 [   1.      90.       0.    ... -100.       0.       1.    ]
 [   1.      69.       0.    ... -100.       0.       7.    ]
 ...
 [   1.      -2.       2.    ... -100.       0.       1.    ]
 [   1.     179.       0.    ... -100.      60.42  -100.    ]
 [   1.      -2.      -1.001 ... -100.       0.       7.    ]]
(43031,)
[0 1 1 ... 0 1 0]

WARNING:root:Finish Process.
```

In [17]:

```
# # 4. 数据归一化
# sc = StandardScaler()
# sc.fit(X_train)
# X_train = sc.transform(X_train)
# X_test = sc.transform(X_test)
```

In [18]:

```
##########################################################
logging.warning("Begin Model.")
result = {}  # 保存一些结果
# lightGBM 模型
model_name = 'lightGBM'
version = 'v0.4_test'
from lightgbm import LGBMClassifier

# 5.1 模型定义
model = LGBMClassifier(
    verbose=1
)

# 5.2 模型训练
import time
time_start=time.time()
model.fit(X_train, Y_train)
time_end=time.time()
print('time cost',time_end-time_start,'s')
# 5.3 模型评估
result['score_train'] = model.score(X_train, Y_train)
result['score_test'] = model.score(X_test, Y_test)
# 5.4 模型保存
joblib.dump(model, f'./model/%s_%s_[%f, %f, %f, %s].model' %
            (model_name, version,
             result['score_test'], result['score_train'],
             test_size, model))
# 5.5 结果保存
print(result)
##########################################################
```

```
WARNING:root:Begin Model.

[LightGBM] [Info] Number of positive: 12297, number of negative: 22127
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overh
ead of testing was 0.135918 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 35246
[LightGBM] [Info] Number of data points in the train set: 34424, numbe
r of used features: 395
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.357222 -> initscore=
-0.587443
[LightGBM] [Info] Start training from score -0.587443
time cost 3.3233299255371094 s
{'score_train': 0.8687834069254009, 'score_test': 0.8416405251539445}
```

In [19]:

```
# ########################################################
# logging.warning("Begin Model.")
# result = {}  # 保存一些结果
# # CatBoost 模型
# model_name = 'CatBoost'
# version = 'v0.4_test'
# from  catboost  import  CatBoostClassifier

# # 5.1 模型定义
# model = CatBoostClassifier(
#     verbose=1
# )

# # 5.2 模型训练
# import time
# time_start=time.time()
# model.fit(X_train, Y_train)
# time_end=time.time()
# print('time cost',time_end-time_start,'s')
# # 5.3 模型评估
# result['score_train'] = model.score(X_train, Y_train)
# result['score_test'] = model.score(X_test, Y_test)
# # 5.4 模型保存
# joblib.dump(model, f'./model/%s_%s_[%f, %f, %f, %s].model' %
#             (model_name, version,
#              result['score_test'], result['score_train'],
#              test_size, model))
# # 5.5 结果保存
# print(result)
# ########################################################
```

```
848:    learn: 0.2919997        total: 19s      remaining: 3.38s
849:    learn: 0.2919033        total: 19s      remaining: 3.36s
850:    learn: 0.2918047        total: 19.1s    remaining: 3.33s
851:    learn: 0.2916512        total: 19.1s    remaining: 3.31s
852:    learn: 0.2915599        total: 19.1s    remaining: 3.29s
853:    learn: 0.2914986        total: 19.1s    remaining: 3.27s
854:    learn: 0.2914383        total: 19.1s    remaining: 3.25s
855:    learn: 0.2913382        total: 19.2s    remaining: 3.22s
856:    learn: 0.2912935        total: 19.2s    remaining: 3.2s
857:    learn: 0.2912220        total: 19.2s    remaining: 3.18s
858:    learn: 0.2911435        total: 19.2s    remaining: 3.15s
859:    learn: 0.2910624        total: 19.2s    remaining: 3.13s
860:    learn: 0.2909642        total: 19.3s    remaining: 3.11s
861:    learn: 0.2908532        total: 19.3s    remaining: 3.09s
862:    learn: 0.2907623        total: 19.3s    remaining: 3.07s
863:    learn: 0.2906677        total: 19.3s    remaining: 3.04s
864:    learn: 0.2905833        total: 19.4s    remaining: 3.02s
865:    learn: 0.2904938        total: 19.4s    remaining: 3s
866:    learn: 0.2904242        total: 19.4s    remaining: 2.98s
```

In [21]:

