

大專新進人員訓練工作報告

姓 名： 王翔禾 單位： 台化工務部自控處



報告類別： ☐ 輪班訓練 ☒ 基層實務工作訓練第（一）次報告

訓練部門： 台化工務部自控處智能專案組

起迄日期： 2022/09/14~2022/12/14

工作項目： 化一部 ARO1 廠萃取塔及 ARO3 廠去庚烷塔 AI 節能優化人機介面建置

報告項目： ARO1-C914 及 ARO3-3C8201 AI 節能優化人機介面建置

評 核 ( 評 語 )	
(2) 一 級 主 管	(1) 二 級 主 管
<p>1. 思路清晰，對於AI前後端數據介接程式開發已具備能力。</p> <p>2. 繼續加油！</p> <p>彭明：公司缺量資源 請看個人學習吸收</p> <p>王翔禾 12/14</p>	<p>1. 經由 ARO1 廠萃取塔及 ARO3 廠去庚烷塔 AI 人機介面設計，學習 AI 模型資料前後端介接程式設計。</p> <p>2. 工作及學習態度認真，學習能力佳，能自主學習 AI 建模程式。</p> <p align="right">職：曹志旭 12/14</p>
(4) 經 營 主 管	(3) 經 理 室
	

表號：P0002703 規格：A4

註：一.報告內容應包括訓練期間之 1.工作心得感想 2.所得之工作觀念及精神 3.自我檢討 4.建議意見。

二.訓練部門主管應詳細評核「訓練工作報告」內容，並批註意見。

# 目錄

一、 前言 .....	2
二、 節能優化 AI 人機介面 .....	3
2-1 系統架構 .....	3
2-2 Django 介紹 .....	5
2-3 PostgreSQL 介紹 .....	8
2-4 Apache 架設 .....	10
2-5 系統開發流程 .....	12
三、 AI 機器學習介紹 .....	15
3-1 XGboost Model 介紹 .....	15
3-2 Bagging 與 Boosting 比較 .....	16
3-3 XGBoost 優缺點比較 .....	18
3-4 C914 XGBoost AI Model 介紹 .....	18
四、 AI 深度學習介紹 .....	21
4-1 AI 神經網路介紹 .....	21
4-2 AI 神經網路訓練 .....	23
4-2 ARO3_3C8201 AI 神經網路模型介紹 .....	24
五、 工作心得 .....	25

## 一、前言

職於 9/15 到自控處報到，目前承辦工作為「化一部 ARO1 廠 C914 萃取油分離塔節能優化 AI 人機介面建置」及「化一部 ARO3 廠 3C8201 去庚烷塔節能優化 AI 人機介面建置」。

此兩案旨在協助各部針對單元製程建置可供單元操作調整之可視化工具，以網頁的方式呈現，即時提供節能操作條件建議，以維持最佳能耗操作並貼近品質管制基準。

C914 萃取油分離塔為將進料分離並從塔頂取出對二甲苯(p-Xylene、PX)、甲苯(Toluene)，塔底取出對二乙苯(PDEB)，其中製程管制基準為塔頂 PDEB<30 ppm、塔底 PX<100 ppm，3C8201 去庚烷塔則用於將 C7 以下的苯(benzene)、甲苯及非芳香烴成分從塔頂蒸餾而出，其中管制基準:塔底 Toluene<0.7 wt%、塔頂 Xylenes(A8、二甲苯)<2 wt%，此兩種單元製程於實務操作上，為避免超過管制基準，製程採取保守操作，因而有過度加熱現象，因此須建立節能優化 AI 模組，並架設 Dashboard 以互動式網頁之方式提供現場人員操作判斷依據。

本季工作心得，以上述二案為主軸，詳如後續說明。

## 二、節能優化 AI 人機介面

此兩案主要將各廠建模完成的 AI 優化模組，依據 AI 模組所需即時數據及操作推薦值以互動式網頁的方式作為人機介面呈現，用戶端可以網頁方式查看操作推薦值(於管制基準下之推薦溫度、流量等)、即時數據(如進料流量、進料組成和塔頂、塔底溫度等)，並將即時數據處理後儲存於客戶指定的廠區品管資料庫，設計網頁程式將常用的參考數據從資料庫擷取出來以動態圖表的方式呈現。同時網頁畫面也建置輸入欄位，可供使用者輸入各塔頂、塔頂底管制成分所期望之品質管制試算值，並以 AI 優化模組預測出試算的溫度流量等，以即時且直觀的方式輔助現場人員操作判斷。

### 2-1 系統架構

以下為本案工程開發時所用到的作業系統、資料庫、使用的程式語言，如表 2-1 所示。系統架構圖如圖 2-1 所示，使用 Python 語言之第三方框架「Django」開發連接企業網路的網頁後端程式及讀取 PI 資料庫中的數據，接續以 Python 語言設計將下載之數據轉存至化一部專用的 PostgreSQL 品管資料庫，後續再以 JavaScript 等前端語言，將資料庫數據及模型預測結果，以網頁之形式架設於 Apache Server 中，以供使用者查看。

作業系統	Windows 10
資料庫	PostgreSQL、PI System
程式語言、框架	Python、JavaScript、Django
網頁伺服器	Apache Server

表 2-1、系統環境

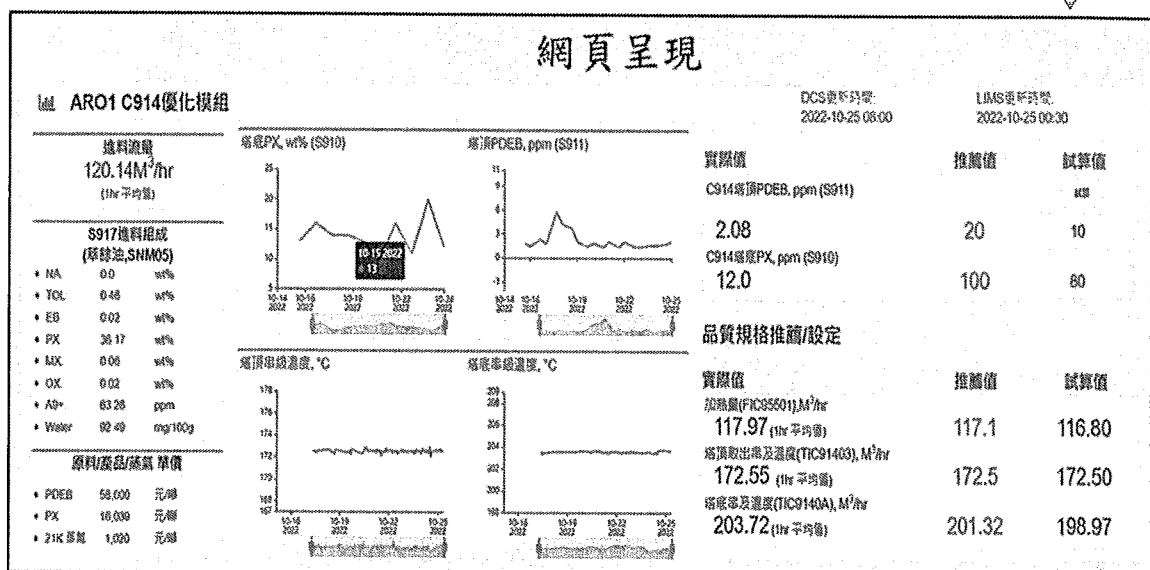
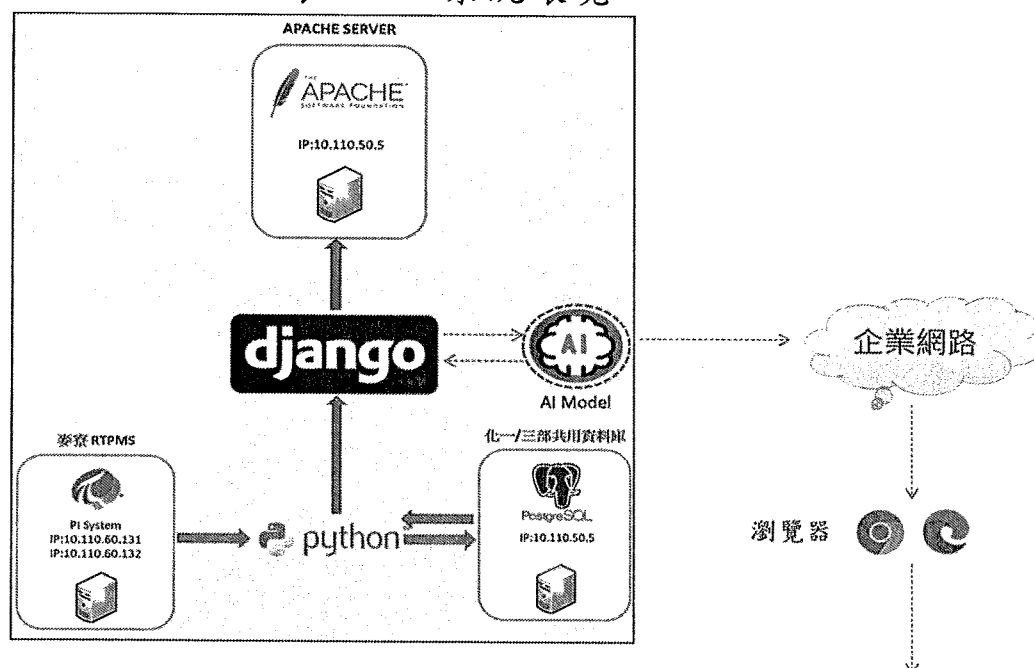


