109550031_Final

Environment Details

	version
python	3.7.10
numpy	1.21.5
pandas	1.3.5
sklearn	1.0.2
joblib	1.2.0

Introduction

在這次的作業中，因為總共有24個特徵，所以花了大部分的時間在做資料處理的部分，包括處理非數字的特徵、填補空格、減少特徵……等等。模型的部分我選擇使用了 `sklearn` 裡面的 `LinearRegression()` 作為我的模型來訓練。

Pre-processing

1. Mapping

在 product_code、attribute0、attribute1 這三項的型態是字串，所以我利用 mapping 的方式，將其轉為可供運算的數字。

以下是關於這部分的程式碼：

```
train_df['product_code'] = train_df['product_code'].map({'A':0, 'B':1, 'C':2, 'D':3, 'E':4,
'F':5, 'G':6, 'H':7, 'I':8})
train_df['attribute_0'] = train_df['attribute_0'].map({'material_5':0, 'material_7':1})
train_df['attribute_1'] = train_df['attribute_1'].map({'material_5':0, 'material_6':1, 'mat
erial_7':2, 'material_7':8})
```

2. Missing Values

一開始的資料集，有許多資料是空白的，為了避免在訓練模型時出現問題，必須將這些空格進行處理，我認為將有空格的資料整筆刪除有點不妥，所以我選擇利用 `sklearn.impute.SimpleImputer`，將有空缺的部分進行填補。

在 `sklearn.impute.SimpleImputer` 中，有四種方式可以填補空缺，分別是中位數、平均值、補零、眾數，我認為以眾數、中位數、平均值去填滿空格是較為合適的作法，因為眾數、中位數、平均值，比較不會是極端值，但使用補零的方式就可能因為太極端，進而對其他

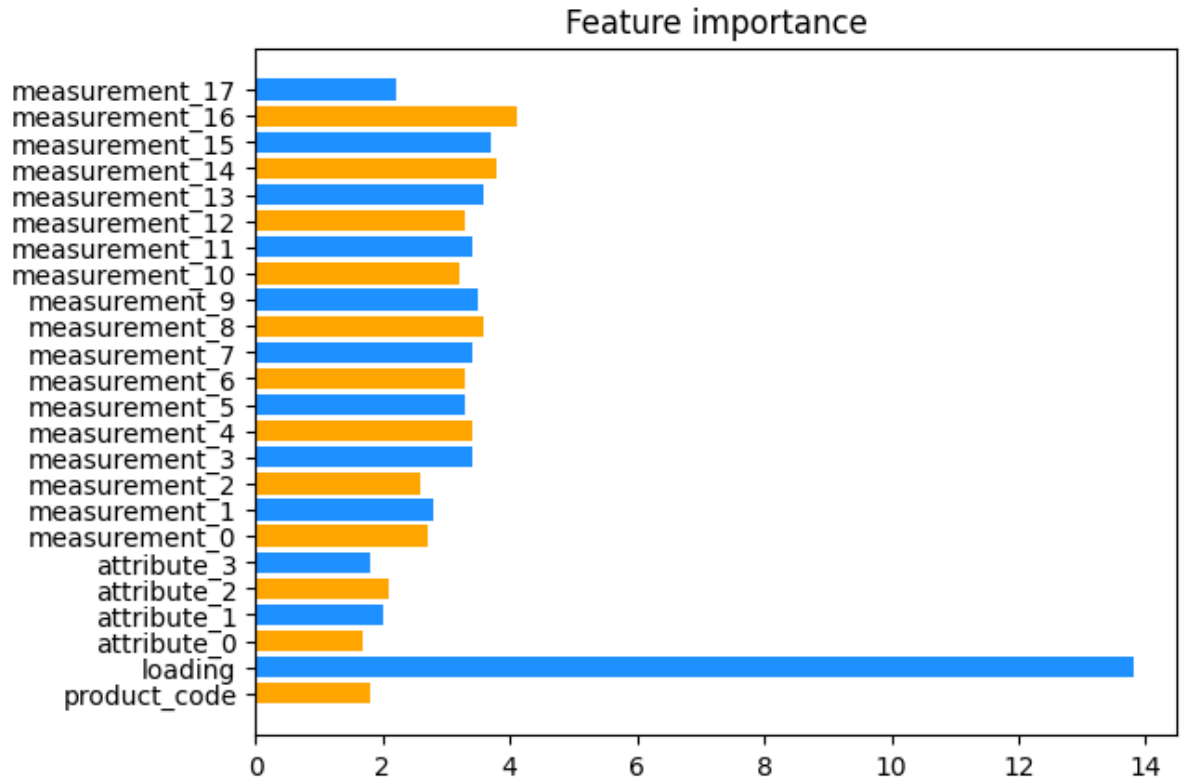
資料造成重大影響。其中，由於使用眾數的值是已經出現過，並且有一定的次數，所以對於其他資料而言，應該會是影響最小的，這也是我為何選擇眾數的原因。

