Pumping It Up: Data Mining the Water Table

隊名：資工水水隊
隊員：B03902031 詹郁珍：資料預處理以及調整Model參數
B03902059 紀典佑：讀取相關paper理解Model並且寫出Code原型
B03902102 廖廷浩(隊長)：voting ＆ 各種 Experiment

I. Preprocessing/feature engineering:

id	int	scheme_management	string, 流程管理機構
amount_tsh	int	scheme_name	string
date_recorded	yyyy-mm-dd	permit	boolean
funder	name	construction_year	int, 建造年份
gps_height	int	extraction_type	string, 測量方式
installer	organization	extraction_type_group	string, 同上，較大的分類
longitude	float, 經度	extraction_type_class	string, 同上，更大的分類
latitude	float, 緯度	management	string, 管理機構
wpt_name	string	management_group	string, 管理機構所屬的族群
num_private	int, 是否私人	payment	string, 薪水給付情況
basin	哪個盆地	payment_type	string, 大致同上
subvillage	哪個村莊	water_quality	string, 水質狀況
region	哪個地區	quality_group	string, 大致同上
region_code	int, 地區代碼	quantity	string, 水量
district_code	int, 區域代碼 範圍>region	quantity_group	string, 同上
lga	string	source	string, 水源
ward	string	source_type	string, 大致同上
population	int, 人數	source_class	string, 大致同上

			，較大的分類
public_meeting	boolean, 有無開放	waterpoint_type	string, 取水方式
recorded_by	string, 記錄者	waterpoint_type_group	string, 同上

A. 觀察：
   1. 有很多關於地區的資料，應該可以用經緯度概括，因此不採用除了經緯度以外的地區相關資料。
   2. id只是編號，理論上不用加入training data裡。
   3. 井水狀態如何跟誰record基本上無關。
   4. funder, installer等資料跟井水狀態基本上也無關。
   5. 最後我們排除了id, recorded_by, wpt_name, subvillage, region, region_code, district_code, ward, scheme_name, installer, funder, installer等資料。

B. 預處理：
   1. DictVectorizer：
      a) 將list中string的部分轉成向量，使用了"one-hot encoding"去完成這件事情，將string轉成向量並且會將feature name存起來。
      b) 例如：一個array裡面，有city分別是A, B, C，以及人口數100, 1000, 10000, 那麼經過DictVectorizer後，會出現的array應該會是[[1, 0, 0, 100], [0, 1, 0, 1000], [0, 0, 1, 10000]]，其中第一項為「city是否為A」，第二項為「city是否為B」，第三項為「city是否為C」，第四項則為人口數。
      c) 我們使用了DictVectorizer將training data中所有非數字的部份轉成向量，並以此為我們的X_train。
   2. LabelEncoder：
      a) 給予list中的data編號，例如把[1, 3, 3, 5]丟進fit_transform裡，再將[5, 3, 1, 1]丟進transform裡的話，應該會出來的是[2, 1, 0, 0]，因為1為encoder裡面的第一項，3為第二項且5為第三項。同理，把['A', 'B', 'A', 'C']丟進fit_transform裡，再將['B', 'A', 'C']丟進transform裡，應該會出現[1, 0, 2]，因為A為encoder裡第一項，B為第二項，C為第三項。
      b) 我們使用LabelEncoder將我們的Y_train全部轉成數字，在要寫出的時候再用inverse_transform轉回文字。

II. Model Description:

A. Random Forest Classification:
   1. 原理：Random Forest Classification (以下簡稱為RFC) 為一種 ensemble 的方法，主要是應用 Bagging + Decision Tree 來做預

測的 Model，整個 Model 中存在許多 Random 的變數，其中有兩種主要用來增加每個Tree 的 Data Diversity 方法：

a) Diversifying by Feature Projection：在 d維的 Training Data 中，只隨機選取其中 d' 個 feature 來當一顆 Tree 的訓練資料，在原始的論文中，作者提到除了能在每棵樹做一次隨機選取 Feature 此動作(又稱 random-subspace)，在樹上的每個 Node 也可以做 random-subspace 來增加隨機性。
b) Diversifying by Feature Expansion：是 random-subspace 的一種延伸，是直接將 d 維度的 Training Data 運用隨機的投影矩陣 P ，直接投影到 d'' 維度(low-dimension) ，此方法又稱 random-combination 。

