

ML Final Project:

Real-world Machine Learning Competition

Tabular Playground Series – Aug 2022

Student ID: 0816201 Student Name: 唐磊

Github Link

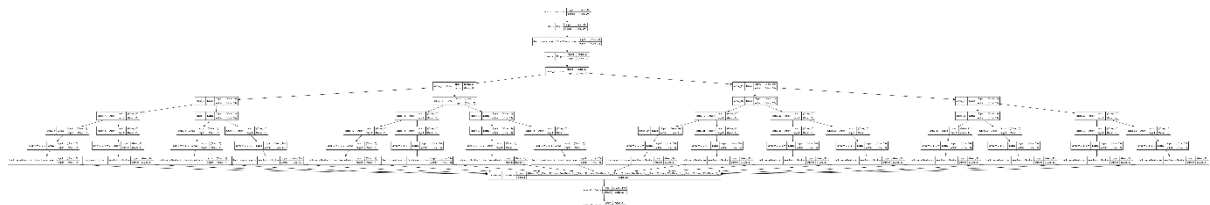
<https://github.com/martintl25/NYCU-2023-Fall-Machine-Learning>

Model Link

https://drive.google.com/drive/folders/1CFlpavrCRPPUtPRqz5KrvZkz_DEO5rd2?usp=sharing

Brief Introduction

對於這次的 ML 問題，起初我因為對 Feature Engineering/Feature Extraction 方法不甚熟悉，所以選擇使用 Neural Network 模型起步。希望能透過 NN 在訓練權重的同時學習特徵，但效果不彰（AUC score 約莫 0.53~0.55 之間），Bottleneck 可能在 Impute 方法以及初始權重方法上。然後，我看到 private LB 第一名的建立的模型架構滿複雜的（如下圖[1]所示），也沒辦法從他的分享裡面得到任何 Insight、為何這樣建模型，因此感到退怯，轉而選擇其他非 NN 模型。



最後，我參考了討論區的多篇 Data Analysis on Training Data 的 Post，並實際實驗後篩選、結合，最後決定出一組對於 LogisticRegression Model() 最有效的數據組合，最後將該數據組合應用在 LogisticRegression, SVM, Naïve_Bayes, GradientBoosting 等預設參數模型上，最後選擇分數最高的 LogisticRegression 進行調參，得到了在 0.59082 的 private score（截圖在 Summary Section）。下文，我會先解釋我使用的數據，介紹我 train model 所使用的 Missing Value Imputation、Scaler、Label Encoder，最後是我測試了所有模型的 Baseline 成績比較。

Methodology

```
label = train.pop('failure')
data = pd.concat([train, test])
data['loading'] = np.log(data['loading'])
data['m_3_missing'] = data.measurement_3.isna().astype(np.int8)
data['m_5_missing'] = data.measurement_5.isna().astype(np.int8)

data['area'] = data['attribute_2'] * data['attribute_3']
#data['measurement_2'] = data['measurement_2'].clip(11, None)
#data['measurement_avg'] would be added after imputation
#data['m_3_5_missing'] would be added after WoEEncoder
```

我分別對既有的 loading, attribute_0 欄位做了轉換，並加入了五個欄位，分別是 m_3_missing, m_5_missing, m_3_5_missing, area 以及 measurement_avg。

