ML小霸王 Survey Report

b05901060賴繹文

b06505022袁肇謙

b06505015秦逸翔

1. Final Score

a. public socre

Rank	Team	Public Score	Description	Entries	Time
1	ML小霸王	0.250000		65	2021-01-10 10:51:34

b. private score

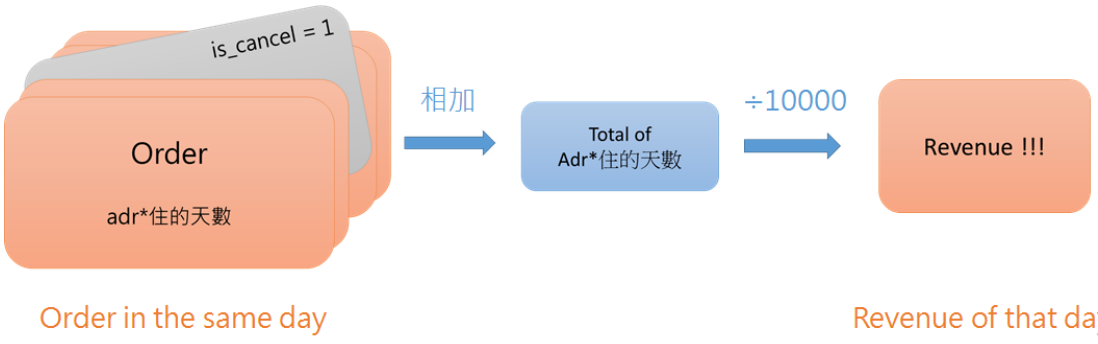
118	ML小霸王	0.545455		65	2021-01-11 01:05:39

2. Our inspection of the training data

a. How the labels generated

在拿到training data後，我們首先觀察了label revenue與資料間的關係，發現最後的revenue是由當天所有訂單去掉cancel掉的部分，再將其adr*住的天數(week_night+ weekend_night)做總和並除以10000便可得到(fig 1.)。而這樣的發現也讓我們很快的定下了後面model的流程，分別是:

1. 用一classification模型來預測is_canceled
2. 用一regression模型來預測adr
3. 再透過如前文所述步驟(如下圖)處理來得到最終revenue



```
[640 rows x 2 columns]
               adr_days  label
date
2015-07-01  20311.186621    2.0
2015-07-02  16530.645277    1.0
2015-07-03  12966.714164    1.0
2015-07-04  17480.654256    1.0
2015-07-05  19591.458478    1.0
...                  ...    ...
2017-03-27  26217.381380    2.0
2017-03-28  16185.177703    1.0
2017-03-29  24002.255525    2.0
2017-03-30  33095.297394    3.0
2017-03-31  36062.103164    3.0

[640 rows x 2 columns]
```

fig 1.

3. Preprocessing

a. feature selecting / preprocessing

在feature的選擇上我們其實沒有一定的標準，而是先照我們自己些微的domain knowledge挑選再慢慢用一個一個feature 增減的方式來決定我們最終的input。至於在preprocessing的部分，則是用老師上課所交的方式，將category類別的feature做one-hot encoding，針對數值範圍較大的feature我們也有想過要做normalization，但經過實際嘗試後發現似乎維持原樣的成績還比較好一點。

另外因為考慮到會使用到多種不同套件的model，因此我們選擇將scalability的重點建立在架構的一致性上，以物件導向的方式得到predict_adr和is_canceled，且統一將data都轉換成numpy的形式，再根據不同model做微調。讓我們在前期建立好架構後很快的就能讓三人同時去做各種不同的嘗試並將成果組合。

b. data pruning:

在任何的機器學習case中，找出noise data並將其排除掉都是十分重要的，而在這次的project中，經過仔細觀察資料我們重點挑出了三種noise data並將其排除，分別是adr為負值、week_night + weekend_night = 0以及adults + children + babies = 0的row，其原因如下圖。透過這樣的data pruning，讓我們的public scores從3.6多大幅度的進步到3.0多，可說是在整個過程中對model進步最大的影響，也讓我們見識到篩選掉noise data 的重要性。

Adr < 0	Week_night + weekend_night = 0	Adults + children + babies = 0
• 依據adr的定義出現負值並不合理	• 訂旅館卻一天也不住?	• 訂旅館但入住人數為0不合理。

(針對第二點入住天數為0的部份我們也想過有可能只是單純用餐而不住的原因造成，但因為實際測試結果發現drop掉後model的performance表現較佳，因此就維持這樣的設定。)

4. **Train Method**

   a. **Predict canceled**

      1. **SVM**

         由於cancel的答案是01形式，最直接想到的預測方法就是 support vector machine。我們認為這是一個具長遠公正性的數學模型，在僅使用linear kernel下的svm，加上 cross validation，適合當作baseline分數。這邊值得注意的是，svm若沒有做cross validation，表現會變得非常差，在參數的選擇上也做了相當多次的實驗與調整，但performance依舊沒有太大的進步，運算時間也相對長就沒有以這個方法為主去做後面的嘗試。

      2. **AdaBoost**

         在上課所教過所有的01 classification中，AdaBoost是較為進階的。我們認為可以較svm有更好的performance，而在這次比賽中也確實是如此。這次是使用sklearn的model，參數如下

