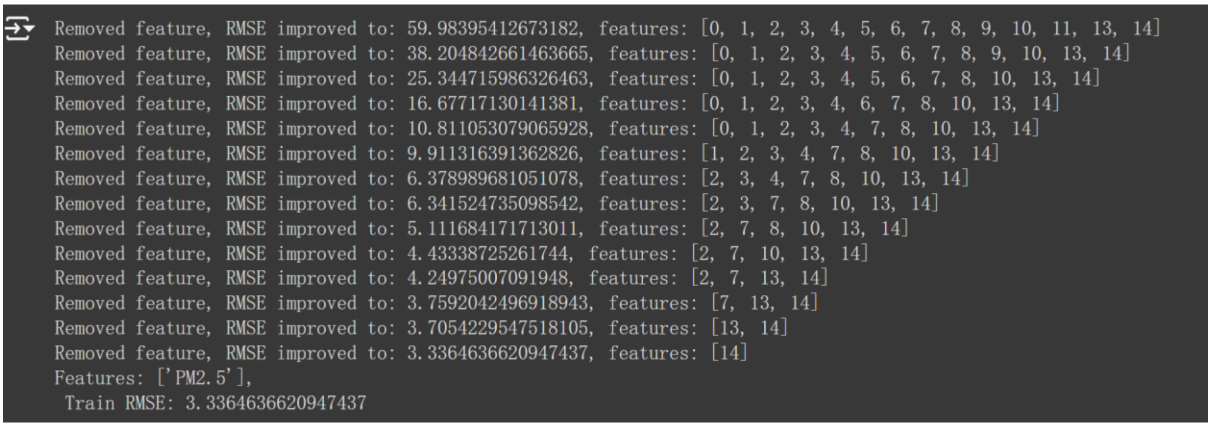
學號：R12227118 系級： 心理所碩二 姓名：張瑋宸

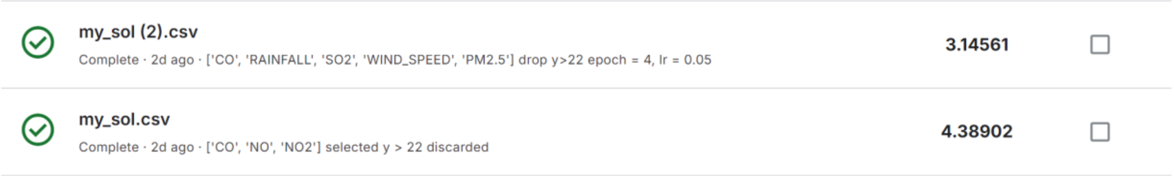
1. (1%) 解釋什麼樣的data preprocessing可以improve你的training/testing accuracy，e.g., 你怎麼挑掉你覺得不適合的data points。請提供數據(例如 kaggle public score RMSE)以佐證你的想法。

   Ans: 在Feature Selection上，採用backward selection的方法，從所有包含所有feature的full model 開始逐步移除使得testing RMSE 降低最多的data points，直到移除feature都無法降低RMSE為止。最後剩下的 features 作為最終模型的挑選的參考，進行training, testing。
   訓練過程中training RMSE有持續下降的過程如下圖一。且Kaggle上 public score也比之前隨意挑選三個Features 之結果還好，見下圖二。此外亦有透過畫箱型圖將如PM2.5 的Q90以上的極端資料點刪除，使得資料fit 更好，刪除PM2.5極端值亦有使Kaggle public score上升。



```
Removed feature, RMSE improved to: 59.98395412673182, features: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14]
Removed feature, RMSE improved to: 38.204842661463665, features: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14]
Removed feature, RMSE improved to: 25.344715986326463, features: [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 13, 14]
Removed feature, RMSE improved to: 16.67717130141381, features: [0, 1, 2, 3, 4, 6, 7, 8, 10, 13, 14]
Removed feature, RMSE improved to: 10.811053079065928, features: [0, 1, 2, 3, 4, 7, 8, 10, 13, 14]
Removed feature, RMSE improved to: 9.911316391362826, features: [1, 2, 3, 4, 7, 8, 10, 13, 14]
Removed feature, RMSE improved to: 6.378989681051078, features: [2, 3, 4, 7, 8, 10, 13, 14]
Removed feature, RMSE improved to: 6.341524735098542, features: [2, 3, 7, 8, 10, 13, 14]
Removed feature, RMSE improved to: 5.111684171713011, features: [2, 7, 8, 10, 13, 14]
Removed feature, RMSE improved to: 4.43338725261744, features: [2, 7, 10, 13, 14]
Removed feature, RMSE improved to: 4.24975007091948, features: [2, 7, 13, 14]
Removed feature, RMSE improved to: 3.7592042496918943, features: [7, 13, 14]
Removed feature, RMSE improved to: 3.7054229547518105, features: [13, 14]
Removed feature, RMSE improved to: 3.3364636620947437, features: [14]
Features: ['PM2.5'],
 Train RMSE: 3.3364636620947437
```

