Machine Learning 2015 Fall

Final Project Survey Report

Team: JoJo 北極熊
組員:電機四 B01901046 謝沅廷
電機四 B01901054 林其政
電機四 B01901077 林奕辰

一、資料描述

本次期末專題是要用使用者的操作紀錄，去預測他會中途退出(dropout)一門課的機率，有兩種評估的方式，分別是 MAP 和有權重的正確率。資料是由老師所提供的跟 KDD Cup 2015 一樣來源的資料，他是一個大量線上課程網站(MOOC)所提供的註冊以及使用者操作的資料，分別有

(1) enrollment:描述使用者和課程及註冊 ID 對應的資料

(2) log:記錄每一個使用者在註冊的課程的三十天的操作紀錄(包含類型以及時間等)

(3) object:描述每樣課程的模組的相對位置，每一個模組是屬於哪一門課，是影片或是問題等

(4) truth_train: training data 的真實結果，觀察有 79%的 dropouts 跟 21%的 non-dropouts

總共有 96434 筆的有 Label 的 training data，還有 24109 筆 testing data，要從這些資料中去預測準確的結果，就需要抽取出有意義且有效的特徵來幫助機器學習。

二、特徵抽取

因為我們相信特徵抽取是本次專案非常重要的一環，因此我們花了非常多力氣在這上面。

下面敘述我們所抽取的特徵。

在本實驗中，特徵可以粗略的區分為三種特性：第一種是與課程相關，第二種是與使用者相關，第三種則是與操作時間相關。單一個特徵可能同時具有此三種特性。

基本特徵抽取：

在本實驗中，我們可以粗略的將每個註冊事件(enrollment)依照使用者、課程、時間區分成幾大類。課程名稱、使用者名稱、各個時間點各個課程的退出率(dropout rate)都屬於此類特徵的範疇。然而，這樣的特徵並無法提供各個註冊事件本身客製化、獨一無二的特性。而這樣的特性將在之後的相關特徵來補足。本階段中，總共抽取了 6 維的特徵。

自操作記錄(log)中所抽取的特徵：

在本實驗中所提供的數據，絕大部分都是操作記錄。因此，如何自操作紀錄抽取有效的資訊，對於最終預測的結果具有十分大的影響力。本部分所抽取的特徵可以分為四大類：第一大類是時間軸上特徵的抽取，在這裡，我們抽取了 30 天之中每天的操作記錄次數、平均的使用時間、最大及最小使用間隔時間(以日為單位)、總操作期間、最後空閒時間、平均每次使用的操作數目（我們定義再間隔一段時間內的操作合是做同一次使用，倘若兩次間隔時間在一小時或是三小時，則視做是不同次使用，不同的間隔時間會產生不同的特徵組合），一日之內各個時刻的操作，一週七天各自的操作數目等此類與時間相關的資料在這裡被抽取出來。同時，我們也對三十天的每天的操作數目做一次及二次微分，希望可以更精準地描繪出操作記錄的變化。而第二部分則是將 30 天的操作紀錄轉換至頻譜上，透過 FFT，我們可以在頻譜上觀察操作記錄的變化。而第三部分則是時頻分析，透過 Hamming Window，我們得以取得一段時間內的頻譜變化。第四大類則是依照不同的課程或依照不同的操作行為，統計不同課程或不同操作行為分別的次數。另外為了

消除課程本身模組數量的多寡可能對操作次數造成影響(如使用者在一門課上的操作次數雖少，實則使用者已無一遺漏的接觸過該課程的每一模組，操作次數少僅因課程本身模組數量較少)，我們也抽取了操作次數占課程總模組數的比例。在本階段，我們共抽取了 430 維的特徵。

擴展特徵：

在本實驗中，我們希望預測 31～40 天是否有操作記錄。因此，我們決定延展操作記錄的統計，設法製造出 31~40 天的操作紀錄。我們設法取得同一使用者在其他註冊事件中的操作記錄，倘若其他註冊事件在本次註冊事件的 31~40 天有操作記錄，那麼我們就將該註冊事件落在此區間內的特徵複製作為本註冊事件 31~40 天的操作記錄。透過這樣的方式，我們在本階段，共抽取了 13 維的操作記錄。

課程相似度：

在本實驗中，我們希望得知各個課程之間的相關度，而這樣的特徵，我們透過統計該兩兩課程之間重複修課的比例之多寡來決定兩兩課程之間的相關度。倘若兩兩課程之間重複修課的人數越多，則兩兩課程之間的相似度越高。同時，我們也可以依此來判斷該註冊事件的課程在同一使用者之中的其他註冊事件的課程之相關度。本階段，共抽取了 3 維的特徵。

