無檔案的惡意程式辨識競賽

設計文件

隊伍: [點點]

成員：[顏天保]

# 摘要

這次比賽我大部分的時間是花在特徵萃取上，剛好這次比賽的資料6G可以讀進sql server(限制是10G)，因此我用sql server2012做出462個變數，之後用這些變數跑lightgbm的分數大概在0.945~0.955分徘徊，最大的突破是顧客信用轉換，我將顧客的軟體使用紀錄，以一個sparse matrix(定義為A)表示，維度是5539813(顧客)x81894(軟體)，有使用就填1，其餘為零，用這個矩陣來對原始的462個特徵變數轉換，轉換方法如下

其中

是A矩陣的列加總的倒數的對角矩陣(維度是5539313x5539313)

是A矩陣的欄加總的倒數的對角矩陣(維度是81894x81894)

概念上的意義是，代表，我將每個顧客曾經使用過那些軟體，這些軟體的462個變數的平均當作是顧客的變數值，公式上來看是變成5539313x462的矩陣，代表5539313個customer的462個變數值。接著作用在，代表，考慮每個軟體曾經被哪些顧客使用過，這些顧客的462個變數的平均當作是軟體的變數值。我之所以這樣做的起源有兩個，其一，我認為這462個特徵是好的特徵，但每個file的log數有多有少，並無法很好的萃取出462個特徵，例如說81894個軟體中有50個軟體只有一筆log，根本無法計算出時間(以秒為單位)的variance，而這又剛好是很強的特徵，因此適度的考慮軟體之間的相關性來輔助特徵的萃取是必須的。

第二個原因是我認為A矩陣的資訊一定有用，在比賽最早期資料探勘時期，我假設如果可以知道所有軟體的好壞Y向量，我可以定義顧客信用，也就是顧客使用過的所有軟體的平均，然後再用信用重新組合成Y，也就是 ，也就是將每個軟體被那些顧客使用過的平均信用。基於這個構想，我設計了一個迭代預測:

重設的1~52518元素為trainning的label

而要預測的部分(52519~81894的Y值)初始值我是用0.5，也就是，的1~52518元素為trainning的label，52519~81894是0.5。跑了100次迭代的分數是0.923，初始值和跌代數會稍微影響，浮動大概是落在0.920~0.927左右，因此這個A矩陣應該是有用的(我有嘗試過將t(A)矩陣直接當作特徵，跟462個變數一起預測，但cv error幾乎不變)。 至於為什麼是九次是cv error的結果，462個變數F加更新一次的變數的cv是0.9723左右，leaderboard的分數是0.9620左右，加入更新兩次是0.9736左右，leaderboard的分數是0.9650左右，加入更新九次是0.9757左右，leaderboard的分數是0.9665左右(注意到我都是用5-fold cv的五個model 平均預測，如果用全部資料train單一模型的預測反而分數會變差)，再往上更新的cv變化不大，而且我的電腦memory不夠，(4620變數幾乎是極限)

為了效率，我篩選4620個變數的前600重要變數(篩選前後cv error大致上不變)，然後盡量跑一堆cv(250次)當作stacking用，然後再用250個預測再跑一層lightgbm，重複10000顆並平均作為最後預測

# 環境

Windows 10 x64

SQL server2012

R (x64 3.4.3)

Python (anaconda3)

# 特徵

# 我用sql創造462個變數，再用顧客信用轉換9次(參考前面章節)，加原本的共有4620個變數，最後再以lightgbm選取最重要的600個變數做為最終變數。

變數的意義、創造方法、顧客信用轉換請參考附檔code1.txt

# 訓練模型

最初我嘗試logistic regression，RF, libSVM，deep learning，GBDT這幾個相當常見的模型，這部分我是用R來做(較熟悉)，套件就是非常有名的libSVM 和h2o，但使用462個變數預測的效果皆不太好(只有GBDT的cv error有達到0.94左右，deep learning大約是在0.925左右)，如果要加入更新的變數(4620個)，根本跑不動。因此我後來是考慮兩個樹類的模型xgboost和lightGBM。這兩個模型的預測效果大同小異，但是考慮效率我選擇後者，這部分我是用python來做(網路上範例較多)

libSVM

<https://c3h3notes.wordpress.com/2010/10/20/r%E4%B8%8A%E7%9A%84libsvm-package-e1071/>

h2o

<http://h2o-release.s3.amazonaws.com/h2o/rel-lambert/5/docs-website/Ruser/Rinstall.html>

xgboost

<http://xgboost.readthedocs.io/en/latest//model.html>

<https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf>

lightgbm

<https://lightgbm.readthedocs.io/en/latest/>

# 訓練方式及原始碼

細節請參考code2.txt

將4620個特徵(原始462個加上顧客信用轉換九次後的)丟入lightgbm，以5fold-cv篩選前600重要的變數後，在跑了250次5fold-cv，並將這250個結果當作feature跑stacking model，重複lightgbm 10000次 (每次隨機抽80%資料)最後平均作為最後預測(注意到每個stacking model出來的預測值不會落在0~1之間，而是會類似0.3~0.8之間，因此要先minmax normalize後再平均才是最後的預測)。

# 結論

雖然我最一開始的特徵萃取有462個變數，但應該是可以控制在100變數以內(ng系列的變數其實用處不大，clust和時間有關的變數是關鍵)，我幾乎沒有畫圖和研究變數間關係是我覺得可改進的地方。而特徵信用轉換是我覺得最有特色的地方，但未來還需要改良以便應用在更大的資料集或是設計成新的模型，而stacking的技術我並不純熟，因此我只是單純的以相同的lightgbm(因為我沒做出其他更好的模型)跑cv做出250組預測當作stacking要用的資料，然後跑10000顆(每次抽80%資料)bagging，平均作為最後的預測。這部分可以改成更多層的stacking，更多的模型，更多的平均方式(我有試過rank 平均)