課 程 ：機器學習
教 授 ：李宏毅 老師
題 目 ：期末報告 Pump it Up: Data Mining the Water Table
組 別 ：大guy4這young
學 生 ：張君澤 張曜庭 陳璽文
學 號 ：R04945022 R05922023 B02901017
日 期 ：20170625

Preprocessing

一開始我們把資料攤開來看，真的是讓人眼花撩亂。但仔細一看會發現一些小端倪，例如裡頭有payment的欄位，卻又有payment_type的欄位，兩者提供的資訊其實是差不多的。因此類似的欄位我們保留一個即可，包括water_quality與quality_ group，waterpoint_type與waterpoint_type_group，extraction_type_group與extraction_type_class...等。此外，我們還發現很多空白欄位，這些空白欄為我們就幫他補上整個欄位的中位數。後來又發現了有些相同人名或是相同地區會有時大寫有時小寫，因此我們在後來進行xgboost的時候也統一把他轉成小寫，這樣一來也減少了我們利用pandas的get_dummies所產生的不必要欄位。

Feature Engineering

起初我們用了dnn還有random forest來通過我們的simple_baseline。後來random forest可以輕易的讓我們過baseline，我們就選擇使用random forest 裡頭的feature_importances來觀察哪些參數是比較重要的。當時我們發現最重要的幾個參數有：quantity_dry、longitude、latitude、data_recorded_offset_days。我們就分別把他們的重要性提高，新增了一個欄位是longitude*latitude，以及longitude的平方，還有latitude的平方，最後還有quantity_dry*payment。提高這些重點欄位的重要性後，我們實作的random forest的validation也有提高，在leader board上的成績從0.8160進步到0.8211。

Model Description

這一章我們會稍微簡介一下我們所使用的模型，以及他的特性，而一些參數的調整我們則會放在第三部份的experiment來討論，這裡我們使用了五種方法來做嘗試，分別是random forest、gradient boosting、xgboost、dnn以及svm，以下分述之。

Random Forest

Random forest 是一種基於decision tree的bagging，簡單描述一下他的演算法的核心，假設原本有N個樣本，那它每次都會隨機取N個samples，但這個過程是取後放回的，所以每次sample的結果會不一樣，建出tree也不相同，用這種方法來增加資料分佈的歧異度，此外除了原本bagging會針對資料sampling外，它也會對於樣本本身的資料維度做sample，假設原本資料有M個feature，它會隨機則m個，m<M，並以此m個feature來做一棵tree，這樣的方法可以減少overfitting的機會。

Gradient Boosting

原始的Boost算法是在算法開始的時候，為每一個樣本賦上一個權重值，初始的時候，大家都是一樣重要。在每一步訓練中得到的模型，會使得數據點的估計有對有錯，我們就在每一步結束後，增加分錯的點的權重，減少分對的點的權重，這樣使得某些點如果老是被分錯，那麼就會被「嚴重關注」，也就被賦上一個很高的權重。然後等進行了N次疊代，將會得到N個簡單的分類器，然後將它們ensemble，得到一個最終的模型。 而Gradient Boost與傳統的Boost的區別是，每一次的計算是為了減少上一次的殘差(Residual)，而為了消除殘差，我們可以在殘差減少的梯度(Gradient)方向上建立一個新的模型。所以說，在Gradient Boost中，每個新的模型的建立是為了使得之前模型的殘差往梯度方向減少。

XGBoost

XGBoost的方法跟GBST的方法很像，只是它做一些修正，主要有三點，第一點是，傳統boosting最耗時的部份是要將feature的值做sorting，來選擇最佳分裂點，選擇影響最大的feature做分裂，所以在疊代過程中，需要不斷的花資源去計算。而XGBoost則是做了一些調整，它預先排序好特徵並把這個結構存起來，之後的疊代過程都是基於這個結構，所以可以平行執行嘗試各個feature，最後再選擇影響最大的，大大的提昇他的速度；第二是它有加上了regularization做為最佳化的目標，以避免出現overfitting的問題；第三點則是它支援，2nd order 的導數，這一部份他是用泰勒展開式的，這也使的它可以擁有更高的複雜度。

