HW01

PM2.5 prediction

313581038 智能系統 蒲品憶

1. How do you select features for your model input, and what preprocessing did you perform?

選擇特徵:

每項特徵的與 PM2.5 的相關係數

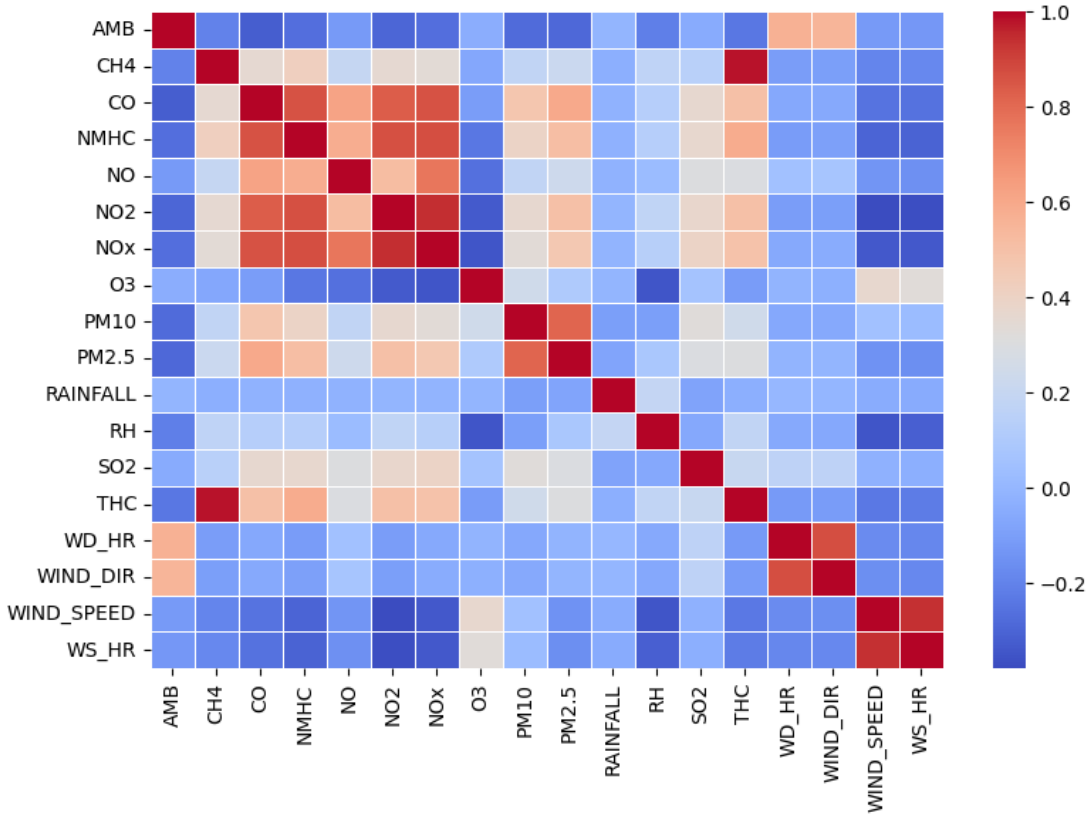


Figure1: 相關係數圖

```
================ Feature Selection - top_n =========
與 PM2.5 排名最的前 17 個特徵：
PM10          0.816310
CO            0.600925
NMHC          0.511133
NO2           0.499470
NOx           0.463238
THC           0.303378
SO2           0.297900
NO            0.233854
CH4           0.218112
O3            0.101997
RH            0.084675
WIND_DIR     -0.011948
WD_HR        -0.017137
RAINFALL     -0.077858
WIND_SPEED   -0.146933
WS_HR        -0.162156
AMB          -0.291249
Name: PM2.5, dtype: float64
================ analysis graph ==================
```

