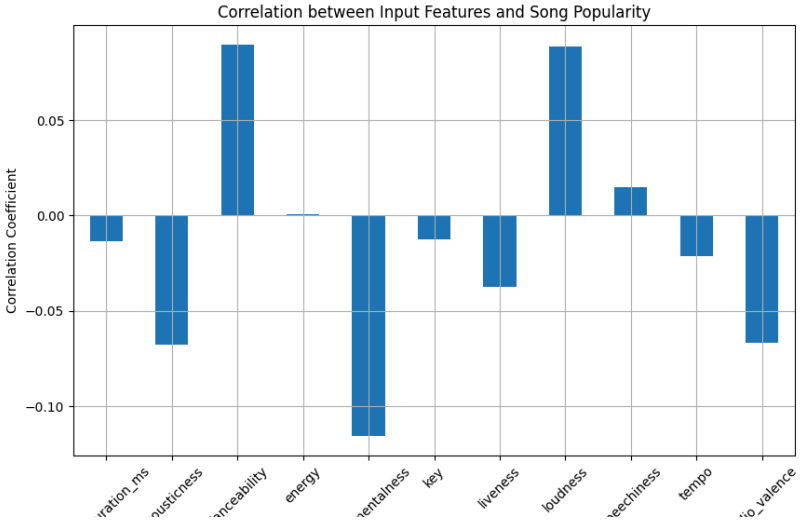
HW1 report

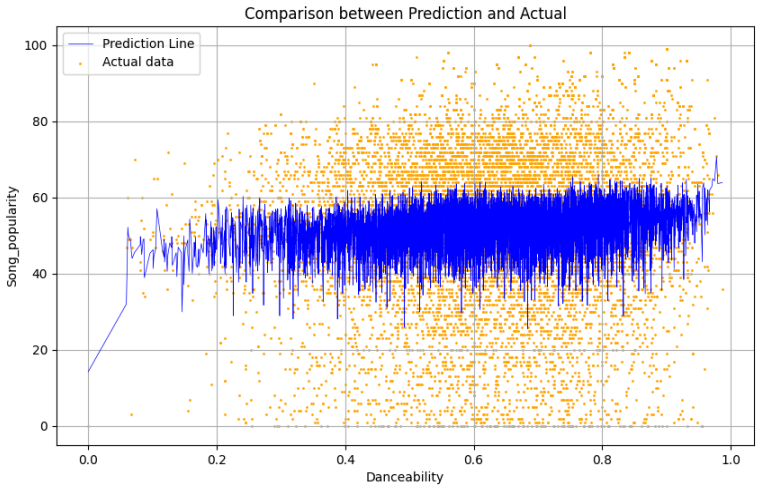
1.Fitting Curve

在本次的作業，訓練方法是使用Maximum Likelihood，透過Design Matrix以及dataset的output, 進行矩陣運算，逆推模型權重。

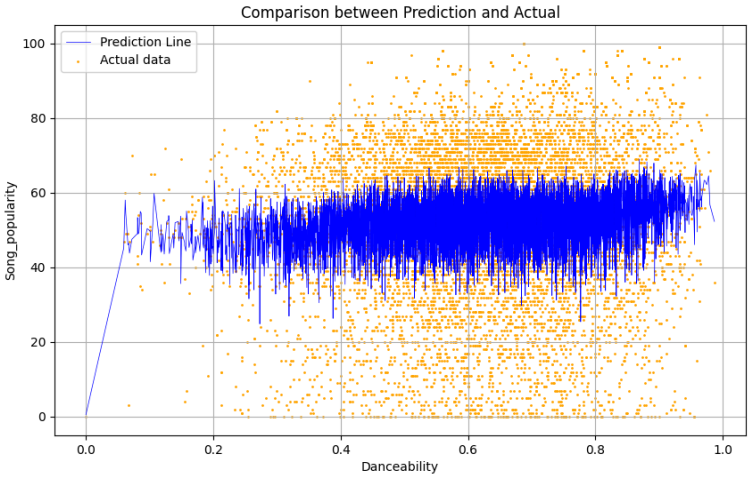


從圖表中可以看到，acousticness、danciability、instrumentalness、loudness以及audio_valence，會是比較具有相關性的數據。(此模型並沒有先進行feature selection)

