Python Machine Learning

Calories Burnt Prediction

IMLP 343 楊閎富

Kaggle

這次的資料是來自Kaggle 的 **Calories Burnt Prediction**

https://www.kaggle.com/datasets/ruchikakumbhar/calories-burnt-prediction/data

以下是我的介紹影片連結
https://youtu.be/_KpLDnzYWzs
https://youtu.be/axfXjXti8iY

Calories Burnt Prediction

Features:

- User_Id
- Gender
- Age
- Height
- Weight
- Duration
- Heart_rate
- Body_temp

使用 Python 開發機器學習模型，該模型可以根據一些生物指標預測一個人在運動期間燃燒的卡路里數。

Target:

- Calories

為甚麼選這份資料

我選擇這份資料是因為自己也有在運動，對於熱量控制以及計算運動燃燒卡路里都有研究，就把這份資料拿來當作這次的機器學習目標。

也希望透過這次的專題讓自己更了解年紀、身高、體重、運動時間、體溫以及心律對於卡路里燃燒有什麼關係。

資料集說明

Calories總共有15000筆資料9個欄位

	User_ID	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
0	14733363	male	68	190.0	94.0	29.0	105.0	40.8	231.0
1	14861698	female	20	166.0	60.0	14.0	94.0	40.3	66.0
2	11179863	male	69	179.0	79.0	5.0	88.0	38.7	26.0
3	16180408	female	34	179.0	71.0	13.0	100.0	40.5	71.0
4	17771927	female	27	154.0	58.0	10.0	81.0	39.8	35.0
...	...	...	...	...	...	...	...	...	...
14995	15644082	female	20	193.0	86.0	11.0	92.0	40.4	45.0
14996	17212577	female	27	165.0	65.0	6.0	85.0	39.2	23.0
14997	17271188	female	43	159.0	58.0	16.0	90.0	40.1	75.0
14998	18643037	male	78	193.0	97.0	2.0	84.0	38.3	11.0
14999	11751526	male	63	173.0	79.0	18.0	92.0	40.5	98.0

15000 rows × 9 columns

資料前處理

- 我將資料的User_ID刪除並加上兩個target分別是Heavy_level及Burn_level
- Heavy_level是透過身高及體重計算出BMI，在利用BMI的大小分類，BMI小於24Heavy_level=0， BMI介於24到27Heavy_level=1， BMI大於27Heavy_level=2
- Burn_level將Calories大於一百的值標示為1 小於的為0
- 將性別轉成數字男性為1女性為0

	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories	Heavy_level	Burn_level
0	1	68	190.0	94.0	29.0	105.0	40.8	231.0	1	1
1	0	20	166.0	60.0	14.0	94.0	40.3	66.0	0	0
2	1	69	179.0	79.0	5.0	88.0	38.7	26.0	1	0
3	0	34	179.0	71.0	13.0	100.0	40.5	71.0	0	0
4	0	27	154.0	58.0	10.0	81.0	39.8	35.0	1	0
...	...	...	...	...	...	...	...	...	...	...
14995	0	20	193.0	86.0	11.0	92.0	40.4	45.0	0	0
14996	0	27	165.0	65.0	6.0	85.0	39.2	23.0	0	0
14997	0	43	159.0	58.0	16.0	90.0	40.1	75.0	0	0
14998	1	78	193.0	97.0	2.0	84.0	38.3	11.0	1	0
14999	1	63	173.0	79.0	18.0	92.0	40.5	98.0	1	0

15000 rows × 10 columns

將資料做Min-Max Normaliaztion

```
def normalization(d):
    df2[d] = (df2[d] - df2[d].min())/(df2[d].max() - df2[d].min())
```

```
for i in df2.columns[1:-3]:
    normalization(i)
df2
```

