ML Final Project

B02902015 梁智泓
B02902051 林昀宣
B01902126 劉家銘

I. Fetch Feature From Row Data :

A. What feature do we fetch?

User	ID	Course
1. num_of_course	1. Total_log_num 2. Each_log_num (有九個) 3. Max_diff_of_log 4. 所有天的 log 的標準差 5. 有 login 天數的 log 標準差 6. log_frequency_迴歸線斜率 7. log_frequency_迴歸線的截距 8. 每兩次上線間時間差的迴歸線斜率 9. login 的時間差 10. average_log(Total_log / 天數)	1. drop_out 的人數比率 2. chapter_count 3. sequential_count 4. video_count 5. course_log_num

1. User:
   一個人修課的數量會影響一個人 dropout rate，因為修越多，就表示 loading 越重，越不容易將每堂課都修好
2. Course:
   一堂課的好壞與重輕會影響學生的 dropout rate，好的課 log 的次數會相對多，而一堂課的輕重與否跟這堂課的要做的事有關，除此之外，我們還發現，在 test data 中與 train data 中，總共只有 39 堂課，而且 test data 中並沒有 train data 中沒出現的 course，所以我們還可以加入這堂課的 dropout rate
3. ID:
   一個 ID 就是某個 user 在一堂課中修課的代號，能直接影響 dropout rate 就是這個 ID 在這堂課的貢獻，也就是他所做的所有事，包含上線、聽課、寫作業、討論，而我們還利用了一些統計（回歸曲線）的方式，為了要表現出一個 ID 的上線狀況，例如：

上線的趨勢，每天 log 的趨勢等

B. Which feature is useful?

1. Choosing Method:

然而，這麼多的 feature 中，究竟哪個 feature 最能 predict 出正確的結果，我們利用 RandomForest 來做實驗，我們先將 24 個 feature 依序抽掉，再利用 CrossValidation 算出 Performance，然後依照 Performance 做 sort，再將原本 24 個 feature 照著順序做 remove 前 N 個，再算出 Performance 看哪個會達到最好

2. Experiment:

我們最後發現，當 remove 掉 server_wiki, id_standard_deviation(login), browser_page_close, server_navigate, video_count, fre_slope 時，整體的 Performance 會達到最好。

在 Eout 也上得到印證，在 track1 的 public score 上拿到了 95.72%，在 track2 的 public score 上拿到了 87.0116，分別都進步了 0.5%左右。

3. What We Find:

在 data mining 的過程中，我們發現一些看起來沒那麼直觀的 feature(ex: server_access)，卻能有意想不到的 Performance，而一些原本一些以為直觀的 feature(ex: Max_diff_of_log)，反而沒那麼有效。

C. Comparison with Sample Feature

然而，雖然我們增加了某些 feature，這跟原本助教所給的 sample feature 仍互相有好跟壞，由下表可以清楚得知，但大部分來說，我們的 feature 在大部分的時候還是比 sample feature 表現來的好一點

	Track 1		Track 2	
	Public	Private	Public	Private
Sample Feature	0.956329	0.956695	0.872709	0.878280
Our Feature	0.957213	0.958243	0.872304	0.879490

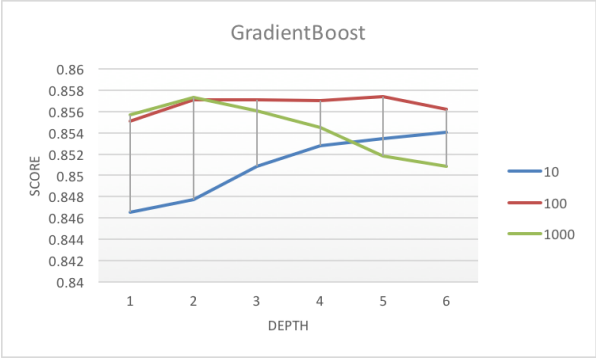
II. GradientBoost

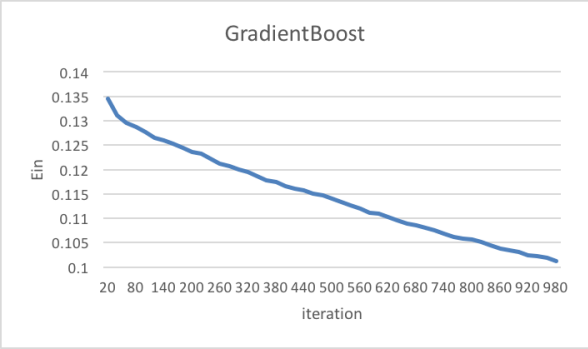
A. Parameter Choosing

GradientBoost 有 n_estimators, max_depth 兩個參數可以調整，我們透過 n_estimators = [10, 1000], max_depth = [1, 6] 來進行實驗，因為怕會有 overfitting 狀況，所以利用 CrossValidation 5 fold 記錄 Performance，並且使用 Performance 最好的 model。