```
# ########################################################
# logging.warning("Begin Model.")
# result = {}  # 保存一些结果
# # lightGBM 模型
# model_name = 'xgboost'
# version = 'v0.4_test'
# import xgboost as xgb

# # 5.1 模型定义
# model = xgb.XGBClassifier(
#     verbose=1
# )

# # 5.2 模型训练
# import time
# time_start=time.time()
# model.fit(X_train, Y_train)
# time_end=time.time()
# print('time cost',time_end-time_start,'s')
# # 5.3 模型评估
# result['score_train'] = model.score(X_train, Y_train)
# result['score_test'] = model.score(X_test, Y_test)
# # 5.4 模型保存
# joblib.dump(model, f'./model/%s_%s_[%f, %f, %f].model' %
#             (model_name, version,
#              result['score_test'], result['score_train'],
#              test_size))
# # 5.5 结果保存
# print(result)
# ########################################################
```

```
WARNING:root:Begin Model.
/home/moegwjawe1/atec_project/train/lmf/lib/python3.6/site-packages/xg
boost/sklearn.py:1224: UserWarning: The use of label encoder in XGBCla
ssifier is deprecated and will be removed in a future release. To remo
ve this warning, do the following: 1) Pass option use_label_encoder=Fa
lse when constructing XGBClassifier object; and 2) Encode your labels
(y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[19:10:54] WARNING: ../src/learner.cc:576:
Parameters: { "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by la
nguage bindings but
  then being mistakenly passed down to XGBoost core, or some parameter
actually being used
  but getting flagged wrongly here. Please open an issue if you find a
ny such cases.


[19:10:54] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'binary:logisti
c' was changed from 'error' to 'logloss'. Explicitly set eval_metric i
f you'd like to restore the old behavior.
time cost 12.514598608016968 s
{'score_train': 0.9287125261445504, 'score_test': 0.8350180085976531}
```

In [ ]:

```
#####################################################################
# 下面是 rank 代码
#####################################################################
```

In [22]:

```
# 2. 数据预处理
pd_data_rank = pd_raw_data.copy(deep=True)
print(pd_data_rank.shape)
# 2.1 去除 label 列
pd_data_rank = pd_data_rank.drop(columns=['label'], axis=1)
print(pd_data_rank.shape)
# 2.2 去除文字列
pd_data_rank = pd_data_rank.drop(columns=['memo_polish'], axis=1)
print(pd_data_rank.shape)
# 2.3 去除离群点
min_max_map = {}
file = open('min_max_map.txt', 'r')
for line in file.readlines():
    line = line.strip()
    line = line.split(' ')
    min_max_map[line[0]] = (float(line[1]), float(line[2]))
file.close()
columns = pd_data_rank.columns.values.tolist()[1:]  # 去除 id 的所有列
for col in columns:
    (d_min, d_max) = min_max_map[col]
    print(col, d_min, d_max)
    pd_data_rank[pd_data_rank[col] > d_max] = np.nan
    pd_data_rank[pd_data_rank[col] < d_min] = np.nan
print(pd_data_rank.shape)
# 2.4 去除取值只有一个值的列
drop_list = ['x2', 'x8', 'x13', 'x18', 'x25', 'x32', 'x33', 'x42', 'x48', 'x55',
             'x79', 'x91', 'x96', 'x97', 'x100', 'x107', 'x110', 'x111', 'x113',
             'x145', 'x152', 'x155', 'x178', 'x184', 'x198', 'x202', 'x207', 'x2
             'x232', 'x239', 'x248', 'x249', 'x258', 'x261', 'x264', 'x265', 'x2
             'x292', 'x300', 'x319', 'x325', 'x341', 'x343', 'x347', 'x351', 'x3
             'x389', 'x394', 'x400', 'x419', 'x427', 'x431', 'x436', 'x441', 'x4
             'x462', 'x475']
pd_data_rank = pd_data_rank.drop(columns=drop_list)
print(pd_data_rank.shape)
```

