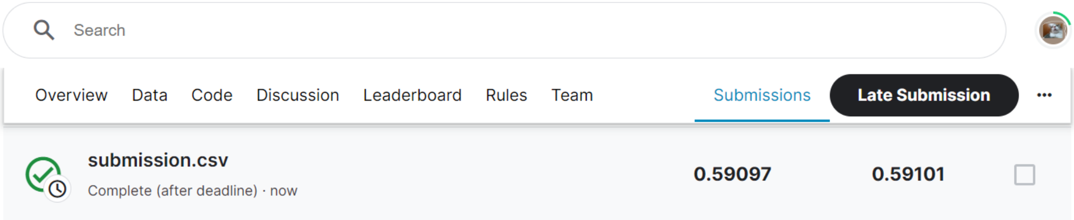
Final 109550119 邵筱庭

Model weight

https://drive.google.com/drive/folders/11QsOMRZtqqbmhv_44RS3scFosfWukHHK?usp=sharing

Score：0.59097, 0.59101



GitHub

https://github.com/ting0602/NYCU_ML_Final

Environment

python==3.10.4

numpy==1.23.3

pandas==1.5.1

IPython== 8.5.0

matplotlib==3.5.2

seaborn==0.12.2

colorama==0.4.4

sklearn==0.0

lightgbm==3.3.4

scipy==1.8.1

imblearn==0.0

Implementation details

- Outline
    I. Problem Introduction
    II. Data Processing
    III. Model Architecture
    IV. Code

I. Problem Introduction

在這次的題目中，主要在解決資料問題

1. 如何處理缺失資料
2. 觀察資料間關係

i. 找出與答案關聯度高的 features

ii. 組合有關連性的 features

II. Data Processing

1. New feature (沒有全部用到，但若要改寫則可以嘗試使用這些 feature)

根據 Kaggle 上的討論，新增多個新的欄位

a. m3_missing, m5_missing：當 measurement_3, measurement_5 為空，missing = 1，否則為 0

b. m3_17_avg：為 measurement_3 和 measurement_17 的平均值

c. m3_17_stdev：為 measurement_3 和 measurement_17 的標準差

d. m3_to_16_avg：為 measurement_3 至 measurement_16 的平均值

e. area：attribute_2 乘上 attribute_3

2. Data impute

使用 HuberRegressor 和 KNNImputer 來補齊欄位內的值

3. SMOTE (最終沒有使用)

由於 label 0 和 1 的比例不均，因此可考慮使用 SMOTE 處理資料不平衡的問題。

結果：一開始獲得較高的分數，但在固定 random_state 後，分數反而更低了。由於較高分數與使用 SMOTE 差距不大，因此選擇較穩定的方法 (不使用 SMOTE)

4. Reference

新增與處理 feature：

1. https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/343939
2. https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/342319
3. https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/342126

III. Model Architecture

LogisticRegression：

Params:

max_iter=1000, C=0.0001, penalty='l2', solver='newton-cg'

max_iter：Maximum number of iterations taken for the solvers to converge

C: Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

penalty: Specify the norm of the penalty ('l2': add a L2 penalty term and it is the default choice)

solver : Algorithm to use in the optimization problem (newton-cg：利用 Hessian matrix 來迭代以優化損失函數)

Intro:

為連續型的機率分佈，有分佈函數及密度函數

$$F(x) = P(X \leq x) = \frac{1}{1 + e^{-(x-\mu)/\gamma}}$$

$$f(x) = F'(X \leq x) = \frac{e^{-(x-\mu)/\gamma}}{\gamma(1 + e^{-(x-\mu)/\gamma})^2}$$

產生出決策邊界，輸入值後根據輸出(是否 > 0)判斷類別

IV. Code

1. Data Process

Data processing

- 新增 columns (m3_missing, m5_missing)：若 measurement_3, measurement_5 為空，此欄位為1
- 使用 np.log1p 對 loading 欄位的值做轉換

```python
# 合併 train & test
data = pd.concat([train, test])
data['m3_missing'] = data['measurement_3'].isnull().astype(np.int8)
data['m5_missing'] = data['measurement_5'].isnull().astype(np.int8)
data['loading'] = np.log1p(data['loading'])
display(data[:5])
```