DNN

DNN則是借鑒了人類的神經網路的概念，它會有很多層layer，每一層node之間都是fully connected，訓練過程是用gradient descent的方式去降低我們的losses，因為很多層所以會使用back propogation的去做最佳化，因為他的參數比較多，而且會利用hidden layer，來尋找features之間交互影響的特性，所以可以做不錯的結果，但

是通常在scalar上的feature上表現會比較好，如果是比較像label上的，可能decision tree類型的，表現會比較好。

SVM

SVM則是基於PLA的演算法去做調整，它除了考慮分類對錯外，它設計了一個叫margin的參數，代表我的資料點，和我分類的線的距離，去最大化這個值。代表我的分類器擁有更高的容錯性，SVM可以去做大部份二元分類的問題，但今天是muti label classification時，就可以採取兩種策略，第一種是一對多，一個label對上其他不屬於這一類的，如果有n種label，就去嘗試n次來分類，另一種方法則是一對一，每次選其中兩個label來做二元分類，最後嘗試n*(n-1)/2次，有一些靠近邊緣的點，則用他的margin大小，來判斷屬於那一類，在這個task上，我們是選擇前者來當作是一對多來做我們的svm。

Experiments and Discussion

在這一章，我們會討論到我們在上一部份所提到的model的表現，以及再進一步說明模型有哪些參數是我們調整的，最後我們如何取得最高分數0.8260。

Random Forest

我們random forest是用scikitlearn裡面的模型來實做，testing分數最高分是0.8211。我們所調整的參數分別描述一下，n_estimator代表我們要去建幾個tree來提供我們做voting，設定為777個，min_sample是代表我們最低一個node要有幾筆data，設定為8，我們才會把它split，max_feature代表我們最多會在一次split考量幾個重要的feature，這裡我們設定為auto，最後是max_depth，代表我們的樹最深會長到多深，設定為50，我們認為最關鍵的兩個參數是min_sample和max_depth，因為他們會決定了我的tree的深度和廣度，因為decision tree本身是一個很容易overfitting的模型，它一定會將最後的trainning accuracy衝到100%，所以適當的調整這兩者是決定我們數的表現的關鍵另外n_estimator，並不是無限制的voting，表現就會一直提昇，因為很多時候它會在大概200以上以上之後，就不會進步了，如果我們單純只有一個estimator，也就是視為一個decision tree，我們的validation 上準確率就只有0.7534而已，這一些參數的取得是我們這裡事先用手動調整之後，再去用grid searching微調，得到不錯的結果。