Figure2: 各項數值

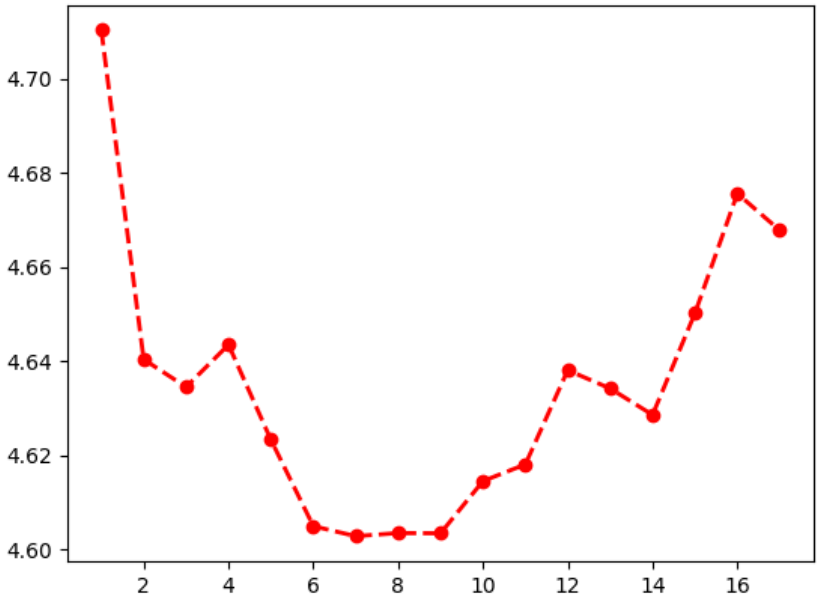


Figure3: 選取不同特徵對 valid loss 的影響

從 Figure1 可以觀察出 PM10 對於 PM2.5 的關係最接近、CO 次之、NMHC、NO2、NO 並列、後面接著是 SO2、THC……。

在 Figure2 能夠清楚知道每項特徵與 PM2.5 對應的相關係數的詳細數值，其中以 PM10 對於 PM2.5 的關係最接近。最初挑選特徵，我是先將小於 0 以下的特徵刪除，效果明顯比使用全部特徵 valid loss 下降許多；後來又嘗試使用 top 6 的相關 features 作為特徵，效果有稍微好一些。

之後，我又想說試試全部不同特徵對於 valid loss 的影響，於是產生 Figure3。從 Figure3 可以看出，選擇 top 6 -top 9 的 features 對於 valid loss 下降效果較佳，經過丟上 kaggle 測試後，最後認定取 top6 features 對於目前任務效果最佳。

前處理:

數據清理：將#,*,x,A 改成用零替換。在實作時發現 test data 還有個不是數值的值'WIND_DIR+D2070EC'也將之進行替換，其實也可以用平均值替換但是比較費工夫，所以這裡我只用補零處理。做數據清理好處是，可以避免丟失大量有價值的信息，從而保留更多的數據以供分析。而且要做要做 linear regression 之前要先將數據都轉換成數值。

數據標準化：將數據縮放到相同範圍，標準化（z-score）或最小-最大縮放。

公式: x = (x – mean_x) / std_x

標準化的好處:

A. **加快收斂速度:** 梯度下降法會受到數據範圍的影響。標準化可以讓所有特徵處於相同的尺度，這樣梯度下降等優化算法會收斂得更快。
B. **提高模型性能:** 標準化能夠讓各個特徵對模型訓練的影響更加均衡，使得每個特徵都在相似的範圍內，這有助於模型更好地學習每個特徵的影響，從而提高模型的性能。
C. **避免數據偏差:** 如果數據集中的某些特徵的數值範圍過大，未經標準化可能會導致模型訓練過程中的不均衡，這可能會引起模型的偏差。標準化有助於減少這種偏差，讓模型在所有特徵上學習的影響更為均衡。

2. Compare the impact of different amounts of training data on the PM2.5 prediction accuracy. Visualize the results and explain them.

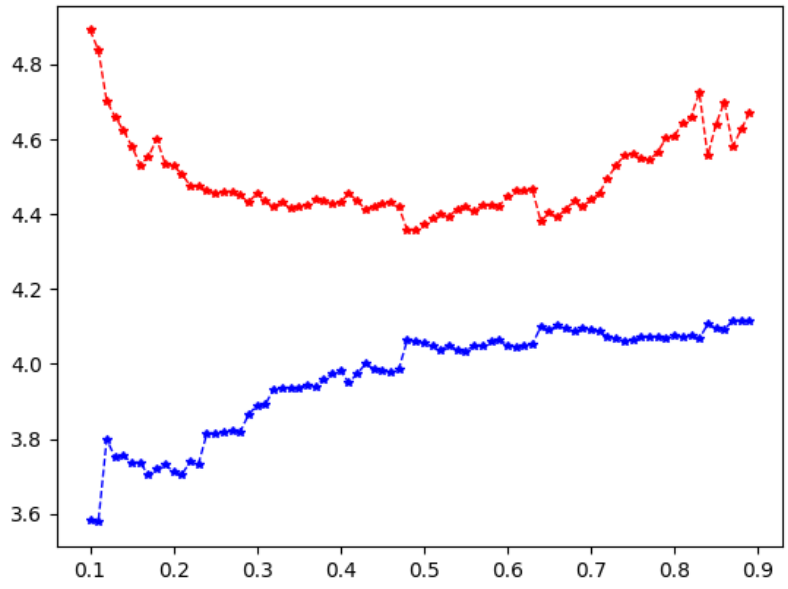


Figure3: 不同訓練集之下 valid and train loss

紅色: valid loss、藍色: train loss、X: training Data、Y: predict

訓練數據的影響：藍色線條代表模型在訓練損失。

隨著訓練資料量的增加，訓練損失穩步下降，這是因為模型有更多資料來學習。

測試數據的影響：紅色代表的是測試損失。

驗證損失初期下降，但在某個點之後開始上升，顯示出過擬合(overfitting)的情形。這意味著當訓練資料過多時，模型開始學習過於專注於訓練資料中的細節，無法更好地處理新的資料。