	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories	Heavy_level	Burn_level
0	1	0.813559	0.676768	0.604167	0.965517	0.622951	0.840909	231.0	1	1
1	0	0.000000	0.434343	0.250000	0.448276	0.442623	0.727273	66.0	0	0
2	1	0.830508	0.565657	0.447917	0.137931	0.344262	0.363636	26.0	1	0
3	0	0.237288	0.565657	0.364583	0.413793	0.540984	0.772727	71.0	0	0
4	0	0.118644	0.313131	0.229167	0.310345	0.229508	0.613636	35.0	1	0
...	...	...	...	...	...	...	...	...	...	...
14995	0	0.000000	0.707071	0.520833	0.344828	0.409836	0.750000	45.0	0	0
14996	0	0.118644	0.424242	0.302083	0.172414	0.295082	0.477273	23.0	0	0
14997	0	0.389831	0.363636	0.229167	0.517241	0.377049	0.681818	75.0	0	0
14998	1	0.983051	0.707071	0.635417	0.034483	0.278689	0.272727	11.0	1	0
14999	1	0.728814	0.505051	0.447917	0.586207	0.409836	0.772727	98.0	1	0

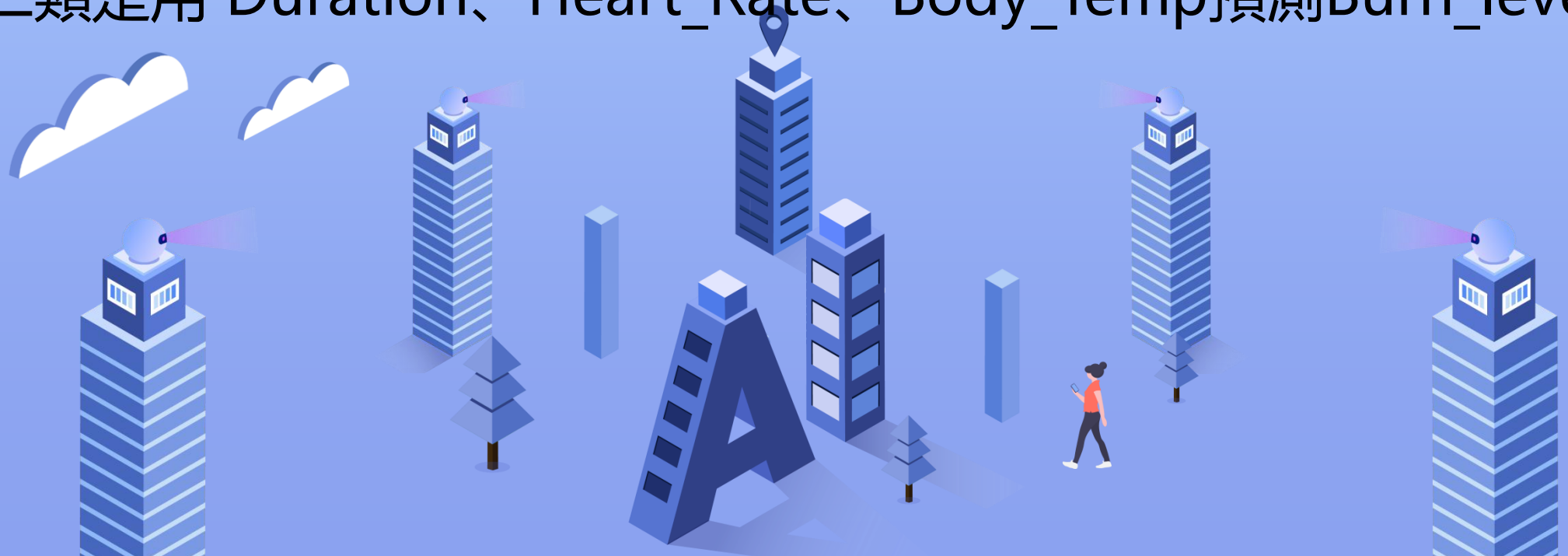
15000 rows × 10 columns

訓練方式

在這次的資料中我分成兩類

第一類是用 Height、Weight預測Heavy_level

第二類是用 Duration、Heart_Rate、Body_Temp預測Burn_level



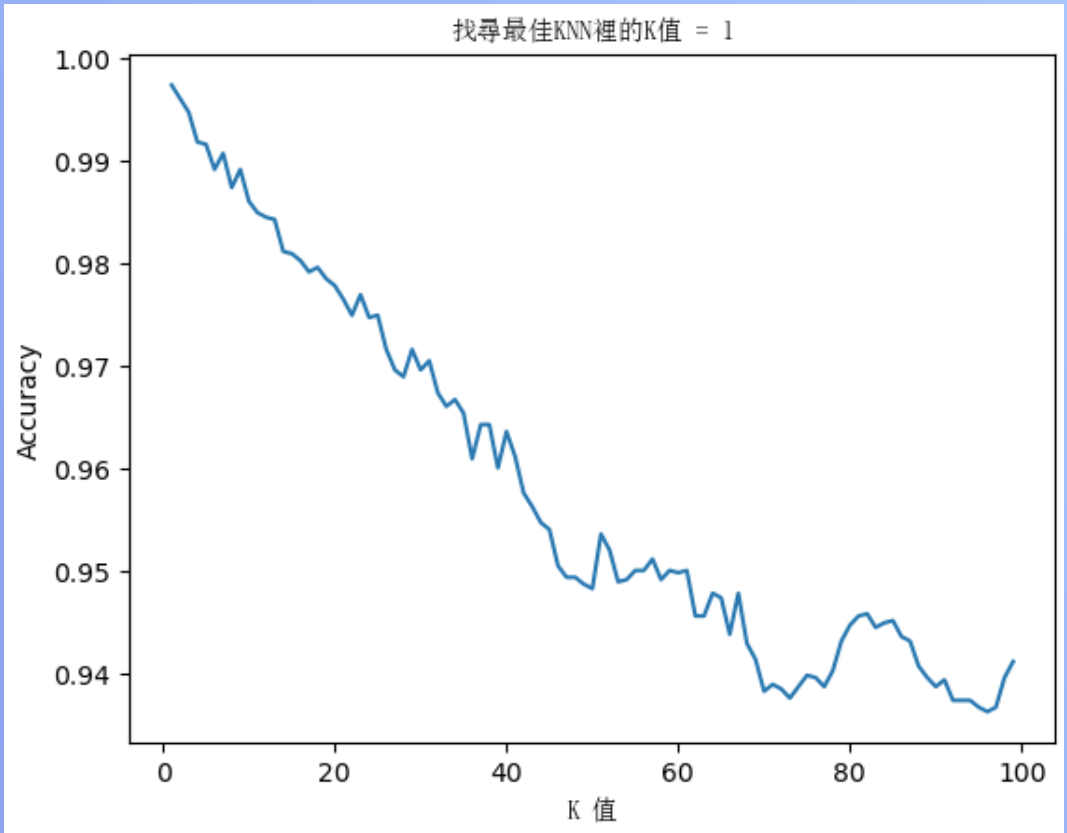