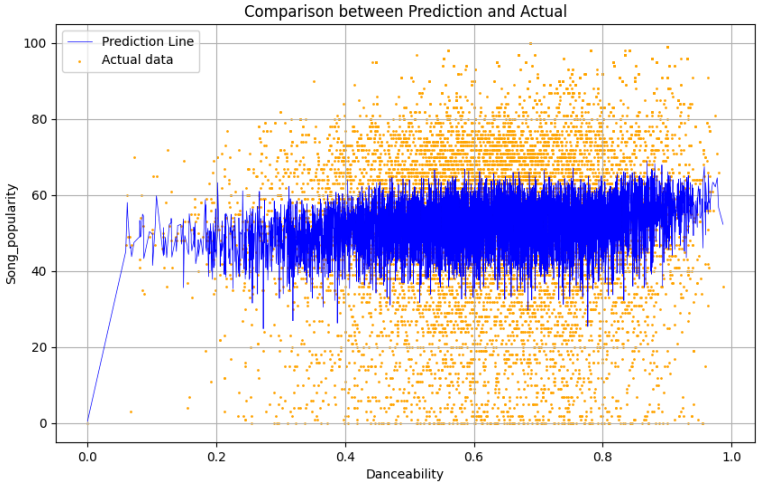
Training:



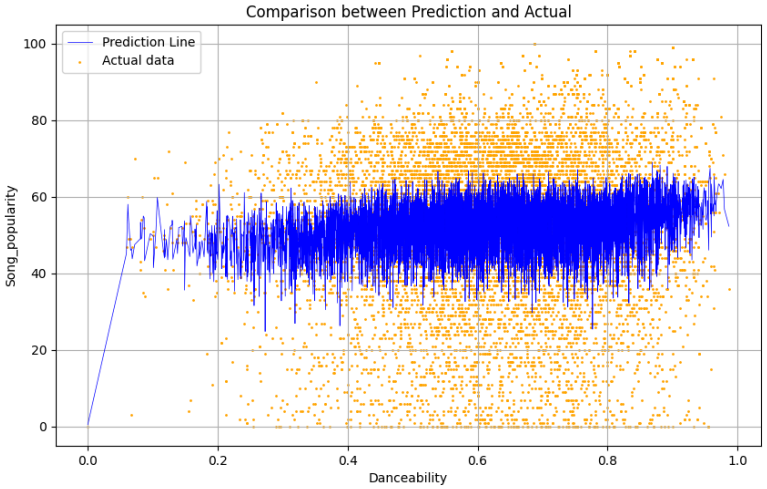
M=5


Comparison between Prediction and Actual
Prediction Line
Actual data
Song_popularity
Danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

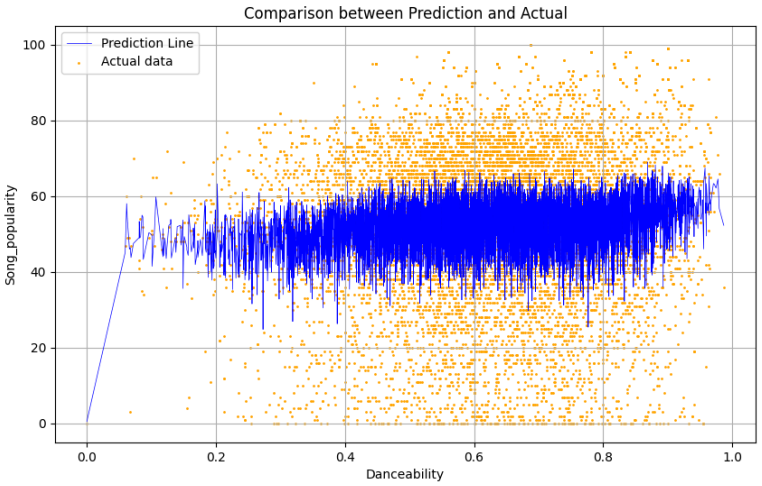
M=10


Comparison between Prediction and Actual
Prediction Line
Actual data
Song_popularity
Danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

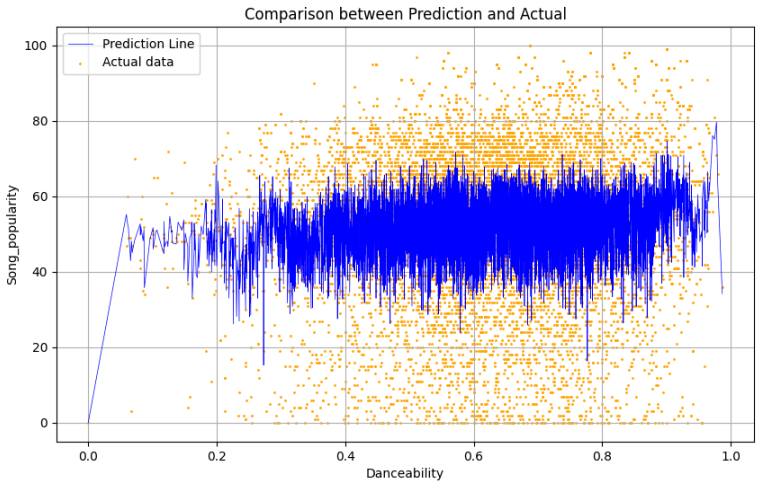
M=15


Comparison between Prediction and Actual
Prediction Line
Actual data
Song_popularity
Danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