總結:

訓練及太多或太少都會造成效果不佳，使用 training data 雖然圖上，0.5-0.7 區間 valid loss 最低，但是考慮到 test Data 的特徵值可能與 valid Data 不這麼相似，所以我選擇讓 training data 調整在 0.7-0.8 區間進行訓練，讓更多的特徵進行學習，有機會在 test Data 有更好的表現。

3. Discuss the impact of regularization on PM2.5 prediction accuracy.

（Regularization）在機器學習中主要是用來防止模型過擬合的技術，對於 PM2.5 預測準確度有著重要影響。當模型訓練時，尤其是在資料較為複雜或者訓練資料量較少的情況下，過擬合的風險會增高，這會導致模型在驗證集或測試集上的表現不佳。正則化技術通過在損失函數中加入額外的項來限制模型的複雜度，從而提高其泛化能力。

以下是我使用的 Regularization 方法，將會提及公式、三種方法說明以及三種方法，最終使用何種 Regularization。

正則化公式：

1. **L2 正則化（Ridge）**：L2 正則化項的公式是 $\lambda \sum w_i^2$，梯度是 $2\lambda w$。
2. **L1 正則化（Lasso）**：L1 正則化項的公式是 $\lambda \sum |w_i|$，梯度是 $\lambda \cdot \text{sign}(w)$。
3. **ElasticNet 正則化**：ElasticNet 正則化是 L1 和 L2 正則化的組合，公式是 $\lambda_1 \sum |w_i| + \lambda_2 \sum w_i^2$，梯度是 $\lambda_1 \cdot \text{sign}(w) + 2\lambda_2 w$。

三種方法說明:

1. L2 正則化（Ridge）

- **L2 正則化的作用**：對模型的權重參數進行懲罰，特別是權重過大時。它的正則化項是權重的平方和，這會使模型更加平滑，從而減少過擬合的風險。
- **λ 的影響**：
  - 當 **λ 較小** 時，正則化的效果較弱，模型的權重較大，可能容易過擬合。
  - 當 **λ 較大** 時，正則化會較強，會使模型的權重變得較小，進而降低過擬合的風險。但如果 **λ 太大**，模型的表現會變得過於簡單，可能會欠擬合（underfitting）。

2. L1 正則化（Lasso）

- **L1 正則化的作用**：L1 正則化會對模型的權重進行懲罰，並且會推動某些權重為零，這樣可以實現**特徵選擇**。L1 正則化的正則化項是權重的絕對值和，這樣某些特徵的權重會被壓縮到零，從而消除不重要的特徵。
- **λ 的影響**：
    - 當 **λ 較小** 時，L1 正則化的效果較弱，模型會包含更多的特徵，可能會導致過擬合。
    - 當 **λ 較大** 時，L1 正則化會強烈地懲罰權重，使得更多的特徵被壓縮到零，模型變得更簡單，避免過擬合，但也有可能造成欠擬合。

3. ElasticNet 正則化

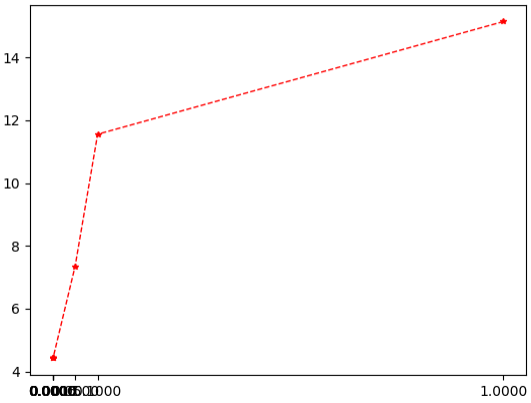
- **ElasticNet 正則化的作用**：ElasticNet 正則化結合了 L1 和 L2 正則化的特點，既能進行**特徵選擇，又能控制模型的複雜度**。它的正則化項包含 L1 和 L2 的加權和，因此既能保持 L1 的特徵選擇能力，又能避免 L2 正則化過度平滑的問題。
- **λ_1 和 λ_2 的影響**：
    - λ_1 控制 L1 正則化的強度（特徵選擇的效果）。
    - λ_2 控制 L2 正則化的強度（控制模型複雜度）。
    - 當 **λ_1 較大** 時，更多的特徵會被壓縮為零，並且特徵選擇的效果會更強。
    - 當 **λ_2 較大** 時，模型會更加平滑，權重會被壓縮到較小的值，減少過擬合的風險。