以下是關於這部分的程式碼：

```
imputer = SimpleImputer(strategy='most_frequent')
train_df = pd.DataFrame(imputer.fit_transform(train_df), columns=train_df.columns, index=tr
ain_df.index)
test_df = pd.DataFrame(imputer.fit_transform(test_df), columns=test_df.columns, index=test_
df.index
```

3. Feature Importance

因為這次的資料集中有許多的特徵值，為了避免有些雜訊會讓整個模型太過 overfitting，所以我希望能夠透過 `sklearn.decomposition.PCA` 來減少特徵值的數目。觀察 feature importance 可以發現，loading 這項非常重要，遠遠的超過了其他特徵值，這也讓我在選擇 n_components 時，決定刪除大部分的特徵值。



以下是關於這部分的程式碼：

```
pca = PCA(n_components=2)
train_df = pca.fit_transform(train_df)
test_df = pca.transform(test_df)
```

Model Architecture

在這次的作業裡，我使用 `sklearn.LinearRegression` 來做為我的模型，經過前面預先替資料做處理後，這邊便可以透過直接呼叫此模型的函式，來完成訓練和預測的部分。

以下是關於這部分的程式碼：

```
model = LinearRegression(fit_intercept = True)
model.fit(X, target)
y_pred = model.predict(test_df)
```

Experimental Results

- SimpleImputer

SimpleImputer’s strategy	score on kaggle private leaderboard
mean	0.59009
median	0.59024
constant	0.58895
most_frequent	0.5903

根據上方圖表可以發現，以眾數、中位數、平均值去填滿空格的分數較高，所以這三個應該是較為合適的作法，因為眾數、中位數、平均值比較不會是極端值，但使用補零的方式就可能因為太極端，進而對其他資料造成影響。其中，眾數又是三者中分數最高的，我認為是因為其值出現了一定的次數，所以對於其他資料而言，應該是影響最小的。

- PCA

n_components	score on kaggle private leaderboard
1	0.51942
2	0.5903
4	0.59004
8	0.58944
16	0.58879
24	0.57706

根據上方圖表可以發現，從一開始的 24 個特徵減少至 2 個特徵的分數都是有在提升的，我認為這確實是因為沒有過於 overfitting 的結果，減少至 2 個特徵的分數是最高的，可以說明有兩個最重要的特徵，當 n_components = 2，這時沒有任何的雜訊，所以分數較高。如果再減少至 1 個特徵，分數會突然驟降，這也說明了兩個特徵都是很重要的特徵。

Result

- on kaggle private leaderboard：準確率可達 0.5903

Summary

整體來說，我認為這次最大的重點並不是在模型，而是資料的前處理，如何找出最有關聯的特徵，決定哪些特徵要被選擇。模型的部分，或許有比 Linear Regression 更適合的，之後可以再試試看其他的模型，除了 sklearn 之外，可能也可以試著使用 NN。

📢 Github: https://github.com/lon5948/Tabular-Playground-Series---Aug-2022

📢 Model: https://drive.google.com/drive/folders/1Jl5dlfCTMIZ8CdsakW3jjyGRHT-fFGuZ?usp=sharing