圖 2-1、系統架構圖

## 2-2 Django 介紹

Django 為由 Python 編寫的一個開放原始程式碼的 Web 應用框架，由於其強大的內建功能，使開發人員使用 Django 做為網站開發框架時，可用少量且簡潔的代碼，輕鬆地完成一個正式網站所需要的大部分內容，並進一步開發出全功能的 Web 服務，Django 本身基於 MVC 模型，即 Model(模型) + View(視圖) + Template(模板) 設計模式，MVC 模式使後續對程式的修改和擴展簡化，並且使程式代碼的可讀性大大增強，MVC 分別特點如下。

- Model：主要負責網頁中與資料有關的直接處理，定義物件關聯對映(ORM，以類的形式創建資料表)，有對資料庫直接存取的權力。
- View：主要負責處理 Model 和 Template(網頁) 之間的邏輯，並做出回應。
- Template：負責資料的顯示，即為使用者所看到的頁面

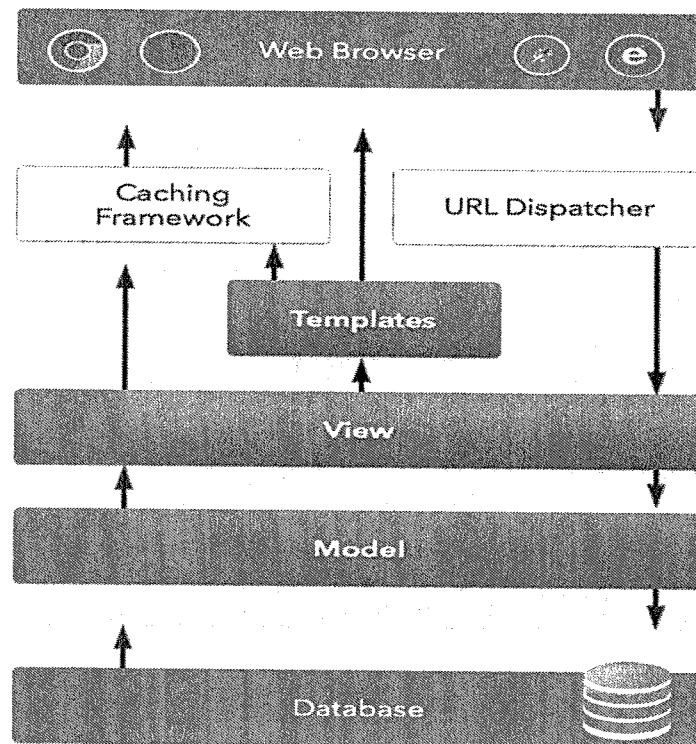


圖 2-2、Django MVC 模型流程圖

C: > ARO1\_C914 > ARO1\_C914 > app > models.py > LIMS\_data

```

1  from django.db import models
2  # import pandas
3  # import numpy as np
4  每個類對應資料庫中的一張表，框選範圍對應 廠區資料庫之DCS data 表
5  # Create your models here.
6  class DCS_data(models.Model):
7      datetime = models.DateTimeField(blank = False, null = False)
8      tag = models.CharField(blank = False, null=False, max_length = 255)
9      value = models.FloatField(blank = False, null = False)
10     create_at = models.DateTimeField(auto_now_add = True)
11
12     class Meta:
13         db_table = "DCS_data"
14
15     class LIMS_data(models.Model):
16         datetime = models.DateTimeField(blank = False, null = False)
17         tag = models.CharField(blank = False, null=False, max_length = 255)
18         value = models.FloatField(blank = False, null = False)
19         create_at = models.DateTimeField(auto_now_add = True)
20         class Meta:
21             db_table = "LIMS_data"
22
23     class Recommend(models.Model):
24         datetime = models.DateTimeField(blank = False, null = False)
25         tag = models.CharField(blank = False, null=False, max_length = 255)
26         value = models.FloatField(blank = False, null = False)
27         create_at = models.DateTimeField(auto_now_add = True)
28
29         class Meta:
30             db_table = "Recommend"
31
  