M=20

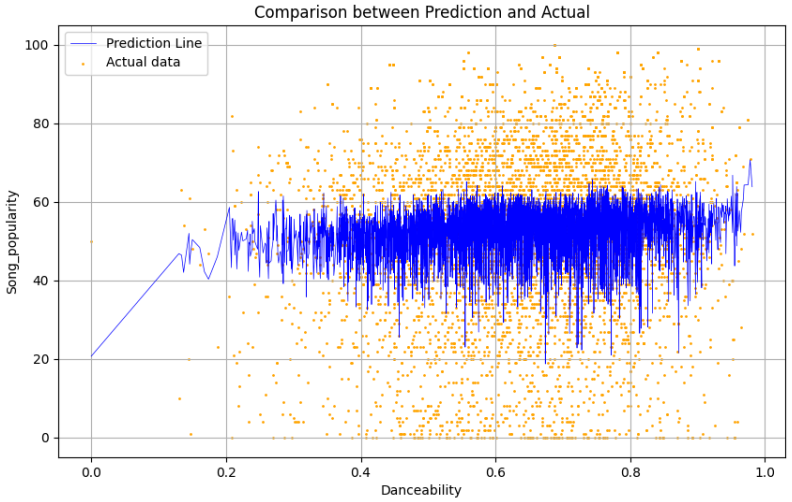


M=25

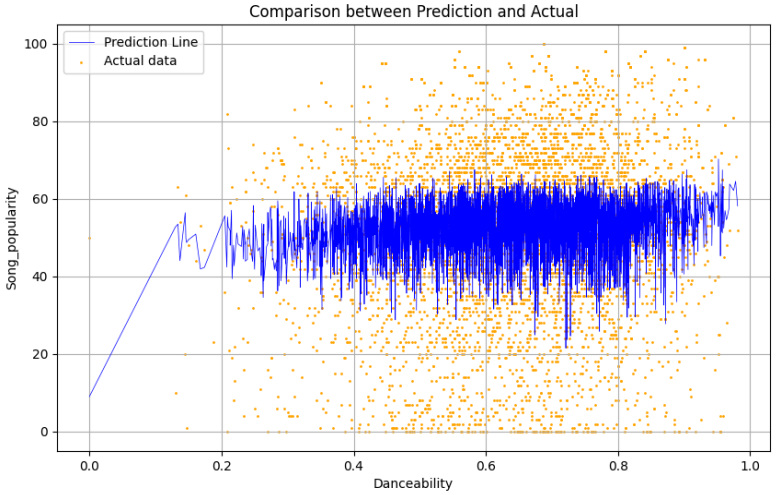


M=30

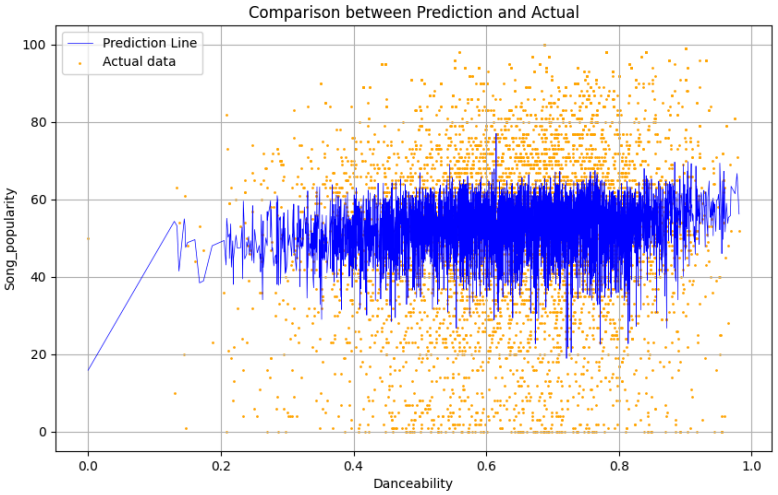
Testing:



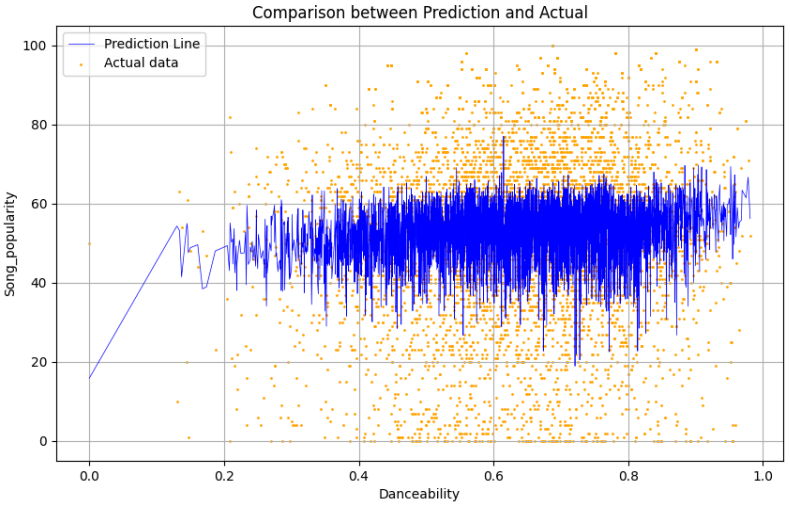
M=5


Comparison between Prediction and Actual
Prediction Line
Actual data
Song_popularity
Danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

