Week2

🩺 XGBoost

🕛 Light Boost

🔒 Azure本身服务器运维所提供的数据，其中包含的数据量更大而广，更有利于模型的泛化。
通过这一组数据建立一个
预测器，以预测机器可能出现failures或者errors的**概率**。
你将接触到不同维度的数据，包括时间序列、属性以及每个ID具备非结构化的数据。
其中包含：
Machine conditions and usage：机器的运行条件，例如： 从传感器收集的数据。
Failure history：机器或机器内组件的故障历史。
Maintenance history：机器的维修历史，例如错误代码、以前的维护活动或组件更换。
Machine features：机器的特性，例如引擎尺寸、品牌和型号、位置。

任务要求
使用这一套数据集，第一阶段独立学习并使用XGBoost完成故障的预测器
第二阶段进一步学习Light GBM，并使用LightGBM构建你的最终模型。

🔒 CSV的数据结构:

1. **PdM_errors.csv**（错误记录）：
   - `datetime`: 错误发生的日期和时间。
   - `machineID`: 机器ID。
   - `errorID`: 错误的ID。
2. **PdM_failures.csv**（故障记录）：
   - `datetime`: 故障发生的日期和时间。
   - `machineID`: 机器ID。
   - `failure`: 故障的ID（对应机器的故障组件）。
3. **PdM_machines.csv**（机器信息）：
   - `machineID`: 机器ID。
   - `model`: 机器的型号。
   - `age`: 机器的使用年限。
4. **PdM_maint.csv**（维护记录）：
   - `datetime`: 维护发生的日期和时间。
   - `machineID`: 机器ID。
   - `comp`: 更换的组件。
5. **PdM_telemetry.csv**（机器数据记录）：
   - datetime | machine ID | volt | rotate | pressure | vibration

🔒 XG Boost 预测器构建:

- **特征工程**
  &
  `time window`**:**failure&maint发生在每天的6点，以24H为时间窗口，计算前一天内组件的**平均值**

&`comp`，即维护记录中的更换组件信息，直接与机器的故障模式相关

添加最近维护时间列向量：计算每台机器每个组件距离上次维护的时间作为连续特征。

&
`errorID:`记构建当前时间发生errorID的时间间隔 和 三天内的发生次数

&
`model:` **标签编码 Label Encoder**

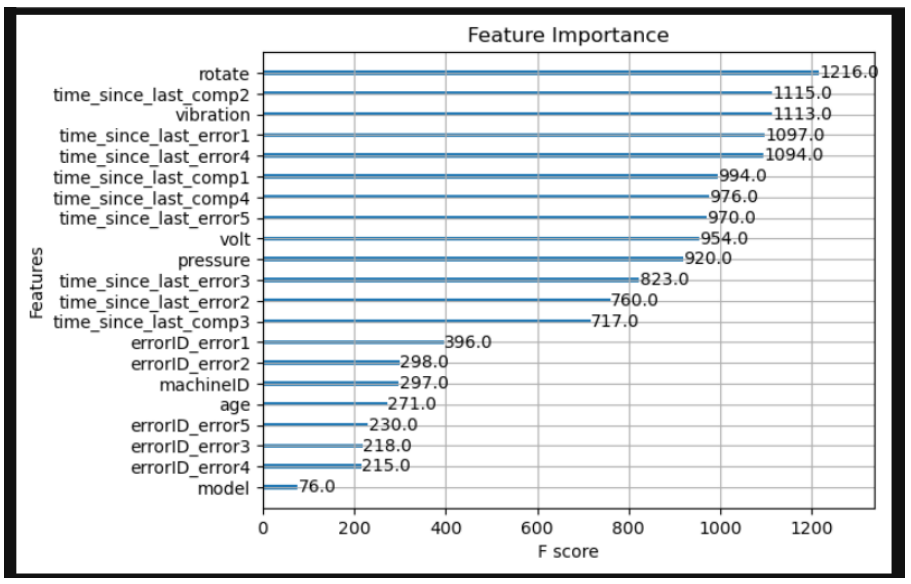
&`target_label:` **取值为0-9或0-5的Column Vector**

