Pump it Up: Data Mining the Water Table

NTU_r05943139_全班最多PhD組

學號	姓名	分工
R05943139	張育瑄	Pump程式 & 研究 Preprocessing and Feature Engineering
F04943103	劉彥君	房地產程式 & 模型survey & report統整
R04921094	葉孟元	Pump程式 & 實驗討論stacking
F04943095	郭玳質	房地產程式 & 模型survey & report統整

1. **Preprocessing and Feature Engineering**

   由於本題目是採用真實的資料來做預測，因此首先要處理的問題就是 training data 以及 testing data 有大量 missing data 的情況，像是 nan、unknown 或是不合理的資料 (e.g. latitude 為 -2e-8)，以下列出幾個較為重要的 missing data 處理方式：

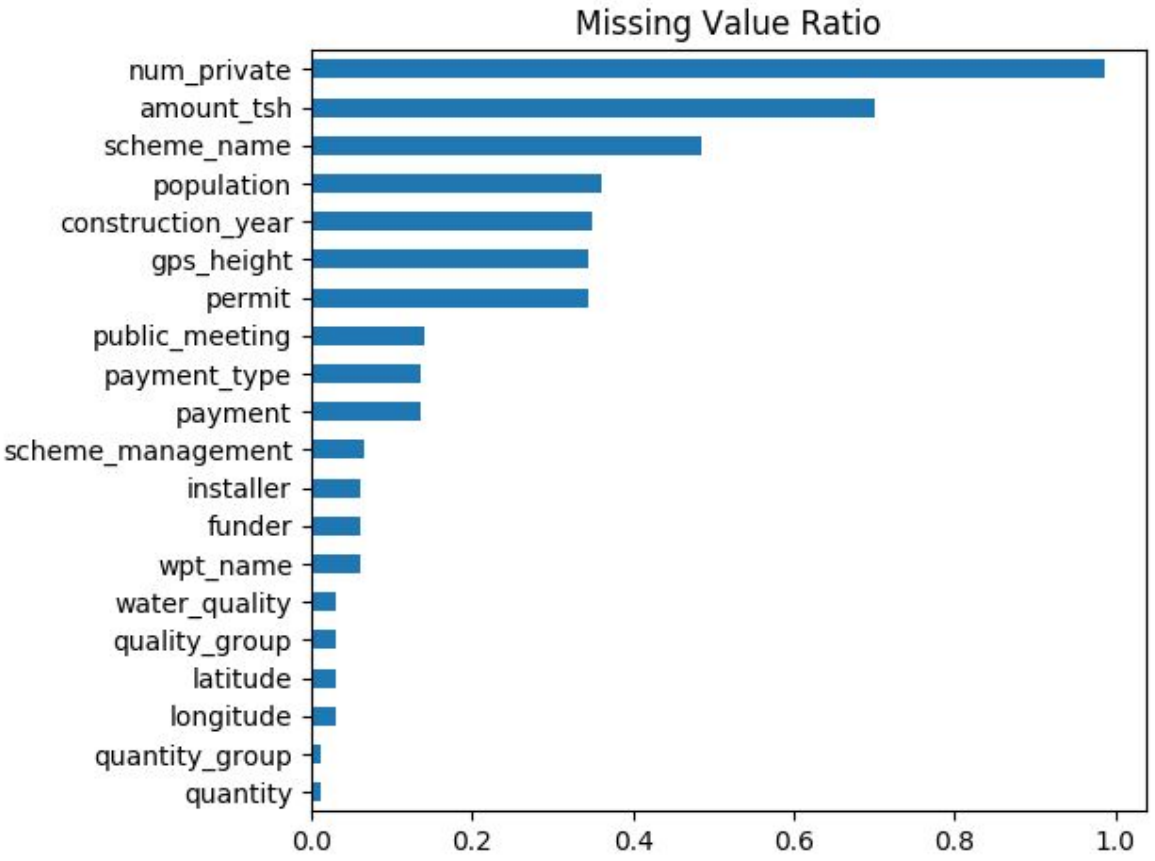
   - longtitude & latitude
     由於經緯度跟地理位置密切相關，因此我們將 missing value 所屬的 lga 找出來，並算出同 lga 下的經緯度平均，再填回去，但若是同 lga 下所有資料的經緯度也都是 missing，但我們就 fallback 回 region 去算平均再填回去。
   - gps_height
     與經緯度的屬性相似，我們也假設同個地區附近的海拔是接近的，因此使用跟經緯度一樣的方式處理。
   - amount_tsh、population 以及 construction_year
     由於這些 feature 與其他 feature 的 correlation 比較沒這麼直覺，因此我們直接用該 feature 的中位數補上 missing value。