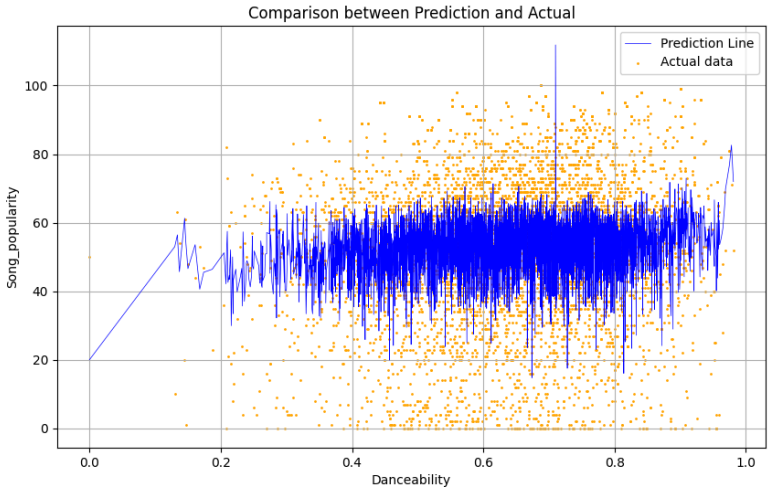
M=10


Comparison between Prediction and Actual
Prediction Line
Actual data
Song_popularity
Danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

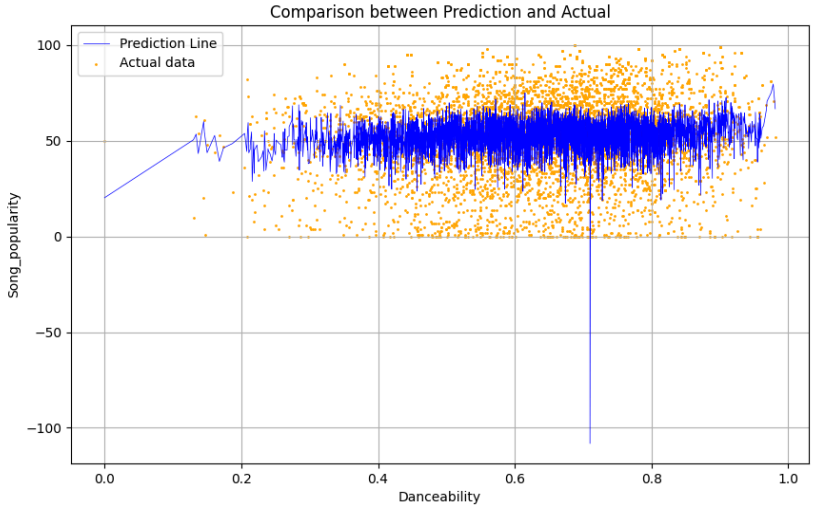
M=15


Comparison between Prediction and Actual
Prediction Line
Actual data
Song_popularity
Danceability
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

M=20



M=25



M=30

2.Mean Square Error & Accuracy

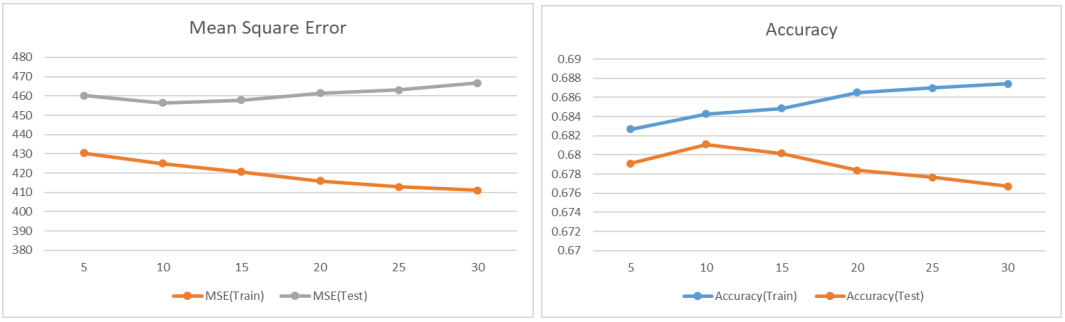
下圖為MSE以及Accuracy的折線圖，橫軸皆為M的數值。可以觀察到，模型在Training set上的表現，無論是MSE還是Accuracy, 皆是M越大，表現越好。這是正常的現象，因為通常參數量越多，模型的靈活性會更好，此時就能更適應訓練數據。