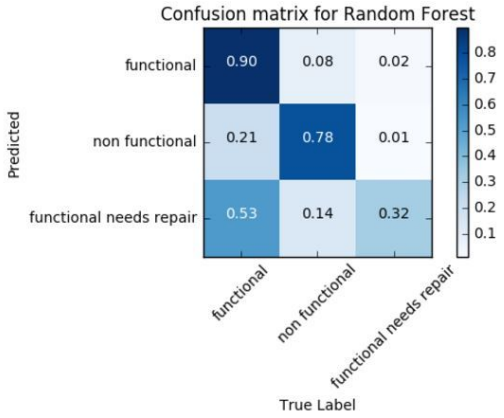


Fig. (3-1) confusion matrix of random forest

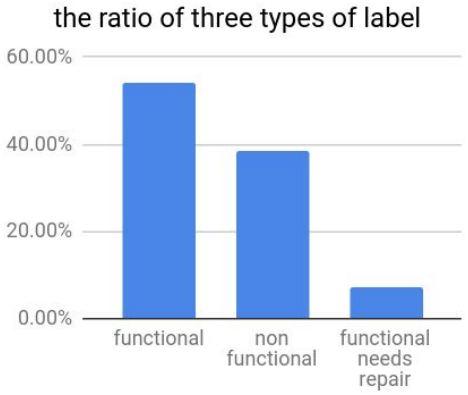


Fig. (3-2) the ratio of three types of label

圖(3-1)代表我們使用randome forest所繪成的confussion matix，可以發現functional needs repair這一類別的判斷最不好，後來我們將各種label的分佈結果統計出來，發現結果是分別佔的比例是54.3% 38.4% 7.3%，圖(3-2)，可以發現相較於前兩者functional needs repair，所佔的比例不到前面的1/5，所以這一種tag訓練起來是比較不好的，比較容易被誤判成functional，另外non functional，也會比較差，也可以假設或許是因為後兩種類型是比較容易混淆的，其原因後續會談到。

圖(3-3)代表我們將在做判斷哪些feature當我在做split時最重要的前十名，可以看出前幾名都是scalar性質的，l經緯度，這一些很好想像是基於地緣關係，另外quantity_dry，代表乾度，一個水井如果快乾了，代表它也快失去功能，而，date_recorded可以想像成當我一天說它壞了之後，它往後一段時間內的表現都是壞的，而contructuon_year則可以想像成他的 建立時間越早，則越有機會壞掉，這也符合直覺的。

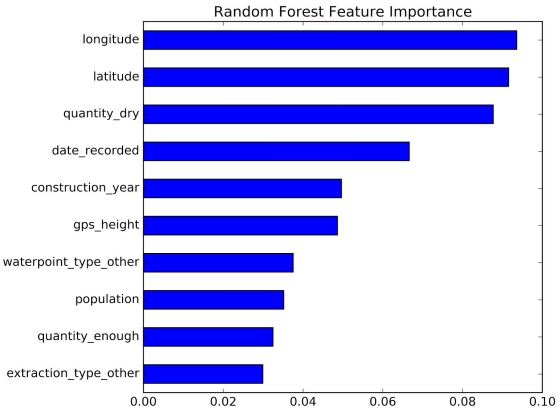


Fig. (3-3) top 10's important feature of random forest

Gradient Boosting

我們Gradient Boosting是用scikitlearn裡面的模型來實做，testing分數最高分是0.8230。我們在這裡有調整的參數有learning rate為0.02，max_depth也就是我們的深度為14，min_samples_leaf為30，也就是一個leaf node最少有30筆data，min_samples_split為100，也就是一個internal node最少要需要幾筆，才會split它，subsample代表每次我要隨機則幾筆資料來做我的classifier，最後n_estimators我們設為800。