   再來是對於一些 feature 我們額外做的 feature engineering，以下列幾個比較重要的：

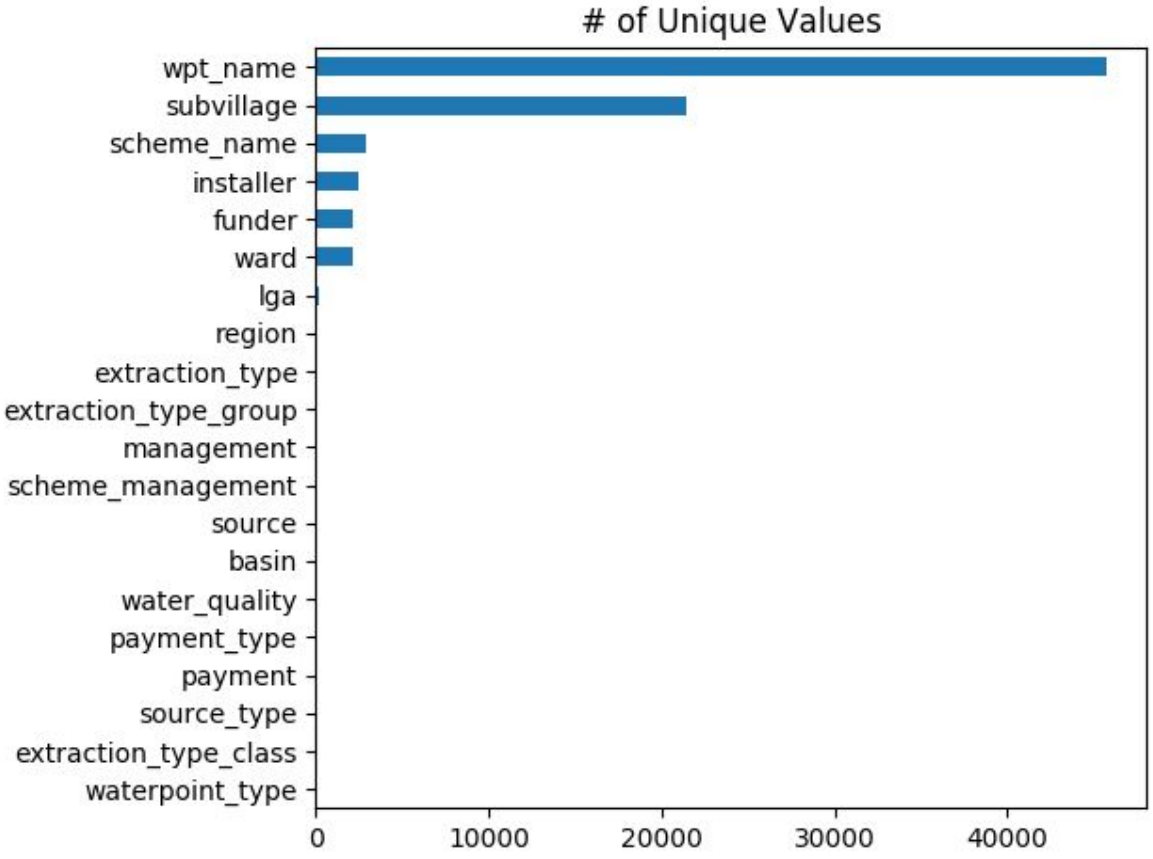
   - date_recorded
     我們將 date_recorded 分為三個 feature：year、month 以及 weekday，因為一天這個範圍實在太小，所以若是分成年、月以及星期幾，資料之間的關聯性才會建立起來，而後面的實驗結果更是證明星期幾這個 feature 影響不小。
   - construction_year
     我們將 construction_year 的最小值找出來，並且將所有的 construction_year 減掉這個最小值，讓這個 feature 的意義代表這個井的先進程度，所以 value 越大代表越近代建造的井，我們就假設它技術越新且越耐用。
   - age
     我們將 date_recorded 抓出來的 year 減去 construction_year 變成全新的 feature，也就是 age，來代表這個井的使用年齡，因為越舊的井故障損壞的機率當然就會越高。
   - population

我們將 population 取 log1p，於是這個 feature 的意義就會是人口的一個數量級，而不是一個絕對的數字，畢竟差幾十個人的影響不大，另外，我們還產生一個新的 binary feature，當經過 log1p 的 population 小於 2 時為 1，其實就是將人特別少的地方標註起來，因為每個地方的地下水資源是有限的，想當然人口就會對井的功能有顯著的影響。