但是參數量並不是越多越好，以下圖為例：我們可以看見模型在Testing set上的表現，以這五個點來說，是M＝10表現最為優異：Accuracy最高，MSE最低。這代表在M=10之後，模型出現過擬合，即雖然在訓練數據上的表現更為優異，但是一但碰到未見過的數據，便會無法精準預測。

產生overfitting的原因除了參數量過多以外，數據量也是關鍵，過少的訓練資料也會導致overfitting。

至於Accuracy，目前是落在0.67~0.69之間，似乎並不是特別高，經過分析，我認為是合理的現象。第一，從下表可以發現：模型的輸入特徵與輸出之間的相關係數之絕對值，幾乎都不是特別的高，在選擇模型特徵時，通常會以相關係數作為依據，因此，若要得到更好的預

測結果，可以考慮進行特徵選擇(Feature Selection)，將相關性低的輸入特徵排除，便可以提高模型準確度。



3. Five-fold cross validation

5 fold cross validation的作法如下:首先，將data切割成5個Block, 並進行五次迭代，在每次迭代中，取其中一個block當作Testing set, 其餘則用於Training。最後，將五次迭代所獲得的權重W_ML 取平均，便可得到最後的權重。

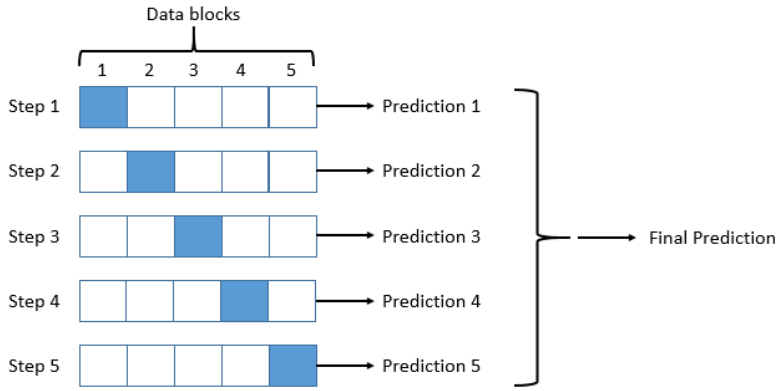
而cross validation的優點如下：

1.防止模型overfitting:

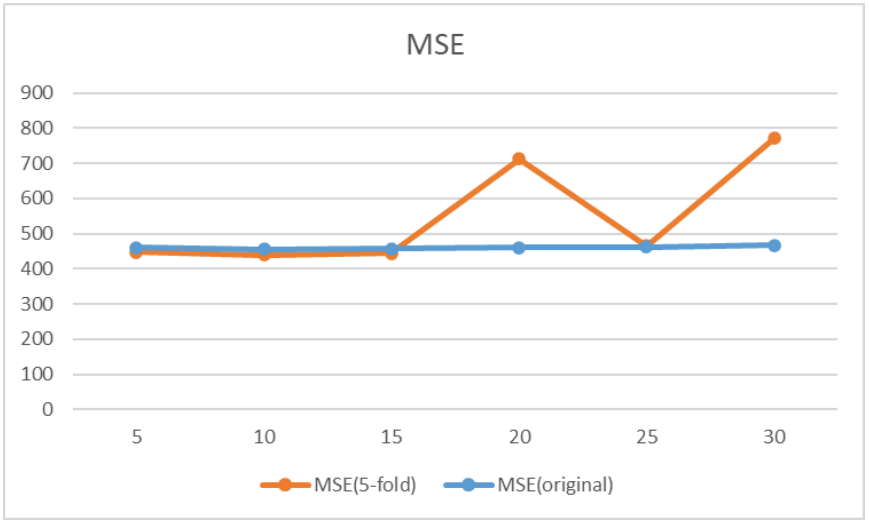
透過cross validation, 可以在同樣參數量的情況下，有效降低模型過擬合，通常除了在Testing data會有較高的準確性，對於其他未見過的資料，也能有較為出色的表現。