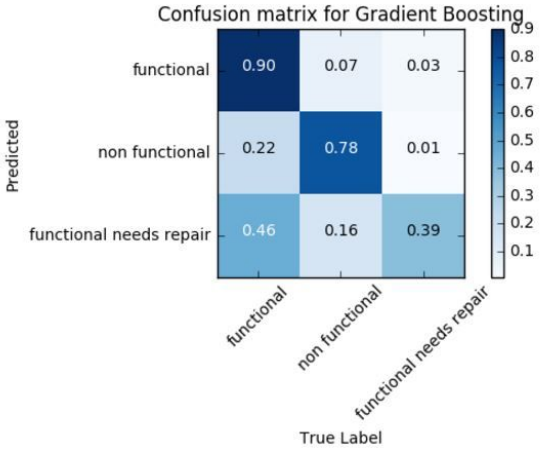


Fig. (3-4) confusion matrix of gradient boosting

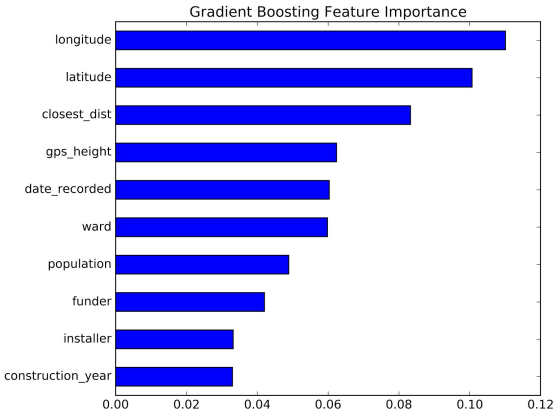


Fig. (3-5) top 10's important feature of gradient boosting

可以從上圖(3-4)看出說如同我的random forest一樣，對於functional needs repair 這一類的誤判情形是最嚴重的，但是它這裡比起我的random forest還是好一點，顯示最後決定我的準確度的關鍵是對於這一類稀少類別的判斷；圖(3-5)則代表那一些feature是我判斷的重點，前幾名都差不多的，只是這裡比較會考量到我的installer和founder，這一點比較有趣，推測可能原因，是因為某一個節點屬於同一個工程人，其鄰近地區，所以它也隱含了地區性的特性，。

XGBoost

我們xgboost是用DMLC所開發的xgboost套件，在testing表現最好的結果是0.8201，它跟前面的gradient boost最大的不一樣是，它跑得速度快非常多，在16個thread情況下，100個epoch下，只需要5分鐘，時間上來說比起gradient boost優異很多，而準確度上並沒有太大的落差，這裡我們調整的參數，主要有eta，也就是我的learnig rate為調成0.4；gama我們調成0.6代表我們最低要判斷來split node的loss，max_depth也一樣，代表我分割程度，有點像decision tree裡面的樹的深度，我們設為14；另外min_child_weight則可以想像成，一個node裡面最少必須要包含幾筆data，如果越高，代表我越不傾向去分割它，模型也會比較保守，這裡我們是設成3，這兩者也是主要控制我overfitting的情形。

我們在xgboost上的表現情況，和前兩者類似，但是對於non reapiar的預測降低，此外對於functional上的表現也提昇了0.02，這一種擅長的預測上的差異，導致我們有了下一階段的想法：essemble，因為每一個模型在預測上的分佈是有落差的，可以利用的一概念，綜合很多model，或許有比較優異的表現。

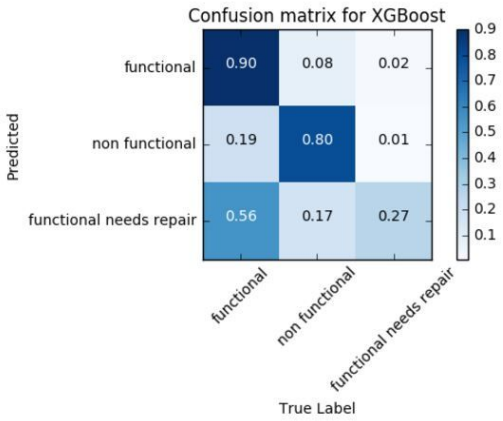


Fig. (3-6) confusion matrix of xgboost

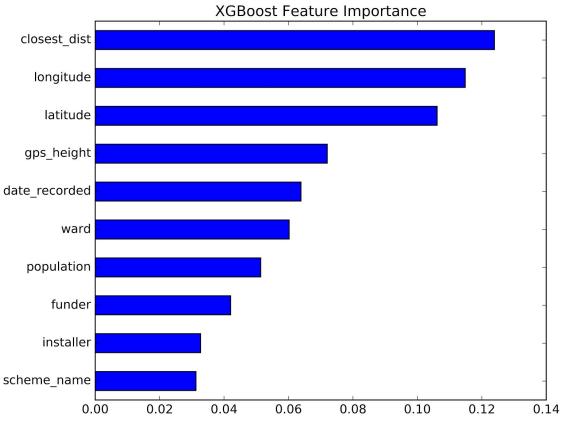


Fig. (3-7) top 10's important feature of xgboost

DNN

我們所使用的DNN，是直接用keras所提供DNN model，模型設計如下，首先第一層是直接接到一個batch normalization，為什麼這樣做的原因，是因為，我們的資料有些是scalar，有些是binary的形式的label，如果直接傳進去，會發現他的weight基本上會調不出來，準確率大概都維持在0.55左右，所以將其normalize當mean等於0，variance等於1時，就可以train的起來，我們在DNN模型上，取得的testing準確率最好是0.791，trainig準確率也只有0.812，並不是很理想，其訓練過程如下圖。