B. Experiment

N_est\dep	1	2	3	4	5	6
10	0.846527	0.847719	0.850861	0.852769	0.853433	0.854076
100	0.855114	0.857063	0.857094	0.857011	0.857384	0.856233
1000	0.855715	0.857312	0.856078	0.854481	0.851806	0.850831





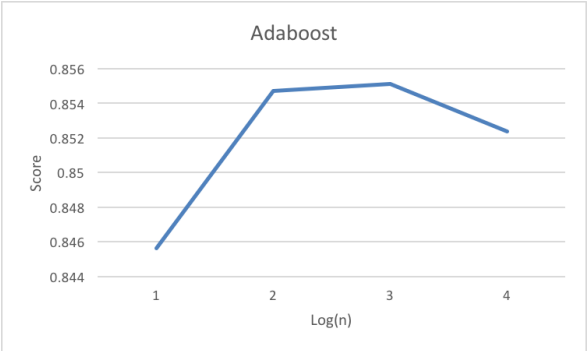
III. AdaBoost

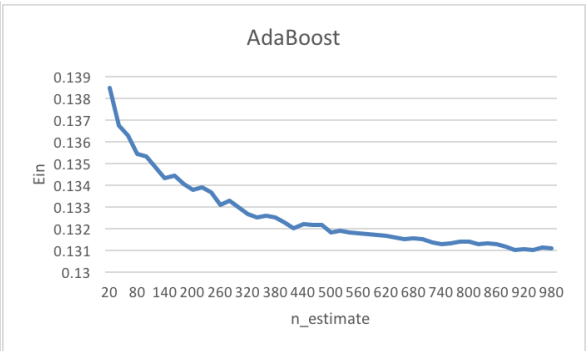
A. Parameter Choosing

GradientBoost 有 n_estimators 這個參數可以調整，我們透過 n_estimators = [10, 10000], 來進行實驗，因為怕會有 overfitting 狀況，所以利用 CrossValidation 5 fold 記錄 Performance，並且使用 Performance 最好的 model。

B. Experiment

N	10	100	1000	10000
Score	**0.845626**	**0.854699**	**0.855134**	**0.852386**





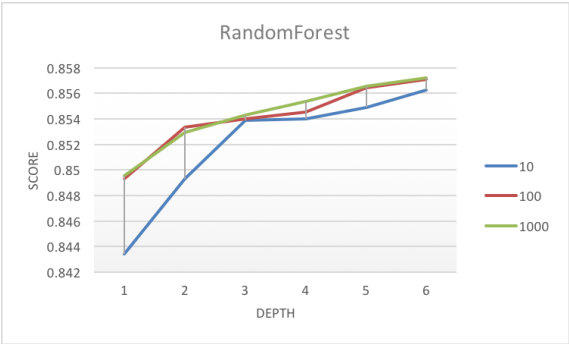
IV. RandomForest

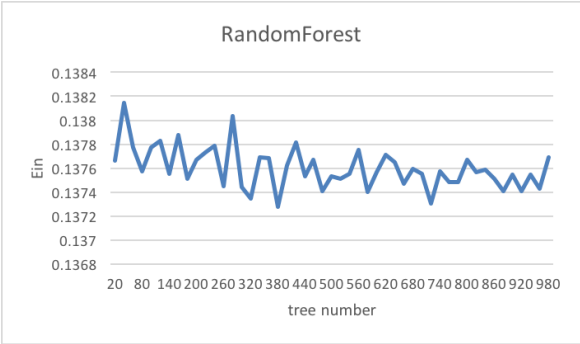
A. Parameter Choosing

GradientBoost 有 n_estimators, max_depth 兩個參數可以調整，我們透過 n_estimators = [10, 1000], max_depth = [1, 6] 來進行實驗，因為怕會有 overfitting 狀況，所以利用 CrossValidation 5 fold 記錄 Performance，並且使用 Performance 最好的 model。

B. Experiment

n_est\max_d	1	2	3	4	5	6
10	0.843447	0.849327	0.853879	0.854014	0.854865	0.856233
100	0.849317	0.853371	0.853983	0.854564	0.856462	0.857094
1000	0.849524	0.852946	0.854315	0.855373	0.856555	0.857208





V. SVM

A. Parameter Choosing

我們使用的是 SVM 的 RBG kernel，有 C 及 Gamma 兩個參數可以調整，我們透過 gamma = [0.000001, 0.01] , C = [1, 100] 的調整來進行實驗，因為怕有 overfittinh 的狀況，所以利用 CrossValidation 5 fold 記錄 Performance，並且使用 Performance 最好的 model

B. Experiment

C \ gamma	0.000001	0.0001	0.01
1	0.794214	0.854306	0.861306
100	0.853788	0.859646	0.861720

