

## I. About this homework:

這份報告共分為五部分。第一部分，我們首先將 15000 筆的資料分為 training, validation, testing data；接下來第二跟三部分，分別使用 Maximum Likelihood Estimation (MLR) 跟 Bayesian Linear Regression (BLR) 進行 prediction；第四部分則是比較這兩種回歸模型(MLR, BLR)的差異；最後，第五部分，我們使用 sklearn 的內建模型 Gradient Boosting Regressor 來跑本次作業，並做討論。

## II. Data splitting:

### 1. Detailed implementation :

我們先將 exercise.csv 跟 calroies.csv 合併成 merged.csv，因為 User\_ID 不重要，所以我們直接將此 column 刪除；另外，第一個 feature，Gender 的部分，為求方便，我們將 Male 更改為"1"，Female 更改為"0"。最後以 training data, validation data, testing data 等於 70:10:20 的比例，將 15000 筆 data，分別分割成 10500, 1500, 3000 筆。

1	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
2	1	68	190	94	29	105	40.8	231
3	0	20	166	60	14	94	40.3	66
4	1	69	179	79	5	88	38.7	26
5	0	34	179	71	13	100	40.5	71
6	0	27	154	58	10	81	39.8	35

(merged.csv)

```

(m1) stave@sd233093 coding % python hw3.py
Gender Age Height Weight Duration Heart_Rate Body_Temp Calories
0 1 68 190.0 94.0 29.0 105.0 40.8 231.0
1 0 20 166.0 60.0 14.0 94.0 40.3 66.0
2 1 69 179.0 79.0 5.0 88.0 38.7 26.0
3 0 34 179.0 71.0 13.0 100.0 40.5 71.0
4 0 27 154.0 58.0 10.0 81.0 39.8 35.0
...
10495 1 43 185.0 91.0 16.0 97.0 40.5 88.0
10496 1 31 203.0 110.0 7.0 84.0 39.4 24.0
10497 1 36 186.0 82.0 24.0 106.0 40.5 152.0
10498 0 55 158.0 60.0 4.0 94.0 38.4 21.0
10499 1 25 170.0 74.0 4.0 90.0 39.2 13.0
[10500 rows x 8 columns]
Gender Age Height Weight Duration Heart_Rate Body_Temp Calories
10500 0 28 156.0 60.0 25.0 102.0 40.9 142.0
10501 1 21 198.0 94.0 24.0 104.0 40.8 133.0
10502 0 44 158.0 57.0 28.0 111.0 41.0 196.0
10503 0 23 155.0 57.0 5.0 91.0 39.0 22.0
10504 1 32 199.0 94.0 3.0 84.0 38.5 9.0
...
11995 0 66 156.0 52.0 21.0 97.0 40.8 125.0
11996 0 28 178.0 75.0 7.0 88.0 39.6 28.0
11997 0 20 163.0 53.0 20.0 102.0 40.8 158.0
11998 1 28 180.0 84.0 25.0 110.0 40.6 165.0
11999 0 40 155.0 51.0 8.0 84.0 39.5 33.0
[1500 rows x 8 columns]
Gender Age Height Weight Duration Heart_Rate Body_Temp Calories
12000 0 55 176.0 70.0 11.0 90.0 40.2 52.0
12001 0 28 178.0 73.0 5.0 84.0 39.1 18.0
12002 1 22 182.0 84.0 4.0 91.0 39.3 14.0
12003 1 32 194.0 94.0 13.0 97.0 39.9 65.0
12004 1 24 185.0 88.0 11.0 90.0 39.9 38.0
...
14995 0 20 193.0 86.0 11.0 92.0 40.4 45.0
14996 0 27 165.0 65.0 6.0 85.0 39.2 23.0
14997 0 43 159.0 58.0 16.0 90.0 40.1 75.0
14998 1 78 193.0 97.0 2.0 84.0 38.3 11.0
14999 1 63 173.0 79.0 18.0 92.0 40.5 98.0
[3000 rows x 8 columns]
  
```

(training, validation and testing data)