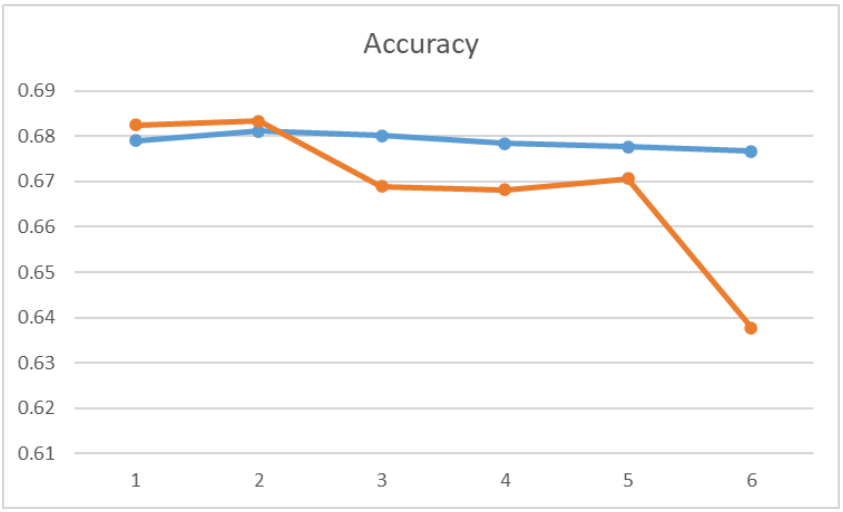
2.充分利用數據

在機器學習中，若是搜集的資料樣本數不夠多，很容易在訓練時產生overfitting的現象，而除了做data augmentation外，使用cross validation來進行模型訓練，也能一定程度的解決數據不足的問題。

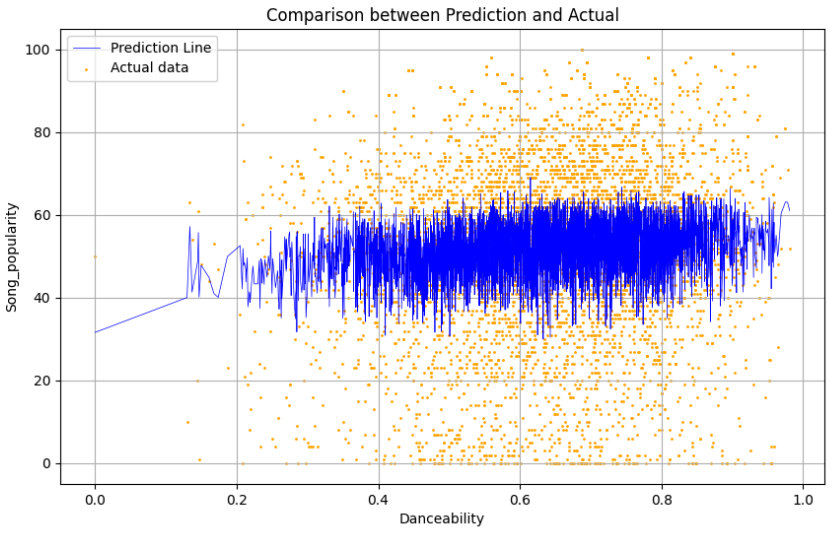


而以下是在與原先方法相同的Testing Set中評估的結果：



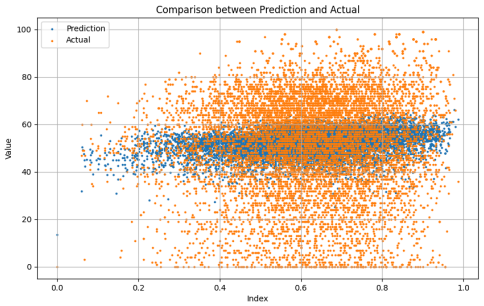


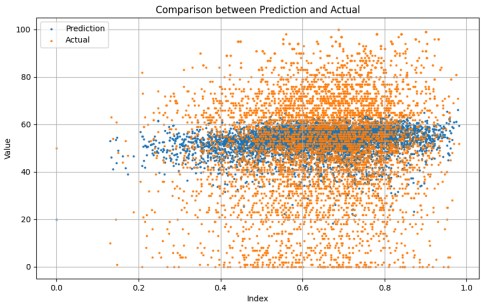
最後，透過以上的圖表可知，在M=10時，進行5-fold cross validation會得到最佳的效果。



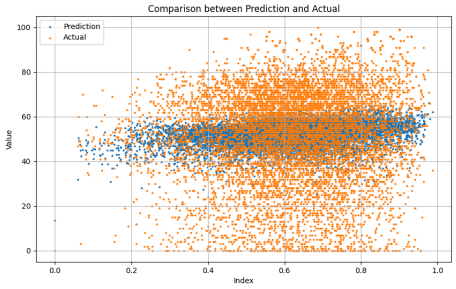
M=10, with 5-fold Cross Validation.

