Hotel daily revenue prediction

Github URL

[machine_learning_final_project]

Problem description

給定 2015/7/1 到 2017/3/31 的近 100000 筆住宿交易明細，目標是預測 2017/4/1 到 2017/8/31 的每日收益等級。

Overview

我們主要以 scikit-learn package 進行實作，整體流程簡述如下：

1. Data cleaning
   - 移除所有 adr 為 0 的 samples
   - 移除所有 總人數為 0 的 samples
   - 若 “children” 欄位為 null，則設為 0
   - 若 "company” 或 "agent” 欄位為 null，則設為 -1

2. Generate new columns
   - 新增 “arrival_date” 欄位，存成 pandas 的日期格式，方便之後計算每日收益
   - 新增 “stays” 欄位，相當於 “stays_in_weekend_nights” 及 “stays_in_week_night” 兩欄位加總，作為 feature 之一
   - 新增 “persons” 欄位，相當於 “adults”, “children”, “babies” 三欄加總，作為 feature 之一

3. Feature selection
   我們計畫建立一個 regression model 來預測 “adr” 以及 一個 soft classification model 來預測 “is_canceled”。針對這兩個 models，我們選用相同的 features 如下：
   - Numerical features
     lead_time, arrival_date_year, stays, stays_in_weekend_nights, stays_in_week_nights, adults, children, babies, persons, previous_cancellations, previous_bookings_not_canceled, booking_changes, days_in_waiting_list, required_car_parking_spaces, total_of_special_requests

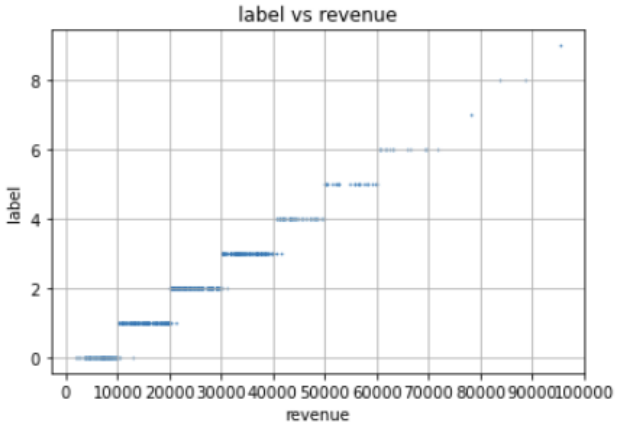
   - Categorical features
     hotel, arrival_date_month, arrival_date_week_number, arrival_date_day_of_month, meal, country, market_segment, distribution_channel, is_repeated_guest, reserved_room_type, assigned_room_type, deposit_type, customer_type, agent, company

4. feature preprocess
   以 one-hot encoder 處理所有的 categorical feature columns。若 optimization 的過程中會用到 gradient descent，則以 standard scalar 來縮放 numerical features，可提升收斂效果。

5. Train regression model to predict “adr”
   為了預測 adr，分別嘗試 Ridge regression, gradient boosting decision tree, 以及 xgboost 等方法來訓練 regression models。

6. Train soft classification model to predict “is_canceled”
   為了預測 is_canceled，分別嘗試 Logistic regression, gradient boosting decision tree, xgboost 及 random forest 等方法來訓練 classification models。

7. Predict the daily revenue rank
   每筆訂單的預期收益 = adr * is_canceled * stays。將同一日期的所有訂單加總即為當日的預期收益。根據以下分析結果可直接判斷每日收益與收益等級之間的關係，收益量每增加 10000 就提升一個等級。



Machine learning models

以下分別討論 regression models 及 soft classification models。若這兩部分都能做好，則應能有良好的結果。

- Regression models
  - Ridge regression
    相當於採用 L2 regularization 的 linear regression，其 objective function 為

$$\min_{w} ||Xw - y||_2^2 + \alpha||w||_2^2$$

    因為資料量不小，故採用 stochastic gradient descent 來做最佳化。需調整的參數為 regularization parameter alpha。
    透過 cross validation 的結果發現 ridge regression 的效果不佳。事實上，即使是 training accuracy 也只有大約 0.6 左右，因此判斷 features 與 adr 之間並非簡單的線性關係。