```

圖 2-3、Django Model 程式示意圖

```

1 import sys
2
3 from django.shortcuts import render
4
5 # Create your views here.
6 from django.http import HttpResponse, JsonResponse
7 from datetime import datetime, timedelta
8 from .models import DCS_data, LIMS_data, Recommend
9 from django.db.models import Sum
10 import pandas as pd
11 from model.C914_Recommend import*
12 # from web_project.task import scheduler
13
14 抓取資料庫中的tag，經處理後交由前端頁面顯示
15 def ARO1_c914(request):
16
17     sqlite_dict = dict()
18
19     query = DCS_data.objects.order_by('tag', '-datetime').distinct('tag')
20     for p in query:
21         sqlite_dict.update({p.tag:p.value})
22
23     query = LIMS_data.objects.order_by('tag', '-datetime').distinct('tag')
24     for p in query:
25         sqlite_dict.update({p.tag:p.value})
26
27     query = Recommend.objects.order_by('tag', '-datetime').distinct('tag')
28     for p in query:
29         sqlite_dict.update({p.tag:p.value})
30
31 >     sqlite_dict.update({...
32
33 >     sqlite_dict.update({...
34
35 >
36 >
37 >
38 >
39 >
40
41     print(sqlite_dict)
42     date_time = datetime.now().replace(minute=0, second=0, microsecond=0)
43     start_time = (date_time + timedelta(days=-14)).strftime("%Y-%m-%d %H:%M:%S")
44     end_time = (date_time).strftime("%Y-%m-%d %H:%M:%S")
45
46
47 >     if request.method == 'POST' and request.POST['action'] == 'main':...
48
49 >     if request.method == 'POST' and request.POST['action'] == 'calculate':...
50
51 >
52 >
53 >
54 >
55 >
56 >
57 >
58 >
59 >
60 >
61 >
62 >
63 >
64 >
65 >
66 >
67 >
68 >
69 >
70 >
71 >
72 >
73 >
74 >
75 >
76 >
77 >
78 >
79 >
80 >
81 >
82 >
83 >
84 >
85 >
86 >
87 >
88 >
89 >
90 >
91 >
92 >
93 >
94 >
95 >
96 >
97 >
98 >
99 >
100 >
101 >
102 >
103 >
104 >
105 >
106 >
107 >
108 >
109 >
110 >
111 >
112 >
113 >
114 >
115 >
116 >
117 >
118 >
119 >
120 >
121 >
122 >
123 >
124 >
125 >
126 >
127 >
128 >
129 >
130 >
131 >
132 >
133 >
134 >
135 >
136 >
137 >
138 >
139 >
140 >
141 >
142 >
143 >
144 >
145 >
146 >
147 >
148 >
149 >
150 >
151 >
152 >
153 >
154 >
155 >
156 >
157 >
158 >
159 >
160 >
161 >
162 >
163 >
164 >
165 >
166 >
167 >
168 >
169 >
170 >
171 >
172 >
173 >
174 >
175 >
176 >
177 >
178 >
179 >
180 >
181 >
182 >
183 >
184 >
185 >
186 >
187 >
188 >
189 >
190 >
191 >
192 >
193 >
194 >
195 >
196 >
197 >
198 >
199 >
200 >
201 >
202 >
203 >
204 >
205 >
206 >
207 >
208 >
209 >
210 >
211 >
212 >
213 >
214 >
215 >
216 >
217 >
218 >
219 >
220 >
221 >
222 >
223 >
224 >
225 >
226 >
227 >
228 >
229 >
230 >
231 >
232 >
233 >
234 >
235 >
236 >
237 >
238 >
239 >
240 >
241 >
242 >
243 >
244 >
245 >
246 >
247 >
248 >
249 >
250 >
251 >
252 >
253 >
254 >
255 >
256 >
257 >
258 >
259 >
260 >
261 >
262 >
263 >
264 >
265 >
266 >
267 >
268 >
269 >
270 >
271 >
272 >
273 >
274 >
275 >
276 >
277 >
278 >
279 >
280 >
281 >
282 >
283 >
284 >
285 >
286 >
287 >
288 >
289 >
290 >
291 >
292 >
293 >
294 >
295 >
296 >
297 >
298 >
299 >
300 >
301 >
302 >
303 >
304 >
305 >
306 >
307 >
308 >
309 >
310 >
311 >
312 >
313 >
314 >
315 >
316 >
317 >
318 >
319 >
320 >
321 >
322 >
323 >
324 >
325 >
326 >
327 >
328 >
329 >
330 >
331 >
332 >
333 >
334 >
335 >
336 >
337 >
338 >
339 >
340 >
341 >
342 >
343 >
344 >
345 >
346 >
347 >
348 >
349 >
350 >
351 >
352 >
353 >
354 >
355 >
356 >
357 >
358 >
359 >
360 >
361 >
362 >
363 >
364 >
365 >
366 >
367 >
368 >
369 >
370 >
371 >
372 >
373 >
374 >
375 >
376 >
377 >
378 >
379 >
380 >
381 >
382 >
383 >
384 >
385 >
386 >
387 >
388 >
389 >
390 >
391 >
392 >
393 >
394 >
395 >
396 >
397 >
398 >
399 >
400 >
401 >
402 >
403 >
404 >
405 >
406 >
407 >
408 >
409 >
410 >
411 >
412 >
413 >
414 >
415 >
416 >
417 >
418 >
419 >
420 >
421 >
422 >
423 >
424 >
425 >
426 >
427 >
428 >
429 >
430 >
431 >
432 >
433 >
434 >
435 >
436 >
437 >
438 >
439 >
440 >
441 >
442 >
443 >
444 >
445 >
446 >
447 >
448 >
449 >
450 >
451 >
452 >
453 >
454 >
455 >
456 >
457 >
458 >
459 >
460 >
461 >
462 >
463 >
464 >
465 >
466 >
467 >
468 >
469 >
470 >
471 >
472 >
473 >
474 >
475 >
476 >
477 >
478 >
479 >
480 >
481 >
482 >
483 >
484 >
485 >
486 >
487 >
488 >
489 >
490 >
491 >
492 >
493 >
494 >
495 >
496 >
497 >
498 >
499 >
500 >
501 >
502 >
503 >
504 >
505 >
506 >
507 >
508 >
509 >
510 >
511 >
512 >
513 >
514 >
515 >
516 >
517 >
518 >
519 >
520 >
521 >
522 >
523 >
524 >
525 >
526 >
527 >
528 >
529 >
530 >
531 >
532 >
533 >
534 >
535 >
536 >
537 >
538 >
539 >
540 >
541 >
542 >
543 >
544 >
545 >
546 >
547 >
548 >
549 >
550 >
551 >
552 >
553 >
554 >
555 >
556 >
557 >
558 >
559 >
560 >
561 >
562 >
563 >
564 >
565 >
566 >
567 >
568 >
569 >
570 >
571 >
572 >
573 >
574 >
575 >
576 >
577 >
578 >
579 >
580 >
581 >
582 >
583 >
584 >
585 >
586 >
587 >
588 >
589 >
590 >
591 >
592 >
593 >
594 >
595 >
596 >
597 >
598 >
599 >
600 >
601 >
602 >
603 >
604 >
605 >
606 >
607 >
608 >
609 >
610 >
611 >
612 >
613 >
614 >
615 >
616 >
617 >
618 >
619 >
620 >
621 >
622 >
623 >
624 >
625 >
626 >
627 >
628 >
629 >
630 >
631 >
632 >
633 >
634 >
635 >
636 >
637 >
638 >
639 >
640 >
641 >
642 >
643 >
644 >
645 >
646 >
647 >
648 >
649 >
650 >
651 >
652 >
653 >
654 >
655 >
656 >
657 >
658 >
659 >
660 >
661 >
662 >
663 >
664 >
665 >
666 >
667 >
668 >
669 >
670 >
671 >
672 >
673 >
674 >
675 >
676 >
677 >
678 >
679 >
680 >
681 >
682 >
683 >
684 >
685 >
686 >
687 >
688 >
689 >
690 >
691 >
692 >
693 >
694 >
695 >
696 >
697 >
698 >
699 >
700 >
701 >
702 >
703 >
704 >
705 >
706 >
707 >
708 >
709 >
710 >
711 >
712 >
713 >
714 >
715 >
716 >
717 >
718 >
719 >
720 >
721 >
722 >
723 >
724 >
725 >
726 >
727 >
728 >
729 >
730 >
731 >
732 >
733 >
734 >
735 >
736 >
737 >
738 >
739 >
740 >
741 >
742 >
743 >
744 >
745 >
746 >
747 >
748 >
749 >
750 >
751 >
752 >
753 >
754 >
755 >
756 >
757 >
758 >
759 >
760 >
761 >
762 >
763 >
764 >
765 >
766 >
767 >
768 >
769 >
770 >
771 >
772 >
773 >
774 >
775 >
776 >
777 >
778 >
779 >
780 >
781 >
782 >
783 >
784 >
785 >
786 >
787 >
788 >
789 >
790 >
791 >
792 >
793 >
794 >
795 >
796 >
797 >
798 >
799 >
800 >
801 >
802 >
803 >
804 >
805 >
806 >
807 >
808 >
809 >
810 >
811 >
812 >
813 >
814 >
815 >
816 >
817 >
818 >
819 >
820 >
821 >
822 >
823 >
824 >
825 >
826 >
827 >
828 >
829 >
830 >
831 >
832 >
833 >
834 >
835 >
836 >
837 >
838 >
839 >
840 >
841 >
842 >
843 >
844 >
845 >
846 >
847 >
848 >
849 >
850 >
851 >
852 >
853 >
854 >
855 >
856 >
857 >
858 >
859 >
860 >
861 >
862 >
863 >
864 >
865 >
866 >
867 >
868 >
869 >
870 >
871 >
872 >
873 >
874 >
875 >
876 >
877 >
878 >
879 >
880 >
881 >
882 >
883 >
884 >
885 >
886 >
887 >
888 >
889 >
890 >
891 >
892 >
893 >
894 >
895 >
896 >
897 >
898 >
899 >
900 >
901 >
902 >
903 >
904 >
905 >
906 >
907 >
908 >
909 >
910 >
911 >
912 >
913 >
914 >
915 >
916 >
917 >
918 >
919 >
920 >
921 >
922 >
923 >
924 >
925 >
926 >
927 >
928 >
929 >
930 >
931 >
932 >
933 >
934 >
935 >
936 >
937 >
938 >
939 >
940 >
941 >
942 >
943 >
944 >
945 >
946 >
947 >
948 >
949 >
950 >
951 >
952 >
953 >
954 >
955 >
956 >
957 >
958 >
959 >
960 >
961 >
962 >
963 >
964 >
965 >
966 >
967 >
968 >
969 >
970 >
971 >
972 >
973 >
974 >
975 >
976 >
977 >
978 >
979 >
980 >
981 >
982 >
983 >
984 >
985 >
986 >
987 >
988 >
989 >
990 >
991 >
992 >
993 >
994 >
995 >
996 >
997 >
998 >
999 >
1000 >

```

圖 2-4、Django View 程式示意圖

對於開發者更方便的是，Django 有其特有的自帶模板語言 DTL(Django Template Language)，意味著不需要再使用前端框架撰寫頁面，可大幅縮短網頁開發時程。

```

<div class="left aligned content">
  <ul class="ts list" style="color:#4D5156;text-align: left;">
    <li>
      <div class='five wide column'>NA:</div>
      <div class="five wide column">{{ARO1_LIMS_s917_744_non_ARO|Round:2}}</div>
      <div class="three wide column">wt%</div>
    </li>
    <li>
      <div class='five wide column'>TOL:</div>
      <div class="five wide column">{{ARO1_LIMS_s917_744_Toluene|Round:2}}</div>
      <div class="three wide column">wt%</div>
    </li>
  </ul>
</div>

```

圖 2-5、Django Template (Template Language) 示意圖



## 2-3 PostgreSQL 介紹

本案有用到 PostgreSQL 資料庫，它是一套功能強大、開放原始碼的物件關聯資料庫系統，歷史悠久，在系統可靠性、資料完整性和正確性上均獲得極佳評價，其也號稱為最先進資料庫系統，如多版本並行控制(Multi-Version Concurrency Control, MVCC)，時間點資料還原、表格空間 tablespaces)、非同步資料複製(Asynchronous Replication)、巢狀交易(儲存點)、線上/熱備份、複雜的查詢規劃器/優化、支援容錯能力的優先寫入日誌檔等高階功能，此外 PostgreSQL 可以在各種常見作業系統上執行，例如 Linux、Unix、Windows 等，也由於上述特性讓其可稱得上是一套企業級資料庫系統，現今也有許多跨國大企業公司使用其作為主要資料庫。

於上述兩案中，AI 人機介面主要將 PI 資料庫中的各資料點定期取出，並整理放入 PostgreSQL 資料庫當中，而後將從 PostgreSQL 資料庫取出資料呈現於化一部 AI Dashboard 中。