## 2. Code :

這部分 code 在 python 的 main() 的第一部分，基本上只要執行一遍即可。

```

65     '''Q1: Data preprocessing (Only do once)'''
66     # To merge the csv file :
67     calories_df = pd.read_csv('calories.csv')
68     exercise_df = pd.read_csv('exercise.csv')
69     exercise_df = exercise_df.drop(calories_df.columns[0], axis=1)
70     calories_df = calories_df.drop(calories_df.columns[0], axis=1)
71     merged_df = pd.concat([exercise_df, calories_df], axis=1)
72     merged_df['Gender'] = merged_df['Gender'].replace({'male': 1, 'female': 0})
73     # print(merged_df)
74     merged_df.to_csv('merged_data.csv', index=False)
75
76     total_samples = len(merged_df)
77     train_size = 10500
78     val_size = 1500
79     test_size = 3000
80
81     train_data = merged_df[:train_size]
82     val_data = merged_df[train_size:train_size + val_size]
83     test_data = merged_df[train_size + val_size:]
84     # print(train_data)
85     # print(val_data)
86     # print(test_data)
87
88     # Save the split datasets as CSV files
89     train_data.to_csv('train_data.csv', index=False)
90     val_data.to_csv('val_data.csv', index=False)
91     test_data.to_csv('test_data.csv', index=False)

```

(第一部分 code)

### III. Maximum Likelihood Estimation (MLR):

#### 1. Detailed implementation:

這部分細節主要參考課本 3.1.1 節，首先將 training data 的 features 跟 label 分開，將 features 寫成 feature vector ( $\Phi$ ) 進行處理，接下來將 training data 的 label 帶入  $t$ ，即可進行 model's weight 的計算，如課本 3.15 公式：

$$\mathbf{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$$

另外 prediction 的部分，需先將 testing data 的 features 分出來，之後直接乘上上面所計算的 weight 即可以得到 prediction。

## 2. Code:

Code 的部份原則上跟上述數學式一樣，但我在 training features 跟 testing features 都先加入了一行數值為 1 的 column，當作 intercept term (或 bias term)這是在 linear regression 常見的做法，主要是在估計時可以更準確訓練出 intercept 值，這樣對於 features 跟目標變數的關係會更準確，也可以得到更好的預測值。另外，根據題目要求，MLR 是要針對 testing data 做測試，而下方 BLR 則是對 validation data 做測試，

```

13 def MLR(train_data, test_data_feature):
14     train_data_feature = train_data[:, :7]
15     train_data_label = train_data[:, 7]
16
17     train_data_feature = np.c_[np.ones(train_data_feature.shape[0]), train_data_feature]
18     coefficients = np.linalg.inv(train_data_feature.T.dot(train_data_feature)).dot(train_data_feature.T.dot(train_data_label))
19
20     test_data_feature = np.c_[np.ones(test_data_feature.shape[0]), test_data_feature]
21     MLR_predictions = test_data_feature.dot(coefficients)
22
23     return MLR_predictions

```

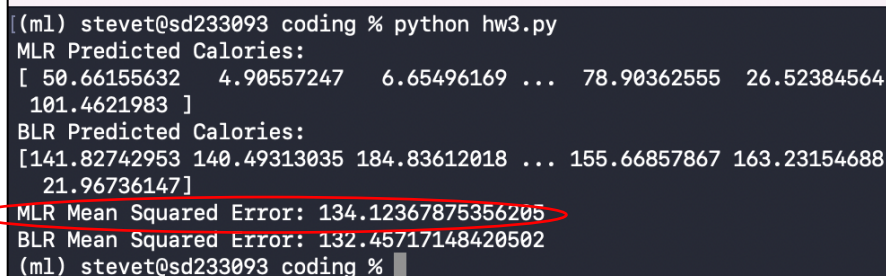
## 3. Mean squared error (MSE) function:

```

55 def MSE(data, prediction):
56     mse = np.mean((prediction - data) ** 2)
57     return mse

```

## 4. The result:



```

coding — -zsh — 98x22
(ml) stevet@sd233093 coding % python hw3.py
MLR Predicted Calories:
[ 50.66155632  4.90557247  6.65496169 ... 78.90362555 26.52384564
 101.4621983 ]
BLR Predicted Calories:
[141.82742953 140.49313035 184.83612018 ... 155.66857867 163.23154688
 21.96736147]
MLR Mean Squared Error: 134.12367875356205
BLR Mean Squared Error: 132.45717148420502
(ml) stevet@sd233093 coding %