- 使用 KNNImputer 補足資料

```python
feature = [f for f in test.columns if f.startswith('measurement') or f=='loading']

fill_dict = {
    'A': ['measurement_5','measurement_6','measurement_8'],
    'B': ['measurement_4','measurement_5','measurement_7'],
    'C': ['measurement_5','measurement_7','measurement_8','measurement_9'],
    'D': ['measurement_5','measurement_6','measurement_7','measurement_8'],
    'E': ['measurement_4','measurement_5','measurement_6','measurement_8'],
    'F': ['measurement_4','measurement_5','measurement_6','measurement_7'],
    'G': ['measurement_4','measurement_6','measurement_8','measurement_9'],
    'H': ['measurement_4','measurement_5','measurement_7','measurement_8','measurement_9'],
    'I': ['measurement_3','measurement_7','measurement_8']
}

for code in data.product_code.unique():
    tmp = data[data.product_code==code]
    column = fill_dict[code]
    tmp_train = tmp[column+['measurement_17']].dropna(how='any')
    tmp_test = tmp[(tmp[column].isnull().sum(axis=1)==0)&(tmp['measurement_17'].isnull())]
    print(f"code {code} has {len(tmp_test)} samples to fill nan")
    model = HuberRegressor()
    model.fit(tmp_train[column], tmp_train['measurement_17'])
    data.loc[(data.product_code==code)&(data[column].isnull().sum(axis=1)==0)&(data['measurement_17'].isnull()), 'measurement_17'] = model.predict(tmp_test[column])

    model2 = KNNImputer(n_neighbors=5)
    print(f"KNN imputing code {code}")
    data.loc[data.product_code==code, feature] = model2.fit_transform(data.loc[data.product_code==code, feature])
```

- 補足資料內容後，建立新的 columns

```python
data['m3_17_avg'] = (data['measurement_3'] + data['measurement_17']) / 2.0
data['m3_to_16_avg'] = (data['measurement_3'] + data['measurement_4'] + data['measurement_5'] + data['measurement_6'] + data['measurement_7'] + data['measurement_8'] + data['measurement_9'] + data['measurement_10'] +
data['m3_17_stdev'] = data[['measurement_3', 'measurement_17']].std(axis=1)
data['area'] = (data['attribute_2'] * data['attribute_3'])
display(data[:5])
```

- _scale(): 對 data 的 features 進行標準化

```python
def _scale(train_data, val_data, test_data, feats):
    scaler = StandardScaler()

    scaled_train = scaler.fit_transform(train_data[feats])
    scaled_val = scaler.transform(val_data[feats])
    scaled_test = scaler.transform(test_data[feats])

    # back to dataframe
    new_train = train_data.copy()
    new_val = val_data.copy()
    new_test = test_data.copy()

    new_train[feats] = scaled_train
    new_val[feats] = scaled_val
    new_test[feats] = scaled_test

    assert len(train_data) == len(new_train)
    assert len(val_data) == len(new_val)
    assert len(test_data) == len(new_test)

    return new_train, new_val, new_test
```

2. Train

a. 選擇 features

根據 kaggle 上的討論，得知資料間關聯

1. loading、measurement_17 和 label 關聯性大
2. measurement_3、measurement_5 有無缺失和 label 關聯性大
3. 不要一次選取過多 feature，效果反而不佳

總結：為了讓許多 feature 都能影響答案，且不要一次性選取過多 feature，因此採取三種 features 組合，最後在設置權重綜合出答案

Features 組合：

```
1. select_feature = ['m3_missing', 'm5_missing',
'measurement_1', measurement_2', 'loading', 'measurement_17']

2. select_feature = ['measurement_1', 'measurement_2',
'loading', 'measurement_17']

3. select_feature = ['loading', 'measurement_17',
'm3_17_avg']
```