用 Height、Weight預測Heavy_level

- 在這類型的預測中KNN的表現是最好的
- 我使用原始資料中的 Height、Weight 用來預測 Heavy_level
- 以下是測試集錯誤報告

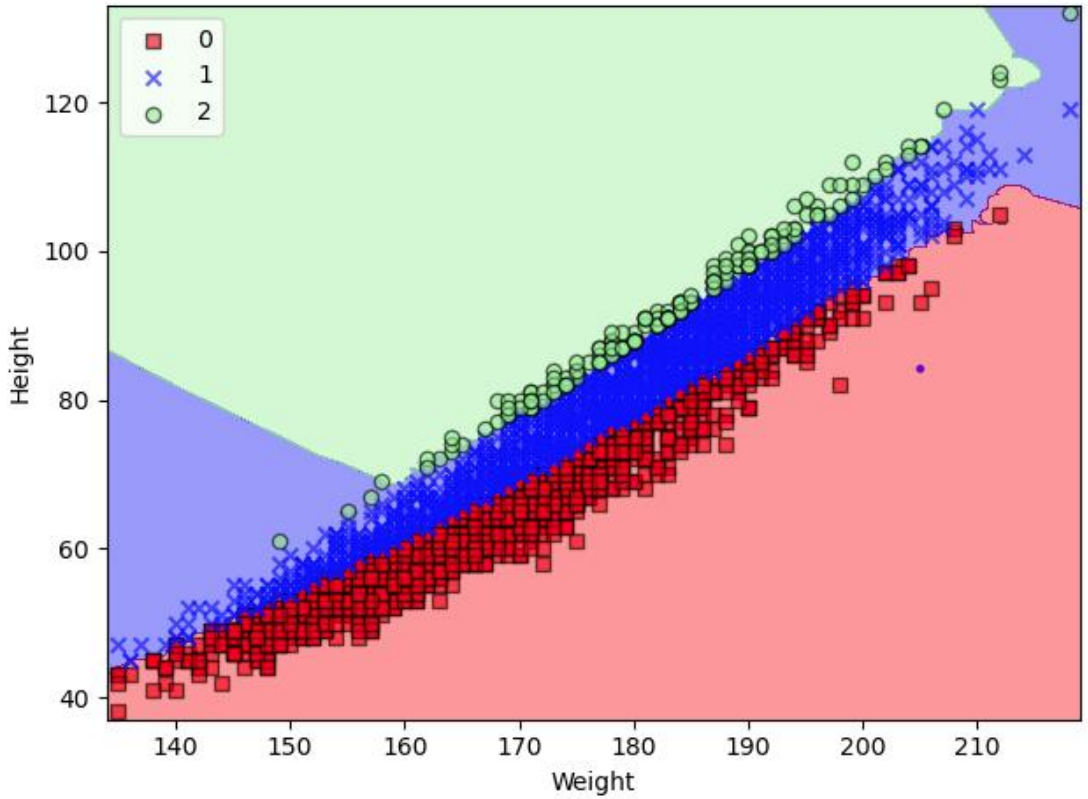
```
accuracy: 0.998
number of errors :9/4500
--------------------------------------------------------------------------------
error_index : [62, 247, 445, 1166, 1760, 2252, 3571, 3599, 3969]
miss classification : [0 1 1 1 0 1 1 1 0]
error data index: Index([8035, 14132, 3653, 9728, 13800, 2098, 5253, 7171, 14875], dtype='int64')
```

