


# Machine Learning Final Project 109550171 陳存佩

**GitHub link:** <https://reurl.cc/EX0ypk>

**Weight link:** <https://reurl.cc/NGM97x>

## Result

| Submission and Description   | Private Score ⓘ | Public Score ⓘ | Selected                 |
|--|-----------------|----------------|--------------------------|
|  <b>submission.csv</b><br>Complete (after deadline) · now | <b>0.5916</b>   | <b>0.58352</b> | <input type="checkbox"/> |

## Reference

- ◆ Frame ref.1 <https://www.kaggle.com/code/anubhavde/tabular-playground-series-august2022>
- ◆ Frame ref.2 <https://www.kaggle.com/code/ambrosm/tpsaug22-eda-which-makes-sense>
- ◆ PCA & Standardization <https://www.jcchouinard.com/pca-with-python/>

## Brief introduction

任務的目標是根據現有資料預測 Keep It Dry 公司產品 Super Soaker 的產品失敗機率，train data 和 test data 分別有不同的 product code。

因為 data 特徵數量多且複雜，且其中有很多 noise，所以我將訓練的重點放在資料的處理和過濾上，模型則是選用 LogisticRegression。

## Methodology (Data pre-process, Model architecture, Hyperparameters, ...)

### Data pre-process

- ◆ Flags of missing values

因為 measurement\_3, 5 是否 missing 和 failure 的相關程度高，因此在原始資料上新增兩欄去紀錄是否 missing。

- ◆ **Dropping some columns**

透過視覺化 data 去觀察每個 feature 和 failure 的關係，進而決定移除一些與 failure 低相關的 column（如 product code, attribute\_0~3）。

- ◆ **Filling the missing values**

Data 內有很多空值，必須進行預測再丟入模型，這邊選用的是 IterativeImputer，比較特別的是 measurement\_17，因為他在眾多特徵中和 failure 是較高度相關，若能成功預測空值將能提升準確率，且發現他可以由 measurement\_3~9 的線性組合推算出來，而不同的 product code 的權重也不相同，因此將每個 product code 都用 LinearRegression 找出 measurement\_17 和 measurement\_3~9 的關係，並用此預測遺失的 measurement\_17（若 measurement\_3~9 本身有遺失，則先用 IterativeImputer 補值）。

- ◆ **PCA(Principal Component Analysis)**

PCA 常用來降低資料維度與分析資料，因為 measurement\_3~16 的資料屬於同個 group，而個別去看的參考價值不高，所以使用 PCA 去做降維，減少需要訓練的特徵，聚焦在較重要的特徵上。

- ◆ **Calculating area**

推測 attribute2 & 3 代表的是 soaker 的面積，因此將兩者相乘。

- ◆ **Standardization**

將要訓練的 feature 都進行標準化。

## **Model architecture & Hyperparameters**

`LogisticRegression(penalty='l1', C=0.01, solver='liblinear')`

## **Summary**

這次的資料較接近現實世界，有很多 noise 和缺失值，要怎麼去處理是訓練的關鍵，其中較為重要的處理是 measurement\_17，因為他與 failure 高度相關，但 train data 中有約 1/10 的缺失，在找到他與其他特徵的線性相關並用來預測後，模型準確度有明顯提升，這次的作業參考了很多 kaggle 討論區的 Insights，也學習到要怎麼利用各種方式（資料背景、視覺化資料等）去分析資料，進而做最好的處理。

## Comparisons of different approaches

Methods of filling missing values

| Method   | Private Score |
|--|---------------|
| IterativeImputer only  | 0.59082       |
| IterativeImputer &<br>LinearRegression for measurement_17                      | 0.59128       |
| KNNImputer only(n_neighbors=10)  | 0.5909        |
| KNNImputer(n_neighbors=10) &<br>LinearRegression for measurement_17            | 0.59101       |
| SimpleImputer only(strategy=most_frequent)                                     | 0.59103       |
| SimpleImputer(strategy=most_frequent) &<br>LinearRegression for measurement_17 | <b>0.5916</b> |

Models

| Models             | Private Score |
|--------------------|---------------|
| LogisticRegression | <b>0.5916</b> |
| LinearRegression   | 0.5909        |

## Comprehensive related works survey

- ◆ <https://www.kaggle.com/code/ambrosm/tpsaug22-eda-which-makes-sense>
  1. 透過大量圖表分析各特徵的分佈、以及和 failure 的關係，透過視覺化資料可以更清楚知道哪些資料跟結果是低相關，進而選擇過濾。
  2. 分析各特徵 missing values 的數量，發現占的比例都偏高，因此 imputer 變得重要。
  3. 發現 attribute0~3 的組合即為 product code。
  4. 使用 GroupKFold 來驗證與調參，調參階段好用，但訓練最終模型我選擇將所有 train data 都加入訓練。
  5. 使用 LogisticRegression model，並計算 feature importance。
- ◆ <https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/341462>

這則文章在探討這個比賽數據的意義，看完會更了解各特徵的含義，進而

幫助資料的 pre-processing（例如知道 loading 是能承受的水量，因此就會是評估海綿好壞的重要特徵、發現 measurements 之間有相關等）

- ◆ <https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/343939>

這則討論發現了 measurement\_17 和 measurement\_3~9 的線性組合有高度正相關，而權重會根據不同 product code 而不同，透過這個關係可以用 measurement\_3~9 去預測 missing 的 measurement\_17，比起直接使用 imputer 套件，準確度會更高。

## Thorough experimental results

Ablation study (SimpleImputer with most\_frequent, LogisticRegression)

| Missing flag | PCA | area | standardization | Private Score |
|--------------|-----|------|-----------------|---------------|
| x            | x   | x    | x               | 0.58175       |
| ✓            | x   | x    | x               | 0.58174       |
| x            | ✓   | x    | x               | 0.58177       |
| x            | x   | ✓    | x               | 0.57165       |
| x            | x   | x    | ✓               | 0.59092       |
| ✓            | x   | x    | ✓               | 0.59158       |
| x            | ✓   | x    | ✓               | 0.59118       |
| x            | x   | ✓    | ✓               | 0.59114       |
| ✓            | ✓   | ✓    | ✓               | <b>0.5916</b> |

KNNImputer n\_neighbors

| n_neighbors | Private Score |
|-------------|---------------|
| 10          | <b>0.5909</b> |
| 30          | 0.59075       |
| 50          | 0.59073       |
| 70          | 0.59072       |

### SimpleImputer strategy

| strategy      | Private Score  |
|---------------|----------------|
| mean          | 0.59082        |
| median        | 0.59097        |
| most_frequent | <b>0.59103</b> |

### LogisticRegression parameters(max\_iter=100)

| penalty | C     | Private Score |
|---------|-------|---------------|
| l1      | 0.03  | 0.59148       |
| l1      | 0.01  | <b>0.5916</b> |
| l1      | 0.008 | 0.59138       |
| l2      | 0.01  | 0.59112       |

## Interesting findings or novel features engineering

第一次接觸到有很多缺失的資料，覺得最有趣的是尋找補值的方法，把最重要的 measurement\_17 特別抓出來處理（利用其他 measurement 和他的線性關係），效果就有顯著的提升，原本以為 KNNImputer 和 IterativeImputer 會有較好的結果，但實驗的結果卻是直接取眾數補值最好，蠻令人意外的。

一開始有點卡關，不論做什麼優化都無法提升，後來發現是因為沒有做 standardization 的關係，做完之後除了準確度提升，其他優化的效果也變得更加明顯（詳見 ablation study），因而了解到 standardization 的重要性。