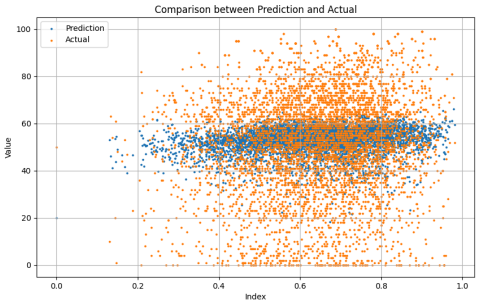
4.Regularization and repeat part 1.(左邊為Training set, 右邊為Testing set)



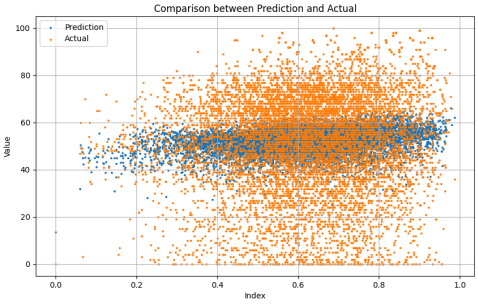


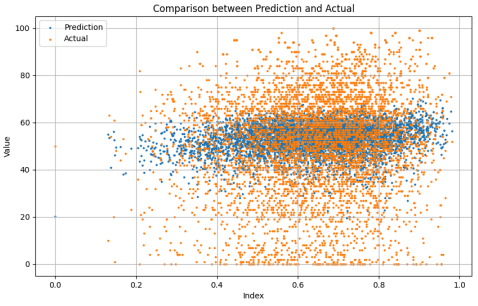
M=5, Lambda = 0.1



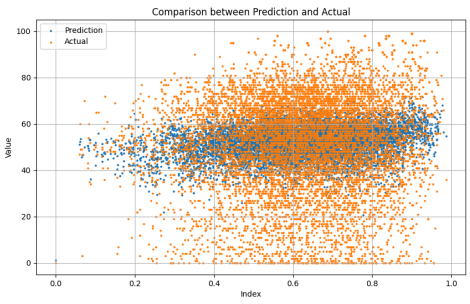


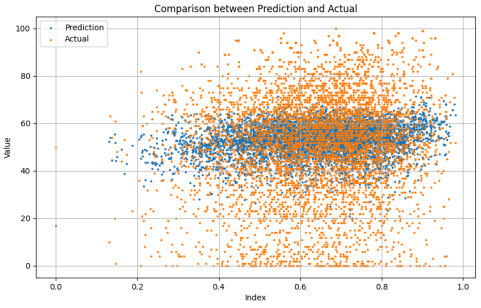
M=10, Lambda = 0.1



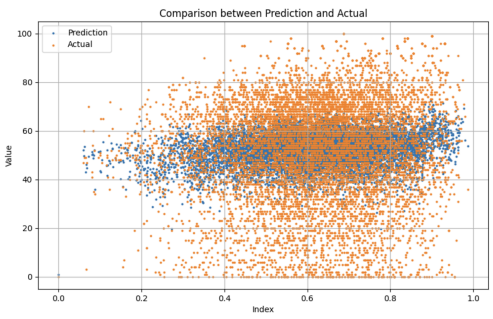


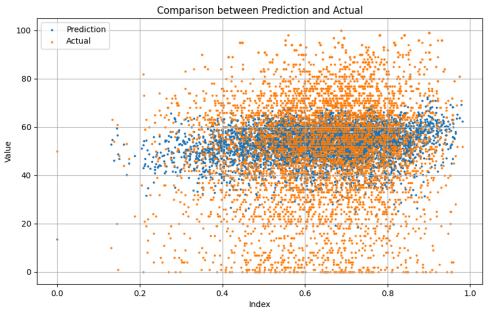
M=15, Lambda = 0.1



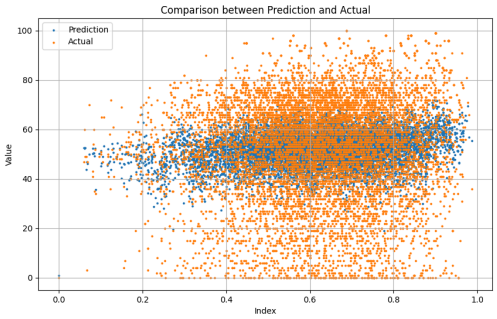


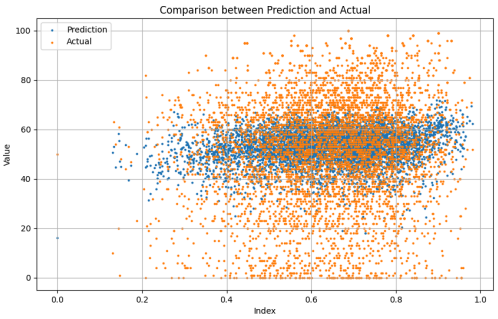
M=20, Lambda = 0.1





M=25, Lambda = 0.1





M=30, Lambda = 0.1

在Regularization中，我們在原先的Error unction中加入約束項

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{y(x_{i,1},\ldots,x_{i,K},\mathbf{w}) - t_i\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