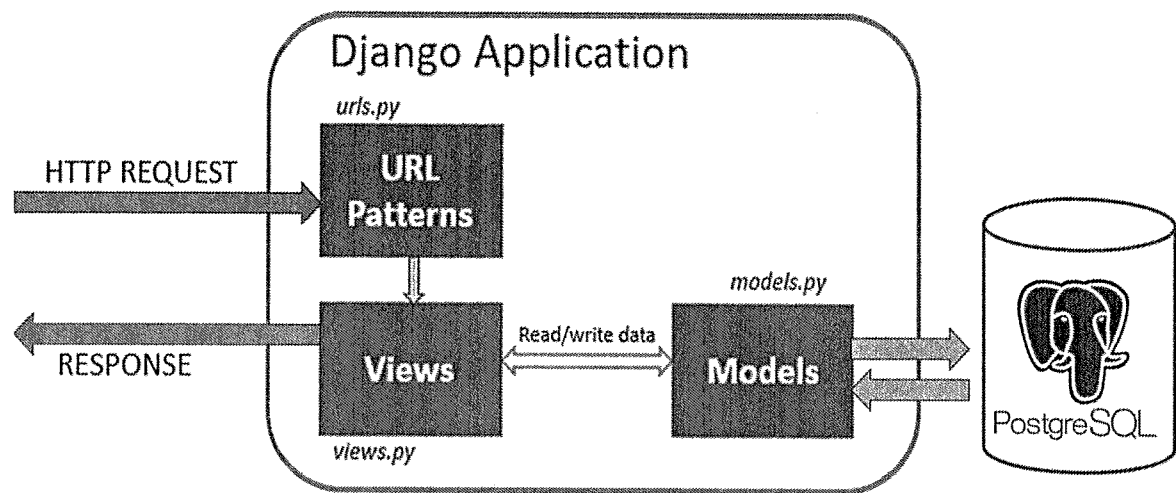


圖 2-6、PostgreSQL 與 Django 應用交互關係圖

```

80  DATABASES = {
81  ✓    'default': {
82        'ENGINE': 'django.db.backends.postgresql',
83        'NAME': 'AR01_C914',
84        'USER': 'fcfc_eng',
85        'PASSWORD': 'fcfc_eng',
86        'HOST': '10.110.50.4',
87        'PORT': '5432',
88    }
89
90  }

```

圖 2-7、Django 連線 PostgreSQL 資料庫設定

## 2-4 Apache 架設

AI 節能優化人機介面建置是以 Python Django 做為後端框架設計，要佈署到 Web 伺服器後才能運用。Apache 是 Apache 軟體基金會的一個開放原始碼的 Web 伺服器軟體，可以在大多數電腦作業系統中運行。由於其跨平台和安全性，被廣泛使用，是最流行的 Web 伺服器軟體之一，可通過簡單的 API 擴充，將 Perl 與 Python 等程式語言編譯到伺服器中。

Apache 源於 NCSAhttpd 伺服器，經過多次修改，成為世界上最流行的 Web 伺服器軟體之一。Apache 取自 “a patched server” 的讀音，意思是充滿補丁的伺服器，因為它是自由軟體，所以不斷有人來為它開發新的功能、新的特性、修改原來的缺陷。Apache 的特點是簡單、速度快、性能穩定，並可做代理伺服器來使用。

Apache 建置版本需與 Python 軟體版本一致，本次安裝需求為：Apache 2.4 vc16 版本、Python 3.7 版本，mod\_wsgi 檔案 mod\_wsgi-4.7.1+ap24vc16-cp37-cp37m-win\_amd64.whl

本案將 Apache 軟體安裝至廠區電腦中(IP:10.110.50.5)，並透過 Python 後端網路框架 Django 進行串接，如圖 2-8 所示，使用者於瀏覽器輸入網址後，即可看到該操作單元相關資訊(包含進料組成、趨勢圖、AI 推薦/試算值等)，於輸入框