最後就是決定要丟掉那些 feature，以下列出幾個決定的方法：



首先觀察每個 feature missing value 的比例多大，如果太大，那麼補一堆上去的話可能毫無意義，還不如直接丟掉，上圖為 missing value 比例前二十高的 feature，意義不明或是我們認為較無用處的先直接丟掉，像是 num_private、scheme_name、permit 以及 public_meeting。



再來是觀察每個 feature 的 unique value 的數量，若是太大，那一樣沒有意義，上圖為 unique value 數量前二十多的 feature，可以明顯看到 wpt_name 根本幾乎是每筆資料都是不一樣的 value，因此我們認為這個 feature 毫無幫助 (就如同 id)，所以將它丟掉，再來 scheme_name 已經被丟掉了，直接跳過，再來 installer 和 funder 我們直覺會有用處，所以將其前十多的 value 維持原狀，其餘的則歸類為 other，最後 ward 和 subvillage 兩個 feature 跟 region 以及 lga 一樣都是地理位置的資訊，只是它們是更詳細的資訊，原本地理資訊應該是很有用的資訊，但像 subvillage 這樣幾乎一種 value 就只有兩筆資料的狀況，我們認為並沒有幫助，甚至還會大幅增加 feature 的數量，嚴重拖慢 training 的速度，所以最後決定直接丟掉，最後至於 ward，為了加速 training，也決定直接丟掉。

2. Model Description

我們主要嘗試了一些模型，包括DNN、RandomForest和XGBxgboost，以下我們分別對不同的模型做討論:

DNN:

利用基本fully connected network架構，並沒有過多的新穎作法，當時只是相信了Deep Learning很強的“謠言”，在不斷調節不同的模型形狀，層數和optimizer的情況下，最終採用了下圖的架構。最終也只達到0.7697的成績，這個成績和simple baseline還有很大的差距，最終就放棄了這樣的做法。

```
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Features (InputLayer)        (None, 218)               0
_________________________________________________________________
Dense-1 (Dense)              (None, 64)                14016
_________________________________________________________________
Dense-2 (Dense)              (None, 128)               8320
_________________________________________________________________
Dense-3 (Dense)              (None, 256)               33024
_________________________________________________________________
Dense-4 (Dense)              (None, 512)               131584
_________________________________________________________________
StatusGroup (Dense)          (None, 3)                 1539
=================================================================
Total params: 188,483
Trainable params: 188,483
Non-trainable params: 0
```

RandomForest:

原理: 顧名思義即是將多個decision tree ensemble起來，且多個decision tree在每次做branching的時候，random選擇某些feature不使用以產生不同的tree。

Training: 從training data中以有放回的抽樣方式取樣(bootstrap取樣)，以形成training set。對於decision tree的branch隨機選擇一組feature做splitting。並使用未取樣到的training data做為validation使用(out-of-bag validation)

效果: (1) 對於不同種資料可以產生高準確率的classifier。 (2) 能夠處理高維度的data，不太需要降維。 (3) 能夠評估各feature在分類問題上的重要性。 (4) 訓練速度很快。

XGBoost:

XGBoost的全名是“Extreme Gradient Boosting”，主要他是使用gradient boosting的技術，也就是說他會建立多個模型，一般來說這些模型會比較弱，然後會針對原本的模型中表現較差的地方再建立新的模型來加強他，最後對各自的模型以一些權重做分配而形成整體的主要模型，如此一來，儘管原本的模型可能只是比較簡單的線性模型，但是經過這樣boosting的動做之後就可以形成較為複雜的模型，藉此來提升預測的準確率。

而XGBoost除了使用原本的boosting的方法之外，還加上了tree ensemble，tree上的每個節點用來判斷某些我們所預期的特性之後再對其評分，其中的概念和

RandomForest有些類似，但是因為XGBoost有其評分的機制可以使得我們再training的過程中能夠刪除一些不必要繼續生長的節點，也就是說我們不需要去列舉出所有可能的tree，我們只需要對不夠強的節點繼續衍生即可。
效果: 由於XGBoost可以藉由樹狀的結構複合出較為複雜的模型，並且可以針對訓練過程中較弱的模型做強化，尤其是這個題目中的label分佈不平衡的現象很嚴重，因此我們希望可以藉由XGBoost來強化資料較少的label，在初期時也取得不錯的效果。

