HW1 Report

312706026 資管碩一 曾雅鈺

Q1:

1. 首先訓練集進行缺失值處理，這邊選擇使用前後數值的平均值來填補，從而可以保留一定的趨勢。

   規則為：對於每個缺失值，會找到前一個和後一個非缺失值，然後計算它們的平均值，將缺失值用平均值來填補。如果第一個值或最後一個值為缺失值，則將旁邊的值直接補進去。

```python
def fill_missing_with_avg(df, row_first_index):
    for i in range(len(df)):
        row = df.iloc[i]
        for j in range(row_first_index, len(row)+row_first_index):
            if  "#" in str(row[j]) or "*" in str(row[j]) or "x" in str(row[j]) or "X" in str(row[j]) or "A" in str(row[j]) or "WIND" in str(row[j]):  # 
                # Find the previous non-null and non-"X" value
                left_idx = j - 1
                while (left_idx >= row_first_index) and("#" in str(row[left_idx]) or "*" in str(row[left_idx]) or "x" in str(row[left_idx]) or "X" in str
                    left_idx -= 1

                # Find the next non-null and non-"X" value
                right_idx = j + 1
                while (right_idx < len(row)+row_first_index) and("#" in str(row[right_idx]) or "*" in str(row[right_idx]) or "x" in str(row[right_idx]) o
                    right_idx += 1

                # Fill missing value with the average of previous and next non-null values
                if left_idx >= row_first_index and right_idx < len(row)+row_first_index:
                    row[j] = (float(row[left_idx]) + float(row[right_idx])) / 2.0
                elif left_idx >= row_first_index:
                    row[j] = row[left_idx]
                elif right_idx < len(row)+row_first_index:
                    row[j] = row[right_idx]
                else:
                    row[j] = 0
        df.iloc[i] = row
    return df
```

圖一 補值程式碼

這邊印出某 row 為例，可以看到在左邊的圖原本有四個缺失值，後續如右圖通過計算缺失值前一個和後一個非缺失值的平均值進行填補。

0.27	0.27
0.25	0.25
0.23	0.23
0.21	0.21
0.23	0.23
0.26	0.26
0.25	0.25
0.21	0.21
0.23	0.23
0.12	0.12
#	0.135
#	0.1425
#	0.14625
#	0.148125
0.15	0.15
0.15	0.15
0.14	0.14
0.16	0.16
0.17	0.17
0.15	0.15
0.14	0.14
0.14	0.14
0.12	0.12
0.1	0.1

圖二 補值前 圖三 補值後

2. 將 Data Frame 轉成 features 為 column，時間為 row 的 dataframe

	AMB_TEMP	CH4	CO	NMHC	NO	NO2	NOx	O3	PM10	RAINFALL	\
0	11.1	2.01	0.31	0.10	11.9	11.9	13.5	21.6	38.0	0.0	
1	11.2	1.99	0.28	0.10	10.4	10.4	11.9	25.1	29.0	0.0	
2	11.4	2.00	0.28	0.08	9.8	9.8	11.2	25.6	27.0	0.0	
3	11.5	2.02	0.33	0.09	12.1	12.1	13.7	22.4	24.0	0.0	
4	11.6	2.03	0.32	0.10	12.4	12.4	13.9	21.1	29.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
5755	18.5	2.35	0.94	0.36	42.2	42.2	49.3	3.0	33.0	0.0	
5756	18.7	2.36	1.06	0.41	44.9	44.9	56.5	0.9	32.0	0.0	
5757	18.7	2.43	1.24	0.47	44.9	44.9	71.7	0.2	41.0	0.0	
5758	18.5	2.42	1.01	0.38	42.8	42.8	65.9	0.3	27.0	0.0	
5759	18.3	2.36	0.89	0.33	39.3	39.3	60.1	0.1	31.0	0.0	