框輸入欲試算之品質管制值後，網頁將輸入值傳遞給後端框架，並連接至 AI 模組運算後將試算結果傳遞至前端頁面中，網頁呈現如圖 2-9 所示。

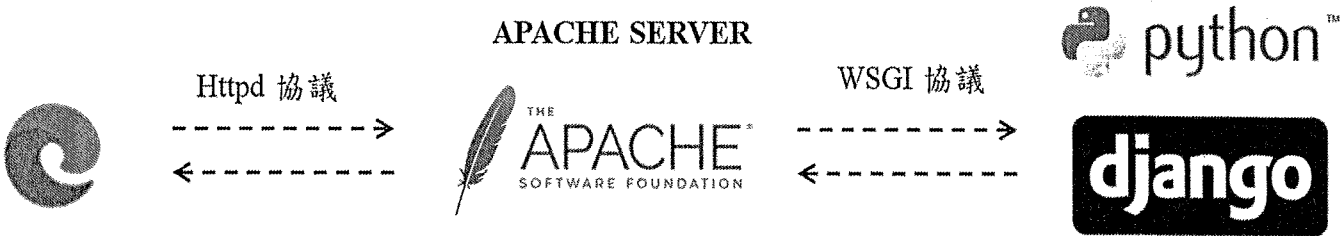


圖 2-8、Apache Web Server 運作流程圖

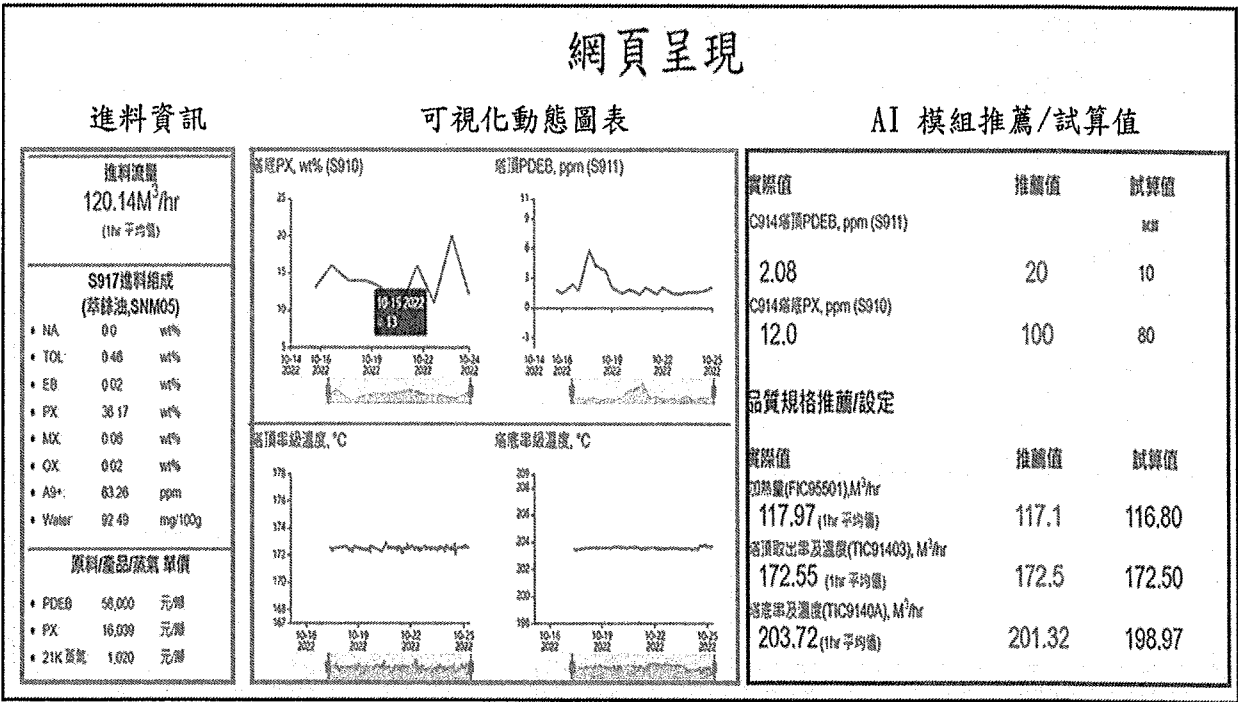


圖 2-9、AI 人機介面互動式網頁呈現

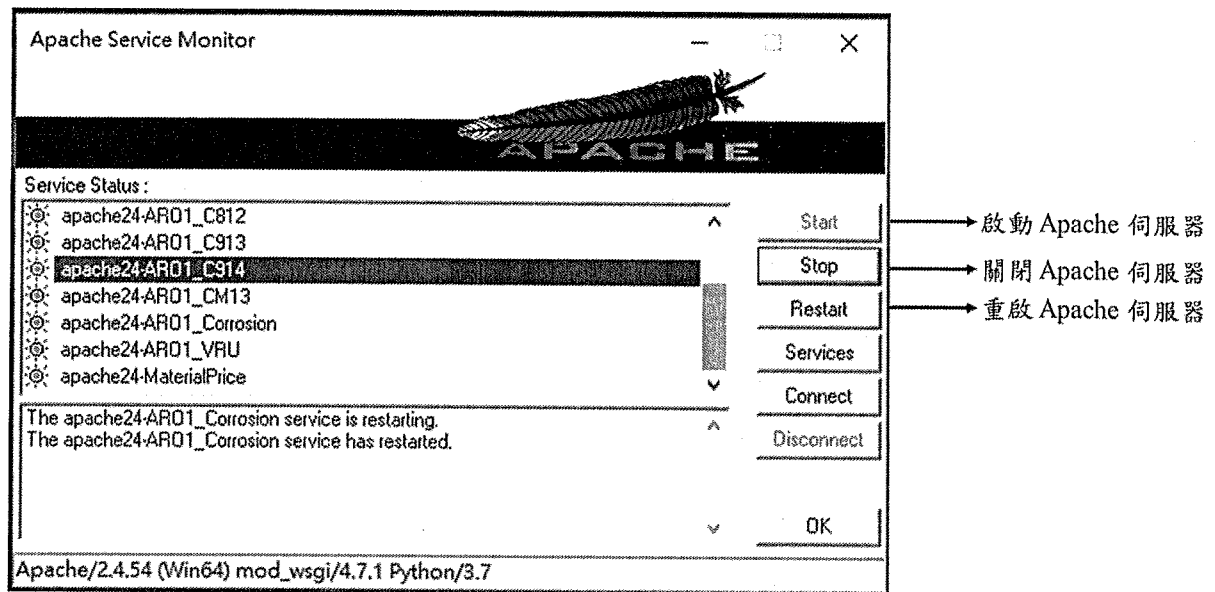


圖 2-10、Apache 伺服器操作介面說明

## 2-5 系統開發流程

程式設計流程如圖 2-11 所示，於瀏覽器輸入網址後，前端 JavaScript 程式會發送資料請求傳遞至後端，後端則透過 Django、Python 開發之程式讀取 PostgreSQL 資料庫的資料，接著將讀取資料進行整理後，將資料以 JSON 型態回傳至前端，最後經 JavaScript 處理完後呈現使用者畫面，而後端 Python 程式也會定期向 PI Server 抓取 DCS/LIMS 數據，設定排程為每小時抓取最新資料，經處理後轉存至 PostgreSQL 資料庫以確保前端呈現之數據均在最新狀態，資料抓取程式畫面如圖 2-12 所示。

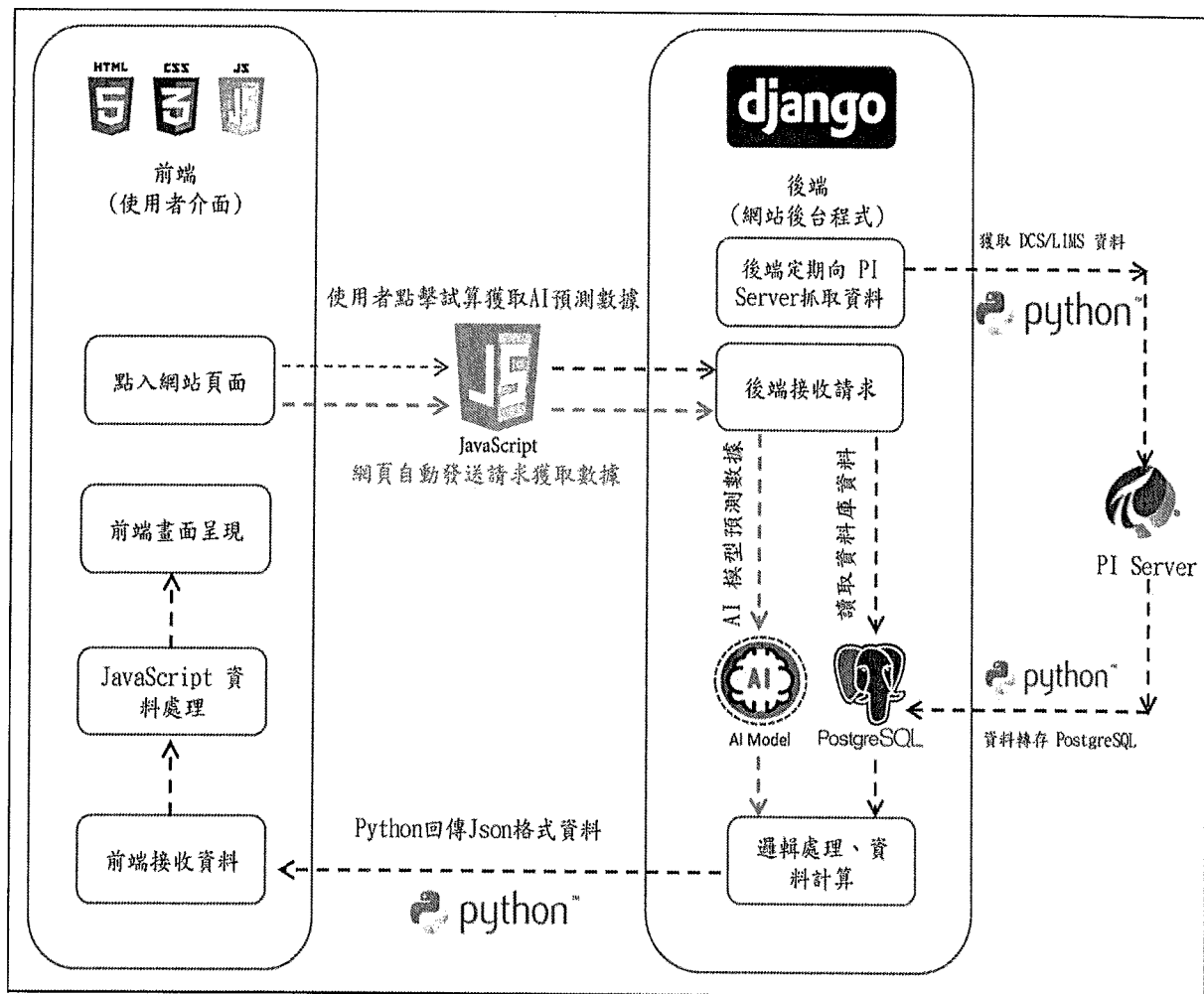


圖 2-11、程式設計流程

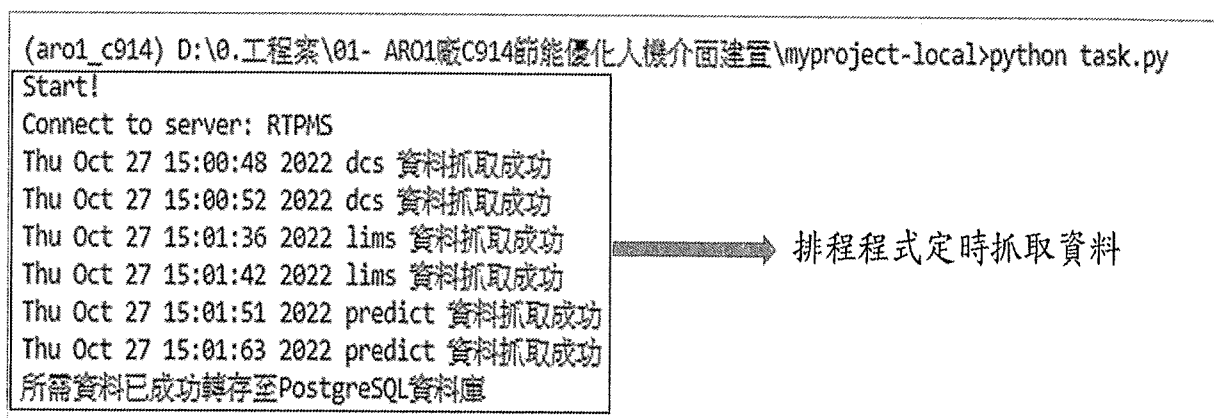


圖 2-12、定時資料抓取式畫面

DCS 資料彙總到 PostgreSQL 資料庫後可藉由 PostgreSQL 自帶的 UI 介面 pgAdmin 查看資料庫內容，如圖 2-13、2-14 所示，化一部開發人員可隨時於資料庫中查看所需資料。

pgAdmin File Object Tools Help

Browser

- > Foreign Tables
- > Functions
- > Materialized Views
- > Procedures
- > Sequences
- > Tables (15)
  - > DCS\_data
  - > LIMS\_data
  - > Recommend
  - > auth\_group
  - > auth\_group\_permissions
  - > auth\_permission
  - > auth\_user
  - > auth\_user\_groups
  - > auth\_user\_user\_permissions
  - > django\_admin\_log
  - > django\_apscheduler\_dje
  - > django\_apscheduler\_dje
  - > django\_content\_type
  - > django\_migrations
  - > django\_session
- > Trigger Functions
- > Types
- > Views

Dashboard Properties SQL Statistics Dependencies Dependents public.LIMS\_data/ARO1\_C914/postgres@ARO

public.LIMS\_data/ARO1\_C914/postgres@ARO

Query Editor Query History

```
1 SELECT * FROM public."LIMS_data"
2 ORDER BY id ASC
```