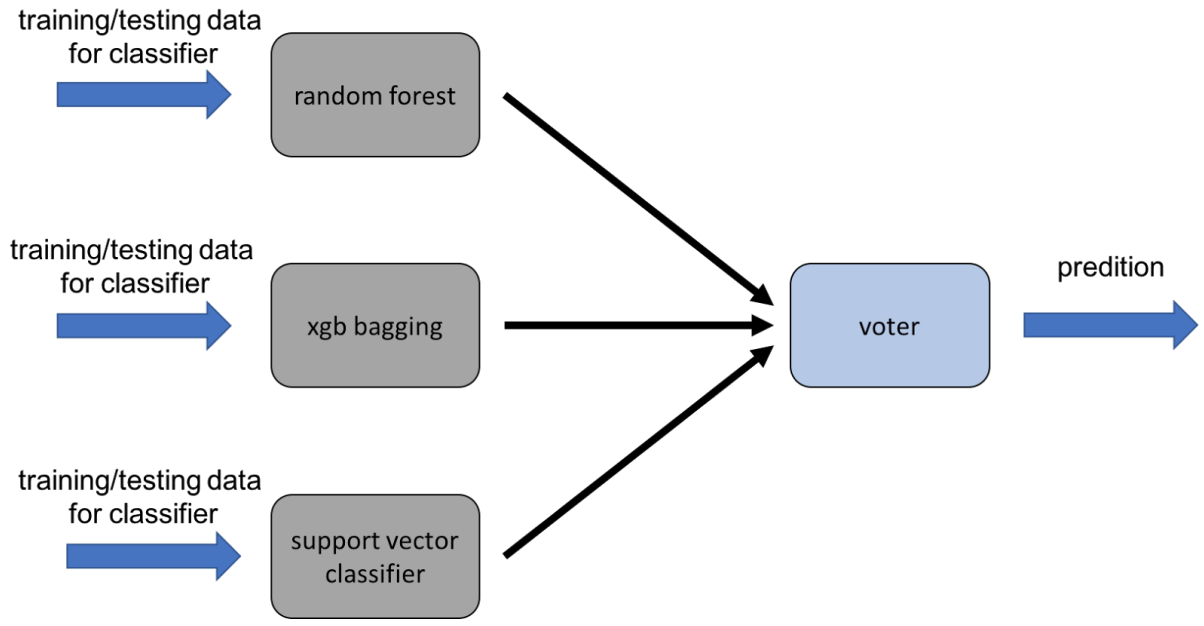
Grid search：

在選擇參數上我們採用了Grid search，最一開始使用的是normal distribution random測試參數，最後因為時間充足，我們採用brutul force利用給定一個範圍內大量的參數的排列組合去train出不同的model，並使用cross-validation的方式選出一個準確率最高的model。

3. Experiments and Discussion

我們在這個部分準備討論的是stacking中不同voter對分數的影響和在我們實作中stacking是否起到決定性的作用，以及stacking的部分特性，最後還有對於feature重要性的探討。

如上所述，我們分別嘗試了xgb bagging和randomforest加上Grid search並成功利用這兩個模型突破strong baseline。但是這兩個方法主要都是利用ensemble中的bagging和boosting來提高預測的準確率。在最後嘗試再拉高分數的時候想到了老師在上課的時候講過的另外一種ensemble的方法：stacking。這個方法主要適用於不同的模型之間如果單純利用average或hard voting來得出最後結果效果會很差，因此需要專門一個voter來決定voting algorithm。stacking模型的基本原理，如下圖所示，首先先取出一部分資料來訓練出不同的單一classifier，比如random foreset， xgb bagging或者support vector classifier。再取出在訓練classifier時沒有用過的其餘資料先通過前面的classifier得到各自的機率預測（predict_proba或用xgb的softprob），concatenate起來，利用這些concatenate過後的資料訓練出最後的voter。這個voter可以決定每一個不同classfier之間的voting權重甚至是非線性的關係，這取決於我們要何種classfier來作為voter。