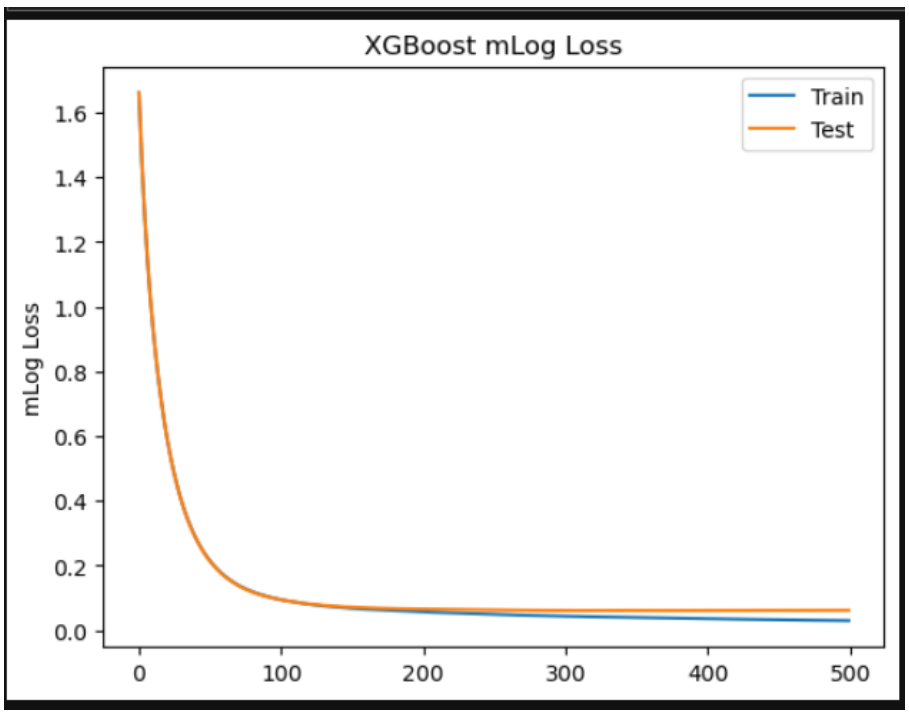
- **数据整合**
  整合所有数据集，确保根据
  `machineID`和`datetime`对齐
- **数据预处理**
  &转换
  时间格式
  &
  编码分类特征（model，failure ID，error ID）
  &
  欠采样&smote过采样