2. 參數：這次實作 RFC 使用的是 scikit learn 的 package，其中有三個參數對於整體 Model 表現有極大的影響力，分別為：
   a) max_features：在每次 split 時所能影響的 feature 數。通常較大的數值會有較好的表現，因為在每個 Node 上考慮的選擇更多，但並不是一定，因為太大的數會使得在 split 時的 diversity 降低。此外，雖然此數值會增加 model 的表現，但因為運算量變多，有可能會降低整體的 performance。
   b) n_estimators：Tree 的數量。較多的 Tree 會讓預測的表現變好，但是讓 Training 的時間變長。
   c) min_sample_leaf：每一個樹最後一個子葉中的 sample 數量。數字小通常會讓整個 RFC branch 太過詳細，容易被 Noise 影響，此數字在單個 decision tree 上影響極大。

B. Xgboost:

1. 原理：Xgboost 是一種改進的 gradient boosting 的訓練模型，相較於傳統的 gradient boosting decision tree，Xgboost 最大的差別是加入了 regularization term 以及設計出可平行化的演算法，迅速加快整個模型的訓練速度，也因為速度的增加，使得可以快速做出更多的 Tree，增加了準確度。
2. 參數：Xgboost 的原始論文作者有在網路上開放open source ，而此 package 有提供 scikit learn 的 API 介面，以下列出幾個較為關鍵的參數：
   a) eta：即 learning rate，在做 gradient boosting 時的每次更新大小(stepsize)。值太大可能無法到 Object funtion 的最佳解，但是太小更新速度會太慢。
   b) max_depth：每顆 gradient boosting tree 的最大深度。深度愈深，越容易overfitting。太淺則可能會 underfitting。
   c) min_child_weight：Node 中最少需要的數量，如果某個 node 中的 sample 數量比此數還小，則不再 partition。增

加此數得值可以避免 overfitting ，但是增加太多會讓分類變得不準確(branch error) ，造成 underfitting。

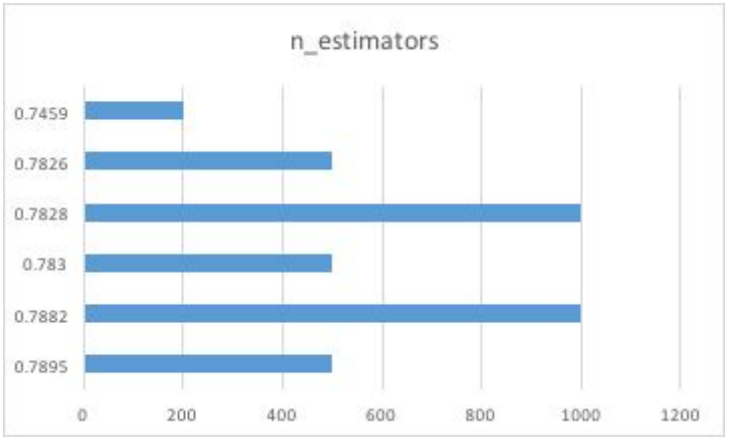
d) gamma：與 loss 有關，最少需要多少 gamma 的 loss 才會繼續做 partition。此數值跟 loss function 有較大的關聯性。
e) subsample：每個 Tree 所拿到的資料量。調整此數能夠增加整個 Model 的隨機性，使得 Model 增加一些 Noise 不會導致 Overfitting。
f) colsample_bytree：每個 Tree 所拿到的 feature 量，調整數值一樣能增加 Noise。

上述參數都可用來防止 Overfitting，機制大概分為減少 Model 複雜度（b,c,d）與增加 Model 的隨機性（e,f）。

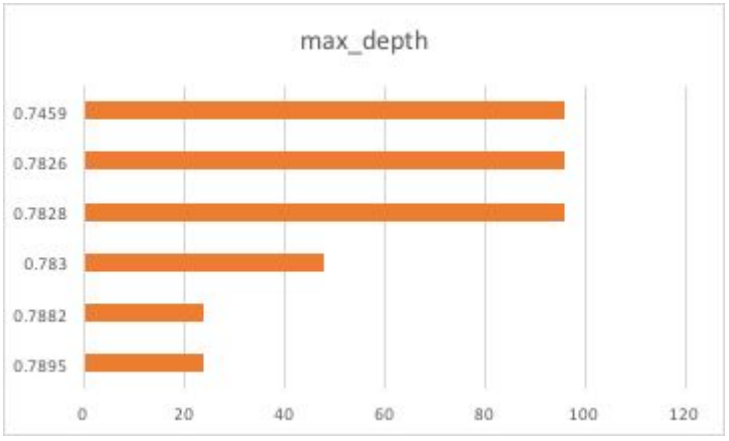
III. Experiment and discusion:

A. Random Forest Tune：

1. n_estimator：



2. max_depth：

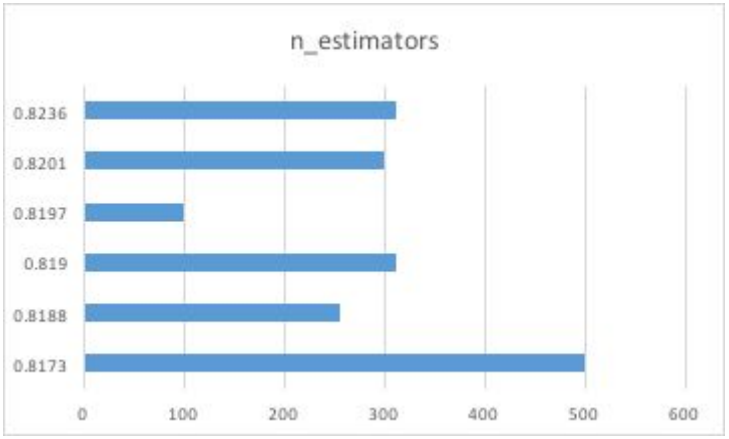


3. discussion：可以觀察到n_estimator並不會因為太大或太小而有更高的提升，大約在五百左右是最好的。太小效果不好的原因是因為樹的量太小，Diversity不足，而太大不好的可能原因是樹太多造成了Overfitting了。另外觀察到max_depth越大，對準確度來

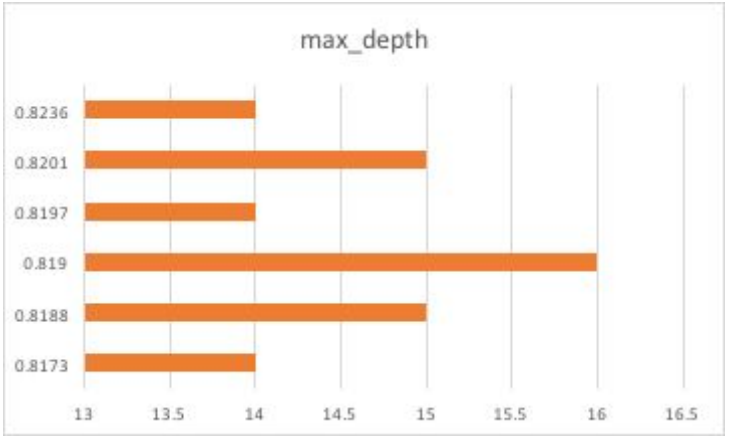
說結果會越差，是因為分得太細造成Overfitting，因此max_depth理論上應該要調到一個不會造成 Overfittin 與 Underfitting 的中間值。