	RH	SO2	THC	WD_HR	WIND_DIERC	WIND_SPEED	WS_HR	PM2.5
0	64.0	2.1	2.11	38.0	53.0	3.0	2.6	25.0
1	65.0	2.1	2.09	41.0	46.0	3.4	2.4	24.0
2	63.0	2.1	2.08	49.0	43.0	2.7	2.5	13.0
3	63.0	1.8	2.11	54.0	54.0	3.0	2.5	14.0
4	63.0	1.1	2.13	50.0	50.0	2.6	2.1	15.0
...	...	...	...	...	...	...	...	...
5755	78.0	1.6	2.71	139.0	44.0	0.3	0.3	20.0
5756	76.0	1.6	2.77	6.0	157.0	0.7	0.1	15.0
5757	78.0	1.6	2.90	143.0	98.0	0.3	0.3	22.0
5758	80.0	1.5	2.80	169.0	161.0	0.5	0.6	19.0
5759	83.0	1.4	2.69	80.0	41.0	0.3	0.4	25.0

圖四

3. 計算出每個數與 PM2.5 的相關係數

```
AMB_TEMP     -0.288179
CH4           0.455671
CO            0.612298
NMHC          0.522467
NO            0.513329
NO2           0.513329
NOx           0.473734
O3            0.105635
PM10          0.835467
RAINFALL     -0.080517
RH            0.083415
SO2           0.304948
THC           0.525400
WD_HR        -0.012395
WIND_DIERC   -0.008428
WIND_SPEED   -0.145598
WS_HR        -0.159818
PM2.5         1.000000
Name: PM2.5, dtype: float64
```

圖五 相關係數

4. 選出相關係數大於 0.5 的特徵：CO、NMHC、NO、NO2、PM10、THC 為 x_train，確保所選擇的特徵與 PM2.5 具有好的相關性

	CO	NMHC	NO	NO2	PM10	THC	PM2.5
0	0.31	0.10	11.9	11.9	38.0	2.11	25.0
1	0.28	0.10	10.4	10.4	29.0	2.09	24.0
2	0.28	0.08	9.8	9.8	27.0	2.08	13.0
3	0.33	0.09	12.1	12.1	24.0	2.11	14.0
4	0.32	0.10	12.4	12.4	29.0	2.13	15.0
...	...	...	...	...	...	...	...
5755	0.94	0.36	42.2	42.2	33.0	2.71	20.0
5756	1.06	0.41	44.9	44.9	32.0	2.77	15.0
5757	1.24	0.47	44.9	44.9	41.0	2.90	22.0
5758	1.01	0.38	42.8	42.8	27.0	2.80	19.0
5759	0.89	0.33	39.3	39.3	31.0	2.69	25.0