使用者學習深度：

在本實驗中，我們希望得知使用者在某一門課上學習到何種程度，我們推測當使用者學習的越深入時，可能越不會退出該課程，是以抽取此特徵。我們將使用者在一門課上接觸過的模組依照時間排序，再算出一模組被使用者接觸的平均順位，藉以表示此模組在這門課中的順位。於是我們便可利用此順位觀察使用者在這門課上學習到了什麼程度。我們抽出了最大學習深度和平均學習深度，共 2 維的特徵。

參考同一使用者其他註冊事件的退出率重新排序：

本階段，我們希望可以優化 MAP 的分數。因為 MAP 的特性，倘若我們有更多的參考資訊來判斷我們前一階段的預測分數，我們便可以更確定前一階段的預測分數是不是應該排在這個位置。本階段中，我們抽取出同一使用者其他註冊事件的退出率，以此產生出 20 維的特徵。10 維分別是 10 個不同時間該使用者的平均退出率。另 10 維則是 10 個不同時間中，分別有幾個註冊時間。經過試驗，倘若以此 20 維透過 GBDT 判斷是否會退出，準確率大約在 80%與全部猜退出的 79%相去不遠，並不是非常有用的特徵。不過倘若在此 20 維中加入上一階段對於此註冊時間退出率的預測。在最後，可以有效提升 MAP 的分數達 0.0005 至 0.001。

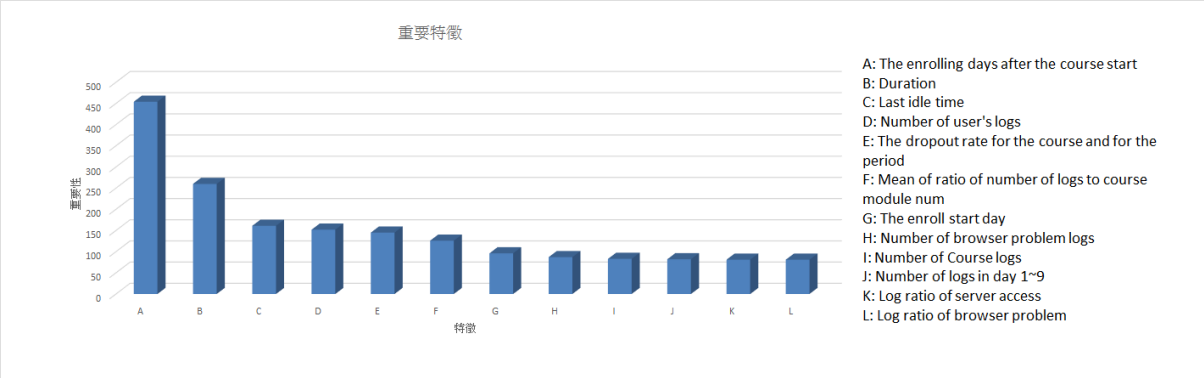
圖形化特徵抽取:

這邊使用 python 的 networkx [1]來做，我們將使用者和課程做成一個 bipartite graph，edge 的權重是讓有中途退出的是-1，沒有的是 1，然後去算每個點的 latapy cluster coefficient、eigen centrality 和 degree centrality，我們原本是認為這些係數可能某方面可以代表著同一個課程中不同使用者中相同的特性，就是去找到使用者之間、或課程之間的相關性。

課程模型混合: 參考了之前 KDD Cup 第二名的投影片[2]，把每一個課程的 training data 分成十份，用其中九份訓練去預測另一份，如此重複得到每一份的預測結果，再將他們接起來當作新的特徵使用。而 test 的部分的延長，則是用全部該課程的 training data 去訓練，用預測 test data 的結果來當成新的特徵，我們認為這樣為每個課程量身訂做可能可以提高準確率。

特徵效果分析：

在研究過程中，我們發現 XGBoost 有函式可以印出機器在訓練時使用到哪些較為有效而重要的特徵，我們便用了這個函式，希望能對我們所研究問題有更深入的了解，也可以幫助思考未來抽取特徵的方向，其結果如下：