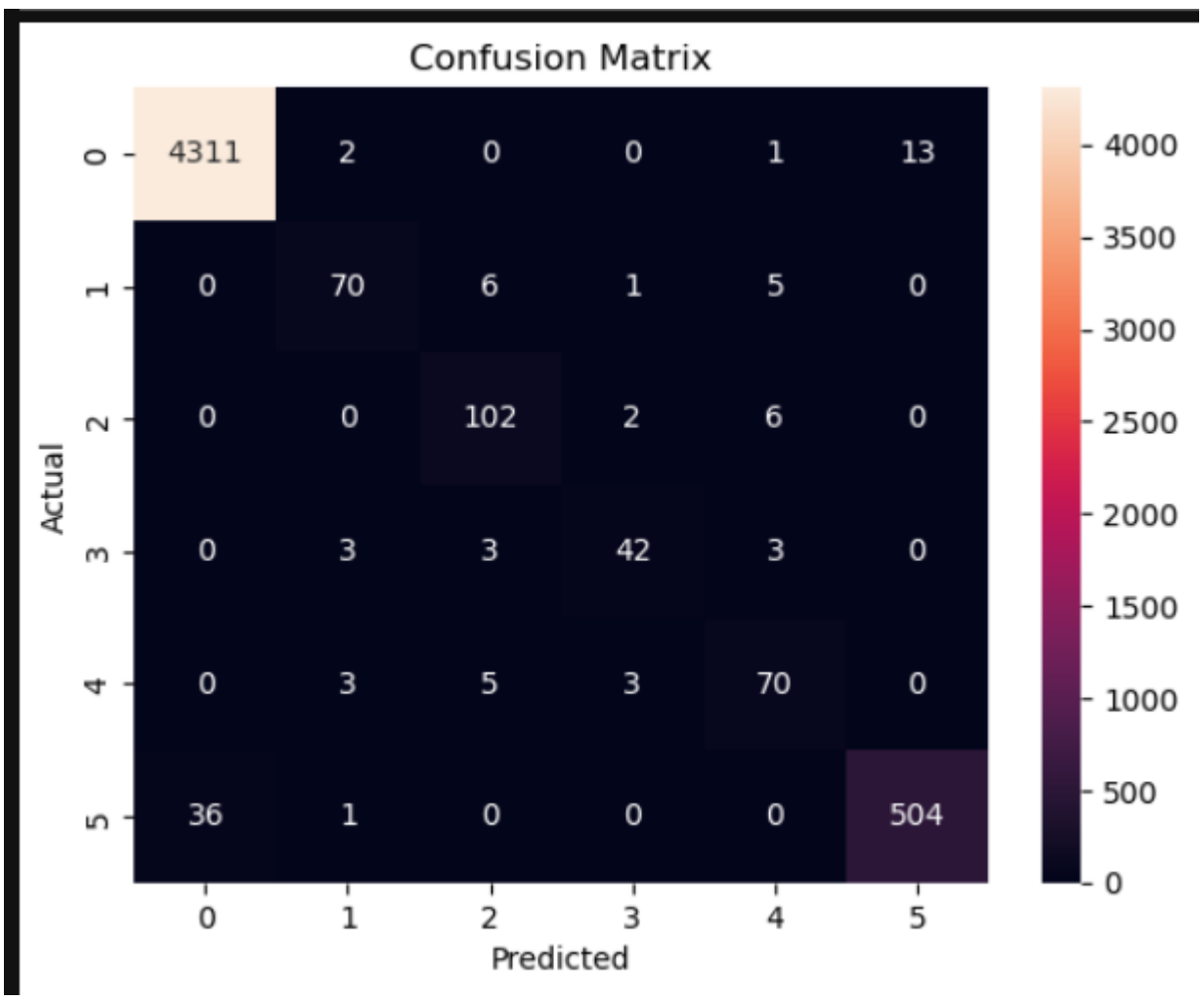
🔒 将所有Error归为一类 XGB model 1

```
undersampled_df = undersample_non_events(merged_df, 'label', no_event_code, undersample_ratio=5)

sampling_strategy = {label:times*sum(y_train == label) for label,times in zip(range(1, 5),[5,3,8

model = xgb.XGBClassifier(
    objective='multi:softprob',
    eval_metric='mlogloss',
    num_class=num_class,
    max_depth = 3,
    n_estimators = 300,
    eta = 0.05,
)

category_mapping = {
    'comp1': 1, 'comp2': 2, 'comp3': 3, 'comp4': 4,
    'error1': 5, 'error2': 5, 'error3': 5, 'error4': 5, 'error5': 5,
    'NoProb': 0
}

#---------------------------------------------------------
precision    recall  f1-score   support

           0       0.99      1.00      0.99      4327
           1       0.89      0.85      0.87        82
           2       0.88      0.93      0.90       110
           3       0.88      0.82      0.85        51
           4       0.82      0.86      0.84        81
           5       0.97      0.93      0.95       541
```

```
    accuracy                           0.98      5192
   macro avg       0.91      0.90      0.90      5192
weighted avg       0.98      0.98      0.98      5192
```