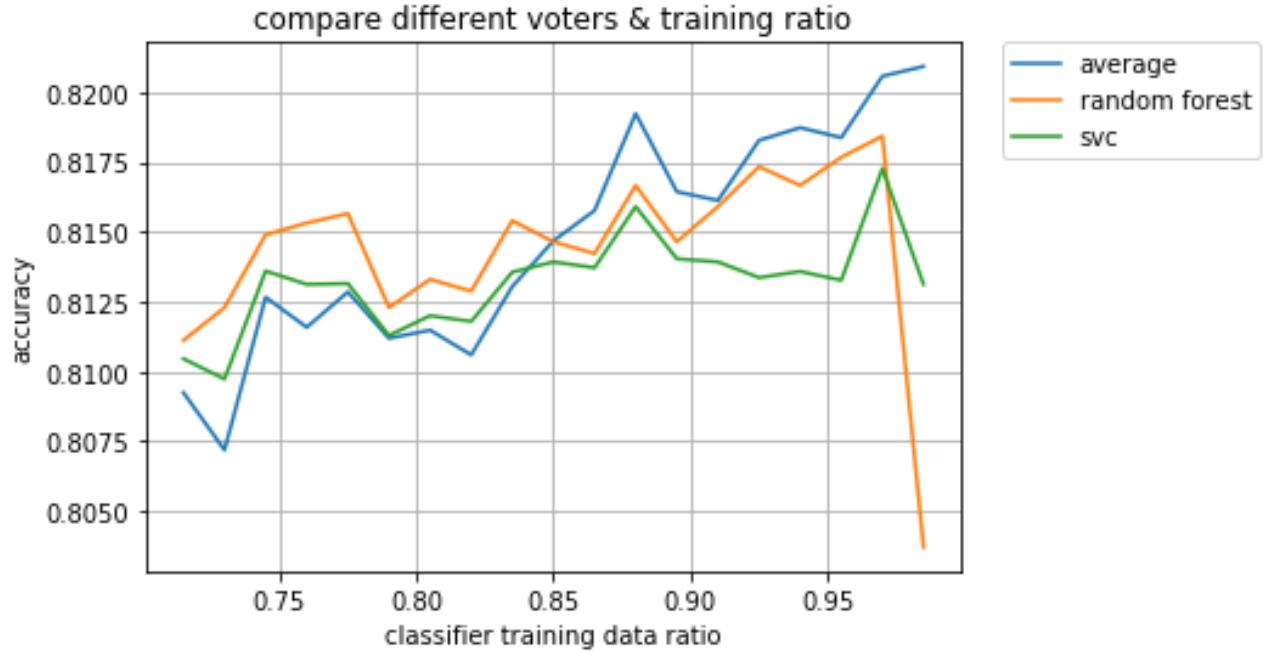
在我們實作stacking發現有很多的問題和困擾，觀察到利用xgb和random forest這兩個都可以各自過strong baseline的模型疊加適用stacking卻只從0.8236提升到0.8239，並沒有比較好的表現。我們一開始懷疑是切割資料部分出現問題，也許是因為我們切了太多的資料去訓練voter，而導致了前端的classifier不夠強於是因此衍生了以下三個問題並做了相關的實驗，並發現了一些有趣的結論，接下來會進行詳細的說明。

首先介紹一下實驗環境和名詞介紹，爲了讓實驗結果不要出現太大的誤差，我們從training set中切出了20%的資料總共11880筆validation data。這部分validation切好後都不會再變動，而data preprocessing是使用我們最後的版本來實作的。“切割比例”：我們認爲用來train前端的classifier的資料佔總資料的比例爲切割比例。如果拿出90%訓練前端classifier，10%來訓練voter，則切割比例爲0.9。

