2019 Data Science Bowl Competition

b05602052 電機四 舒泓諭

r06522709 機械碩三 鄭呈毅

r07543069 應力碩二 潘俊霖

Introduction & Motivation

這一個比賽主要是依據手機遊戲**PBS KIDS Measure Up! app**的資料，去預測0~3歲小朋友的學習狀況，對於data更詳細的資料請見**Data preprocessing&feature engineering**

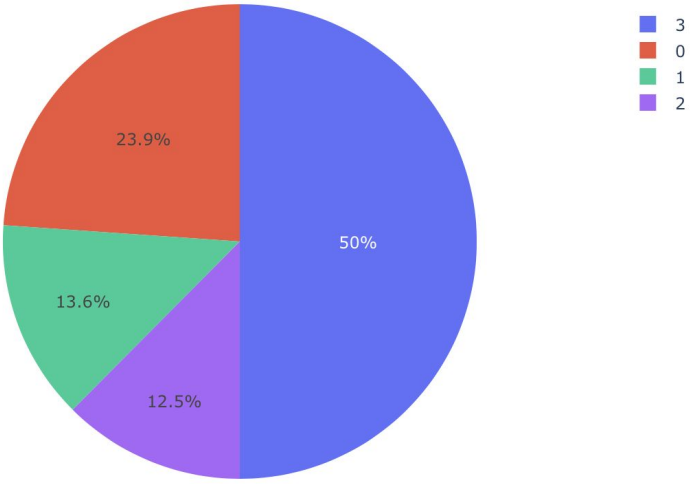
開始這個比賽前，就先去查了以往kaggle上資料科學的比賽獲得比較好結果的work。發現前幾名都有用gradient boosting的model，例如:xboosting、adaboosting.....所以我們最後的模型使用的是2種**gradient boosting**加上**NN ensemble**在一起的模型架構，讓public score到達0.546，final project的ranking是全班第三名。

Data preprocessing/Feature engineering

a.data visualization

```
Train Data Size  (11341042, 11)
Specs Data size  (386, 3)
Train Lables size  (17690, 7)
Test size  (1156414, 11)
Sample_submission  size  (1000, 2)
```

一開始把所有data讀進來的時候，真的讀超久的.....所有的data加起來大約有4GB。



training data kids group 分佈

在一開始我們有參考一些notebook視覺化的方式進行data visualization(參考資料見reference)。我們可以發現在training data中，有超過50%的小孩第一次就通關成功，而有24%的小孩沒有通關成功。

b.接著觀察train.csv資料內部有幾個feature，可以發現在train.csv/test.csv裡面有以下幾個feature。

event_id	game_session	timestamp	event_data	installation_id	event_count
event_code	game_time	title	type	world	

1.其中event_id在specs.csv內部又有著更多資訊。

2.對於app提供的內容又有不同的type:Clip、Activity、Assessment、Game。

3.其中若feature若是文字不是數字的話，例如：event_id，我們是使用one-hot encoding的方式，進行data的preprocessing。

c.train label

我們的train target主要是，**kids經過幾次嘗試，才成功完成一個assessment**，而這一部分要依據資料將**kids**分成4個類型。

Train target	level
the assessment was solved on the first attempt	3
the assessment was solved on the second attempt	2
the assessment was solved after 3 or more attempts	1
the assessment was never solved	0

d.feature engineering

會針對train.csv/test.csv每一個installation_id，把game_session一樣的挑出來，利用**one-hot encoding**將train.csv/test.csv化簡成reduced_train.csv/reduced_test.csv，其中若有一樣的資料就將其累加起來，所以每一個train_data，會變成一筆**m*(10+n)**維的向量，reduced後的data如下表。

installation_id	#of Clip	#of Activity	#of Assessment	#of Game	x
acc_Bird Measurer (Assessment)	acc_Mushroom Sorter (Assessment)	acc_Chest Sorter (Assessment)	acc_Cart Balancer (Assessment)	acc_Cauldron Filler (Assessment)	Event ID game sessiom

Model Description

1. Ensemble

由**xgboost, LightGBM, Catboost,Neural network** ensemble 出最後的model，將不同model預測出來的結果進行weighted sum，最後的public score可以達到0.546。

model	xgboost	LightGBM	Neural network
weight	0.2	0.6	0.2

ensemble weight

2. Neural network

在本次的model中，也有用keras兜一個NN (模型因為有礙於排版，所以放置於appendix)，使用adam作為optimizer，learning rate=1e-4，public score為0.483。

3. Xgboost

為某一種的 GBDT (Gradient Boosting Decision Tree)，在public score可以到達0.54

i. GB (Gradient Boosting)

機器學習中的學習算法的目標是為了優化或者說最小化loss Function， Gradient boosting的思想是疊代生多個（M個）弱的模型，然後將每個弱模型的預測結果相加，後面的模型Fm+1(x)基於前面學習模型的Fm(x)的效果生成的。

iii. GBDT (Gradient Boosting Decision Tree)

將Boosting 應用於分類樹 (Decision Tree)上面，也就是GBDT是GB和DT的結合，可改善原本模型不夠準確的問題。

iv. Xgboost

在GBDT的計算上，誤差函數中增加了正規化(L2 norm)項來簡化學習難度。另外，也利用特徵列排序後以塊的形式存儲在記憶體中，在疊代中可以重複使用；雖然boosting算法疊代必須串行，但是在處理每個特徵列時可以做到並行，藉以提高運算效率，但很耗費記憶體。