最佳的K值

我做了一個迴圈來測試出最好的K值，
並且發現K等於1的時候效果最好



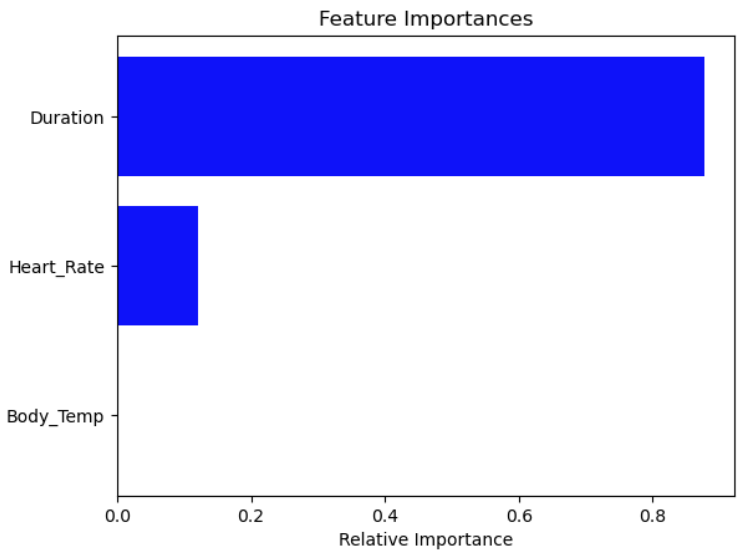
圖像化KNN



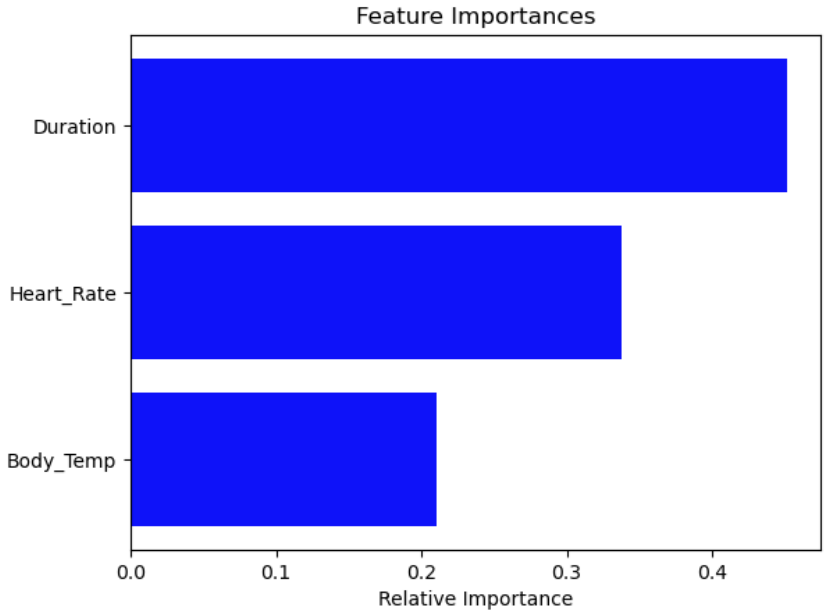
用 Duration、Heart_Rate、BodyTemp預測Burnlevel

在使用Decision Tree、RandomForest以及AdaBoost訓練模型後，我發現BodyTemp的重要性都是很低的，主要影響Burnlevel的是Duration、Heart_Rate