```

#### IV. Bayesian Linear Regression

##### 1. Detailed implementation:

參考課本 3.3 節，Bayesian 的 feature vector 用法跟上方 Maximum Likelihood 很像，如果參考課本的(3.50), (3.51)，用  $S_0^{-1} = \alpha I$  帶入來計算 weight，會發現最後 weight 跟上面 MLR 的 weight 只差加入一個  $\lambda I$ ，其中  $\lambda = \frac{\alpha}{\beta}$ 。

$$\begin{aligned}\mathbf{m}_N &= \mathbf{S}_N (\mathbf{S}_0^{-1} \mathbf{m}_0 + \beta \Phi^T \mathbf{t}) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta \Phi^T \Phi.\end{aligned}$$

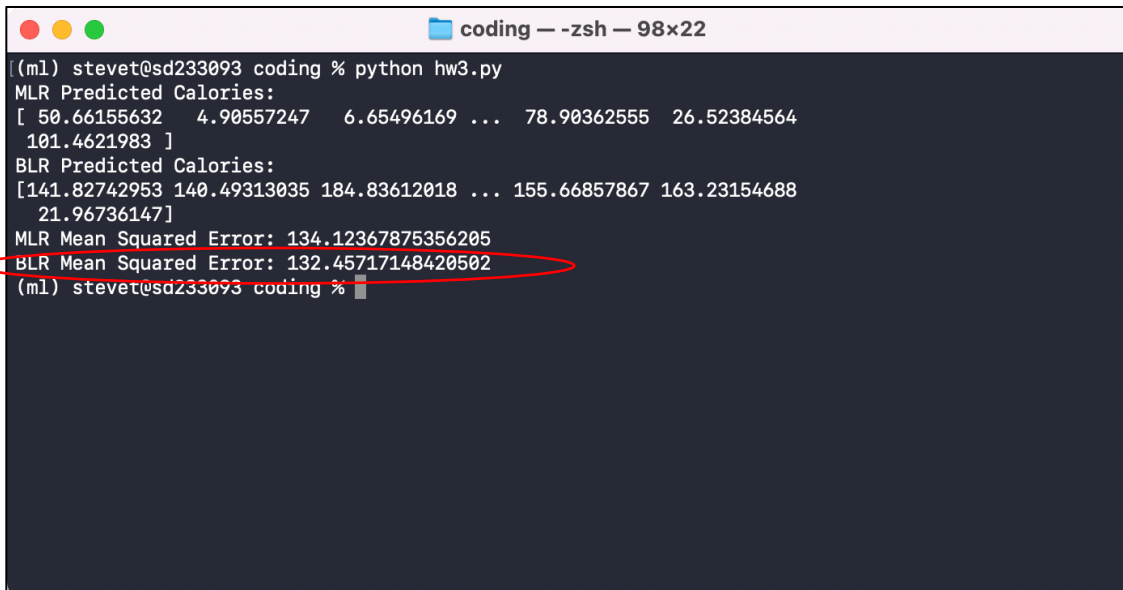
Weight:

$$\mathbf{w} = (\lambda \mathbf{I} + \Phi^T \Phi)^{-1} \Phi^T \mathbf{t}.$$

##### 2. Code:

```
26 def BLR(train_data, test_data_feature):
27     train_X = train_data[:, :-1]
28     train_y = train_data[:, -1]
29
30     train_X = np.c_[np.ones(train_X.shape[0]), train_X]
31     X_transpose_X = train_X.T.dot(train_X)
32     X_transpose_y = train_X.T.dot(train_y)
33     posterior_precision = np.linalg.inv(X_transpose_X + np.eye(X_transpose_X.shape[0]))
34     posterior_mean = posterior_precision.dot(X_transpose_y)
35
36     test_X = np.c_[np.ones(test_data_feature.shape[0]), test_data_feature]
37     BLR_predictions = test_X.dot(posterior_mean)
38
39     return BLR_predictions
```