數據顯示

Id	datetime	tag	value	create_at
[PK] integer	timestamp with time zone	character varying (255)	double precision	timestamp with time zone
1	2022-10-12 00:30:00+08	ARO1_LIMS_s917_744_non_A...	0.005499999970197678	2022-10-12 14:08:32.427454+08
2	2022-10-12 00:30:00+08	ARO1_LIMS_s917_744_Toluene	0.23669999837875366	2022-10-12 14:08:32.458858+08
3	2022-10-12 00:30:00+08	ARO1_LIMS_s917_744_Ethylbz	0.019600000232458115	2022-10-12 14:08:32.490457+08
4	2022-10-12 00:30:00+08	ARO1_LIMS_s917_744_p_Xyle...	34.908199310302734	2022-10-12 14:08:32.516812+08
5	2022-10-12 00:30:00+08	ARO1_LIMS_s917_744_m_Xyle...	0.06350000202655792	2022-10-12 14:08:32.532465+08
6	2022-10-12 00:30:00+08	ARO1_LIMS_s917_744_o_Xyle...	0.01590000092983246	2022-10-12 14:08:32.563659+08
7	2022-10-12 00:30:00+08	ARO1_LIMS_s917_744_C9	64.74800109863281	2022-10-12 14:08:32.594901+08
8	2022-10-12 00:30:00+08	ARO1_LIMS_s917_WATER	87.05000305175781	2022-10-12 14:08:32.610523+08
9	2022-10-10 16:00:00+08	ARO1_LIMS_s911_798_p_DEB	1.5226999521255493	2022-10-12 14:08:32.641765+08
10	2022-10-11 00:30:00+08	ARO1_LIMS_s911_798_p_DEB	2.3090999126434326	2022-10-12 14:08:32.673007+08
11	2022-10-11 16:00:00+08	ARO1_LIMS_s911_798_p_DEB	1.5693000555038452	2022-10-12 14:08:32.688672+08
12	2022-10-12 00:30:00+08	ARO1_LIMS_s911_798_p_DEB	2.0992000102996826	2022-10-12 14:08:32.719916+08

各資料表

圖 2-13、PostgreSQL 資料表示意圖

Data Output Explain Messages Notifications

Id	next_run_time	job_state
[PK] character varying (255)	timestamp with time zone	bytea
1 web_project.task.renew_3C8201	2022-11-01 11:06:00+08	[binary data]

下次排程執行時間

圖 2-14、資料庫定時資料抓排程預計時間

## 三、AI 機器學習介紹

### 3-1 XGboost Model 介紹

C914 萃取油分離塔節能優化 AI Model 利用經典的 XGboost 演算法進行回歸數值預測，XGboost 全名為 eXtreme Gradient Boosting，是目前競賽中最常見到的算法，同時也是多數得獎者所使用的模型。此機器學習模型是由華盛頓大學博士生陳天奇所提出來的，它是以 Boosting 為基礎下去實作，並結合 Bagging 的優點作為實現，Bagging 及 Boosting 將於 3-2 小節詳述。

XGboost 保有 Gradient Boosting 的做法，每一棵樹是互相關聯的，目標是希望後面生成的樹能夠修正前面一棵樹判斷錯誤的地方。此外 XGboost 是採用特徵隨機採樣的技巧，和隨機森林一樣在生成每一棵樹的時候隨機抽取特徵，因此在每棵樹的生成中並不會每一次都拿全部的特徵參與決策。

此外為了讓模型不過於複雜，XGboost 在目標函數添加了標準項正則化。因為模型在訓練時為了擬合訓練資料，會產生很多高次項的函數，但反而容易被雜訊干擾導致過度擬合。因此 L1/L2 正則化之目的是讓損失函數更佳平滑，且抗雜訊干擾能力更大。最後 XGboost 還用到了一階導數和二



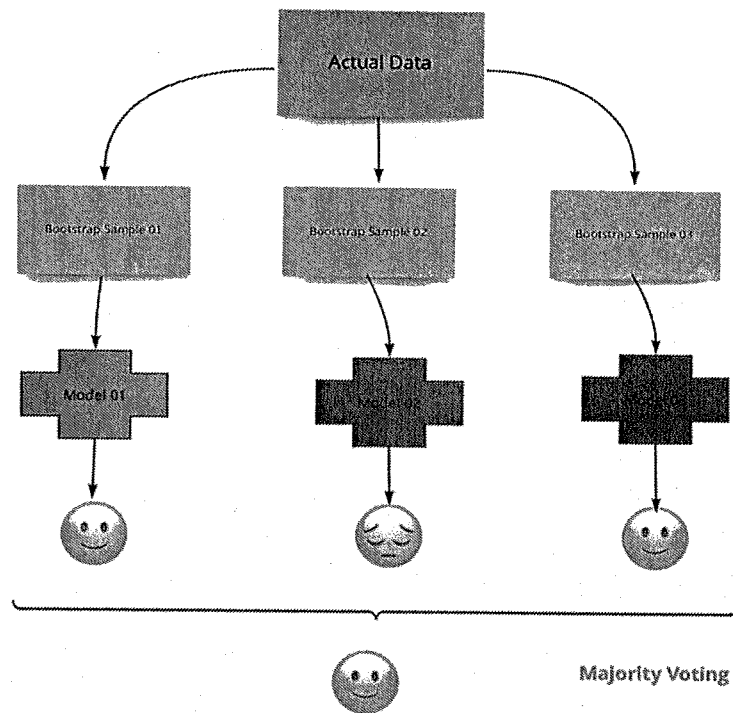
階導數來生成下一棵樹。其中 Gradient 就是所謂的一階導數，而 Hessian 即為二階導數。

### 3-2 Bagging 與 Boosting 比較

在機器學習中須了解 Bagging 與 Boosting 兩者間的差異，Bagging 透過隨機抽樣的方式生成每一棵樹，最重要的是每棵樹彼此獨立並無關聯。上一小節提到的隨機森林就是 Bagging 的實例。另外 Boosting 則是透過序列的方式生成樹，後面所生成的樹會與前一棵樹相關。本章所提及的 XGBoost 就是 Boosting 方法的其中一種實例。正是每棵樹的生成都改善了上一棵樹學習不好的地方，因此 Boosting 的模型通常會比 Bagging 還來的精準，而決策樹通常為一棵複雜的樹，而在 Boosting 是產生非常多棵的樹，Boosting 希望新的樹可以針對舊的樹預測不太好的部分做改善。最終再把所有樹的值合在一起才能當最後的預測輸出。

1. Bagging 透過抽樣的方式生成樹，每棵樹彼此獨立
2. Boosting 透過序列的方式生成樹，後面生成的樹會與前一棵樹相關

### Bagging Ensemble Method



### Boosting Ensemble Method

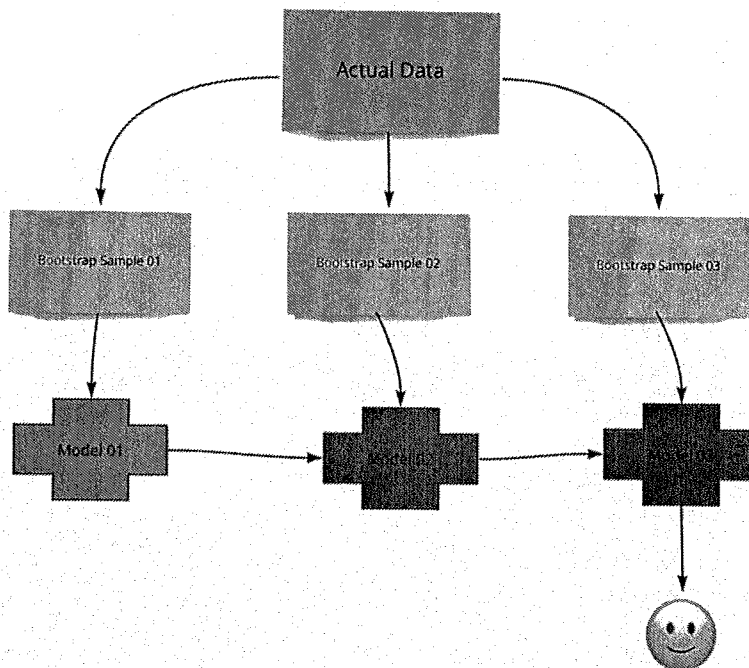


圖 3-1、Bagging 與 Boosting 比較示意圖

### 3-3 XGBoost 優缺點比較

XGBoost 除了可以做分類也能進行迴歸連續性數值的預測，而且效果通常都不差。並透過 Boosting 技巧將許多弱決策樹集成在一起形成一個強的預測模型，而其優缺點比較如下。

#### 1.優點

- 利用了二階梯度來對節點進行劃分
- 利用局部近似算法對分裂節點進行優化
- 在損失函數中加入了 L1/L2 項，控制模型的複雜度
- 提供 GPU 平行化運算