總結:

- L2 正則化（Ridge）：適用於所有特徵都對預測有貢獻的情況，並且 λ 值過大會導致欠擬合。
- L1 正則化（Lasso）：適用於特徵數量很大且懷疑某些特徵不重要的情況，並且 λ 值過大會過度縮小權重，導致過度簡化。
- ElasticNet 正則化：當數據中有冗餘特徵時，ElasticNet 是一個很好的選

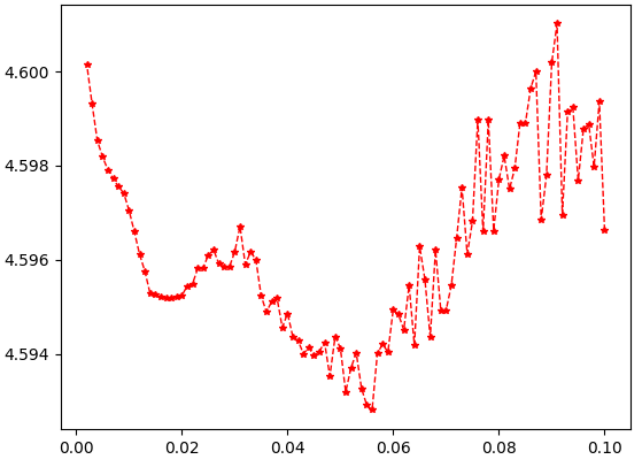
實作比較:

1. L2 正則化（Ridge）



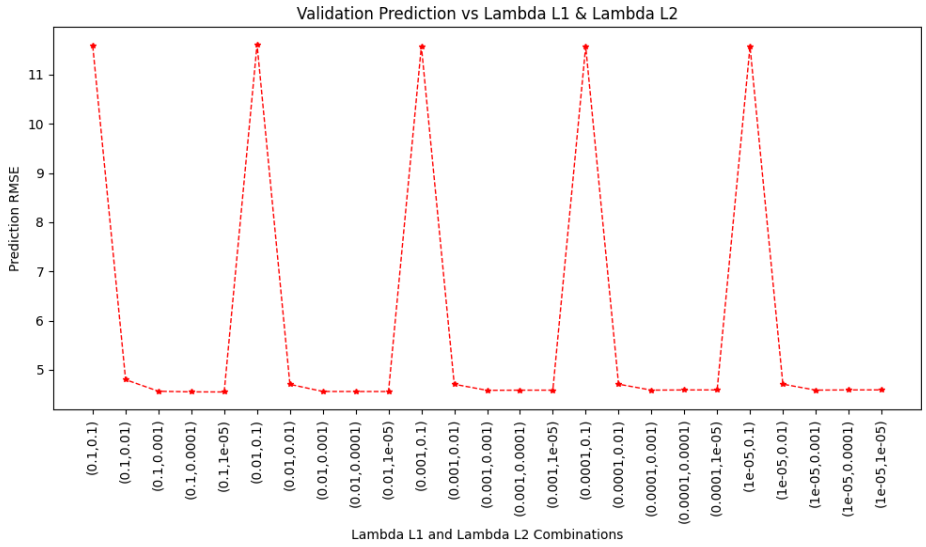
測試 lambda_reg=0.00001~1，測試結果可以看出在 lambda_reg=0.01 時，valid loss 數值最低，valid loss=4.6535841555

2. L1 正則化（Lasso）



測試 lambda_reg=0.01~0.1，測試結果可以看出在 lambda_reg=0.56，valid loss 值最低，valid loss=4.587490810568106

3. ElasticNet 正則化



測試 lambda_reg_L2 = [0.1, 0.01, 0.001, 0.0001, 0.00001]，
lambda_reg_L1 = [0.1, 0.01, 0.001, 0.0001, 0.00001]，
測試結果可以看出在(lambda_reg_L1, lambda_reg_L2) =(0.1, 1e-05)時，valid loss 數值最低，valid loss=4.6972905330245

結論:

並沒有一定是哪個方法好壞，只是在此任務中時做測試的結果下，我使用 L1 正則化（Lasso），在 valid loss 的效果最佳因此使用此方法。推測是因為 L2 正則化（Ridge）的平方損失方式對於簡單的梯度下模型效果太強烈，導致成效不好；ElasticNet 是 L1+L2 的混合版本，因此可能也有相同問題。