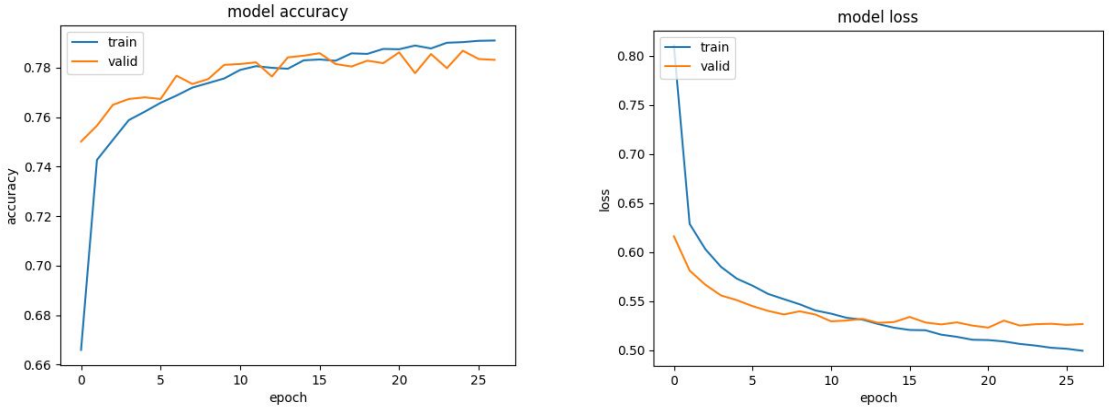


Fig. (3-8) the training and validaiotn accuracy/loss v.s. epoch

可以發現這個模型很快就到達他的準確率極限，之後就開始上下震盪，推測可能做不太起來的原因，是因為DNN對於有連續性質的fearture比較會訓練比較好，但是我們的feature是從原本的類別轉成binary的檔案，所以會train不太起來，但是DNN是一個比較不容易overfitting的模型，所以可以由此評估，這個dataset準確率的極限，也大概在0.8左右。

圖(3-9)則代表我用DNN做出的confusion matrix，可以看出後兩者的預測情形，都下降了，其中，non functional 有三成都誤判成了functional，這也顯示了，另一方面我的funtional的準確率確略有提昇，可以想見，DNN是比較傾向將模型判斷成

functional，可能原因是因為，DNN每一層，都是用前面一層的layer的值做線性組合，所以不會像tree base的方法，會在切割過程中，比較難換成別的feature判斷，導致最後出現overemphsize特定幾個features的情形這也說明他的training也無法提升到100%。