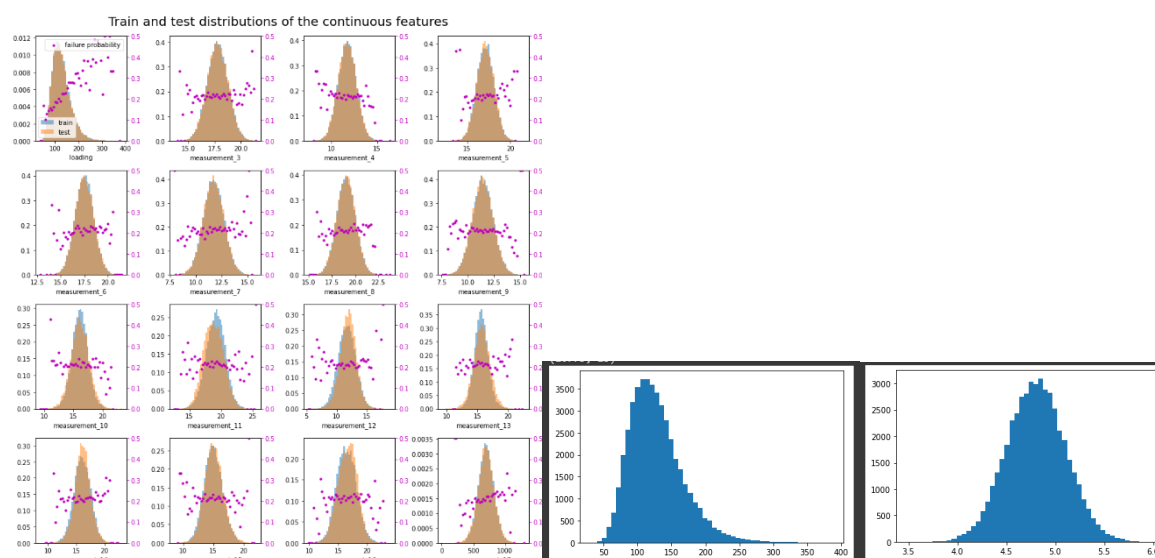


首先，先介紹新增的欄位。討論版上"TPSAUG22 EDA which makes sense"這篇文章的作者 AMBROSM[2]，他在文章內分析了每個有 missing value 的數據，當該數據缺失時發生 failure 的機率分布，發現當 measurement_3 或 measurement_5 消失時發生 failure 的機率，在整體機率分布明顯離均。因為此特性判斷，在 Impute 前加入表示 measurement_3 以及 measurement_5 消失的欄位能幫助判定 failure 發生機率。並以此作為出發點，我加入了表示 measurement_3 與 measurement_5 如果在一筆數據同時消失的欄位，經實驗發現有更好的 AUC 表現，0.59066->0.59073。

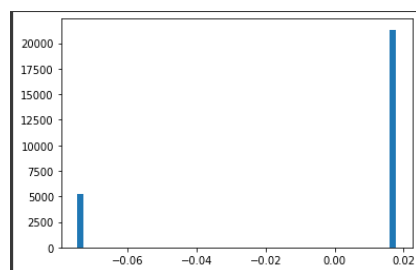
```
'''
    Encode the attribute_0 to float number
'''
woe_encoder = WoEEncoder(variables=['attribute_0'])
woe_encoder.fit(X, y)
X = woe_encoder.transform(X)
test = woe_encoder.transform(test)

# I don't know why if a put the snippet here,
# I would get a better result on LB @@
X['m_3_5_missing'] = X['m_5_missing'] * (X['m_3_missing'])
test['m_3_5_missing'] = test['m_5_missing'] * (test['m_3_missing'])
```

area 與 measurement_avg 的想法則是來自於"Less can be more: Feature Engineering Ideas"[3]的看法。Area 是為一筆資料 attribute 2 與 attribute 3 的乘數。Attribute 參數是與 Product Code 綁定的數據，也就是相同的 Product Code 的資料會有相同的 Attribute 組合。這篇文章只有說這是實驗結果，並未提出可信的數據分析、Proof of Concept，但是從這篇文章釋出後，不少人分析這些 Data 可能的背景，例如可能是肥皂、海綿生產商，attribute 數值是為產品規格，因此兩個 attribute 間的乘數能夠表示一個產品，能夠作為分析 failure rate 的依據。並且這篇文章發現，同樣的 Product Code 會有相近的 measurement3~16 的平均以及離均差。但我實驗後發現離均差在加入後反而效果相較較差，因此，我最後只有取 area 以及 measurement_avg。



介紹完新增的欄位後，再來是對原有數據的轉換。AMBROSM[2]印出每個 float column 的 distribution，發現 loading 欄位的數據 distribution 有偏度，疑似是 log normal distribution。因此，我對數據做了 log 轉換，上圖可見轉換前後的數據分布。轉換後的數據能在經過 StandScaler 標準化後保有原有的數據資訊量。並且我們能看到 loading 數據有與 failure rate 有高度正向關係。



再來是，我對 attribute_0 使用 WoEEncoder (上頁圖三)，將分類資料從字串映射到一個 float 數值，WoEEncoder 會透過監督式學習，學習出對應用在該 training dataset 適合的映射關係。然而，我發現我加入 m_3_5_missing 後進行 WoEEncode 的化，雖然在 training data 上有更好的表現，但似乎在 test data 反而效果較差，但也沒差多少影響的 AUC score 數量級在 $1e-5$ 。

