

HW2 Report

R10946013 劉馨瑄

● 步驟說明：

1. 讀取資料表

讀取資料表

```
df = pd.read_csv("GDSC_PDX_Paclitaxel.csv"), index_col=0)
df = df.rename(columns={'Unnamed: 0': 'CELL_LINE_NAME'}) # 把cell_line_name欄位名稱統一 (方便做合併)
df_info = pd.read_csv("GDSC_PDX_Paclitaxel_info.csv", index_col=0)
df_test = pd.read_csv("CCLE_PDX_Paclitaxel.csv"), index_col=0)
df_test = df_test.rename(columns={'CCLE.Cell.Line.Name': 'CELL_LINE_NAME'})
```

[2] ✓ 23.7s

2. 取欄位交集 (兩資料表都有的欄位 features)

取欄位交集 (兩個資料表都有的欄位)

```
train_features = set(df.columns)
test_features = set(df_test.columns)
# 把非交集的欄位從資料表中去除 (減去差集)
(variable) df_test: DataFrame es.difference(test_features), axis=1)
df_test = df_test.drop(test_features.difference(train_features), axis=1)
```

[3] ✓ 0.2s

3. 計算 IC50 與 MAX_CONC_MICROMOLAR 欄位的關係

用IC50與MAX_CONC_MICROMOLAR欄位計算S或R 並將結果併入訓練資料表中

```
ans = []
for i in range(len(df_info["IC50"])):
    if list(df_info["IC50"])[i] < list(df_info["MAX_CONC_MICROMOLAR"])[i]:
        ans.append(0) # S
    else:
        ans.append(1) # R
df_info['ans'] = ans
df_bind = df_info[['CELL_LINE_NAME', 'ans']] ## 合併用的
df_train = pd.merge(df_train, df_bind) ## 訓練用的
print(df_train.shape) # 處理過後有的行列數量
```

[4] ✓ 0.2s

... (399, 15175)

4. 資料標準化

資料標準化

```
X = df_train.drop("ans", 1) #Feature Matrix
y = df_train["ans"] #目標變量
df_train = df_train.drop(['CELL_LINE_NAME'], axis=1)

df_ = df_train.drop("ans", axis=1)

normalized_df=(df_ - df_.mean())/df_.std()
normalized_df['ans'] = y # 把標準化前刪掉的ans欄位併回來
```

[5] ✓ 13.2s

+ 程式碼 + Markdown

5. 計算 correlation 以便做 Feature Selection (使用 Pearson 方法) :

計算各欄位與欲預測欄位的correlation (pearson方法)

```
df_train = normalized_df
cor_df = df_train.corr(method='pearson') # 使用皮爾森方法
cor_df
```

[6] ✓ 3m 33.7s Python

	TSPAN6	TNMD	DPM1	SCYL3	C1orf112	FGR	CFH	FUCA2	GCLC	NFYA	...	OR4N3P	MRPL46	...
TSPAN6	1.000000	0.046208	0.053007	-0.365482	-0.233656	-0.465248	0.149307	0.227825	0.120466	-0.223184	...	-0.215747	-0.130318	-0.1571
TNMD	0.046208	1.000000	0.040708	0.070092	0.079292	0.044540	-0.108413	-0.103665	-0.030382	0.034691	...	0.058246	0.043290	-0.0031
DPM1	0.053007	0.040708	1.000000	-0.050125	-0.099479	0.032664	0.047902	0.070631	0.015885	-0.120462	...	-0.034802	0.075925	0.016
SCYL3	-0.365482	0.070092	-0.050125	1.000000	0.207594	0.364232	-0.095711	-0.160789	0.029839	0.241952	...	0.099734	0.203343	0.098
C1orf112	-0.233656	0.079292	-0.099479	0.207594	1.000000	0.039565	-0.144687	-0.298227	-0.034905	0.351181	...	0.045729	0.169987	0.014
...
OR1D5	-0.256798	-0.019272	0.058854	0.092487	0.089290	0.177333	0.004501	-0.066179	-0.017884	0.077607	...	0.084624	0.124597	0.104
MYH4	0.067981	0.080862	0.023932	0.061193	-0.068675	0.006781	-0.036972	0.095648	0.118941	-0.021334	...	0.019490	0.019301	0.015
KRT18P55	0.009864	0.063532	0.083644	0.088698	0.058666	0.072268	-0.058323	0.018816	0.119819	-0.109738	...	-0.010065	0.107776	0.102
TBC1D3P5	-0.170854	0.058706	-0.086360	0.103085	0.145893	0.076177	-0.114808	-0.111958	-0.070690	0.042536	...	0.056023	0.088320	0.245
ans	0.068623	-0.007491	-0.033391	-0.053902	-0.119629	0.006937	-0.056578	-0.028594	0.020921	-0.060083	...	0.003051	0.058840	0.092

15174 rows x 15174 columns

6. 設定閾值選取 features

(經測試後使用 cor 絕對值大於 0.17 的欄位組合模型表現最佳)