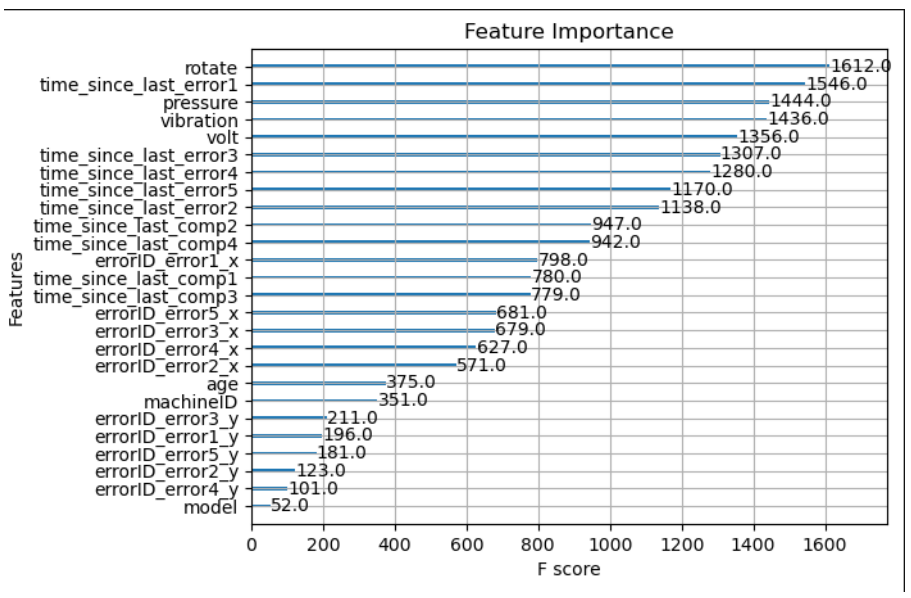


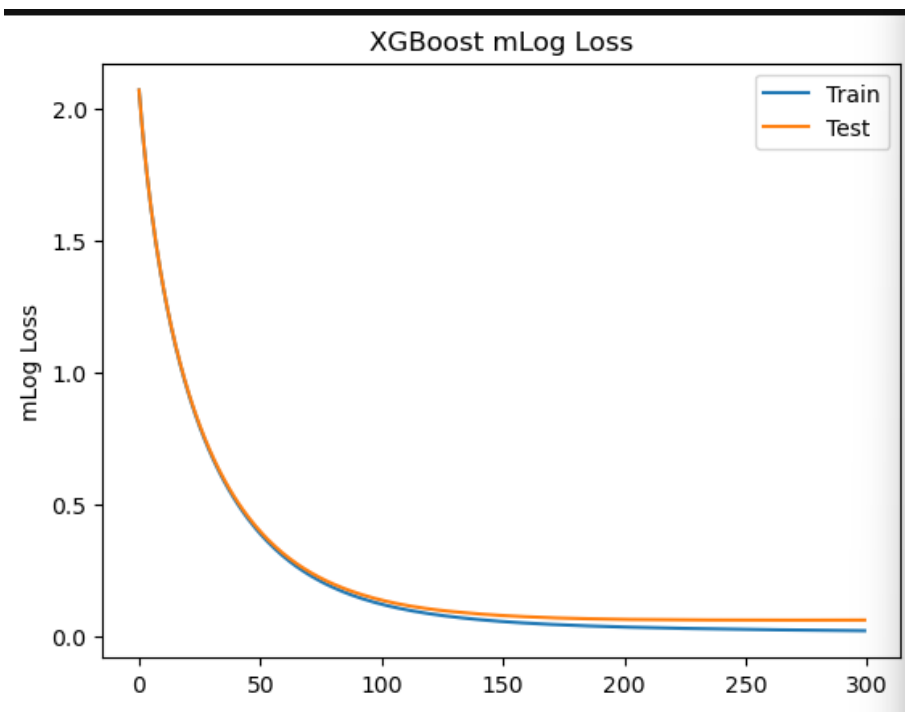


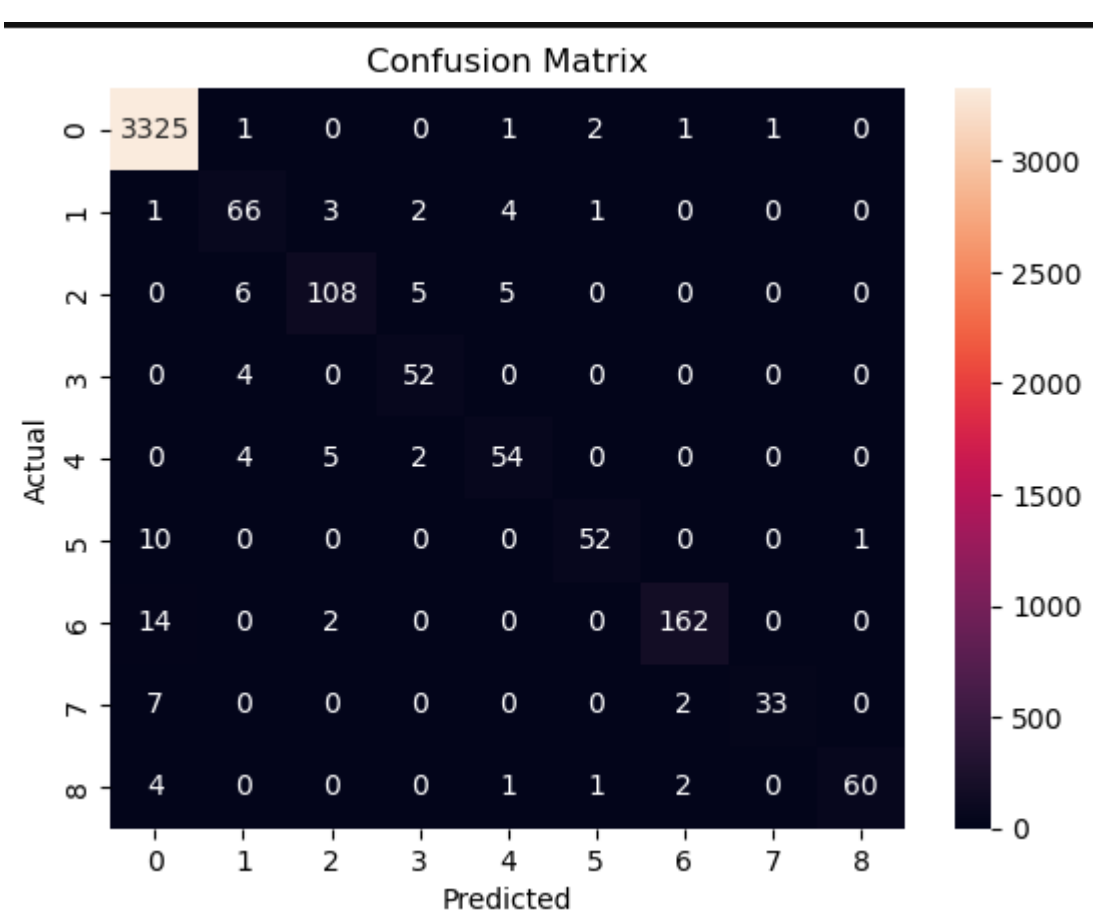
🔒 XGB model 2

```
sampling_strategy = {label:10*sum(y_train == label) for label in range(1, 10)}

num_class = len(y.unique())

model = xgb.XGBClassifier(
    objective='multi:softprob',
    eval_metric='mlogloss',
    num_class=num_class,
    max_depth = 3,
    n_estimators = 200,
```

```
    eta = 0.05,
)
#-----------------------------------------
precision    recall  f1-score   support

           0       0.99      1.00      0.99      3331
           1       0.81      0.86      0.84        77
           2       0.92      0.87      0.89       124
           3       0.85      0.93      0.89        56
           4       0.83      0.83      0.83        65
           5       0.93      0.83      0.87        63
           6       0.97      0.91      0.94       178
           7       0.97      0.79      0.87        42
           8       0.98      0.88      0.93        68

    accuracy                           0.98      4004
   macro avg       0.92      0.88      0.89      4004
weighted avg       0.98      0.98      0.98      4004
```







🔒 **Light Boost**
训练了两个模型:1.未过采样 2.过采样

过采样后的模型拟合效果很好

```
import lightgbm as lgb
from sklearn.metrics import classification_report


train_data = lgb.Dataset(X_train, label=y_train)
test_data = lgb.Dataset(X_test, label=y_test, reference=train_data)

#-----------------过采样
train_smote = lgb.Dataset(X_train_smote, label=y_train_smote)
test_smote_data = lgb.Dataset(X_test, label=y_test, reference=train_smote)

# 设置
# 设置参数
params = {
    'objective': 'multiclass',
    'num_class': num_class,
    'metric': 'multi_logloss',
    'boosting_type': 'gbdt',