問題一：不同的voter和切割比例是否會影響最後的準確度？

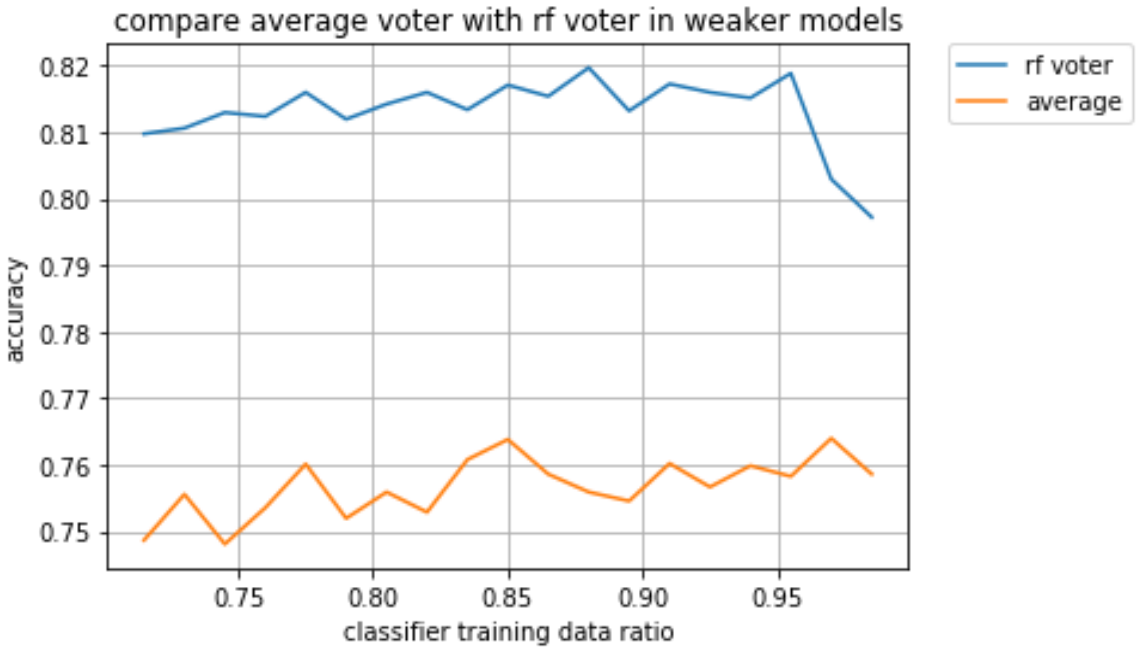
結論：會，但是結論和我們預期的有很大的差異。

前端classifier利用的是3個random forest和3個xgb，我們這個實驗比較的只是random foreset和svc兩個不同voter之間和切割比例之間的關係。我們最初預測，準確度可能會不斷增大在比例是0.9左右出現最大值之後再慢慢下降，但是實驗結果卻在0.97的地方才出現了最大值，盡管在0.985第地方出現了下降，我們認爲有可能是取樣時的variance導致的。我們懷疑，會不會越多資料給前端classifier訓練會更好？其實那些切割來做訓練voter的資料在前端classifier反而更能發揮他們的價值。於是我們大膽猜測如果使用全部資料都放在前端classifier，voter只是簡單average會不會有不一樣的結果？（雖然這樣與bagging也很像，但是我們偏向熱為這更像是stacking，因為前端還是有不同的模型）於是我們就有了下面這張圖，average voter在0.87左右開始超過svc和random forest voter，在最後也沒有出現下降的趨勢。為什麼一開始是比較差的，而後來會越來越好呢？我們推測是因為在average voter在切割比例小的時候相當全部個模都少了資料。（因為不用train average voter，只是單純平均）但是當切割比例大的時候就average就超過了另外兩個方法，也就是說，這些資料都放在前端classifier會比放在訓練voter更有價值。我們進而進一步大膽猜測，也許當模型都類似的時候，都有不錯的表現的時候，train出來的svc／random forest voter和average voter或許是很類似的。那些data與其來訓練voter，不如放到前端classifier，然後再利用一個簡單的average voter來整合就足夠了。利用這個觀察我們採用簡單的average方法結合5個random forest和xgb達到了0.8246的最終成績。（其實因為時間不夠，我們認為利用drivendata 的分數做 weighted average會能進一步提升成績）



問題二：既然stacking在xgb和random forest上沒有很好的表現，如果現在有兩個差異很大模型，stacking是否有意義？和average voter之間的比較呢？

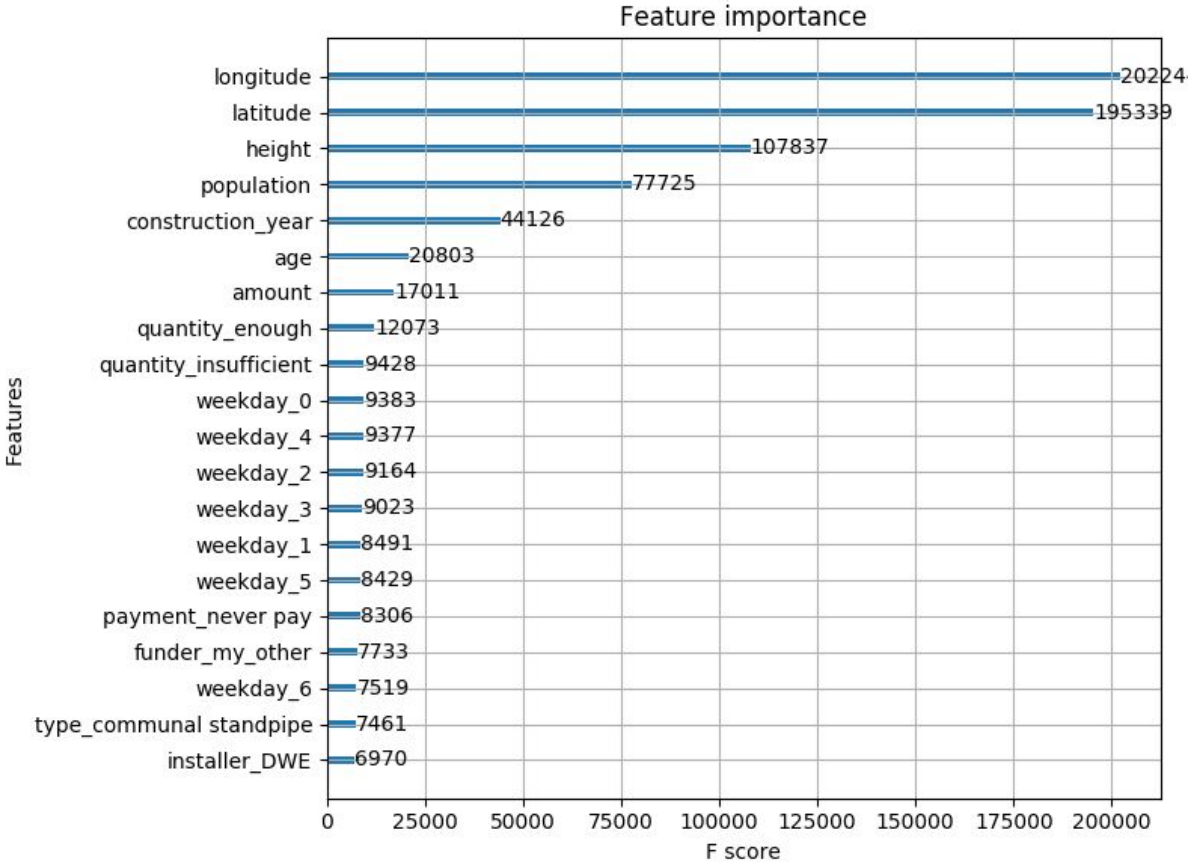
結論：stacking在差異比較大的模型上會有更好的效果。這次實驗的前端classifier是svc和random foreset。為了實驗室比較快，就都各只有一個。我們主要比較的是random forest voter和average voter之間的差異。結果如下圖，趨勢和上圖是類似的，rf voter這次在最後有明顯的下降，average也是從一開始一直在提升效果。可以看到的是，rf voter確實有效的選到了正確的值來作為最後的輸出。說明stacking voter的機制是有意義的，但是可能僅限於差異較大的classifier上。這和我們預測的是很接近的。