```
x402 -100.0 5.0
x403 -100.0 0.0
x404 -1.0 91656.79684362357
x405 -1.0 1.0
x406 0.0 2.0
x407 -1.0 0.05225940799998332
x408 -1.001 6.624499999998079
x409 0.0 23.0
x410 -1.001 5.0
x411 0.0 76251.99999994948
x412 -100.0 12.0
x413 -1.001 3.0
x414 -100.0 1.0
x415 0.0 1.0
x416 -1.001 17903.476899999645
x417 0.0 6.0
x418 0.0 129821.6106030297
x419 -1.002 -1.002
x420 -100.0 0.0
x421 -100.0 30.35000000000582
```

In [23]:

```
# 3. 数据转 numpy
np_data = pd_data_rank.values
# 去除索引列
np_data = np_data[:, 1:]
# 分成 input 和 output，output 降维到一维
np_data_input = np_data[:, :np_data.shape[1]-1]
np_data_input = np_data_input.astype(np.float64)

print(np_data_input.shape)
print(np_data_input)
```

```
(43031, 398)
[[   1.      -2.       0.    ... -100.       0.       1.    ]
 [   1.      90.       0.    ... -100.       0.       1.    ]
 [   1.      69.       0.    ... -100.       0.       7.    ]
 ...
 [   1.      -2.       2.    ... -100.       0.       1.    ]
 [   1.     179.       0.    ... -100.      60.42  -100.    ]
 [   1.      -2.      -1.001 ... -100.       0.       7.    ]]
```

In [26]:

```
#15. 模型预测
output_predictions_path = './predictions.jsonl'

model_path_list = [
#     ('./lightGBM_(0.010000, 0.858685, 0.853608).model', 1.0),
#     ('./CatBoost_(0.010000, 0.876104, 0.845361).model', 1.0),
#     ('./gbc_(0.010000, 0.830576, 0.820619).model', 1.0),
#     ('./rfc_(0.010000, 0.999979, 0.830928).model', 1.0),
#     ('./XGBoost_(0.010000, 0.902796, 0.837113).model', 1.0),
#     ('./lightGBM_(0.100000, 0.858832, 0.848185).model', 1.0),
#     ('./CatBoost_(0.100000, 0.877650, 0.845916).model', 1.0),
#     ('./XGBoost_(0.100000, 0.905705, 0.840553).model', 1.0),
#     ('./gbc_(0.100000, 0.831282, 0.836634).model', 1.0),
#     ('./rfc_(0.100000, 0.999977, 0.840347).model', 1.0),

#     ('./rfc.model', 0.003945), # 0.003945
#     ('./lightGBM.model', 0.054045), # 0.054045
#     ('./model/lightGBM_v0.1_test_[0.846638, 0.860215, 0.200000, LGBMClassifie
    ('./model/lightGBM_v0.4_test_[0.841641, 0.868783, 0.200000, LGBMClassifier(verbo
]

pred_list = []

for model_path in model_path_list:
    model = joblib.load(model_path[0])
    model_score = model_path[1]
    pred = model.predict_proba(np_data_input)
    pred_list.append(
        (pred, model_score)
    )

#16. 保存结果
with open(output_predictions_path, 'w') as fp:
    for i in range(np_data.shape[0]):
        label = 0
        score = 0
        for pred in pred_list:
            label += pred[0][i][1] * pred[1]
            # print(pred[0][i][1], pred[1])
            score += pred[1]
        label /= score
        # print("label:", label)
        fp.write('{"id": "%d", "label": %f}\n' % (i, label))
        logging.warning('{"id": "%d", "label": %f}\n' % (i, label))
```

```
WARNING:root:{"id": "2078", "label": 0.102900}

WARNING:root:{"id": "2079", "label": 0.773469}

WARNING:root:{"id": "2080", "label": 0.296777}

WARNING:root:{"id": "2081", "label": 0.013172}

WARNING:root:{"id": "2082", "label": 0.735827}

WARNING:root:{"id": "2083", "label": 0.154012}

WARNING:root:{"id": "2084", "label": 0.877750}

WARNING:root:{"id": "2085", "label": 0.451492}
```

WARNING:root:{"id": "2086", "label": 0.703396}

WARNING:root:{"id": "2087", "label": 0.627192}

In [ ]:

```
1
```