  - Gradient boosting decision tree regression
    由於 features 與 adr 之間呈非線性關係，故考慮採用tree based aggregation model 來解決 underfitting 的問題。這裡選擇廣為使用的 gradient boosting。為了提升 training 效果，這裡採取幾項措施：
    1. 以 huber loss function 取代 least square loss function
       實驗結果發現，採用 least square loss 容易發生 overfitting，而選用較為 robust 的 huber loss 之後，雖然 converge 速度較慢，但能夠明顯改善 overfitting 的情形。huber loss 如下所示：

$$Loss\left(y,\widehat{y}\right)=\begin{cases}\frac{1}{2}\left(y-\widehat{y}\right)^2, & \left|y-\widehat{y}\right|\le\delta\\ \delta(\left|y-\widehat{y}\right|-\frac{1}{2}\delta), & O.W.\end{cases}$$

δ 值（在 sklearn 中相當於參數 alpha）設定越小，loss function 越 robust。huber loss 結合了 square loss 和 absolute loss，當 y 和 yhat 差距較大時，採用 absolute loss 可以防止過度的 penalty，因此不會受到 outlier data 的嚴重影響。

2. Early stopping
   事先從 training data 中取出一部分作為 validation 之用，每一次的 iteration 都會評估 validation error，當 validation error 經過幾個 iterations 之後不再大幅變化，則停止 training。透過 early stopping 可防止 overfitting。以上步驟可藉由設定 validation_fraction、n_iter_no_change、tol 三項參數達成。

3. Shrinkage
   Gradient boosting 每回合根據以下方式更新 model：
   $$F_m(x)=F_{m-1}(x)+\nu h_m(x)$$
   根據 sklearn 文件的建議，將 learning rate ν設為較小的常數（0.01～0.1）使每個 weak learner 的影響力減小，讓之後產生的新 learners 保有發揮空間，可有效減小 test error。

4. Subsampling
   每回合只選取特定比例的 samples 做 training，理論上可增加各個 weak learner 的差異性，搭配 shrinkage 使用，可達到減小 test error 並且防止 overfitting 的目的。此外，採取 subsampling 的同時，還可以透過 out-of-bag estimation 來評估 test error。

5. Pruning
   Tree depth 太大顯然容易造成 overfitting，且不符合 gradient boosting 的精神。因此調整 max_depth 至適當數值對於 training 的結果相當重要。

- XGBoost regression
  與 gradient boosting 大致類似，是內建於 XGBoost package 的演算法，但針對原始的 gradient boosting 進行了若干優化，例如：
  1. 最佳化時，針對 loss function 做了二階泰勒展開。為了確保二階可微，將 huber loss 改為 pseudo huber loss。

  2. 在 loss function 中加入了 regularizer，防止 overfitting。
  3. 提供 column subsampling 的選擇，每回合只隨機選擇特定比例的 features 做 training，同樣可增加 learners 之間的差異性，同樣可防止 overfitting。此外，該方法也提升了演算法的效率。

在參數的選擇上，除了增加了 regularization parameter 以及 column subsample ratio 外，與 gradient boosting 沒有太明顯的差異。

下表為上述各項 regression model 透過 5-fold cross validation 取得的參數以及 validation error。這裡採用 least square error 來評估結果。

	ridge regression	gradient boosting regression	xgboost regression
Parameters	alpha = 100	loss = 'huber' alpha = 0.9 learning_rate = 0.1 n_estimators = determined by early stopping max_depth = 6 subsample = 0.8 validation_fraction = 0.2 n_iter_no_change = 5 tol = 0.005 random_state = 1126	booster = 'gbtree' objective = 'reg:pseudohubererror' learning_rate = 0.04 n_estimators = 10000 max_depth = 6 subsample = 0.8 colsample_bytree = 0.6 gamma = 5.0 reg_lambda = 1.0
Validation error	0.395	0.2119	0.2055
Training error	0.4181	0.2309	0.223
Training time (sec)	1.4	288	228

Ridge regression 具有執行效率高的優點，但XGBoost 及 gradient boosting 的 error rate 明顯較 ridge regression 低。
XGBoost 和 gradient boosting 是相當類似的演算法，故 error rate 差距不大。但 XGBoost 每回合只取部分的 features 做訓練，且做了較多系統優化，故效率稍高一些。
透過比較 validation error 和 training error 可以發現 overfitting 並不嚴重。

- Soft classification models
    - Logistic regression
      基本的 classification model，需調整的參數為 l2 regularization parameter C。
      透過 cross validation 進行參數調整，最終得到的 cross entropy error 約為 6.05。

    - Gradient boosting classification
      與 gradient boosting regression 的作法大致相同，但 loss function 改為 cross entropy error。

    - XGBoost classification
      與 XGBoost regression 的作法大致相同，但 loss function 改為 cross entropy error。

    - Random forest
      為了做出更為正確的分類邊界，嘗試採用 random forest。相較於gradient boosting，random forest 直接對所有的 base learners 做 uniform blending，較不容易發生 overfitting，因此我們在並未限制 tree depth。