圖一: Train RMSE隨著backward selection下降



my_sol (2).csv
Complete · 2d ago · ['CO', 'RAINFALL', 'SO2', 'WIND_SPEED', 'PM2.5'] drop y>22 epoch = 4, lr = 0.05
3.14561 □

my_sol.csv
Complete · 2d ago · ['CO', 'NO', 'NO2'] selected y > 22 discarded
4.38902 □

圖二: Kaggle Public Score的上升，其中my_sol (2) 為使用backward selection後，my_sol.csv則是未使用前隨意挑的Features。

2. (1%) 請實作 2nd-order polynomial regression model (不用考慮交互項)。

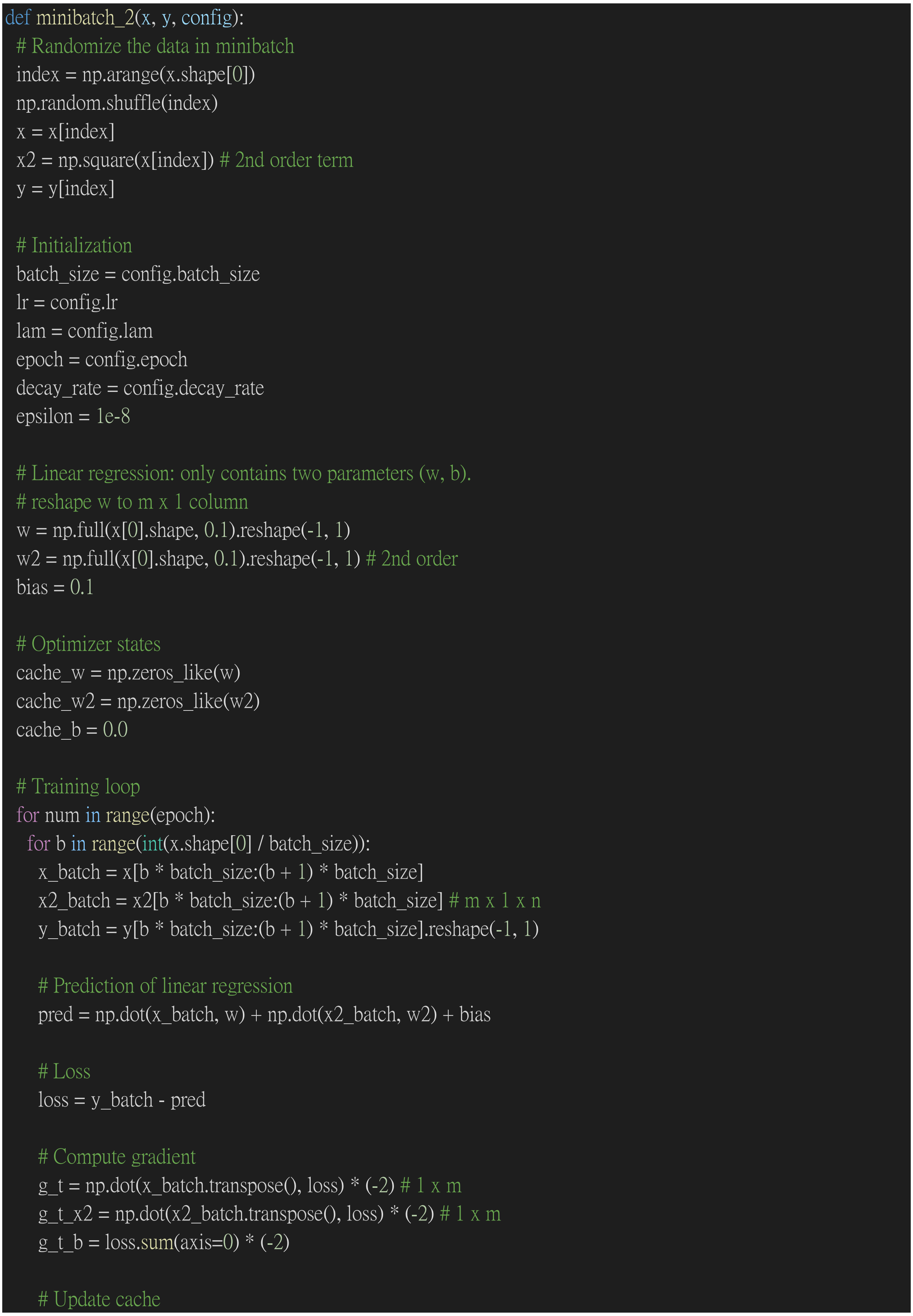
$$y = \beta_0 + \boldsymbol{\beta_1 x} + \boldsymbol{\beta_2 x^2} \quad \text{其中 } \boldsymbol{x^2} = [x_1^2, x_2^2, \ldots, x_n^2]$$