b. 使用 KFold

使用 KFold 以及 LogisticRegression Model，並且計算出使用的 feature 的重要程度

```
# lr_oof_1：LogisticRegression 在每個驗證集上預測的預測機率。
# lr_oof_2：LogisticRegression 在每個驗證集上預測的預測類別。
# lr_test：LogisticRegression 在測試資料集上預測的預測機率。
# lr_auc：LogisticRegression 在交叉驗證過程中的平均 AUC 值。
# lr_acc：LogisticRegression 在交叉驗證過程中的平均準確度。
```

```
# importance_list：LogisticRegression 在各個 fold 上的特徵重要性。
```

```python
lr_oof_1 = np.zeros(len(X))
lr_oof_2 = np.zeros(len(X))
lr_test = np.zeros(len(test))
lr_auc = 0
lr_acc = 0
importance_list = []
model_list = ['./models/model3-1.pkl', './models/model3-2.pkl', './models/model3-3.pkl', './models/model3-4.pkl','./models/model3-5.pkl']

kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
for fold_idx, (train_idx, val_idx) in enumerate(kf.split(X, y)):
    print("Fold:", fold_idx+1)
    x_train, x_val = X.iloc[train_idx], X.iloc[val_idx]
    y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]
    x_test = test.copy()

    x_train, x_val, x_test = _scale(x_train, x_val, x_test, select_feature)
    #x_train, x_val, x_test = x_train.round(2), x_val.round(2), x_test.round(2)

    model = LogisticRegression(max_iter=1000, C=0.0001, penalty='l2', solver='newton-cg') # , class_weight='balanced'
    model.fit(x_train[select_feature], y_train)
    importance_list.append(model.coef_.ravel())

    val_preds = model.predict_proba(x_val[select_feature])[:, 1]
    lr_auc += roc_auc_score(y_val, val_preds) / 5
    y_preds = model.predict(x_val[select_feature])
    lr_acc += accuracy_score(y_val, y_preds) / 5
    lr_test += model.predict_proba(x_test[select_feature])[:, 1] / 5
    lr_oof_1[val_idx] = val_preds
    lr_oof_2[val_idx] = y_preds
    with open(model_list[fold_idx], 'wb') as pkl_file:
        pickle.dump(model, pkl_file, protocol=pickle.HIGHEST_PROTOCOL)

print(f"{Fore.GREEN}{Style.BRIGHT}Average auc = {round(lr_auc, 5)}, Average acc = {round(lr_acc, 5)}{Style.RESET_ALL}")
print(f"{Fore.RED}{Style.BRIGHT}OOF auc = {round(roc_auc_score(y, lr_oof_1), 5)}, OOF acc = {round(accuracy_score(y, lr_oof_2), 5)}{Style.RESET_ALL}")
```

```python
importance_df = pd.DataFrame(np.array(importance_list).T, index=x_train[select_feature].columns)
importance_df['mean'] = importance_df.mean(axis=1).abs()
importance_df['feature'] = x_train[select_feature].columns
importance_df = importance_df.sort_values('mean', ascending=False).reset_index().head(20)
plt.barh(importance_df.index, importance_df['mean'], color='lightgreen')
plt.gca().invert_yaxis()
plt.yticks(ticks=importance_df.index, labels=importance_df['feature'])
plt.title('LogisticRegression feature importances')
plt.show()
```

將使用 rankdata，將三個組的 lr_test 變成該組合對於 label 的預測結果並以權重(0.7, 0.05, 0.3)得到預測結果

```python
submission['rank0'] = rankdata(submission['lr0'])
submission['rank1'] = rankdata(submission['lr1'])
submission['rank2'] = rankdata(submission['lr2'])
```
✓ 0.5s

```python
submission['failure'] = submission['rank0']*0.70 + submission['rank1']*0.05 + submission['rank2']*0.30
```
✓ 0.3s

Reference

https://www.kaggle.com/code/takanashihumbert/tps-aug22-9th-solution/notebook?fbclid=IwAR0_uaztxUxw_pHXL74TZVjN-26DG_r5UCSAROUtrjfxGa0iUzjD1ekZE3c

https://www.kaggle.com/code/ambrosm/tpsaug22-eda-which-makes-sense