#### 2.缺點

- 預排序過程的空間複雜度過高，不僅需要存儲特徵值，還需要存儲特徵對應樣本的梯度統計值的索引，等會消耗較端記憶體

### 3-4 C914 XGBoost AI Model 介紹

C914 AI mode 以經過篩選後之六個特徵變數帶入 XGBoost 模型中進行深度學習運算，並預測出加熱量、塔頂取出串級溫度、塔底串級溫度等三個目標值，其流程圖如下圖 3-2 所示，其中塔頂對二乙苯及塔底對二甲苯濃度為 C914

萃取塔之管制標的，製程人員可依模型預測得到特定標濃度下之製程操作值。

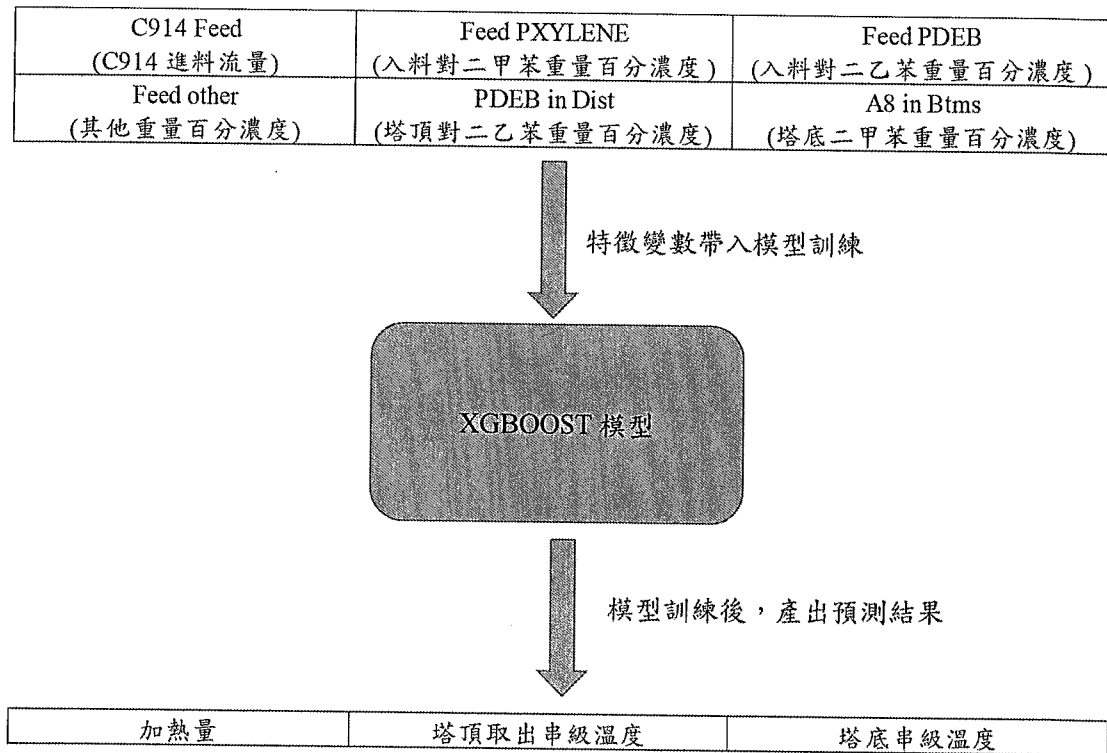


圖 3-2、C914 AI mode 流程示意圖

接著進一步將化一部所提供之訓練資料共 3550 筆，劃分訓練集及測試集比例為 8:2，使用化一部建模時使用之六個特徵變數分別帶入 XGBoost 及現場常用之 Linear-regression 兩種 model 訓練，並以 R-Square、MSE、MAPE 等三種指標，判斷其訓練結果，從圖 3-3 可看出在訓練相同資料的情況下 XGBoost 之訓練結果三種指標均優於 Linear-regression，且 R-Square 可達到 0.99，足以見到 XGBoost 之預測模型預測結果較 Linear-regression 好。

XGBoost-Predict VALUE	MSE	MAPE	R2	Linear-Regression-Predict Value	MSE	MAPE	R2
E955換熱器加熱量	0.000048	0.063606	0.999958	E955換熱器加熱量	0.036575	1.881222	0.968149
C914塔頂串級溫度	0.002802	0.023211	0.999922	C914塔頂串級溫度	3.202031	0.867864	0.910717
C914塔底串級溫度	0.000052	0.002468	0.999988	C914塔底串級溫度	0.473605	0.251846	0.893853
AVG	1.40328	0.023544	0.999967	AVG	155486.43689	1.190996	0.940660

圖 3-3、XGBoost 與 Linear-Regression 結果比較

於 3-2 小節有提到 XGBoost 訓練過程中會產生多棵的決策樹，新的樹可以針對舊的樹預測不足的部分做改善，每棵決策樹將影響下一棵決策樹之複雜度，藉以改善上一輪訓練之缺陷，持續優化模型，也是藉由此方式讓 XGBoost 仍舊在訓練樣本有限、訓練時間短、等場景下具有獨特的優勢。相比深度神經網路，XGBoost 能夠更好地處理表格資料，並具有更強的可解釋性，另外具有易於調整參數、輸入資料不變性等優勢。

## 四、AI 深度學習介紹

### 4-1 AI 神經網路介紹

ARO3\_3C8201 去庚烷塔 AI Model 利用近年崛起的人工神經網路(Artificial Neural Network)進行回歸數值預測，人工神經網路是一個資訊處理系統，它是由許多人工神經元(artificial neuron) 所組成，模擬人腦神經的架構及原理來解決一些決策判定上的問題，也可以理解為深度學習(Deep Learning)下的一個分支。

一個 AI mode 至少包含三種不同的層，輸入層(input layer)、隱藏層(hidden layer)及輸出層(output layer)。模型可以擁有多個隱藏層。每層可以包含一個或多個 neuron，每個 neuron 個別得到來自上一層的輸入後計算並產生輸出，最後會由輸出層產生最終的輸出，如圖 3-4 所示。

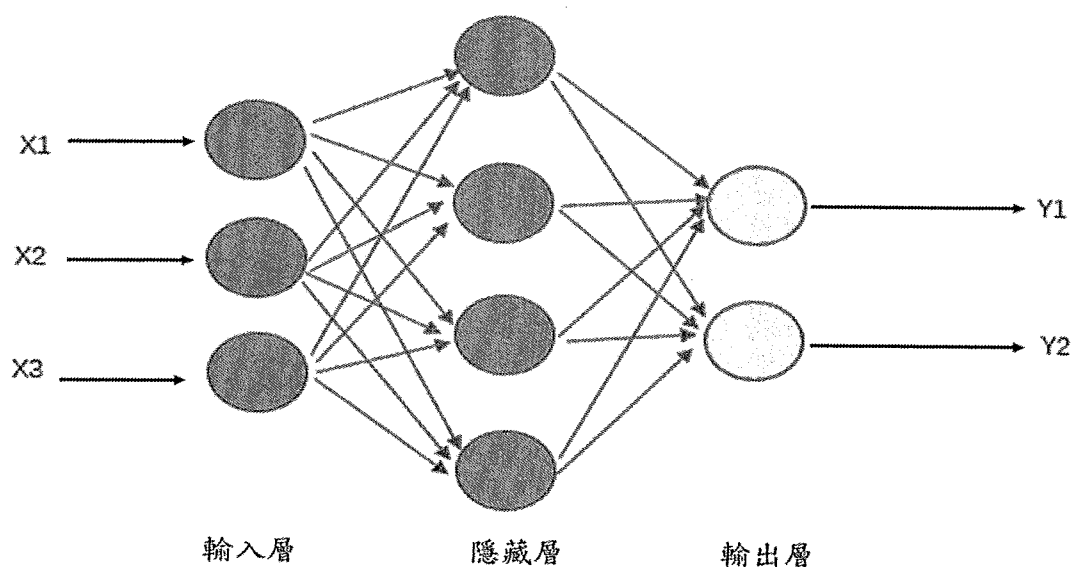


圖 4-1 神經網路模型基本架構圖

### 1. 輸入層

- 在神經網路的第一層被稱作輸入層，這層取得在外部數據，傳至這層的輸入就是常說的特徵值。
- 這層 neurons 的數量和傳入特徵值的數量相同。

### 2. 隱藏層

- 在輸入層與輸出層之間的層面被稱作隱藏層，一個神經網路至少要有一層隱藏層，這裡就是學習開始的地方。隱藏層的 neuron 為了學習需求而做運算，模型需自行決定需要的層數。當隱藏層數越多，計算的複雜度與計算時間也會隨之增加。

### 2. 輸出層

- 在神經網路的最後一層就是輸出層。輸出層會得到在最後一個隱藏層的輸出。輸出層的 neurons 數量由使用者想要神經網路輸出幾組值決定。
- 回歸問題，當網路需要預測一串連續的數值，像是溫度預測，這樣的輸出層只會有一個 neuron。
- 二元/多元問題，當網路需要預測多個中其中一個類別(one of many classes)，輸出層的 neurons 會有和所有可能類別(class)的相同數量。假如網路被訓練用來預測四種之中某一種可能的動物，貓、狗、獅或牛，那麼輸出層就會有四個 neurons，一個 neuron 代表一個 class。