    'max_depth': 4,
    'eta': 0.05,
    'n_estimators': 300,
    'bagging_fraction': 0.8,
    'bagging_freq': 3
}

# 训练模型
modelLGB1 = lgb.train(params, train_data, valid_sets=[train_data, test_data])
modelLGB2 = lgb.train(params, train_smote, valid_sets=[train_smote, test_smote_data])

#--------------------------------------------------------------------
precision    recall  f1-score   support

           0       0.99      1.00      0.99      3331
           1       0.83      0.87      0.85        77
           2       0.92      0.88      0.90       124
           3       0.85      0.91      0.88        56
           4       0.83      0.83      0.83        65
           5       0.93      0.81      0.86        63
           6       0.97      0.92      0.95       178
           7       0.94      0.74      0.83        42
           8       0.97      0.90      0.93        68

    accuracy                           0.98      4004
   macro avg       0.91      0.87      0.89      4004
weighted avg       0.98      0.98      0.98      4004


#--------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      3331
```

```
              1       0.86      0.91      0.89        77
              2       0.92      0.94      0.93       124
              3       0.89      0.89      0.89        56
              4       0.92      0.85      0.88        65
              5       0.90      0.89      0.90        63
              6       0.97      0.96      0.96       178
              7       0.93      0.93      0.93        42
              8       0.94      0.93      0.93        68

    accuracy                            0.98      4004
   macro avg       0.93      0.92      0.92      4004
weighted avg       0.98      0.98      0.98      4004
```