4. LightGBM

LightGBM使用的是histogram算法，佔用的記憶體更低，數據分隔的複雜度更低。其思想是將連續的浮點特徵離散成k個離散值，並構造寬度為k的Histogram。然後遍歷訓練數據，統計每個離散值在直方圖中的累計統計量。在進行特徵選擇時，只需要根據直方圖的離散值，遍歷尋找最優的分割點。

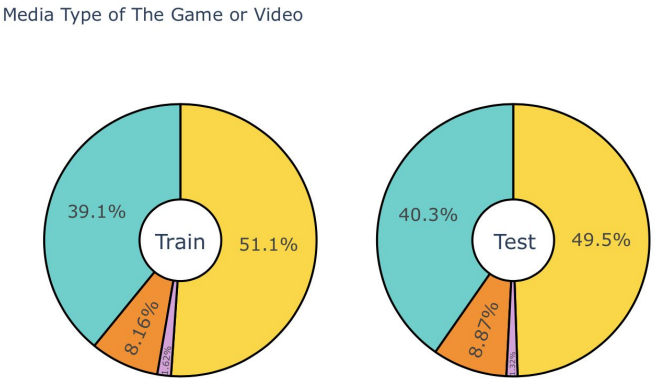
LightGBM也是一種改善的GBDT，而且在很多方面比Xgb更

優秀，在本次的task public score到達0.54。

Experiment and Discussion :

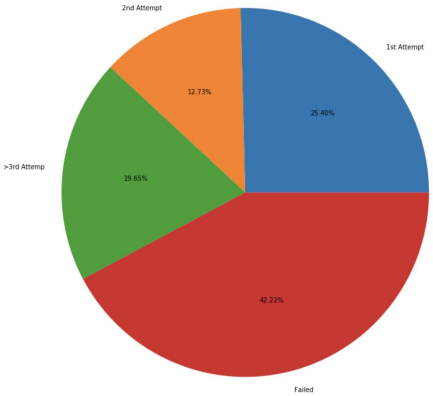
1.資料視覺化

a.Media type of Game or video



可以發現在Train data和Testing data，Media type of Game or video的分布其實是差不多的分佈，這個圖是使用**plotly**這一個套件所畫出來

b.Time Consumed in Assessment



kids解題時間與被分類的group

由以上圖表可以發現，其實觀察到kids嘗試解題的時間和其最後會被歸類在哪一個group其實是很有關係的，若把這一個feature加入training，

可以發現在原本xboosting model的架構下，public score會從0.507提升至0.512。

class	best Accuracy
adding new feature	0.512
non-adding new feature	0.507

2.比較feature engineering 對於public score的影響

class	best Accuracy
feature engineering	0.546
non-feature engineering	0.5

在model都是使用ensemble的架構下，可以發現有做feature engineering的public score會比原本直接拿資料硬train，提高0.046，kaggle排名可以提升大約1000名左右。

3.比較ensemble 前後Accuracy的差距

Model	xgboost	LightGBM	NN(keras)	ensemble
Accurracy	**0.512**	**0.540**	**0.483**	**0.546**

Accurracy comparison

在本次所有嘗試的model中，可以發現在ensemble之前，LightGBM是public score最高的model，可以發現NN在層數不足的時候

表現會不如GBDT架構；而推測 LightGBM相較於xgboost， 會在適當的節點增長決策樹，而表現較佳。

在多次嘗試後，發現將xgboost、LightGBM以及NN最後所predict出來的結果，用0.2、0.6、0.2，的比例進行weighted sum，得到public score 為0.546。

Conclusion

這是我們所有組員，第一次打資料科學的競賽。一開始我們花了很多時間在研究data，後來才發現有別人release出來的notebook，有把data preprocessing寫好，甚至還有很多data visualization的程式，看完真的可以加理解data。此外，也有別人release出來的model，後來我們是參考**Convert to Regression**，並修改**gradient boosting ensemble**的權重，才成功讓public score到達0.546，final project的rank是全班第三名。

做完這個project才發現，打這種kaggle的比賽，可以多花點時間研究別人的notebook。讀別人的notebook真的是一件cp值很高的事，可以藉此更加了解data，此外還可以更加知道有什麼樣的model比較適合比賽的data。

Peer evaluation

組員都超級棒，都做了超多事，討論都非常積極參與，希望有機會能再次合作，分數大家都給對方100分。

分工表:

舒泓諭	coding、撰寫report
鄭呈毅	coding、撰寫report
潘俊霖	實驗設計、撰寫report

Reference

Data visualization

Data Science Bowl 2019Data Visualization

https://www.kaggle.com/fatihbilgin/data-science-bowl-2019-data-visualization

https://www.kaggle.com/c/data-science-bowl-2019/discussion/117019#latest-680222

Feature Engineering

890 features:

https://www.kaggle.com/braquino/890-features

2019 Data Science Bowl - An Introduction:

https://www.kaggle.com/robikscube/2019-data-science-bowl-an-introduction

Model Architecture

Convert to Regression:(final model)

https://www.kaggle.com/braquino/convert-to-regression

Model description

https://www.biaodianfu.com/lightgbm.html

Appendix

Neural network model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 200)               74200
_________________________________________________________________
layer_normalization (LayerNo (None, 200)               400
_________________________________________________________________
dropout (Dropout)            (None, 200)               0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               20100
_________________________________________________________________
layer_normalization_1 (Layer (None, 100)               200
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
layer_normalization_2 (Layer (None, 50)                100
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 25)                1275
_________________________________________________________________
layer_normalization_3 (Layer (None, 25)                50
_________________________________________________________________
dropout_3 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 26
=================================================================
Total params: 101,401
Trainable params: 101,401
```