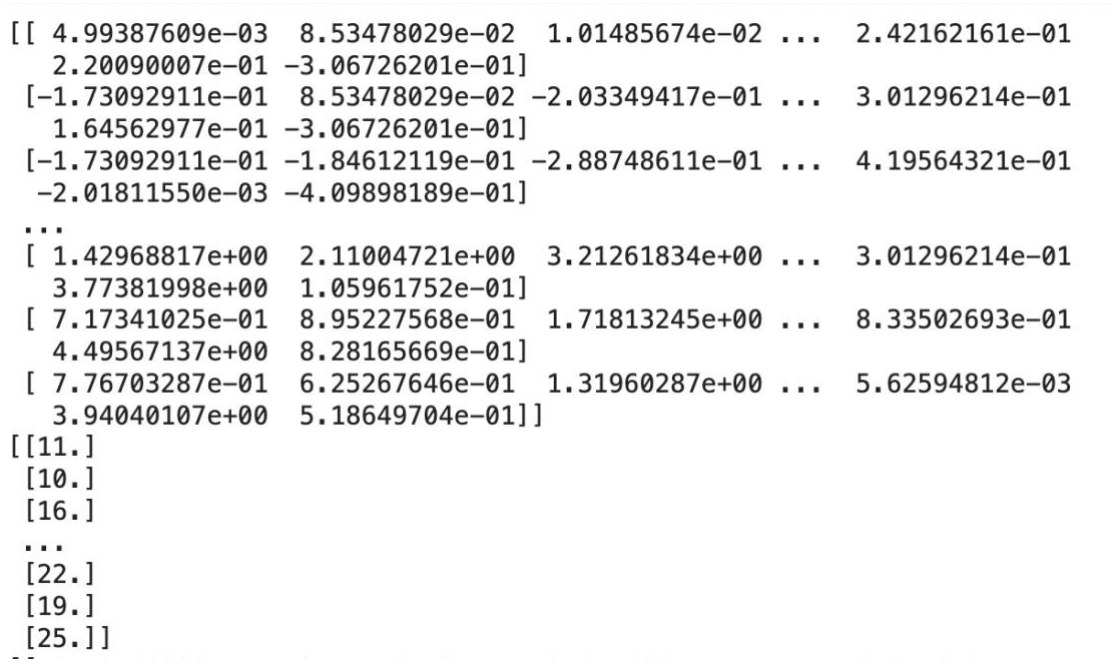
圖六

5. 因為特徵之間的分佈可能有所不同，並且具有不同的尺度，因此將數據進行 standardization，通過將每個特徵的值減去平均值，然後再除以其標準差，從而使特徵的值更符合標準常態分佈，並保留一些特徵的分佈信息。

	CO	NMHC	NO	NO2	PM10	THC	PM2.5
0	0.004994	0.085348	0.010149	0.010149	0.656101	0.109036	1.137682
1	-0.173093	0.085348	-0.203349	-0.203349	0.123894	-0.002018	1.034510
2	-0.173093	-0.184612	-0.288749	-0.288749	0.005626	-0.057545	-0.100382
3	0.123718	-0.049632	0.038615	0.038615	-0.171776	0.109036	0.002790
4	0.064356	0.085348	0.081315	0.081315	0.123894	0.220090	0.105962
...	...	...	...	...	...	...	...
5755	3.744816	3.594827	4.322808	4.322808	0.360430	3.440658	0.621822
5756	4.457164	4.269727	4.707104	4.707104	0.301296	3.773820	0.105962
5757	5.525684	5.079606	4.707104	4.707104	0.833503	4.495671	0.828166
5758	4.160352	3.864787	4.408207	4.408207	0.005626	3.940401	0.518650
5759	3.448005	3.189887	3.910045	3.910045	0.242162	3.329604	1.137682