B. Xgboost tune：

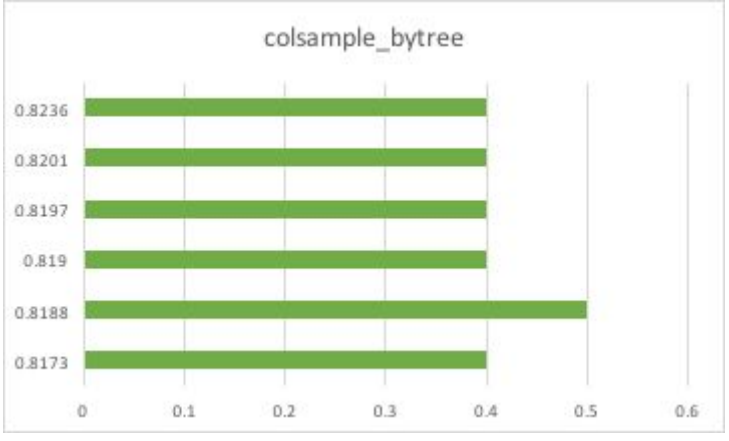
1. n_estimator：



2. max_depth：



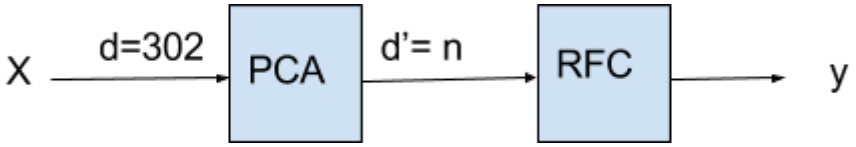
3. colsample_bytree：



4. discussion：n_estimator的參數對準確度來說，太大太小都不是很好，300左右對準確度來說是最好的範圍。做過幾次的training，可以發現當max_depth開得越大，train的時間會變非常非常的久，但結果並不盡然會比較好，在14時反而是做出來準確度最高

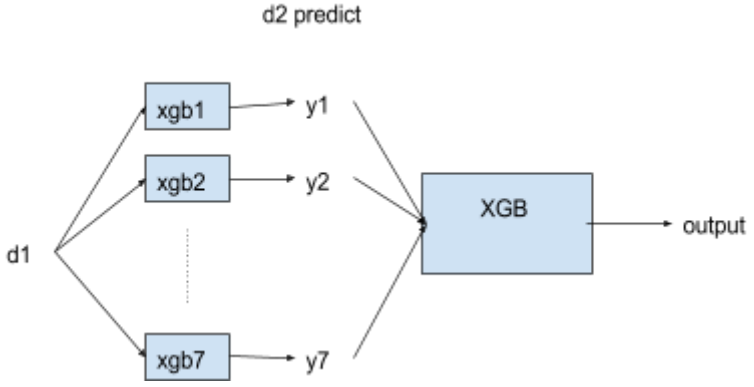
的一組。至於colsample_bytree和mid_child_weight，對於準確度來說似乎沒有太大的差別。

C. PCA & RFC：
   1. 想法： Training 的 Data Feature 有接近 300 個左右，想試看看用 PCA 先降低維度，應該能取出較具有意義的數值，再把降維後的結果丟進 RFC 裡面 Training，或許會得到比較好的結果。
   2. 做法 ：



   3. 結果：
      a) n = 100 ： 在 drivendata 上只有 0.7874 的準確率。
      b) n = 20：在 drivendata 上只有 0.7853的準確率。
   4. Discussion：結果比直接用 RFC 出來的結果還差，大概是因為降低維度後，把太多的資訊簡化了，所以結果變差。

D. Xgboost two level stacking：
   1. 想法：ensemble 的方法中有種方式是先在第一層做出許多種 prediction ，之後運用這些 prediction 的結果當作新的 data feature，在丟進一個新的 Model 裡面去得到最終的結果。
   2. 做法：先將 training data 切成要在第一層及第二層要用的data (d1 , d2) ，在第一層做 7 個 xgboost 的 prediction ，再將這 7 個結果對 d2 做 predict，然後用這 7 個 d2 prediction 當作第二層的 training data ，train 出最終的 y，示意圖如下：



   3. 結果： 在 drivendata 上只有 0.8119 的準確率。
   4. Discussion：此種 ensemble 的方法在網路上的討論中，寧增加滿多的準確率，然而在這個 Task 中沒有提升多少準確率，跟單一個 Model 差不多，甚至更差，可能是因為 test data 的問題也可能是因為 diversity 不足，但是根據單一架構，無法確切得知原因。

E. Ensemble with linear voting：
   1. 想法：將在 datadriven 上較高的 prediction 拿來當 voting 的 input，可能會得到比較好的結果。

2. 做法：簡單的 linear voting，每個 candidate 有一票，當票數相同時，取準確度最高的值。
3. 結果： 0.8244 (Best from 3 diffirent xgboost)
4. Discussion：這是我們這個 Task 得到最高的結果，在 voting的實驗中，我們得知有時候voting的結果準確度並不一定會比較高，發現這些會降低voting準確度的model會有某些參數很接近。而voting之後準確度比較高的大部分是選取的model參數差距會比較大。所以不能接太多相似model得到的結果做voting，必須選取參數差別較大的model作為voting的候選人。