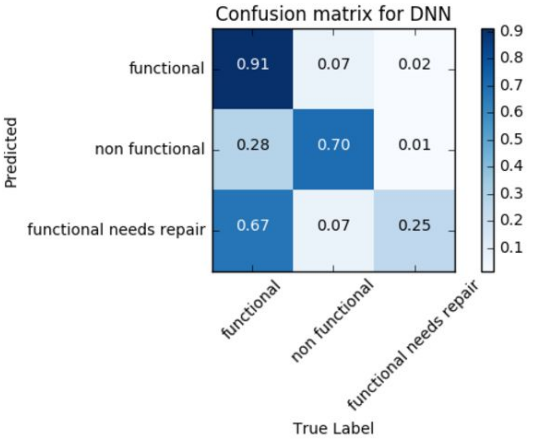


Fig. (3-9) confusion matrix o DNN

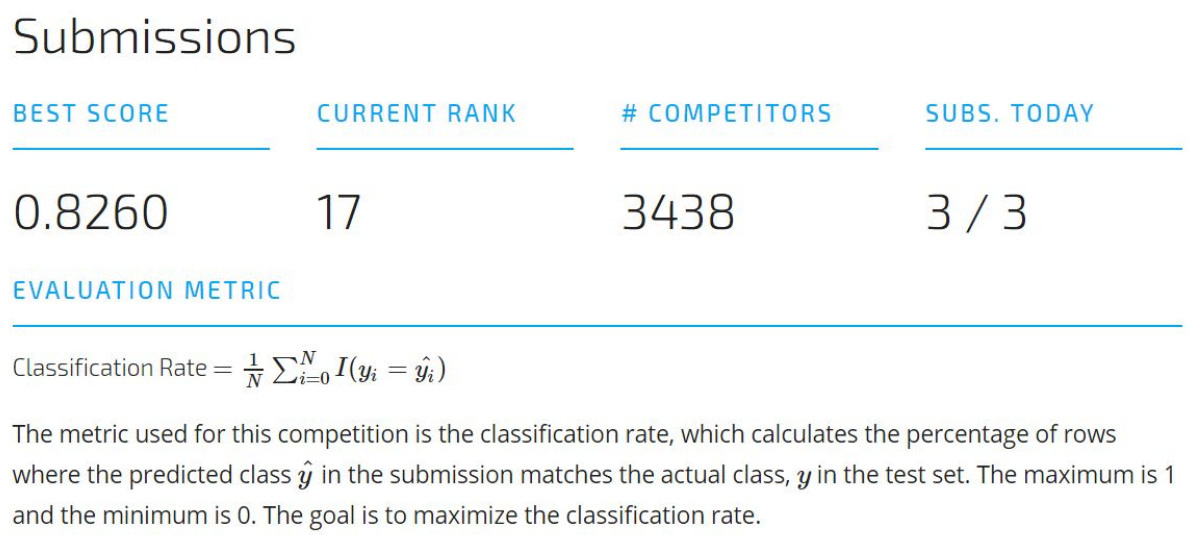
SVM

我們直接使用了sklearn裡頭的LinearSVC的套件，利用Grid-Search發現了Class_weight在balanced的時候會比較好，但如此大的data_set使用svm跑非常慢，sklearn的SVM沒有做multithread的處理，30分鐘都跑不完。而且不管參數怎麼調，我們在Validation的結果也一直無法突破0.75。在datadriven的討論區也看到SVM對於這樣的classification效果很差的相關討論，因此果斷放棄使用SVM。

Voting

最後我們得出最佳結果的方法，是利用hard voting的方式，使用了2組random forest 、gradient boost、xgboost三種去做票選，取多數決，如果結果是二比二，例如兩個選functional，另外兩個non functional，則用gradient boost當作是結果，稍微講解一下為什麼前面的三種方法，都已經是boosting或bagging的方法，裡面都已經隱stacking的觀念了，為什麼這裡esembling還會有用，因為這三種模型，可以從他們的confussion matrix和important features可以看出，其實他們所看到，用來判斷的方法是有差異的，利用的靶心理論來講，就是就是他們預測的分佈，事實上是有所不同的，即使準確率相同他們事實上會有接近10%的label預測不一樣的，所以如果今天我們將不同模型，再去essemble後，因為三角不等式的關係，是有機會得到比較好的模型的，如果都只有使用同一種模形所作做出來的結果去做voting，反而不太會進步，甚至會離最佳解更不好，因為一些錯誤，比較原理靶心的答案，把它的預測從中心給拉走了。

Ranking and Conclusion



在寫報告的當下我們的leaderboard排名在所有參賽者的第17名，我們有利用了兩次上傳扣打來探討testing set整體的分布。全部的label都是functional needs repair的分數為0.0719，換算下來的個數是1067，全部的label都為non-functional的個數為5672。相較起來我們在上頭最好的結果functional needs repair也只有498個，non-functional只有4972個，或許如何將這兩者從functional中區分出來，才是更進一步的關鍵。

我們的程式碼執行方式可以從我們的readme.md裏頭找到相關資訊，也謝謝所有助教們的耐心批改。