## 4-2 AI 神經網路訓練

上圖 4-1 中每一條箭頭，均代表神經元間相對應之「權重」神經網路的運作可以簡單理解成，神經元輸入值乘以權重並進入隱藏層運算，最後輸出預測值，在這個過程中權重是一個很重要的關鍵，他決定了每個特徵的重要性，所以一個好的神經網路，必須要有完善的「計算權重」的環節。

權重的計算無法一步到位，要反覆訓練、檢驗誤差並且修正，才能得到一個好的網路，因此，計算權重的流程如下：

- 1.隨機初始化權重，使得權重無限接近 0 但不等於 0
- 2.將第一組觀察數據輸入"輸入層"，每個變量特徵輸入對應神經元
- 3.正向傳播：神經網路從左至右進行計算，每個神經元得到來自上一層神經元的輸入與權重的運算，並計算預測值和實際值的差異
- 4.反向傳播：神經網路從右至左進行計算，依據誤差函數相對於權重的梯度，對每個權重進行更新，過程中學習速率與梯度將決定更新的速度。
- 5.當所有數據都輸入神經網路後，稱之為一期 (epoch) 的



訓練執行若干個 epoch(次數需自己設定),並對每一組新的觀察數據,重複步驟 1-4 (batch learning)持續訓練模型,直到最後一個 epoch,模型即訓練完成。

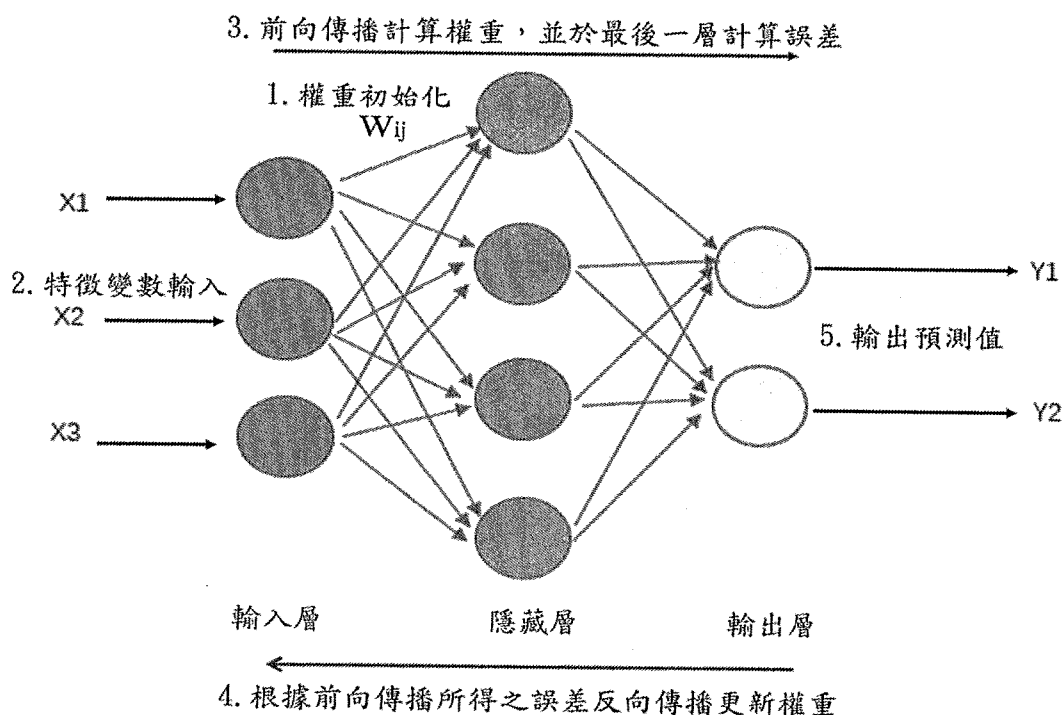


圖 4-2 神經網路模型訓練流程

## 4-2 ARO3\_3C8201 AI 神經網路模型介紹

ARO3\_3C8201 AI model 框架如圖 4-2 所示，包含一層輸入層、一層隱藏層(全連接層)及一層輸出層，藉由輸入溫度、流量、進料組成等 22 個特徵變數進入神經模型中進行深度學習運算，並預測出熱油氣用量、塔頂串級溫度、塔底溫度等三個目標值，其中塔頂二甲苯及塔底甲苯濃度為 3C8201 去庚烷塔之管制標的，製程人員可依模型預測得到特定標的

濃度下之製程操作值，圖 4-3 為 3C8201 去庚烷塔神經網路模型架構。

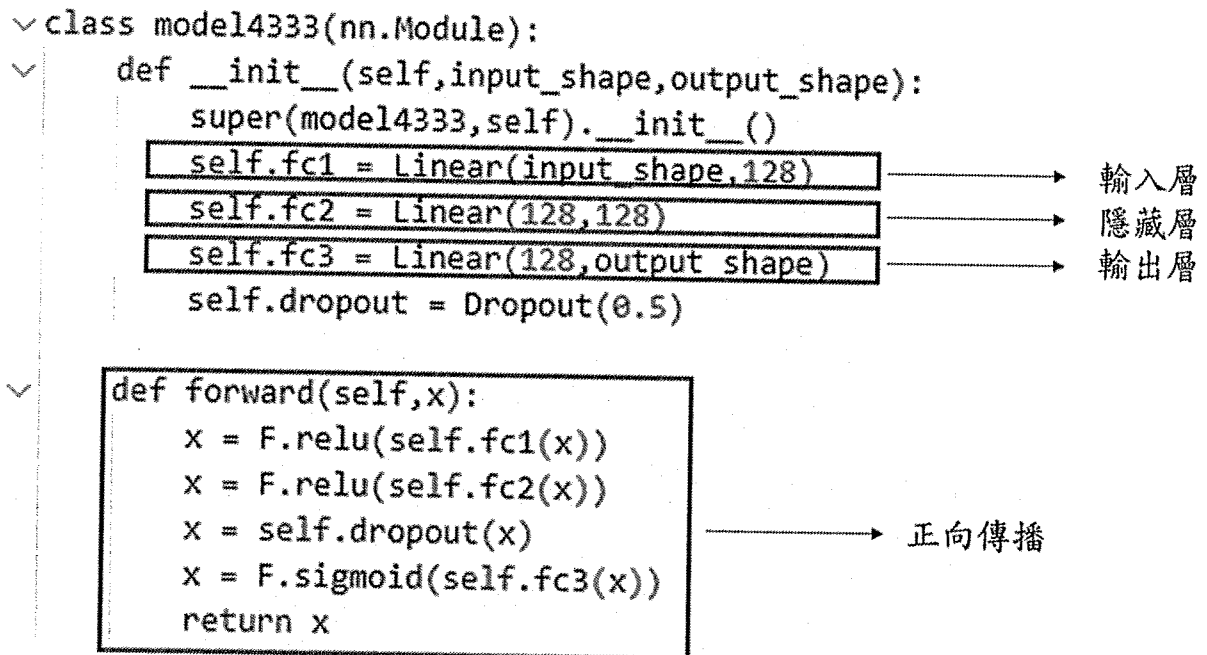


圖 4-3 3C8201 去庚烷塔神經網路模型架構

## 五、工作心得

職於九月中從嘉義新港廠區輪班訓練完成，回到台北工務部自控處智能專案組報到，來到這邊從四班三輪的輪班生活與現場轟鳴吵雜的機台聲到台塑大樓安靜有紀律的生活，差異非常大，雖然需要重新熟悉日常與工作內容，但主管與同事都很親切，有問題或是工作上不懂的地方都很願意說明與幫忙。

於報到第一週，我承接了「ARO1 廠 C914 節能優化人機介面建置」，待第一案完成後又安排「ARO3 廠去庚烷塔操作

AI 預測畫面建置」工程案，雖然之前有寫過相關後端程式的經驗，但還是有許多地方需要理解與處理，像是目前公司主要所用的 PI 系統、資料庫連接，或是如何架設 Apache 等，這些都是先前未接觸過的，為了解決這些困難，有請教前輩與同事們後慢慢解決，也有搜尋相關官方文件或網路資料等，將這些困難點在專案過程中一一釐清，並順利完成這些案件。

在專案過程中我了解到，最重要的一部分就是有效的雙向溝通，無論身處何位都必須學習如何與他人溝通，傾聽客戶實際需求，在設計案件時，從接到案件、理解消化案件、到與業主溝通、確認所需資料及業主訴求，都必須與業主進行大量的溝通才能真正地開始設計案件，藉由雙方有效的交流，既能節省彼此時間，也能加速專案的推進。

在這段時間學習到不少公司制度與規範，更從主管與同仁之間學習到不一樣的專業與處理事情的角度，在工務部需與不同領域的專人互相學習、溝通與合作，發現自己還有許多不足的地方需要更努力，也感謝公司給予資源與環境讓我學習與增進自己的能力。