圖七 數據標準化

6. 將 9 天的數據包在一起，y_train 從第 10 天開始，上面的陣列為 x_train，下面的陣列為 y_train

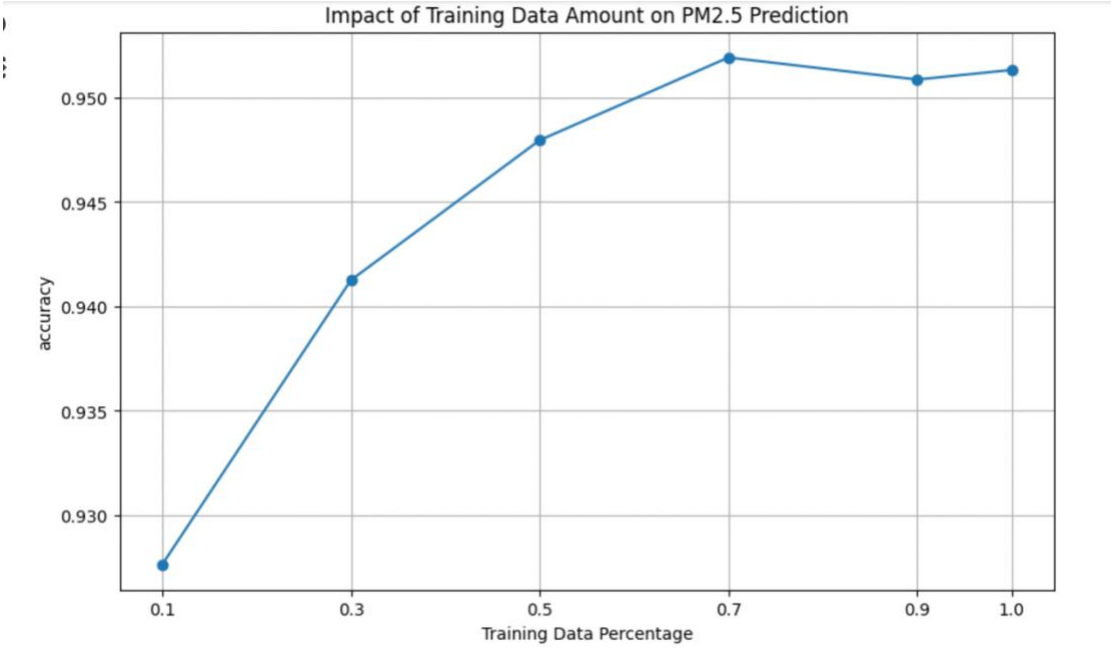


圖八

7. 測試集除了不用計算相關係數，其他步驟都相同。

Q2：

這邊分別以 10%、30%、50%、70%、90%、100%的訓練集進行模型訓練，當使用更多的訓練數據時，模型預測的 PM2.5 accuracy 不斷上升，到使用 70%數據集以後開始下降，100%有稍微回升一點。一開始上升很快，後面越來越慢，開始下降可能的原因是：



圖九

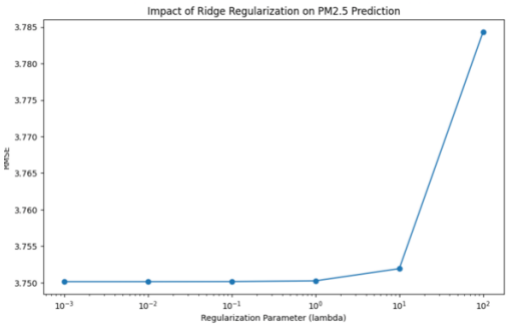
模型無法捕捉到數據中更複雜的模式或規律。因為當增加了更多的訓練數據，模型會需要更多的複雜性來得到規律，但因為這個模型過於簡單，複雜性有限，導致無法完美擬合數據，使模型的 accuracy 下降。

但當增加訓練數據量時，模型可能會達到一個平衡點，在這個平衡點，模型可以很好地泛化新數據，因此在 100%訓練集上有些許改善。

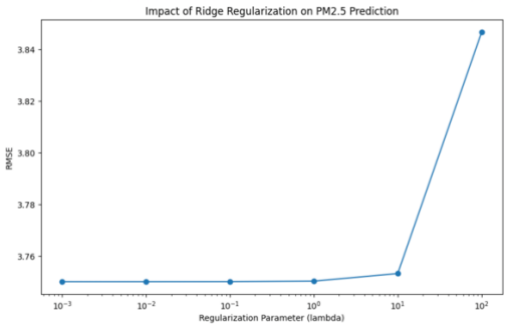
因此這邊使用 70%的訓練數據進行模型訓練，learning rate 為 0.01，epochs 為 3000，得到的 RMSE 為 3.744483529486429。

Q3：

在模型中分別添加了 L1 Lasso 正則化和 L2 Ridge 正則化進行嘗試，並使用參數 alpha 來控制他們的強度，這邊分別設定了 0.001、0.01、0.1、1、10、100 進行測試，在這邊為了判斷正則化是否有效，使用了 70%的訓練數據進行模型訓練，且其他參數也相同。以下兩張圖為它們的測試結果。左圖為使用 L1 正則化的結果，可以知道測試下來最好的結果為使用 0.001，得到的 RMSE 為 3.750143295370。右圖為使用 L2 正則化的結果，可以知道最好的結果為使用 0.001，得到的 RMSE 為 3.75014340524。



圖十 L1 正則化



圖十一 L2 正則化

與 Q2 的 3.744483529486429 相比，可以發現這兩種正則化對模型並沒有帶來太多變化。推論造成這種情況的可能原因是該數據集所使用的特徵數量沒有很多，數據集相對較小，且在模型訓練上沒有太多過擬合現象，因此加入正則化不會帶來太大改變。