我們可以看到第一項特徵遠遠高過其他項。第一項特徵是最初操作的時間減去課程開始的時間，我們認為它之所以會是一個重要特徵的原因是，當這項特徵的數值很小，表示使用者可能一開始就預定要修這門課了，所以會好好地把它修下去。如果這項數值很大，那麼可能是使用者在學期中才突然發現這門課而對其進行操作，只是走馬看花而已。第二項比第一項少了不少。它指的是操作的總期間長度，也就是最後一次操作的時間減第一次操作的時間。這項特徵之所以會是重要特徵是很直觀的，當總操作期間越短，表示在第一到第三十天中，他有越長的時間是沒有使用該課程的，從他在這三十天的最後的乏於操作的行為來看，可以自然的類推到第三十一到第四十天也很可能是沒有使用該課程的。第三項特徵是最後空閒時間，即課程結束時間減最後一次操作的時間，其中課程結束時間是以該課程最後一個模組上傳的時間做代表。第四項是使用者在所有課程中的總操作次數，我們推測操作次數越多，可能表示使用者月勤於使用此線上課程系統，也就越不會退出。第五項是該課程的退出率，退出率越高，表示該課程越不受歡迎，其他修習該課程的使用者退出的機會也就越大。而第九項課程被操作的總次數和其受歡迎程度也是類似的意思。在後面的幾項特徵中，我們可以發現有不少特徵都與特定類別的操作次數有關，故我們可以知道，在眾多不同類別的操作行為中，只有其中幾項是較為重要的，如 browser problem、server access 等。因此我們後來有想過，將不同類別的操作行為配上之前做過的時間、微分、頻譜等分析再做一次(先前只有對操作的總次數做而沒有分類別)，但因時間不足而作罷。

三、學習方法

抽取完大量的特徵後，首先是把他們用 python sklearn [3]這個套件裡面的 StandardScaler 去做預處理，他會把每一維度的特徵平均值變成零，變異度變成一，再來因為之前特徵抽取沒有處理課程和使用者的 ID，所以用 sklearn 的 LabelEncoder 將這兩維資訊加入，一開始有試過使用 Principal Component Analysis(213 講)去降維，但是不管是哪個方法結果都變差，所以最後沒有使用。接下來就開始去跑各種機器學習的方法，因為我們的評分標準有兩種，分別對應到機率以及辨識正確率，所以我們有用專門辨識的方法，也有用回歸的方法，我們這邊嘗試過 Logistic Regression、SVM、RandomForest，以及 GradientBoostedDecisionTree，下面針對各演算法以及實際使用的套件來介紹

1. Logistic Regression:

   由於將老師的話謹記在心，所以決定先用比較簡單的模型來試試看，原理如同上課所講的(第 10 講)，這邊使用的是 sklearn 裡面的 Logistic Regression，設定 dual 為 false，因為 data 的個數遠比特徵維度來的大，如果解 dual problem 的話，維度會變得跟 data 個數一樣大，這樣會造成運算上太慢，再來是用 sklearn 裡面 GridSearchCV 去找到最適當的參數，它的

原理就是你可以指定一些格子點，然後他用網狀的搜尋，去找到一組參數使得交叉驗證的效果最佳，但是這方法效果做出來並不是太好，可能是因為這模型真的太簡單，所以之後就放棄他了。

2. SVM

   我們使用 sklearn 裡面的 SVM.LinearSVC，沒有用 non-linear 的原因是因為我們的電腦跑一次 non-linear 需要很久時間，因為必須去解 dual form，效果也並沒有很強，這樣如果之後要去調到好參數的話，我們認為需要的 overhead 會太高，所以採用線性的版本，這其實就是上課說過的 soft-margin SVM(詳見 204 講)，我們也有使用 SVM.LinearSVR 來做回歸的計算，參數的調整這邊一樣都使用 sklearn 裡面的 GridSearchCV 去找尋。

3. Random Forest

   隨機森林我們是採用 sklearn 的 RandomForestClassifier 和 RandomForestRegressor，它的原理就像上課說的一樣，他會先 bootstrap 選出一部分資料，再來長出一顆 decision tree，一直重複前面的步驟直到 tree 的數量符合我們設定的值，最後再用這些樹去做 uniform 的 blending。參數的選擇一樣是利用網格搜尋搭配交叉驗證。

4. Gradient Boosting Decision Tree

   我們使用的套件是 xgboost [4]，這個套件本身很快也很有效率，美中不足的是他在 python 端的文件闕漏還滿多的，所以很多參數我們就要去一直搜尋，或是要直接去看他的 source code 在寫什麼，造成一部分的麻煩，它的原理就是上課中有提過的 GBDT(211 講)，它裡面可以調整的參數非常多，也可以根據不同的 error function 去優化，他有 MAP、AUC、accuracy 等等，但是他沒有像 sklearn 裡面有很方便的網格搜尋函數可以呼叫，所以我們自己寫了 shell script 搭配 sklearn 裡面 StratifiedKFold 去找出他較好的參數，而這模型的 output 是機率，所以藉由設定 0.5 為 threshold，就可以也拿來分類，由於這個模型所跑出來的結果大幅超越前面的，所以最後主要都是使用這個模型去預測及調整。