##### 3. The result:

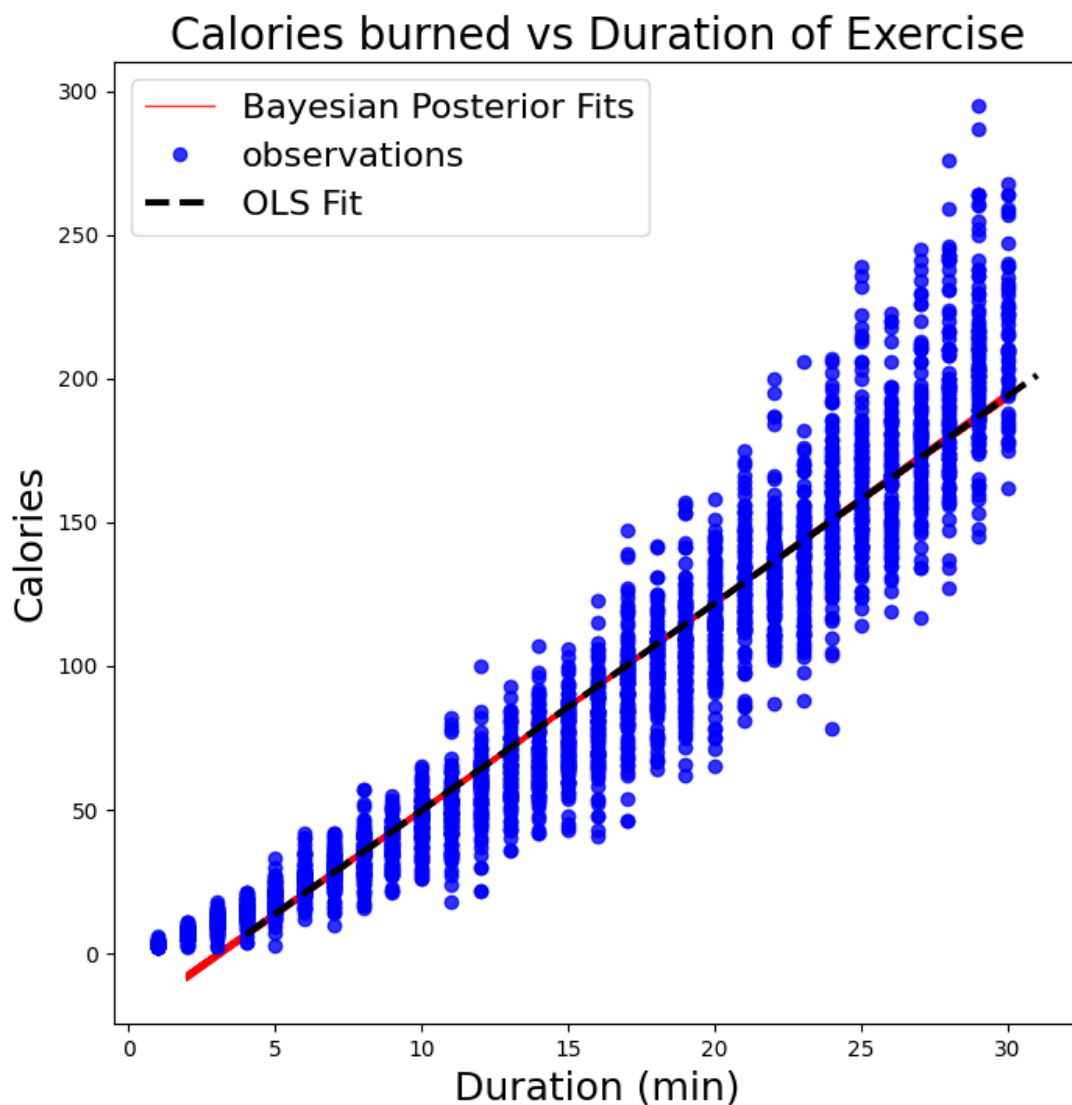


The terminal window shows the execution of a Python script. The output displays predicted calories for both MLR and BLR models. The BLR Mean Squared Error is highlighted with a red circle.

```
(m1) stevet@sd233093 coding % python hw3.py
MLR Predicted Calories:
[ 50.66155632  4.90557247  6.65496169 ... 78.90362555 26.52384564
 101.4621983 ]
BLR Predicted Calories:
[141.82742953 140.49313035 184.83612018 ... 155.66857867 163.23154688
 21.96736147]
MLR Mean Squared Error: 134.12367875356205
BLR Mean Squared Error: 132.45717148420502
(m1) stevet@sd233093 coding %
```

