

說明：請各位使用此 template 進行 Report 撰寫，如果想要用其他排版模式也請註明題號以及題目內容（請勿擅自更改題號），最後上傳至cool前，請務必轉成PDF檔，否則將不予計分。

備註：

- 所有 advanced 的 gradient descent 技術(如: adam, adagrad 等)都可以用
- 第2題請以題目給訂的model來回答

學號：R14921A13 系級：電機所資安組碩一 姓名：鄭皓中

1. (0.4%) 解釋什麼樣的 data preprocessing 可以 improve 你的 training/testing accuracy, e.g., 你怎麼挑掉你覺得不適合的 data points。請提供數據(例如 kaggle public score RMSE)以佐證你的想法。

以下這是在使用了feature selection + 去除極端值之後的結果

	stepwise_e200_lr1e-3_wonorm.csv Complete · 3d ago	3.14115	<input type="checkbox"/>
---	--	---------	--------------------------

這是沒有去除極端值（但有feature selection）的結果

	stepwise_e200_lr1e-3_wovalid.csv Complete · now	4.19286	<input type="checkbox"/>
---	--	---------	--------------------------

這是沒有進行feature selection（但有去除極端值）的結果

	stepwise_e200_lr1e-3_wofeat.csv Complete · 4m ago	4.76144	<input type="checkbox"/>
---	--	---------	--------------------------

我這邊使用 stepwise selection 來進行特徵選擇，透過逐步地將特徵加入考慮範圍，如果AIS有降低的話，就將特徵加入，否則就丟掉，然後再考慮移除特徵，如果該特徵移除後會降低AIS的話，則將特徵移除。

可以看到沒有進行 feature selection 的話，RMSE 會較使用後高非常多，這可能是因為我們將比較不相關的特徵篩選掉，避免他們誤導模型進行錯誤的權重改變。

去除極端值的部分是使用基本的 IQR，透過將高於第75百分位數3倍IQR，或低於第25百分位數3倍IQR的部分去除，而 y 的部分則是直接判斷如果  $y > 50$  就刪除，減少極端值對模型的影響。

可以看到如果沒有去除極端值的話，RMSE也會增加很多，這可能是極端值讓整個模型的預測偏離，透過去除極端值可以避免極端值主導模型的走向。

2. (0.8%) 請實作 2nd-order polynomial regression model (不用考慮交互項)。

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 \text{ 其中 } x^2 = [x_1^2, x_2^2, \dots, x_n^2]$$

(a) 貼上 polynomial regression 版本的 Gradient descent code 內容。

(b) 在只使用 NO 數值作為 feature 的情況下，紀錄該 model 所訓練出的 parameter 數值以及 kaggle public score。

(a)

```
def minibatch_2(x, y, config):
    # 打亂資料
    index = np.arange(x.shape[0])
    np.random.shuffle(index)
    x = x[index]
    y = y[index]

    x_poly = polynomial_features(x, degree=2)

    # Initialization
    batch_size = config.batch_size
    lr = config.lr
    epoch = config.epoch
    decay_rate = config.decay_rate
    epsilon = 1e-8

    w = np.full((x_poly.shape[1], 1), 0.1) # 權重
    bias = 0.1

    # Optimizer states
    cache_w = np.zeros_like(w)
    cache_b = 0.0

    # Training loop
    for num in range(epoch):
        for b in range(int(x_poly.shape[0] / batch_size)):
            x_batch = x_poly[b * batch_size:(b + 1) * batch_size]
            y_batch = y[b * batch_size:(b + 1) * batch_size].reshape(-1, 1)

            # Prediction
            pred = np.dot(x_batch, w) + bias

            # Loss
            loss = y_batch - pred

            # Gradient
            g_t = np.dot(x_batch.T, loss) * (-2)
            g_t_b = loss.sum(axis=0) * (-2)

            # Update cache (RMSProp)
            cache_w = decay_rate * cache_w + (1 - decay_rate) * g_t**2
            cache_b = decay_rate * cache_b + (1 - decay_rate) * g_t_b**2

            # Update params
            w -= lr * g_t / (np.sqrt(cache_w) + epsilon)
            bias -= lr * g_t_b / (np.sqrt(cache_b) + epsilon)

    return w, bias
```

```
def polynomial_features(X, degree=2):
    n, d = X.shape
    features = [X] # 一階
    for j in range(d):
        features.append(X[:, j:j+1] ** 2) # 平方
    return np.hstack(features)
```

由於在使用 polynomial regression 需要將 test 也進行 polynomial features，所以另寫一個函式將 dataset 改成  $[x, x^2]$ 。

(b) 僅使用 NO 進行 2nd-order polynomial regression model 的權重如下

```
w = [[-0.2031377 ] [ 0.45217   ] [ 0.43107988] [ 0.70075698] [ 0.76486358]
[ 0.79979138] [ 0.42908355] [-0.44784197] [ 0.04562753] [-0.00634792]
[-0.01656999] [-0.05146903] [-0.0382639 ] [-0.03136936] [ 0.00210981]
[ 0.15853816]]
```

bias = [9.80706835]

最終則是獲得了這樣的分數



NO.csv  
Complete · 12h ago

4.65456



3. (0.8%) 請實作 feature normalization。

(a) 貼上 normalization 的 code 內容。

(b) 在只使用 WD\_HR 和 PM 2.5 數值作為 feature 且固定 train\_config 的情況下，紀錄 model 在有無使用 normalization 之下的 kaggle public score 變化。並試著解釋其原因。

(a)

```
# Your implementation here
norm_params = {}
for column in df.columns:
    if column == 'PM2.5':
        continue
    mean = df[column].mean()
    std = df[column].std()
    norm_params[column] = {'mean': mean, 'std': std}
    df[column] = (df[column] - mean) / std

return df, norm_params
```

(b)



WDHR\_norm.csv  
Complete · now

3.07195



WDHR\_wonorm.csv  
Complete · 20s ago

3.06720



這是使用normalization跟不使用的差別，由於code上設計normalization不會apply到PM2.5上，所以在使用完normalization可以看到 RMSE 變大的情形，可能是因為只對某一個特徵做normalization會讓模型整體向另一邊傾斜，導致模型判斷變得不準確。