預處理的最後是 Missing Value 的 Imputation 方法以及 Scaler。

```

model1 = HuberRegressor(epsilon=1.9)
model2 = KNNImputer(n_neighbors=3)
#model2 = LGBMImputer(n_iter=50)
#model2 = IterativeImputer(random_state=0)

feature = [f for f in data.columns
            if f.startswith('measurement') or f=='loading']
nullValue_cols = [col for col in train.columns
                   if train[col].isnull().sum()!=0]

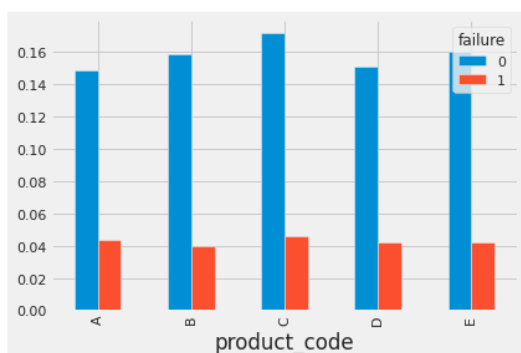
# For each product code group,
# impute the missing value via HuberRegressor(model1) first
# then impute the rest via KNNImputer

for code in data.product_code.unique():
    # HuberRegressor
    for measurement_col in list(full_fill_dict.keys()):
        tmp = data[data.product_code==code]
        column = full_fill_dict[measurement_col][code]
        tmp_train = tmp[column+measurement_col].dropna(how='any')
        tmp_test = tmp[((tmp[column].isnull().sum(axis=1)==0)
                        &(tmp[measurement_col].isnull()))]

        model1.fit(tmp_train[column], tmp_train[measurement_col])
        data.loc[((data.product_code==code)
                  &(data[column].isnull().sum(axis=1)==0)
                  &(data[measurement_col].isnull())), measurement_col] = model1.predict(tmp_test[column])

    # KNNImputer
    data.loc[data.product_code==code, feature] = model2.fit_transform(data.loc[data.product_code==code, fe

```



我在一開始選擇 NN Model 起手，在訓練 NN 的時候，我嘗試過 KNNImputer 以及 SimpleImputer 的平均、中位數與眾數等等 Impute 方法，效果都不太好。NARUKE[4] 發現每組相同 product code 的數據都有他的數據分布特性以及群組的 failure rate，因此選用 HuberRegressor 回歸的方法去填補數值。首先，計算所有數據內的相關係數，取前 10 個與其他欄位最相關的欄位。從這些欄位中建立 full_fill_dict，尋找每組數據中，最有關係的四個欄位。最後依據該 full_fill_dict 去回填，該 10 個欄位以及 measurement_17。其他欄位的空缺則用 KNNImputer 回填。

我嘗試使用 LGBMImputer 以及 Iterative Imputer 效果都沒有 KNNImputer 來得好。預處理的最後一步，因為大多都為 Normal Distribution，我使用 StandardScaler 將所有數據標準化到 -1 與 1 之間，有測過 RobustScaler 但效果不好。測試 MultinomialNB 的話，要改為 MinMaxScaler，或再對 StandardScaler 的數據進行處理，使值域為正數。

最後，我先選用 LogisticRegression 的預設參數 Model，進行訓練，印出 feature_importance_將數據組合不斷刪減。最後找到了下列數據組合。

```
selected_cols = [
    'loading',
    'attribute_0',
    'measurement_17',
    'measurement_0',
    'measurement_1',
    'measurement_2',
    'area',
    'm_3_missing',
    'm_5_missing',
    'measurement_avg',
    # 'measurement_stddev',
    'm_3_5_missing'
]
```