故經過化簡，根據Maximum Likelihood的方式求權重時，必須在做Pseudo Inverse時加入 Lambda * I, 其中 Lambda即為正則化參數，I為Identity Matrix，如下:

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

Code:

```
//==============Self Defined Pseudo Inverse=========================
Eigen::MatrixXd ATA = Design_Matrix.transpose() * Design_Matrix;
ATA += Lamda * Eigen::MatrixXd::Identity(M*K, M*K);
Eigen::JacobiSVD<Eigen::MatrixXd> svd(ATA, Eigen::ComputeFullU | Eigen::ComputeFullV);
Eigen::MatrixXd ATA_inversed = svd.solve(Eigen::MatrixXd::Identity(M*K, M*K));
Eigen::MatrixXd W_ = ATA_inversed * Design_Matrix.transpose() * y;
Eigen::MatrixXd *W = new Eigen::MatrixXd(M, K);
//==================================================================
```

雖說c++ 的Eigen package是可以直接對矩陣進行pseudo inverse, 但是由於要加入 Regularization term, 所以便要將步驟拆開來做。而在求Inverse時，若是直接用.inverse()，會

遇到數值變成nan的情況，所以我的作法是利用Single Value Decomposition來進行以下的方程求解：

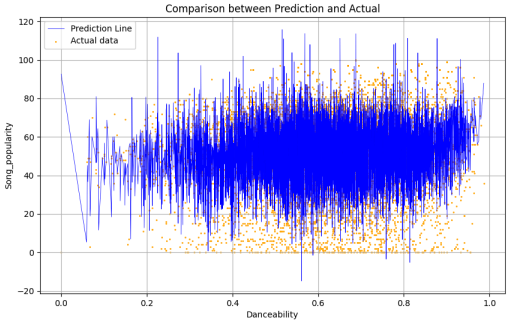
$$AX = I \rightarrow X = A^{-1}$$

正則化的意義在於，在Error Function中將weights作為變數，此時Error的大小除了output的正確性外，Weights的大小也成為影響的因素。如此一來，能夠限制模型產生過於擬合於訓練資料的參數，有助於防止overfitting。並且，我們可以透過調整lambda，來改變正則化的強弱，以下也比對了不同的Lamda所造成的影響。

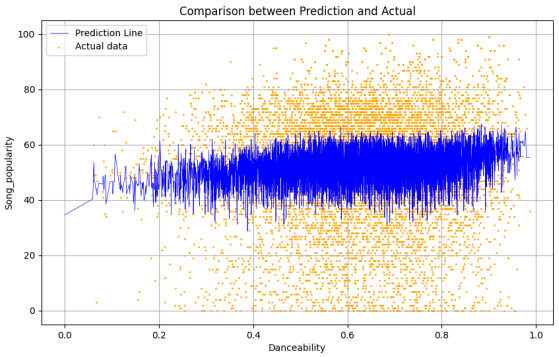
首先是做了Regularization後的MSE以及Accuracy:

以上幾張圖有幾點特別值得觀察:當Lambda(正則化參數)上升時，對於Training set的MSE以及Accuracy影響並不大，頂多在M較大時能觀察到performance稍微下降(由此可見權重有受到約束，不會過度依賴訓練資料)，但是到了Testing set，可以從Accuracy中觀察到，，較大的Lambda，雖說在M=10時幾乎不受影響，但無論是將M調大或調小，模型在Testing set的擬合能力都較強。這就是做Regularization的意義:約束權重避面過度擬合Training set，提升對於新鮮資料的辨識能力。

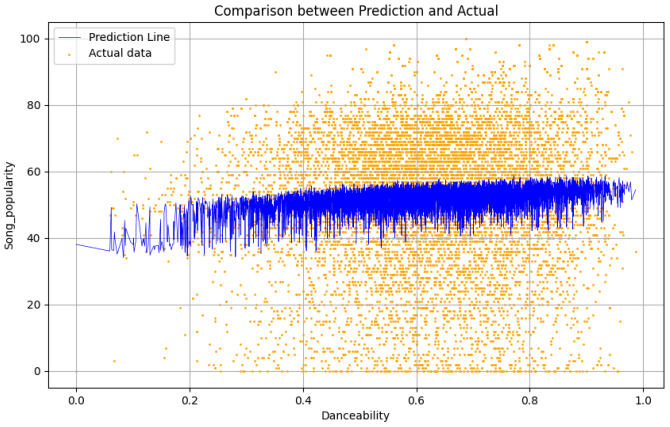
除此之外，也用了一些比較極端的數值觀察Regularization的運作方式，如下圖:
當Lamda 從 -100提升至 100時，明顯看到預測資料會趨近平滑。



Lambda = -100, M=30



Lambda = 10, M=30



Lambda = 100, M=30