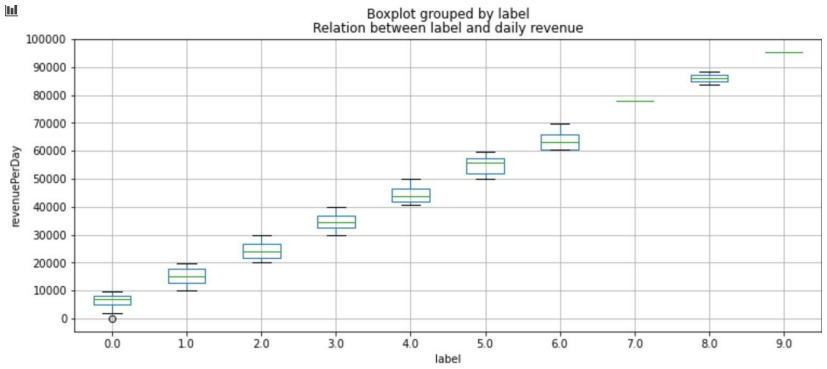
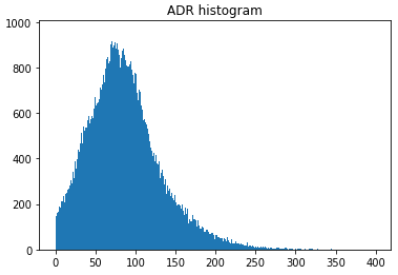
# ML Final Project - Hotel booking demand

Team Name: HTML5566

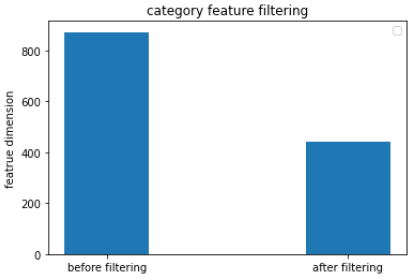
## A. Feature Analysis and Pre-Processing




1. 首先必須確立 revenue 和 label 之間的 mapping 關係，從圖中可以看到當天 **revenue 的總和和 label 幾乎是線性關係**，因此我們初步模型預測時的 label prediction 為當日 revenue 總和再除以 10000，再對 label prediction 取 floor 得其 label。

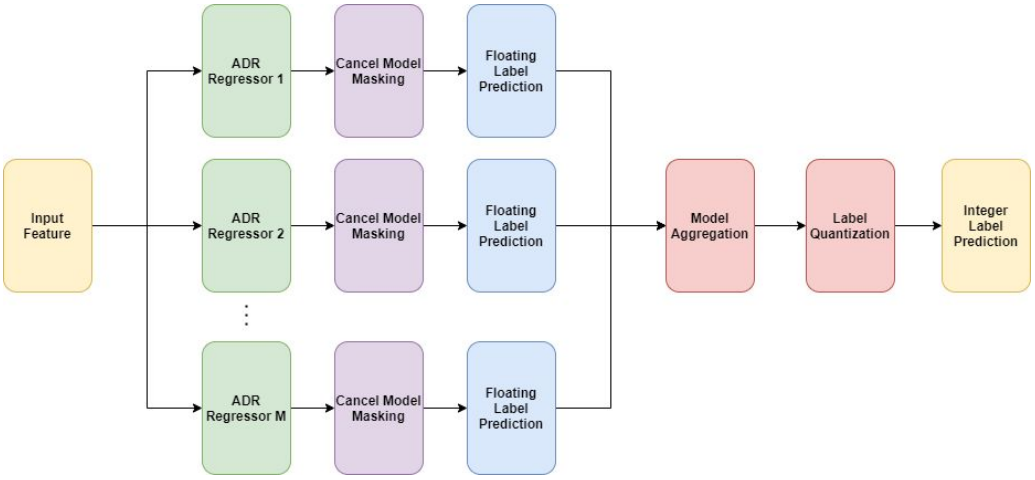



2. 我們發現 ADR 只有不到十筆訂單的 ADR 數值超過 370，為使模型不被 outlier 影響大部分表現，我們**對 ADR 數值做 -10 至 370 的 truncation**。




3. 部分 category feature 種類非常多，尤其 agent、company、country，原轉換成 one-hot 後共有 872 維，但大多數的種類出現次數非常低，加上 train 和 test 的存在大量不會在交集的種類，我們**將出現頻率低於 1% 或是沒有出現在 train 和 test 交集的種類都 filter 掉**，可以將 872 維降至 443 維。
4. 我們發現部分 category feature 存在極高的 unknown 比例，然而有可能存在多筆異質性極高的資料同時被歸類在同個 unknown 族群的情況，會對學習表現造成影響，因此我們去除 unknown 的選項，**在資料遇到 unknown 的情況時，將其指定為最頻繁出現的種類**。
5. 我們模仿 RFR 的 random subset 機制，將 csv 原 table 的 28 種 attribute，**逐一拔掉觀察對指定模型訓練後對 validation 的影響**，我們最好的 feature subset 版本拔除了「**arrival_date_year**」和「**ID**」。

# B. Approach




我們在表現上最好的方法為 ADR prediction，我們主要 approach system 分為三個階段：**ADR regression**、**cancel masking**、**label-level aggregation**，最後將整天 revenue 總和除以 10000，得到 label prediction 後去掉小數部分。

