Pump it Up:

Data Mining the Water Table

Team name: NTU_R05922130_LiebeFat31cmRay1

Member : R04922169 楊智偉

R05922130 王瀚磊

R05944019 張嘉豪

B05901189 吳祥叡

零、資料預處理 ＋ 模型訓練 (optional)

預處理中因為 unique()本身有 random 的效果因此每次產生之預處理結果會不一樣，本次作業中表現最好的 11 個 model 皆為同一筆預處理完的 csv 所訓練出來，然而我們沒有將 random seed 存起來，因此我們有將預處理完的結果附在 GitHub 中提供下載。在 reproduce 中，可以直接從第一部分開始執行。

Step1 : 執行 preprocess_data.sh [$1] [$2] [$3]
$1: [train_value.csv]
$2: [train_label.csv]
$3: [test.csv]
Step2 : 執行 train.sh

壹、程式執行方式

Step1 : 執行 download_models.sh
Step2 : 下載 pre_train_data.csv /pre_test.csv
(已附在 GitHub 的 final 資料夾中)
Step3 : 執行 test.sh [$1]
$1: [output_file.csv]

貳、分工內容

楊智偉：Random forest 實作、特徵處理
王瀚磊：Random forest 實作、XG-Boost 實作、ensemble
張嘉豪：撰寫報告、產生與整理實驗數據
吳祥叡：特徵處理、撰寫報告

參、特徵處理

我們的特徵處理有下列四點，其中 random forest 的部分，特徵處理用到了 1、2、3；而 XG-Boost 用到了 1、2、3、4，也就是說 XG-Boost 多了一個欄位 date_recorded。

1. 將 train_label 編號：
   functional→0
   non functional→1
   functional needs repair→2
2. 將 train_value 和 test_value 中的非數值欄位編號：
   這些欄位為 id, amount_tsh, date_recorded, gps_height, longitude, latitude, num_private, region_code, district_code, population, construction_year 以外的欄位。
3. 生成 pre_train_data.csv 和 pre_test.csv，即繳交的兩個檔案：
   將 2.編號的 test_value 輸出為 pre_test.csv。
   將 1.的 label 接在 2.編號的 train_value 之後，輸出為 pre_train_data.csv。
4. date_recorded 的 preprocessing：
   將 date_recorded 欄位轉成從全部資料中最早日期起算的天數，由於是最後靈機一動提出的方法，因此這部分最後是在 train_xgboost.py 以及 train_randomForest.py 中才實作進去。

肆、模型敘述

1. Random forest（五個 random forest model 做 ensemble）

五個 random forest model 我們只調整了 random state，其他參數都是一樣的，參數如下：

n_estimator = 1200, criterion = entropy, max_features = sqrt,
min_samples_spilt = 2, min_samples_leaf = 2

1st 參數： random_state = 123
2nd 參數： random_state = 234
3th 參數： random_state = 345
4th 參數： random_state = 456
5th 參數： random_state = 567

Ensemble 方法：將這五個 model 跑出的機率做幾何平均

2. XG-Boost（六個 XG-Boost model 做 ensemble）

六個 XG-Boost model 我們只調整 random state，而每一個 model 的 random

state 都是一個 1~100000 之間的亂數，除了 random state 外每一個 model 擁有相同的參數。

n_estimators = 500, learning_rate = 0.2, objective = 'multi:softmax',
booster = 'gbtree', colsample_bytree = 0.4, max_depth = 14

Ensemble 方法：將這六個 model 跑出的機率做幾何平均

3. Random forest + XG-Boost

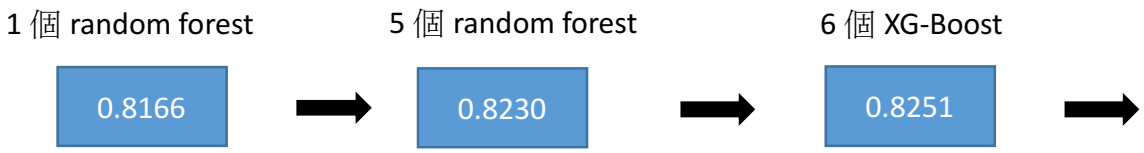
這邊我們是將五個 random forest model 再加上六個 XG-Boost model 一起做 ensemble。這邊 ensemble 的方法兩種：

(1) 幾何平均：直接將 11 個 model 跑出的機率做幾何平均
(2) 眾數：記錄下這 11 個 model 會將各筆 data 分在哪一類，最後在預測該筆 data 結果的時候，用這 11 個 model 的眾數當作最後的預測結果。