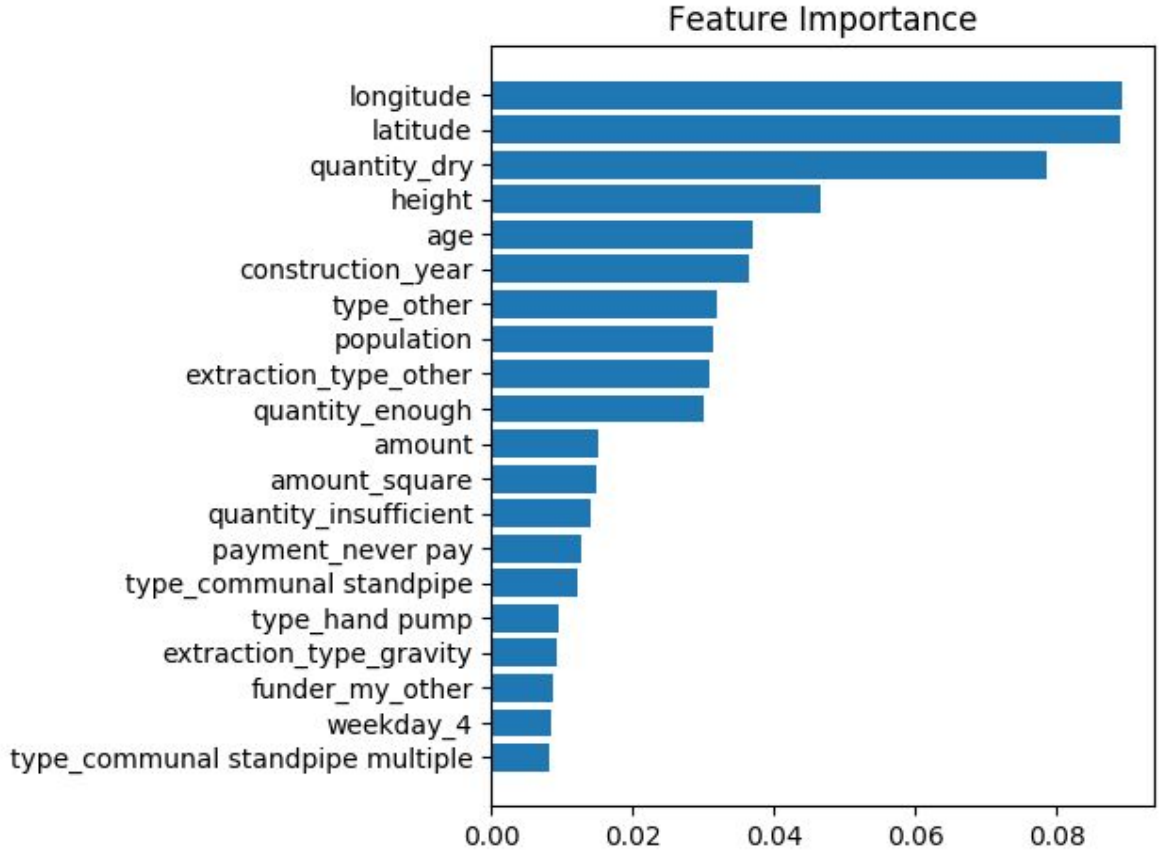
問題三：stacking voter的在對待差異較大的前端classifier的作用？