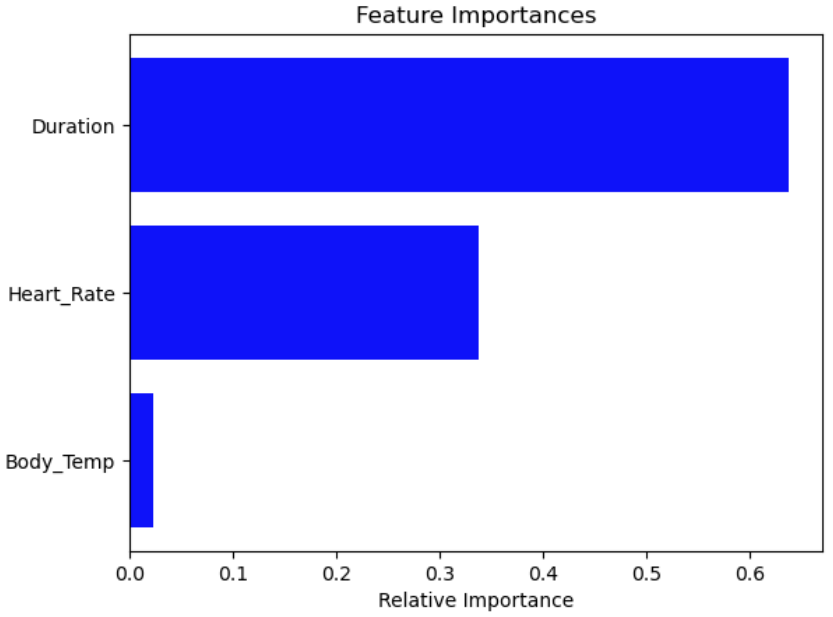
Decision Tree



RandomForest



AdaBoost



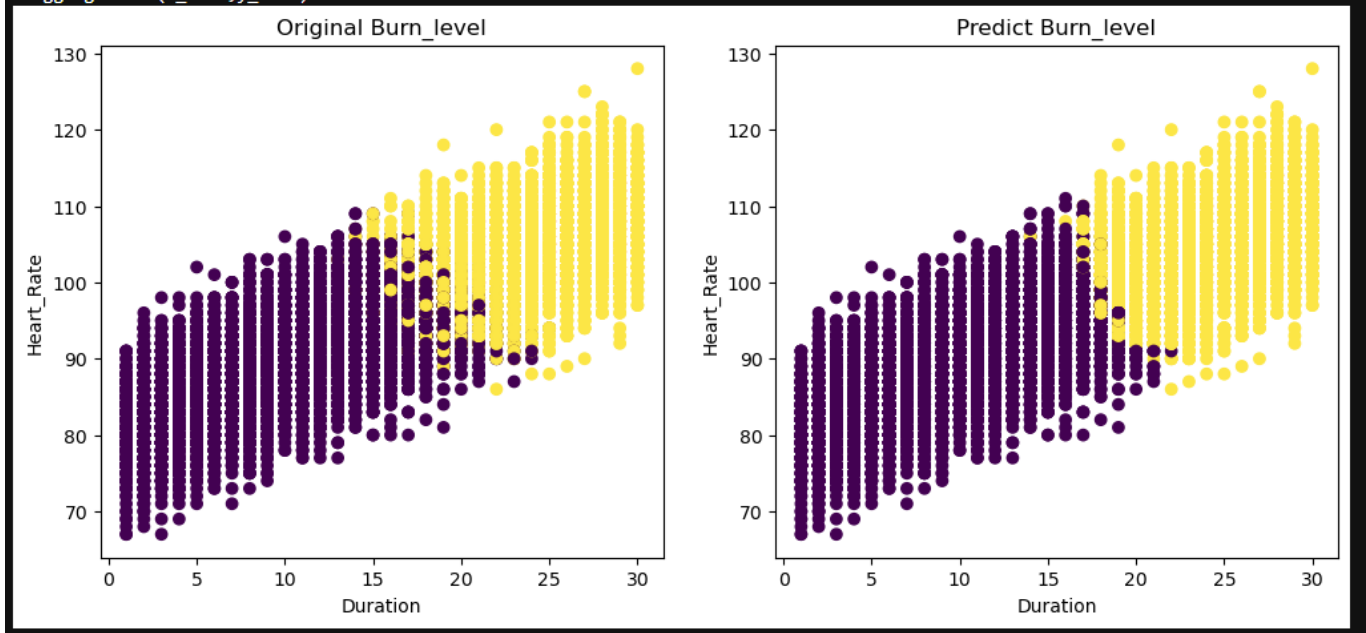
用 Duration、Heart_Rate、BodyTemp 預測Burnlevel