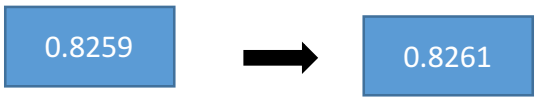
伍、實驗與討論

一、Model 改進過程

在一開始，我們使用的是 random forest，用了一個 random forest 然後過了 simple baseline。後來我們用了五個 random forest 的 model 做了 ensemble 之後，得到了不錯的效果，過了 strong baseline。在聽完上台分享的組別報告之後，發現 XG-Boost 這個厲害的東西，於是我們也決定來試試看 XG-Boost。起初我們用了六個 XG-Boost model 做 ensemble 結果準確率突飛猛進到 82.51%；後來我們決定試試看用之前的五個 random forest 的 model 加上這六個 XG-Boost 的 model 分別做了兩種不同的 ensemble，結果也都比單純用 XG-Boost 要好上許多。



5 個 random forest	5 個 random forest
6 個 XG-Boost	6 個 XG-Boost
Ensemble: 幾何平均	Ensemble: 眾數



二、Model 的優缺點

在試的過程當中，我們發現如果是將相同方法的 model 做 ensemble 的效果會比不同方法的 model 做 ensemble 的效果要差，這可能是因為不同方法的 model 優點不同，所以才會有這種結果。為了要知道 random forest 跟 XG-Boost 兩種方式個別的優點是什麼，我們決定將這兩個方式的 confusion matrix 畫出來。

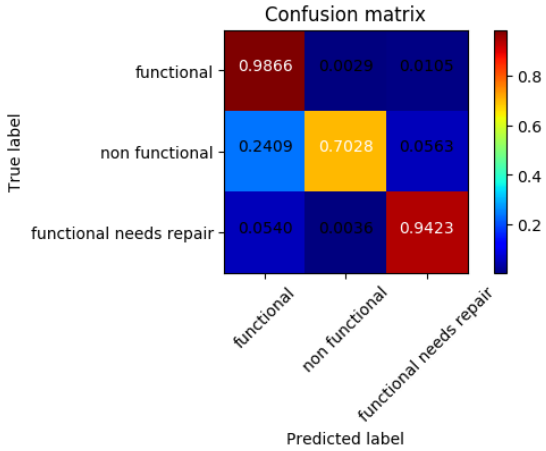


圖 1：confusion matrix of random forest

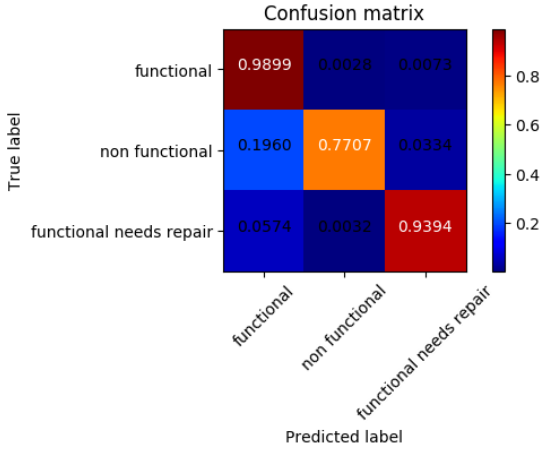


圖 2：confusion matrix of XG-Boost

將這兩種方式的 confusion matrix 畫出來之後，可以發現到 random forest 的 model 容易將 non functional 的 data 辨識為 functional，而在辨識 functional needs repair 的部分，random forest 的效果是比 XG-Boost 要好的；而雖然在辨識 functional needs repair 的部分 XG-Boost 是比較差一點點，但其在辨識 non functional 的準確率是比 random forest 高出許多的。

了解到這兩種 model 的優缺點之後，我們決定將他們 ensemble 起來，發現 functional needs repair 這類預測的結果居然比單純用 random forest 預測的結果還要高，而且 non functional 的部分準確度也接近單純使用 XG-Boost。

