Final project

Brief introduction:

這次作業的目標是讓 model 從 product 的各種 feature 中，得到它會 failure 的機率，submission 的準確度會從預測機率和實際 target 之間的 roc curve 來做判斷，因此我認為這次作業的 training 部分可以分為 preprocessing、choosing best algorithm、choosing important features 這三部分，preprocessing 運用 EDA 來分析資料、imputer 來補齊 training data 的缺漏值，best algorithm 選擇了 Logistic Regression，最後選出了 model.coef 最高的 5 個 feature 來做最後 training，以防止 overfitting 的問題。

Methodology:

Preprocessing:

預處理主要為了清理一些導致 model 降低準確率的資料，找到 feature 之間的相關性來預防將太多 feature 加入導致 overfitting，這部分透過判斷 feature 的 correlation 和 kaggle 討論區，得到了幾個 feature，像是 measurement3-17 值很接近，因此直接計算他們的平均作為新 feature，attribute2 和 3 是 product 的 width 和 height，attribute 0 和 1 可以做 one-hot encoding，imputer 可以先用 product code 來分組，imputer 嘗試過 SimpleImputer、KnnImputer、InteractiveImputer，選擇 InteractiveImputer，最後將資料做 Standard Scaler，方便之後 model 運用。

Choosing best algorithm:

在比較前先用 StratifiedKFold(n=5)來做交叉驗證，之後選擇三種之前學到的 model(adaboost、random forest、logistic regression)，和兩個在 kaggle discussion 看到的 classifier(XGBRFClassifier、LGBMClassifier)

	AdaboostExtraTree	RandomForestClassifier	XGBRFClassifier	LGBMClassifier	LogisticRegression
avg	0.506852	0.546703	0.588033	0.58631	0.590659

最後選擇 logistic regression。

Dealing with imbalanced data:

在做 EDA 時發現 training data 的結果 imbalanced，

```
train_dfs.failure.value_counts()
```

```
0    20921
1     5649
Name: failure, dtype: int64
```

因此引入 smote 來平衡。

```
train_dfs.failure.value_counts()
```

```
0    20921
1    20921
Name: failure, dtype: int64


FOLD 0: 0.6038772817618971
FOLD 1: 0.5981082356328273
FOLD 2: 0.6029241773501876
FOLD 3: 0.5865106050558335
FOLD 4: 0.593783215848309

 Total Average: 0.5970407031298108
```

平衡後 logistic regression 的準確率從約 0.591 進步到 0.597。

Choose 5 important features:

從中選出 5 個比較重要的 feature 防止 overfitting，有試過 5、10 和全部的 features，最後選擇 5 個的 private score 較高。

```
withM5                  2.534256
loading                 1.550390
measurement_17          0.445112
measurement_1           0.178416
measurement_2           0.163968
attribute_1_material_6  0.111323
attribute_0_material_5  0.104382
withM3                  0.093530
measurement_avg         0.081801
attribute_0_material_7  0.057786
withM3&M5               0.044510
attribute_1_material_5  0.033315
2*3                     0.006404
measurement_0           0.002446
dtype: float64
```

Choose:

```
Index(['withM5', 'loading', 'measurement_17', 'measurement_1',
       'measurement_2'],
      dtype='object')
```

Choosing hyperparameters (penalty = l1 works better):

Logistic regression 有許多 hypermeters，引入 optuna 來找到 accuracy 最高的一組 hypermeters。

```
Best trial: score 0.5939954133217078, params {'penalty': 'l1', 'C': 0.018921011773659426, 'solver': 'saga'}
```

Summary:

將存好的 model 給 inference.ipynb 生成最後的 predict probability:

Model link:https://drive.google.com/file/d/1LjcYAse3otPVcob4ixxsKgN7YgWHXVzU/view?usp=sharing

id	failure
26570	0.452214
26571	0.414097
26572	0.431080
26573	0.435558
26574	0.640442
...	...
47340	0.553326
47341	0.388574
47342	0.387117
47343	0.501844
47344	0.417604

最後的 private score 為 0.59189。

Reference:

Measurement3 和 5 是否有缺漏:
https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/343368
Attribute0 和 1 one-hot encoding:
https://www.kaggle.com/code/samuelcortinhas/tps-aug-22-failure-prediction
Measurement 3-17 avg
https://www.kaggle.com/code/desalegngeb/tps08-logisticregression-and-some-fe
Optuna:
https://www.kaggle.com/code/mohamedmagdy11/tps-aug22-eda-logisticregression-with-optuna/notebook?scriptVersionId=103541473
Smote:
https://www.kaggle.com/code/usamabalochhh/data-is-imbalanced-use-smote
Iterative imputer:
https://www.kaggle.com/code/samuelcortinhas/tps-aug-22-failure-prediction#1.-Introduction

Github link:

https://github.com/yangalt/109550046_ML_Final