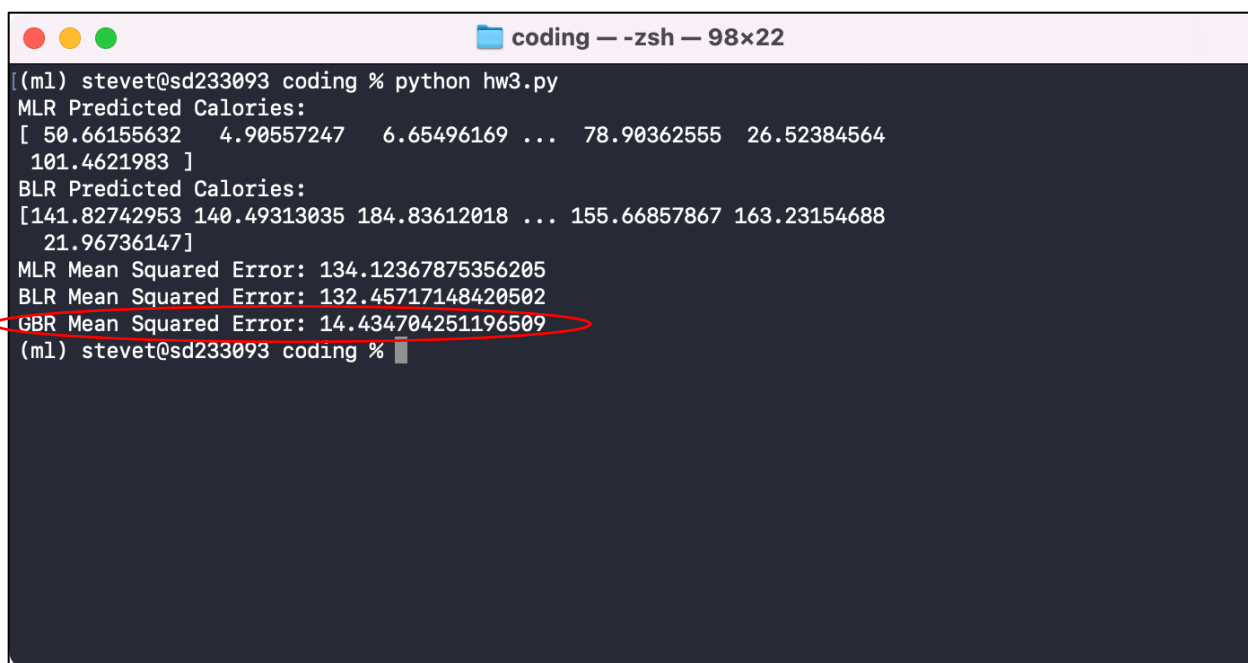
### V. Comparison of BLR & MLR:

這部分需要分別畫出 MLR 跟 BLR 的 fit line，我參考[1]的做法，並按照助教提供的題目要求，針對兩個已經處理好的 regression models，使用單一 feature: “duration”跟 label: “calories”畫出 Fit line。我自己的想法，我認為這兩種回歸的方法其實非常相似，因為兩者在 weight 的計算上只差了一項  $\lambda I$ ，而加入這項 regularization term ( $\lambda I$ ) 主要是讓參數估計值較不會趨近於 0，且可以讓值更趨近 prior distribution，因此訓練出來的模型相較於 MLR 可以產生更準確的估計值，歸功於多了一項 regularization 的步驟。



## VI. The other model:

我這裡直接選擇 sklearn 中的 GradientBoostingRegressor() 並使用 training data 做 fit，並對 testing data 做預測。Gradient Boost 是非常好用的方法，概念就是先用一個基本的 regression tree 當做 prediction (可能非常不準確)，之後在幾次迭代中使用 Residual value，也就是 prediction 跟真實值的差異，來持續修正 model，當 Residual 降低，model 的預測值也會更好。如果要更進一步說明 Residual，其實他的值就是針對 loss function (using MSE) 做 Gradient descent 後的值，所以當降低 Residual，也代表模型預測更準確。我最後得到的 MSE，相較於 MLE, BLR 好非常多。

A terminal window titled 'coding - -zsh - 98x22' showing the output of a Python script. The output compares three models: MLR, BLR, and GBR. MLR and BLR have high Mean Squared Errors (134.12 and 132.45 respectively), while GBR has a significantly lower error (14.43). The GBR result is circled in red.

```
(ml) stevet@sd233093 coding % python hw3.py
MLR Predicted Calories:
[ 50.66155632  4.90557247  6.65496169 ... 78.90362555 26.52384564
 101.4621983 ]
BLR Predicted Calories:
[141.82742953 140.49313035 184.83612018 ... 155.66857867 163.23154688
 21.96736147]
MLR Mean Squared Error: 134.12367875356205
BLR Mean Squared Error: 132.45717148420502
GBR Mean Squared Error: 14.434704251196509
(ml) stevet@sd233093 coding %
```

## VII. Reference:

[1] <https://www.kaggle.com/code/sathi557/bayesian-linear-regression-demo/notebook>