下表為上述各項 classification model 透過 5-fold cross validation 取得的參數以及 validation error。這裡採用 cross entropy error 來評估結果。

	logistic regression	gradient boosting classification	xgboost classification	random forest classification
Parameters	C = 10	loss = 'huber' alpha = 0.9 learning_rate = 0.1 n_estimators = determined by early stopping max_depth = 3 subsample = 0.8 validation_fraction = 0.2 n_iter_no_change = 20 tol = 0.005 random_state = 1126	booster = 'gbtree' objective = 'reg:pseudohubererror' learning_rate = 0.04 n_estimators = 2500 max_depth = 3 subsample = 0.5 colsample_bytree = 0.5 gamma = 0.0 reg_lambda = 1.0 random_state = 1126	n_estimators = 500 max_depth = None max_features = 'sqrt' random_state = 1126
Validation error	0.3381	0.2716	0.2517	0.2406
Training time (sec)	19.2	53.6	39.5	1170

Validation error 由最小至最大依序為 random forest、xgboost、gradient boosting、logistic regression。
執行效率由最好到最差則依序為 logistic regression、xgboost、gradient boosting、random forest。
Random forest 具有正確率高、不易 overfitting，且參數少的優點，但由於 tree depth 較大，不論是 training 或 prediction 都需要相對長的時間。

我們嘗試用不同的組合來預測 adr 及 is_canceled，並且進一步計算 test data 的每日收益等級以及 testing error（mean absolute error），結果如下：

1. gradient boosting regression + gradient boosting classification
    - public score = 0.421053 / private score = 0.350649
2. gradient boosting regression + XGBoost classification
    - public score = 0.407895 / private score = 0.311688
3. XGBoost regression + gradient boosting classification
    - public score = 0.407895 / private score = 0.337662
4. XGBoost regression + XGBoost classification
    - public score = 0.407895 / private score = 0.311688
5. gradient boosting regression + random forest classification
    - public score = 0.421053 / private score = 0.363636
6. random forest regression + random forest classification
    - public score = 0.421053 / private score = 0.337662

不難發現，利用 validation error 最小的 model（例如 random forest classification）做預測，最後求得的 testing error 未必會最小。我們認為造成該情形的其中幾項原因在於 ordinal ranking 問題具有以下性質：

1. 預測 is_canceled 時，交易金額越高的 sample 預測錯誤的損失越嚴重。

2. daily revenue 接近兩個等級的交界時，預測錯誤的損失越嚴重。

而針對 adr 和 is_canceled 的預測進行 validation 時，顯然沒能考慮到這些情形，故容易造成 final prediction 的效果不如預期。

Recommendation

在前述幾種 models 中，我們推薦採用 error rate 最小的 XGBoost regression + XGBoost classification 組合。該方法的優缺點分別陳述如下：

- 優點
  - XGBoost 演算法（不論 regression 或 classification）提供多種方式有效遏止 overfitting 發生，例如使用 robust 的 loss function 或是運用 shrinkage、subsampling 等技巧。
  - 和類似的 gradient boosting 相較之下，XGBoost 不論是在執行效率或是正確率上都略勝一籌。
  - 在 classification 的問題上，XGBoost 的執行效率比 random forest 好得多，因此更容易做 validation
  - 由於是 tree based algorithm，XGBoost（或 gradient boosting）完成訓練後，可根據執行的過程分析每個 feature 的重要性。我們透過該項功能將重要性極低的 features 刪除，提升之後執行的效率。此外，也能透過這項功能評估新增 features 的效果好壞。
  - 我們採用 soft classification 而非 hard classification 來預測 canceled 的機率，可以避免一旦預測錯誤就損失一整筆訂單的差距。實驗結果也顯示 soft classification 確實較適合這項 project。
- 缺點
  - 處理 regression 問題時，我們採用的 pseudo huber loss function 雖然能夠減少 overfitting，但實驗結果發現，相較於 least square loss，huber loss 的收斂速度緩慢許多，因此需要相對更多的 base learners 才能得到較低的 error rate。
  - 該 model 預測 is_canceled 時，並未對交易金額高的 samples 增加權重。此外也沒有嘗試在 daily revenue 接近兩個等級交界時提昇 is_canceled 及 adr 的預測精度。

上述缺點其實是我們多數 models 的共同缺點。在各方面的表現上，XGBoost 確實優於我們使用的其他 models，因此我們認為採用 XGBoost 來處理該 project 的 regression 及 classification 是較佳的選擇。

Reference

1. scikit-learn user guide
2. XGBoost document
3. XGBoost: A Scalable Tree Boosting System