         AdaBoostClassifier(n_estimators=120, random_state=0)

         AdaBoost的表現在所有model中是最好的，連課程後其所教的gradient boosting decision tree的public model也無法明顯超越。個人猜測此次競賽可能著重於clean data noise及避免ovefitting的部分，使model的重要性反而降低。

      3. **Decision Tree**

         在嘗試此方法前，我們就知道這是一個非常容易overfitting的模型。而實務上也確實如此。在training data可以做到accuracy 95%的成果，在scoreboard的public score卻只有0.8的分數，在同個predict adr

的模型情況下，decision tree確實不是好的選擇。我們同樣也是使用sklearn的model，名稱如下

DecisionTreeClassifier(), RandomForestClassifier()

4. **Comparison**

對於以上三個model不管是efficiency及performance adaboost都有明顯的勝出，而在後面的實驗中，我們也有嘗試使用Gradientboost以及lightgbm這樣其延伸的model，但因為在performance上面並無明顯的進步，因此依舊以adaboost這個方法為主，而其他延伸或嘗試性的model則是到後期使用aggregation的方式時再以variety model的方式來組合使用。

b. Predict adr

adr為連續的資料，因此我們選用regression model。

1. **Linear Regression**

這邊我們是用sklearn的LinearRegression()，不過我們的data並非linearly separable，因此performance並不好。

2. **Multi-layer Perceptron regression**

這邊我們是用sklearn的MLPRegressor(max_iter=500)，透過SGD去optimize我們的model。

3. **K Nearest Neighbors**

經過預處理的資料非常高維，這邊我假設在這高維度的空間中，相近的資料點會有相近的數值，因此我們採用多用於classification的KNN，這邊我們是用sklearn的KNeighborsRegressor(n_neighbors=10)，將資料空間上距離最近的10個點的adr，取平均。

4. **Comparison**

綜觀以上三個方法，linear regression與k Nearest Neighbors算是在現實應用中較常見的model，其中K Nearest Neighbors更因為可以用圖表cluster的視覺呈現來讓人明白而有了更好的可解釋性，因此在面對

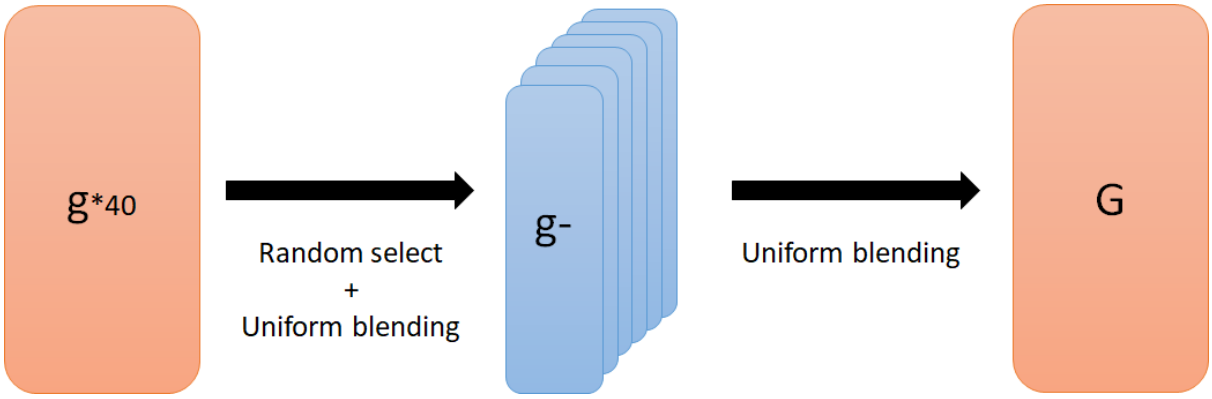
這樣實際案例的應用時，當出現差不多的performace及efficiency時我們傾向使用K Nearest Neighbors這樣的model來作為我們的最終選擇。

5. BEST ONE approach

總結來說，面對 predict canceled 的情況時，AdaBoost是個好的選擇，參數不會像gradient boost decision tree 一樣繁雜。 Predict adr部分，KNN提供了比linear regression更精確的預測。最後再用上aggregation及bagging優化分數。

c. Comprehensive strategy:

在衝刺排行榜的最終階段，我們使用了aggregation中 blending的概念來做整個model的優化，過程中當然也有考慮使用其他形式的aggregation method，但因為其實行上較簡單，因此選擇先用過去我們在嘗試中取得較好結果的模型與參數設定得到將近40個的g，透過隨機取樣再做兩層的uniform blending以期許能達到linear blending的效果，並找到分數較高的組合以得到G。而透過這樣的方式也成功讓我們最後在public score中取得進步來到第一名的位置，但後來檢討我們組別在private score表現不佳原因的過程中，發現有可能正是因為在這個步驟中，我們一開始取樣的g都是以較好public score的模型為主，因而造成了blending後overfitting情況加劇的問題。



5. Github link

https://github.com/YiWenLai510/HTML_Final

6. 分工

b05901060 賴繹文	b06505022 袁肇謙	b06505015 秦逸翔
● structure design ● predict cancel	● preprocessing ● predict adr	● final tuning ● aggregation