設定一個correlation閾值來選取要用來訓練的feature(取絕對值後大於閾值)

```
#Correlation with output variable
cor_target = abs(cor_df["ans"])

#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.17] #0.17 #設定閾值
print("用來訓練模型的feature數量: ", len(relevant_features))
```

[19] ✓ 0.4s

... 用來訓練模型的feature數量: 108

7. 選定 Feature 後整理為建模資料

將feature選定後整理為建模用資料

```
df_train_new = df_train[list(relevant_features.index)]
X = df_train_new.drop("ans",1) #Feature Matrix
y = df_train_new["ans"] #目標變量
```

[20] ✓ 0.5s

8. 調整參數與建模

我一共使用了三種模型進行測試，分別為：RandomForest Classifier、XGBoost Classifier、KNN Classifier。

針對模型我也進行了調整參數，調整模型最主要、影響通常最大的參數："n_neighbors"，並觀察準確度的變化，以選取最佳的參數值。

此外，我也使用了 K-fold cross validation 來進行觀察，並取五次的平均作為 Accuracy 的參考值。

結果如下：

a. RandomForest Classifier：

✓ Cross Validation 的結果(n_neighbors= range(10,290,20))

[0.7125	0.7125	0.725	0.8	0.72151899]
[0.75	0.7625	0.8	0.75	0.7721519]
[0.725	0.775	0.775	0.8125	0.75949367]
[0.7	0.725	0.825	0.7875	0.79746835]
[0.7375	0.7625	0.8125	0.7625	0.81012658]
[0.725	0.75	0.825	0.7625	0.75949367]
[0.725	0.775	0.8	0.8	0.78481013]
[0.7625	0.725	0.8125	0.8	0.79746835]
[0.7375	0.75	0.8	0.775	0.81012658]
[0.7625	0.7625	0.825	0.775	0.79746835]
[0.7375	0.7375	0.8	0.7875	0.79746835]
[0.725	0.7625	0.825	0.7625	0.78481013]
[0.7375	0.75	0.8125	0.8	0.81012658]
[0.7625	0.7625	0.8125	0.8	0.81012658]

✓ 取表現最佳的參數組合：

Accuracy = 0.7895253164556962 (N_neighbors = 270)

b. XGBoost Classifier：

✓ Cross Validation 的結果(n_neighbors= range(10,290,20))

[0.675	0.775	0.7875	0.725	0.81012658]
[0.6375	0.775	0.8	0.7375	0.75949367]
[0.65	0.775	0.7875	0.75	0.75949367]
[0.6625	0.8	0.8	0.7625	0.7721519]
[0.675	0.7875	0.8	0.75	0.7721519]
[0.675	0.7875	0.825	0.75	0.7721519]
[0.6875	0.7875	0.825	0.7375	0.78481013]
[0.7	0.8	0.8125	0.7375	0.79746835]
[0.725	0.7875	0.8125	0.7375	0.78481013]
[0.725	0.7875	0.8	0.7375	0.78481013]
[0.725	0.7875	0.775	0.7375	0.78481013]
[0.725	0.7875	0.775	0.7375	0.78481013]
[0.7125	0.7875	0.775	0.75	0.79746835]
[0.725	0.7875	0.775	0.75	0.78481013]

✓ 取表現最佳的參數組合：

Accuracy = 0.7694936708860759 (N_neighbors = 150)

c. KNN Classifier :

- ✓ Cross Validation 的結果($n_neighbors = \text{range}(10, 290, 20)$)

[0.7125	0.725	0.725	0.75	0.75949367]
[0.725	0.725	0.7125	0.7125	0.70886076]
[0.7	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]
[0.7125	0.7125	0.7125	0.7125	0.70886076]

- ✓ 取表現最佳的參數組合：

Accuracy = 0.7343987341772152 (N_neighbors = 10)

● 結論與說明：

觀察上述三者模型，我們可以發現 Random Forest 在參數為 $n_neighbors = 290$ 時的表現為三者中最佳。且在 Validation 時，我們也發現模型並未有 Overfitting 的狀態（如下圖）。因此我們選用此組合作為預測 CCLE 資料時使用的模型。

最好的模型與參數組合

```
forest = ensemble.RandomForestClassifier(n_estimators = 270) #參數設為最好的
forest_fit = forest.fit(train_X, train_y)
# 預測
test_y_predicted = forest.predict(test_X)
# 績效
scores = cross_val_score(forest, X, y, cv=5, scoring='accuracy')
print(scores)
sum = 0
for i in scores:
    sum+=i
print(sum/5)
```

[51] ✓ 5.9s

... [0.725 0.7625 0.8125 0.775 0.7721519]

0.7694303797468354

```
> accuracy = metrics.accuracy_score(test_y, test_y_predicted)
print(accuracy)
```

[52] ✓ 0.1s

... 0.75

↑ Training accuracy 為 0.7694，Testing accuracy 為 0.75 → 未 overfitting

而針對欲預測的 CCLE 資料表，我們也做相同的處理步驟，需選取訓練模型時使用的特定 Feature 組合，並進行資料的標準化處理等，最後丟入模型預測，並輸出結果為 ans.csv 資料檔。