VI. Model Comparison

A. Efficiency: 在實驗的過程中，我們也漸漸發現了四個 model 之間的差異，我們發現 RandomForest 及 Adaboost 比其他兩個 model 都來的有效率，可以在很短的時間之內就能讓 Ein 收斂，而最沒效率的就是 GradientBoost 及 SVM，也就是他需要話相當大的運算量才能讓 Ein 降到一定程度，尤其是當 complexity 相當大時，就會讓跑的時間變得非常的長，所以需要做相當多次的實驗時，就比較適合用 RandomForest 及 Adaboost。但相對的 GradientBoost 在 complexity 越大時，Performance 會越好，所以可以在下面的表中看到在大部分的時候，GradientBoost 都會比其他的 model 來的好，或是差不多。

B. Scalability:

我們稍微做了一些實驗，我們使用不同大小的 train_data 進行 training，再透過 CrossValidation 進行 test，透過這樣的實驗可以輕易的看出 model 的 Scalability。

	1/5	2/5	3/5	4/5	1
Adaboost	0.864591	0.867754	0.868459	0.869351	0.869953
Gradient	0.866458	0.868117	0.868418	0.868843	0.868884
RF	0.865576	0.865825	0.865888	0.866168	**0.857094**

可以看到在 Scalability 上 AdaBoost < GradientForest < RandomForest 也就是說，RandomForest 可以用相對少數的資料量就能達到一定程度的好，而 AdaBoost 則需要較為多的 Data 才能表現得較好。

C. Interpretability:

在進行 feature selection 時，我們有利用了各個 model 的對各個 feature 的 importance 進行 refine，然而我們也發現了，SVM 在每個 feature 的詮釋上，也就是透過 support vector，會比其他的 model 來得好，而最差的則是 RandomForest，因為他會不斷的做 Random Selection，這也讓 feature 本身在 random 的過程中變得難以預測其重要性。

VII. Blending

我們利用 Uniform Blending 的方式來將四個 model 的結果做 voting 或 mean，希望能將三個 model 的 overfitting 跟 underfitting 互相補齊，找到一個最為適中的 prediction，但事實上，Blending 的結果卻比 GradientBoost 的來的不好，這很可能是因為被較為不好的 model 所影響的關係，由下表可以看到結果

此表都是使用 private score 來做比較，因為 private 比較接近 CrossValidation 的結果

VIII. What Method We Suggest

首先，我們用個以上實驗的結果，所選出的最好的 model 及 feature 去做測試，下表是整理

完後的測試結果

	Track1		Track2	
	Sample Feature	Our Feature	Sample Feature	Our Feature
AdaBoost	**0.953229**	**0.954319**	**0.874958**	**0.878877**
GradientBoost	**0.956316**	**0.952182**	**0.878909**	**0.879854**
RandomForest	**0.955648**	**0.957065**	**0.872503**	**0.876218**
SVM	**X**	**X**	0.856103	0.861720
Blending	**0.956362**	**0.954076**	**0.876519**	**0.879086**

綜合以上所有的因素得考慮之下，我們認為

track1 比較適合使用 **RandomForest**

優點：因為透過不斷的 Random，讓 Data 可以變得比原本更加隨機，更能 predict 出較為適中的 probability，類似做了一個大型的 Blending，使 model 相對 robust，在預測 Probability 上會有較好的表現，而且 RandomForest 在 efficiency 及 scalability 上相較於其他 model 都有較好的表現。

缺點：但在要預測確切值時，就會相對差，這可以在 track2 中清楚的看到。

track2 比較適合使用 **GradientBoost**

優點：因為 track2 比較需要的是 Label Prediction，也就是比較需要對資料做 fitting，對於模糊地帶的確認則是關鍵，而 GradientBoost 可以讓模糊的地帶，透過 Gradient boost，讓其 Ein 不斷下降，最後趨近到 0，所以在某種程度上來說，是一種透過接近 Overfitting 的方式，使其能在 track2 能有較好的表現

缺點：容易 overfitting，而這也在 track1 上得到了印證，Performance 是最後一名。

IX. Conclusion

我們發現，對同一組 feature，不管用何種 model 去做 train，其實不會差很多，甚至到最後幾乎都是看 random 的好或壞，誤差大概都在 1%以內，真正造成%大幅前進的主要是因為 feature 的差異，好的 feature 就可以讓結果差很多，不過可惜的是我們在一開始並沒有特別在意這個差異，直到後來才漸漸開始發現 feature 的重要，也漸漸意識到每個 feature 的得來不易，從 Human Learning 出可能的 feature，再從 data 中抽取出來，最後還要做 feature selection，並不像是平常作業那樣是給好的 feature，只需要 implement model 就好。

雖然我們最後的 Performance 並沒有非常高，但我們卻在過程中，漸漸理解到 Machine Learning 在真實的應用層面會遇到的問題，也在過程中漸漸地進步，track1 從 89%到 95.82%，track2 從 81%到 87.98%，可以真實地感覺到我們的確在一點點的進步，雖然最後還是沒有取出關鍵的 feature，做了相當多的嘗試，而這個過程讓我們很享受。