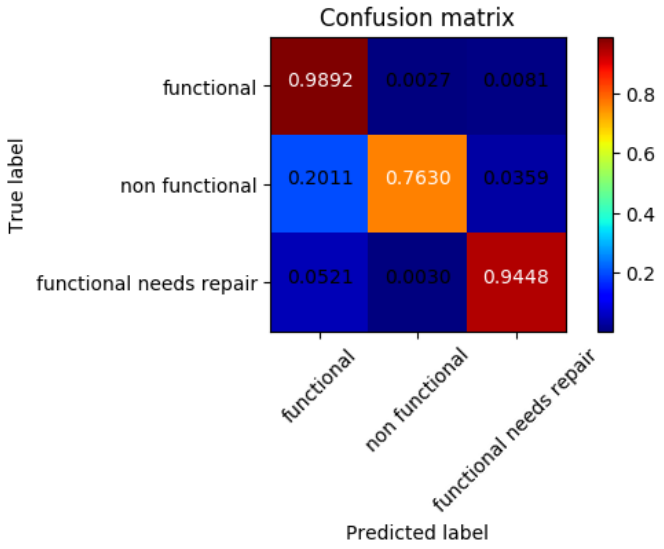


圖 3：confusion matrix of random forest and XG-Boost ensemble

三、Ensemble 方法討論

最一開始我們使用的方法是幾何平均，因為我們在網路上看到有人說在機率模型要做 ensemble 用幾何平均是比較合理的。在機率模型當中，幾何平均代表的是機率模型之間機率的分佈會落在哪，也就是說每個模型輸出的機率都會對這個平均有影響。

後來我們決定使用 hard label，先讓每個 model 預測結果分類，再用投票的方式決定這筆 data 應該是被分在哪一個類別。結果發現這個準確率比用幾何平均還要高，我們的猜想是因為兩種 model 的優點不同，也許 random forest 裡面會將很多的 non functional 誤判，而誤判的 non functional 機率又非常高，所以如果用幾何平均的話，就會將 XG-Boost 預測的準確結果變成錯誤的，所以 hard label 的效果才會比較好。

四、Feature Importance

我們想要知道在 XG-Boost 中哪個 feature 的重要性是比較高的，藉由瞭解 feature 的重要性，才有辦法利用這個資訊來調整我們的 model。由下表可以發現經度的重要性是最高的，其次是緯度；這兩個 feature 會比較重要其實並不難推測，因為在不同的氣候區的結果一定會有差異。

另一個有趣的發現是 funder 意外地扮演著重要的角色，起初我們沒有記錄下 random seed，所以導致 funder 轉換出來的結果不一樣，準確率自然也相差得很多，後來我們才發現這是一個重要的 feature。

而另一個重大發現是我們新增進去的 date_recorded（該筆 data 建造日期與所有 data 中最早的建造日期相差多少天）的重要性居然名列前茅，這也不難想像，因為建造日期跟功能性雖然不一定呈現線性關係，但是這兩者間很明顯是呈現正相關的，當建造時間久了，變成 non functional 或者 functional needs repair 的機率自然會上升，所以 XG-Boost 的預測準確率才會這麼高！

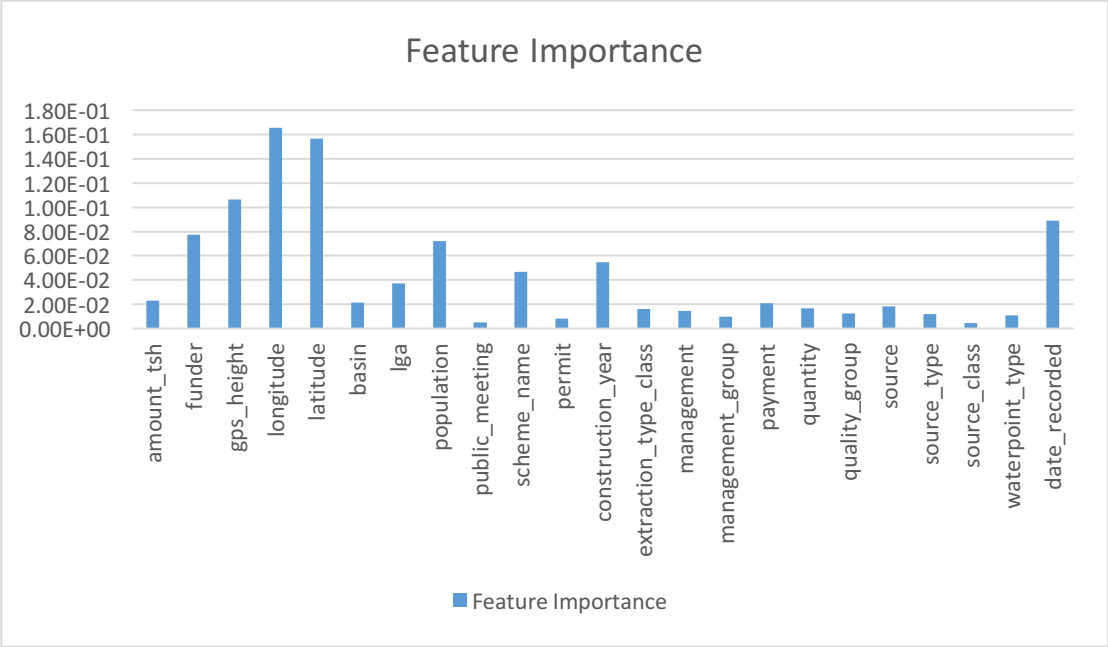


圖 4：feature importance