四、實驗結果

(1) 我們最後抽了 482 維，這邊作最後各模型的比較。(track2)

SVM 參數: C-linearSVC, C =1e04, tol =0.1

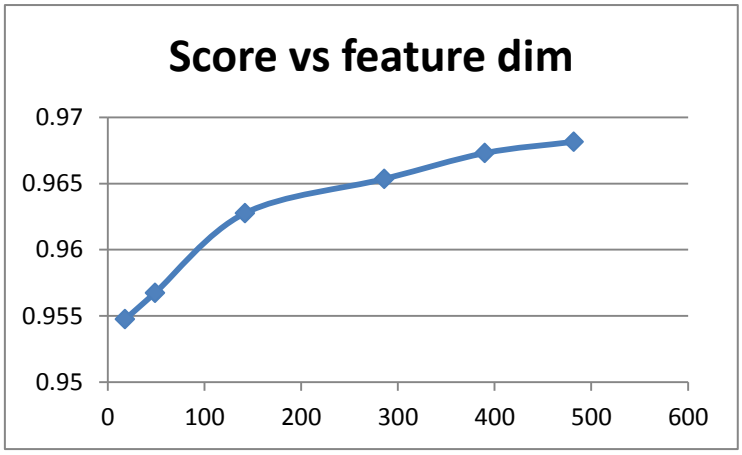
RF 參數: n_estimators=1000,max_depth = 19

GBDT 參數:num_round=500, max depth=5, subsample =0.8, colsample = 0.7

	SVM	RF	GBDT
Public Score	0.875895	0.877943	0.882262

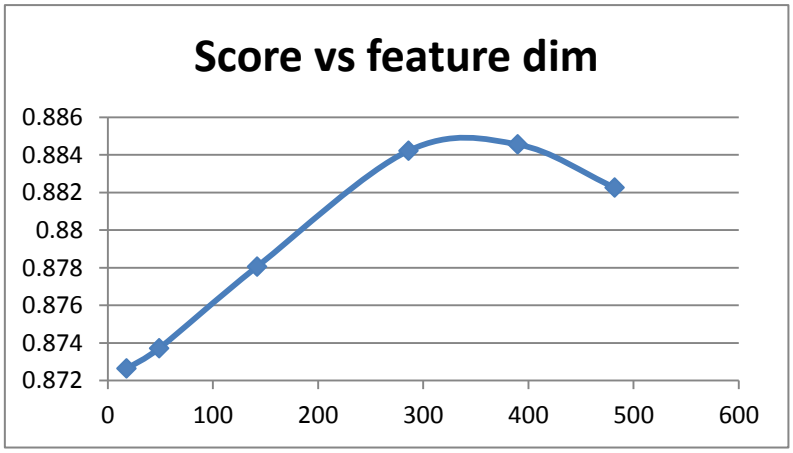
(2) 由於 xgboost 在很早的時候就已經領先別人很多，因此有些中間的 feature 只有 xgboost 的模型有 train 到，所以我們可以比較加入特徵的效果，這邊是 track 1 的結果

dim	Num rounds	Max depth	Colsample_bytree	Public Score (Track 1)
單純助教給的 18	300	4	0.7	0.954746
加以周為單位的 49	300	4	0.7	0.956746
加時間間隔的 142	300	5	0.7	0.962765
加課程相似、頻譜 286	300	4	0.6	0.965347
加周為單位頻譜 390	400	5	0.7	0.967298
全部 482	450	5	0.7	0.968152



(3) 這邊是 track 2 的結果，因為兩個 track 的參數設定不同
Subsample(也就是一次 sample 多少 data)固定在 0.8。
Learning rate 固定 0.05。Objective 固定為 Logistic。

dim	Num rounds	Max depth	Colsample_bytree	Public Score (Track 2)
單純助教給的 18	300	4	0.7	0.872636
加以周為單位的 49	300	4	0.7	0.873718
加時間間隔的 142	300	5	0.7	0.878057
加課程相似、頻譜 286	300	4	0.6	0.884217
加周為單位頻譜 390	500	5	0.7	0.884551
全部 482	500	5	0.7	0.882262



五、Blending:

在本實驗中，單純的 blending 或是 stacking，在大多數的情況，最終的結果表現大多不如 blending 中最好的結果。然而，倘若將最好與次好的兩個結果，加至「參考同一使用者其他註冊事件的退出率重新排序」中產生的 20 維特徵，可以比單使用一種前一階段的分數與 20 維搭配的效果來的些微高上一些。最終，可以再將 MAP 得分 些微的提升 0.0001。

然而，以上的兩種方法，對於 weighted accuracy 來說，都會些微的下降。因此，此二種方法，只能達到 MAP 的優化，對於整體的準確度，並沒有顯著的效果。

六、討論：

綜合以上結論，我們認為，本實驗中最重要的部分是在特徵抽取的部分，由於本實驗提供的資料大多數都是操作記錄，因此如何有效地自操作紀錄中抽所特徵，對於最終機器學習的結果有顯著

的差異。相較之下，許多上課教的方法，我們嘗試 PCA、KNN、K-Means 等特徵抽取或是預處理方式，在最終的表現都不如單用我們自己設計的 feature 來得好，而 blending 也沒有達到預期的效果。我們猜測原因可能來自於，真正具影響力的特徵有限，使得無論使用哪種方式所得到的預測結果，差異都不夠大。

同時，因為在本實驗中，我們最終所選用的模型 XGB，該類的模型對於特徵的多寡敏感性不高，因此在特徵選擇時，移除重要性較低的特徵，並沒有辦法對於效果有所提升。換言之，這類型的模型，本身對於不重要資料所產生的雜訊敏感度並不高。

而在資料的預處理上，我們有嘗試試圖將資料區分為兩群，希望可以降低資料之間的歧異度。希望測試資料與訓練資料的分佈更為相近。在本實驗中，我們有嘗試過將只修一門課與修兩門課以上的使用者相關的資料區分開來，原因在於，我們有在特徵之中，有使用到參考其他課程的資訊。對於只修一門課的使用者來說，這樣的特徵，我們都將這些特徵視做是 0。相較於其他修超過一門課的使用者，這些特徵真正是 0 的情況，我們沒有辦法真正區分出來。不過最終實驗的結果，這樣的操作必沒有辦法產生更好的預測結果。還有其他相關的構想，如我們觀察到這些資料操作記錄的時間，可以粗略的區分為兩大群（如圖），倘若能依照時間將訓練資料區分為兩群，或許在最終的預測結果上會有所進步。不過這樣的想法想到的時間太晚，並沒有足夠的時間落實，效果尚未可知。

在模型選擇方面，由於此問題在面對一個學生及一門課程時，只考慮與目標相關的資訊是較直覺的想法，例如另一個學生在另一個課程的操作紀錄可能就對欲預測的目標較無影響。這時我們就需要能處理 categorical 類型的資料的模型。如 XGBoost、RF 等，可看出其表現優於 Linear Regression、SVM 等模型。

而最後，在寫報告的同時，我們也想到了一些比較特殊的訓練方式。由於本實驗是要預測 31~40 天的是否退出。因此如果我們能將特徵所參考的特徵共三十天的長度，切割成十天。用前十天的資料去預測 11~20 天是否會退出。11~20 天的資料預測 21~30 天是否會退出。21~30 天的資料預測 31~40 天是否會退出。因此每筆資料，我們都可以轉換成 3 個分數，之後再用其他模型去預測綜合以上結果 31~40 天是否會退出。如此一來，或許就可以加權不同時期對於最終結果的影響程度。這是在本實驗操作時，我們發現過的問題，當時卻找不到好的方法，來加權不同時間點的資料。

七、工作分配

謝沅廷: 跑機器學習的演算法、調參數、找尋跟抽特徵有關的文獻/投影片 、抽以課程相關特徵

林奕辰: 跑 blending、抽各式頻譜特徵、抽擴展特徵、退出率排序

林其政: 抽各式基本特徵、時間間隔特徵、學習深度、特徵分析

八、參考資料

以下列出資料包含上面有明確提出的，以及我們有看過的啟發我們想法的東西。

[1] networkx https://networkx.github.io/

[2] KDD Cup 2nd Prize~10th Prize http://kddcup2015.com/information-winners.html

[3] sklearn http://scikit-learn.org/stable/

[4] Xgboost https://github.com/dmlc/xgboost/tree/master/python-package

[5] azure https://github.com/Azure/Azure-MachineLearning-DataScience/blob/master/Misc/KDDCup2015/kddcup2015-sample-experiment-step-by-step-tutorial.pdf

[6] Yehuda Koren, The BellKor Solution to the Netflix Grand Prize