在這類的預測中，分數最高的是Bagging搭配DecisionTree做出來的結果

使用DecisionTree

```
clf=tree.DecisionTreeClassifier()
bagging_compare(clf)
```

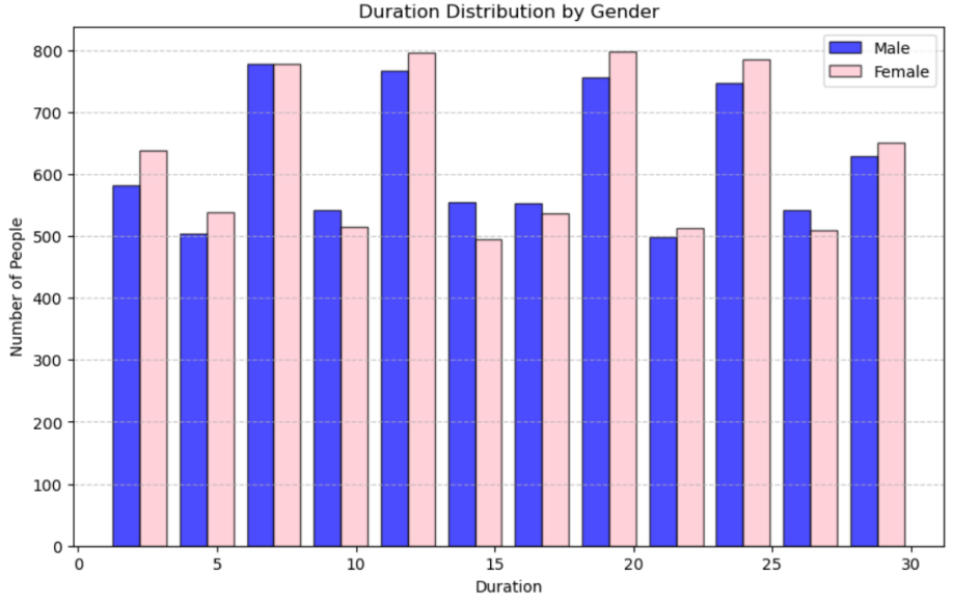
```
bagging.predict(X_test) =  [1 1 0 ... 0 1 1]
--------------------------------------------------------------------------------------------------------
bagging.score(X_train,y_train) =  0.9617142857142857
--------------------------------------------------------------------------------------------------------
bagging.score(X_test,y_test) =  0.9551111111111111
```

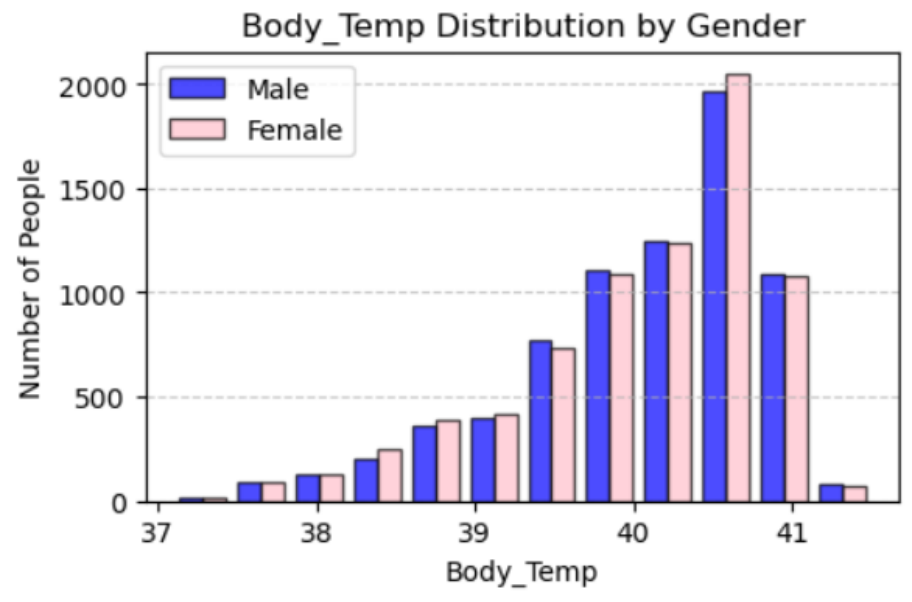


Duration介於15到20間錯誤率比較高

遇到的問題

1. 在做Logistic Regression的時候，資料經過Standardization後，train score跟test score比沒有經過標準化的時候來的低
2. 我認為這是因為我找的這筆卡路里的資料的某些資料分佈非常不對稱，像是Duration(運動持續時間)跟BodyTemp(體溫)所造成的，還有部分特徵對分類結果的影響較大，標準化可能會削弱這些特徵的影響力，導致 error 數量增加





心得

在這次的專題中，讓我對整個SciKit-Learn機器學習的流程都更加的熟悉，從下載資料、使用numpy、pandas分析資料、整理資料、區分訓練集、測試集、到建模、預測、準確程度評估、調整參數，還有用matplotlib、seaborn來視覺化都操做了好多遍。