結論：作用在於“取長補短”，直接比較前端是一個svc和一個random forest和3個xgb和3個random foreset的結果圖。發現的是，就算只有一個svc和一個random forest。它的最大值是0.8197甚至高於 5 * xgb + 5 * rf的0.8184。不過由於這邊在切訓練voter的資料的時候有所差異，所以趨勢並不像之前圖表那麼同步。但是至少可以說明一件1 * svc + 1 * rf不會比那麼複雜的差，在前端甚至還高於xgb + rf。這說明的是，stacking需要的不是兩個預測同一類型都很準的模型，而是兩個分別能夠預測對不同類型資料的classifier。這在最後非常符合邏輯和一個學期所學到的知識，並且對stacking有一個全面和嶄新的認識。

問題四：各 feature 對 model 的重要性？
結論：



上圖為 xgboost 的 feature importance 圖。

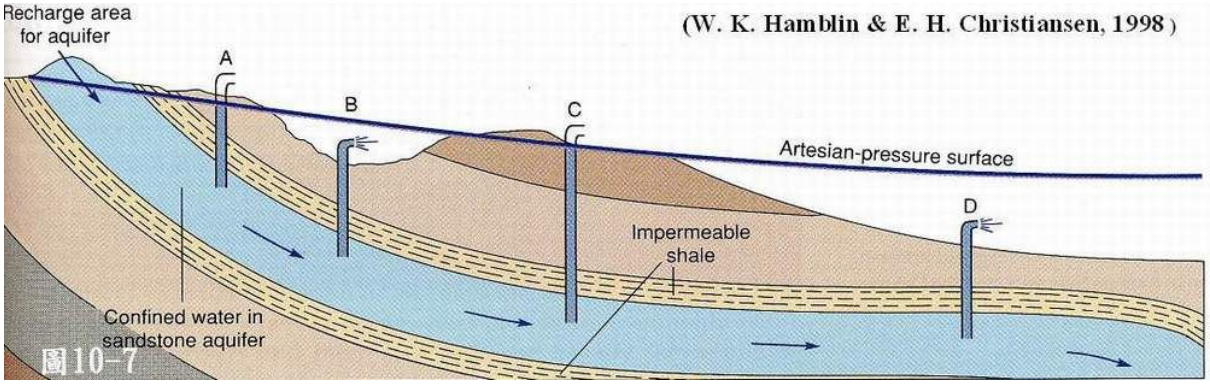


上圖為 random forest 的 feature importance 圖。

根據實驗我們分析了一些 feature 的特性以及其重要性，主要這次是分析井的功能性，井水的主要來源是地下水，而一個區域的地下水在某種程度上是有限的，所以可以看到經緯度有決定性的影響力，同一個地方可能會有很多口井，但是其中所採集的地下水是共同所有的，而這樣的特性最主要就是反應在經緯度上，在相近的區域，也就表示相似的經緯度，會很大程度的對該地區的井有所影響，因此經緯度在這題中成為了一個很重要的因素。

其次則是高度，從上面的推測同樣可以得到高度對井水的影響，原因有二，第一是相近的區域的高度也會相似，除非出現斷層或是峽谷，而這些地形是很少見的，所以高度和經緯度有相似的效果，第二則是地下水，顧名思義水是水在地下，如果原本井的高度就比較高，則可能需要較深的井才可以挖到足夠的水量，而一般的井可能不會故意開鑿太深，因此有可能導致地理位置較高的井更為容易失去其功能性。

可以參考下圖中的井C和D，C所在的地理位置較高，但是地下水層(面)和D是相同的，使的C需要開鑿得更深才能夠取得井水，而D井則不需要那麼深即可得到井水，由此我們認為高度對井是有重大的影響，而實驗分析也得到相同的結論。



[地下水的作用]

除了地理位置的影響之外，人為的因素也很重要，其中包括人口數、建造年份和井的年齡等等。人口的部分在上文中有概略提到，其中主要的考量是地下水的蓄含量是有限的，如果一個地區的人口數量越多，也就代表著水的需求越高，人的身體組成中有70%是水，因此水對人的重要性也就不言而喻，換句話說，當該地區的需求長期大於供給時則較有可能使得該地區的井水消耗殆盡，所以可以推論人口對井的功能有決定性的影響。