找到上述數據組合後，分別對多個模型進行訓練，分別為 SVC, MultinomialNB, BernoulliNB, GaussianNB, Adaboost, RandomForest，求其 predict_proba 結果的最好的 AUC 結果的模型。最後發現，還是 LogisticRegression 在預設參數、沒有調參的狀況下，效果最好。調參後，AUC score 達到 0.59073。另外，官方文檔建議當 sample size 遠大於 feature 數量的時候建議選用新版 scikit-learn v1.2 的 'newton-chlesky'[5]，嘗試過後其結果卻與 newton-cg 相同，生成的 submission 也無異。

```
'''
LR no Hyperparameter Tuning
Average auc = 0.59065
OOF auc = 0.59051
'''

'''
Best Performance
Average auc = 0.59073
OOF auc = 0.59059
'''
```

```
'''
SVM
Average auc = 0.50926
OOF auc = 0.5079
'''
```

```
'''
MultinomialNB() MinMaxScaler
Average auc = 0.54283
OOF auc = 0.53971
'''

'''
BernoulliNB() StandardScaler
Average auc = 0.57341
OOF auc = 0.57322
'''

'''
GaussianNB() StandardScaler
Average auc = 0.58397
OOF auc = 0.55722
'''
```








```
'''
default parameter
Average auc = 0.54735
OOF auc = 0.54728
RandomForestClassifier(n_estimators=100, max_depth=6)
Average auc = 0.58618
OOF auc = 0.58572
'''

'''
default parameter
Average auc = 0.5815
OOF auc = 0.58141
AdaBoostClassifier(n_estimators=50, learning_rate=0.1)
Average auc = 0.58818
OOF auc = 0.58742
'''

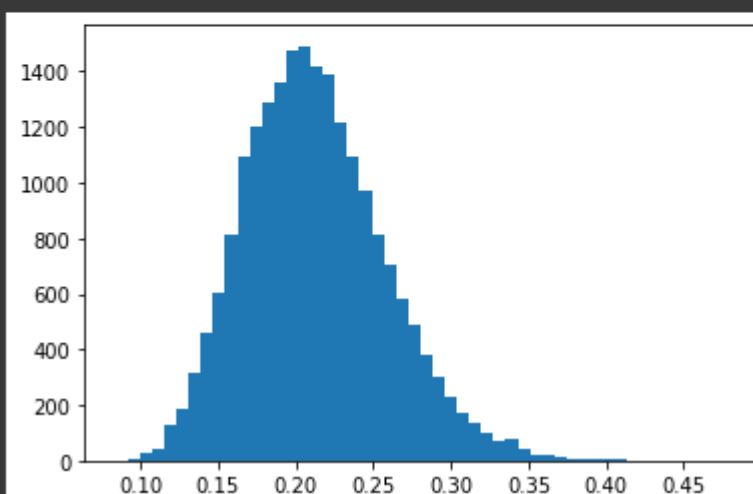
'''
GradientBoostingClassifier()
Average auc = 0.58294
OOF auc = 0.58276
'''
```

Summary

總結以上，我結合 HuberRegressor 以及 KNNImputer，對整體訓練資料集合高度相關的欄位用組內回歸的方便填補 missing value，剩餘則用 KNN 回補。參考其他參賽者的 FE Analysis，加入了五個欄位。最後，從多個 Model，選用了 LogisticRegression Model，進行了調參，得到最佳 private LB 成績 0.59082 的成績。

All Successful Selected Errors				Recent ▾
Submission and Description		Private Score ⓘ	Public Score ⓘ	Selected
 0816201 (2).csv Complete (after deadline) · 1h ago		0.59082	0.59165	<input type="checkbox"/>
 0816201 (1).csv Complete (after deadline) · 1h ago		0.5908	0.59164	<input type="checkbox"/>
 0816201.csv Complete (after deadline) · 8h ago		0.59082	0.59165	<input type="checkbox"/>
 submission (10).csv Complete (after deadline) · 1d ago		0.59051	0.5895	<input type="checkbox"/>
 submission (9).csv Complete (after deadline) · 2d ago		0.59082	0.59165	<input type="checkbox"/>
 submission (8).csv Complete (after deadline) · 2d ago		0.59062	0.59117	<input type="checkbox"/>
 submission (7).csv Complete (after deadline) · 2d ago		0.59035	0.59129	<input type="checkbox"/>

```
lr_test = np.zeros(len(test))
import joblib
path = '/content/drive/MyDrive/ML/Project/model'
for key in [0, 1, 2, 3, 4]:
    model = joblib.load(f'{path}{key}.sav')
    lr_test += model.predict_proba(X_test)[:, 1] / 5
plt.hist(lr_test, bins=50)
plt.show()
```



Reference

- [1] R. Gillbert, "What a surprised !!" 2022 [Online]. Available: <https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/349385> [Accessed: Jan. 10, 2023]
- [2] Ambrosm, "TPSAUG22 EDA which makes sense " 2022 [Online]. Available: <https://www.kaggle.com/code/ambrosm/tpsaug22-eda-which-makes-sense/notebook> [Accessed: Jan. 10, 2023]
- [3] DES., "Less can be more: Feature Engineering Ideas" 2022 [Online]. Available: <https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/342126> [Accessed: Jan. 10, 2023]
- [4] Cabaxiom, " [TPS-AUG-22] EDA + Logistic Regression Baseline " 2022 [Online]. Available: <https://www.kaggle.com/code/cabaxiom/tps-aug-22-eda-logistic-regression-baseline/notebook> [Accessed: Jan. 10, 2023]
- [5] Scikit-learn, "sklearn.linear_model.LogisticRegression" 2023 [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html [Accessed: Jan. 10, 2023]