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

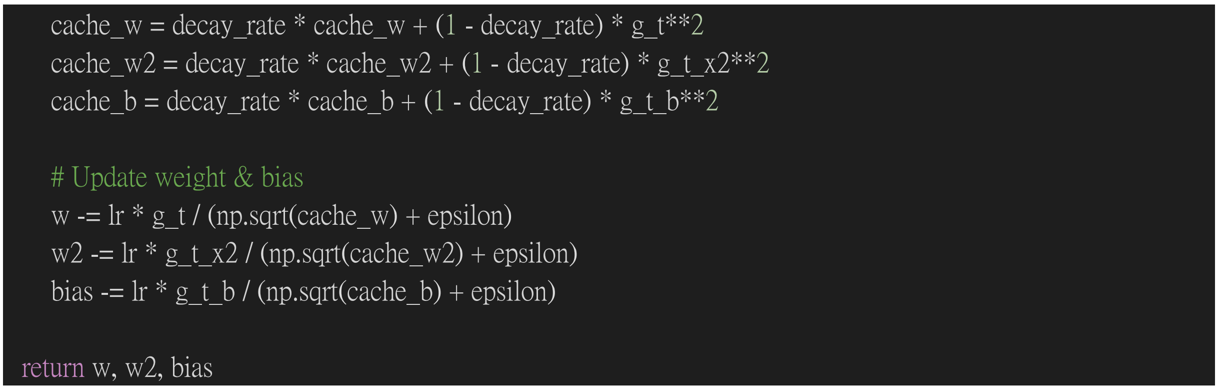
(b) 在只使用 NO 數值作為 feature 的情況下，紀錄該 model 所訓練出的 parameter 數值以及 kaggle public score.

Ans:

(a)



```
def minibatch_2(x, y, config):
  # Randomize the data in minibatch
  index = np.arange(x.shape[0])
  np.random.shuffle(index)
  x = x[index]
  x2 = np.square(x[index]) # 2nd order term
  y = y[index]

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

  # Linear regression: only contains two parameters (w, b).
  # reshape w to m x 1 column
  w = np.full(x[0].shape, 0.1).reshape(-1, 1)
  w2 = np.full(x[0].shape, 0.1).reshape(-1, 1) # 2nd order
  bias = 0.1

  # Optimizer states
  cache_w = np.zeros_like(w)
  cache_w2 = np.zeros_like(w2)
  cache_b = 0.0

  # Training loop
  for num in range(epoch):
    for b in range(int(x.shape[0] / batch_size)):
      x_batch = x[b * batch_size:(b + 1) * batch_size]
      x2_batch = x2[b * batch_size:(b + 1) * batch_size] # m x 1 x n
      y_batch = y[b * batch_size:(b + 1) * batch_size].reshape(-1, 1)

      # Prediction of linear regression
      pred = np.dot(x_batch, w) + np.dot(x2_batch, w2) + bias

      # Loss
      loss = y_batch - pred

      # Compute gradient
      g_t = np.dot(x_batch.transpose(), loss) * (-2) # 1 x m
      g_t_x2 = np.dot(x2_batch.transpose(), loss) * (-2) # 1 x m
      g_t_b = loss.sum(axis=0) * (-2)

      # Update cache
```

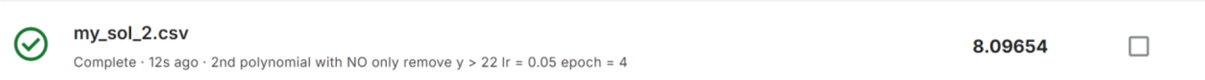


```
    cache_w = decay_rate * cache_w + (1 - decay_rate) * g_t**2
    cache_w2 = decay_rate * cache_w2 + (1 - decay_rate) * g_t_x2**2
    cache_b = decay_rate * cache_b + (1 - decay_rate) * g_t_b**2

    # Update weight & bias
    w -= lr * g_t / (np.sqrt(cache_w) + epsilon)
    w2 -= lr * g_t_x2 / (np.sqrt(cache_w2) + epsilon)
    bias -= lr * g_t_b / (np.sqrt(cache_b) + epsilon)

return w, w2, bias
```

(b) Parameter數值

```
Parameter estimates of 2nd order polynomial with parameter `NO` only:
1st order weights:
[[0.33519698 0.20576142 0.19924521 0.1559158  0.20264756 0.20456135
0.27821887 0.58191698]],
2nd order weights:
[[-0.00941009  0.04068815 -0.02528402 -0.00614913  0.01843553 -0.07489079
-0.00897536  0.00692572]]
Bias: [2.87111462]
```

Kaggle Public Score: 8.09654



my_sol_2.csv

Complete · 12s ago · 2nd polynomial with NO only remove y > 22 lr = 0.05 epoch = 4

8.09654

□