## 1. adopted ADR regression methods overview：

ADR regressor 採用的學習算法分為三種，分別是 Random Forest Regressor(RFR)、Histogram Gradient Boosting Regressor(HGBR)、Deep Network Regressor(DNR)，下述為各學習算法參數設定和簡介以及針對算法之間的統整比較：

**(1) Random Forest Regressor**

RFR 的運作原理是找到好的單一維度的 decision stump，整數數值大小不影響結果，意即**將 category 的 feature 映射成 label 會和映射成 one-hot 效果相同，因此我們輸入 RFR 的 category feature 設為 label integer，能大幅降低原 one-hot feature 對於模型的運算量**。

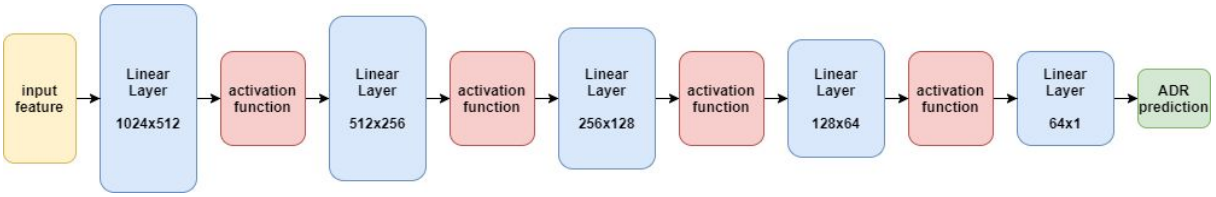
此外，基於 evaluation 採用的是 MAE，我們 criterion 採 MAE 而非 MSE，我們表現最好的模型在 n_estimators 上設為 200，其餘為 sklearn 的 default setting。

**(2) Histogram Gradient Boosting Regressor**

相較傳統 GB 僅使用一階微分並且無法平行化，**extreme GB(XGB) 系列對泰勒二階展開做近似來減緩壞代次數和提高表現，在改善過擬合問題上傳統 GBR 的做法為懲罰過大的 magnitude 項，而 XGB 在損失函數內參考子樹葉數以制定正規項。**

在原本 XGBR 的方法中，若以排序的資料為 N 筆且 d 個維度，需要切 N 次，對每個維度都要切，複雜度為 O(Nd)，而 sklearn 的 HGBR **在建樹前會先對每個維度每筆資料以 histogram 的方式分成 h 堆**，以其為切分點只需要切 h 次，而 h 為一常數，因此**可將複雜度降至 O(d) 獨立於資料的數量**，其表現與 XGBR 可比，但 efficiency 相較 XGBR 或是 GBR，有很大優勢，因此我們在 gradient boosting based 的算法採用 HGBR。

**(3) Deep Network Regressor**



DNR 架構圖如上，實驗結果來說，不同架構差異不大，超參數表格如下所示：

| batch size | optimizer | learning rate | checkpoint mode |
|---|---|---|---|
| 64 | Adam | 0.01 | save best model |

**one-hot category feature 非常 sparse，為限制模型較能夠學到 category feature 提供正確資訊，第一層 linear layer 不設置 bias 項，使 one-hot feature 為 0 的數值不影響輸出。**

此外，針對中間層激活函數做比較，**ReLU 因沒有 zero-centered，得改善 gradient vanishing 問題，但對初始化或是學習率過於敏感，使部分神經元沒被激活，得不到 gradient flow**；LeakyReLU/Swish 針對該問題做概念性調整，給予原本 ReLU 為 0 的部分一微小正斜率，使其 gradient flow 得以流通；Swish 相較 LeakyReLU 更具平滑性，我們針對 ReLU、LeakyReLU、Swish 三個激活函數為變因比較 board 上 public score 結果，表格如下：

| ReLU | LeakyReLU | Swish |
|---|---|---|
| 0.407895 | 0.381579 | 0.394737 |

因為看不到顯著差異，LeakyReLU 表現較高，之後 DNR 皆固定使用 LeakyReLU。

### (4) Algorithm-wise Comparison

**Popularity 因為採用的都是經典算法，對該評量比較有意義的定義為「遇到 final project 的問題類型，會偏好先嘗試哪一個？」：**

因 final project 的資料類型為 concrete categorical feature，以個人經驗來說，算法輸入若為單筆資料不考慮不同訂單的相依性的話，會優先嘗試 HGBR，而 RFR 或 DNR 優先序會依據HGBR 實驗結果決定。

**Interpretability 我們定義為 「是否能依據模型的輸出或是預測行為來回饋人類可容易判讀的知識？」：**

RFR 和 HGBR 為 tree-based 算法，在 categorical feature 的情況能從模型依屬性來判斷目標，將訓練資料依據目標進行「歸納」，以相似族群做平均預測，也許能從歸納的族群觀察人類的可識讀的共通點，而 **RFR 在 sampling 的過程中亦能從 random selected subset 推估 feature 的重要性；而 DNR 為深度結構，直觀可視化的部分為第一層的權重大小。**

我們對 reasoning 的排行為 RFR > HGBR > DNR，但**現行 explainable AI 技術可將模型視為黑盒，利用對輸入的改動觀察輸出的變化，能做到的解釋性和上述無異，再者是人類針對模型的行為的解讀是否超譯無從判斷，而表現上的穩定度，可以體現在大量資料壓力測試的結果，至少這評量對我們選擇 recommended approaches 上不會是優先參考的標準。**

**Efficiency 我們針對各演算法的分別對空間和時間複雜度做討論：**

**RFR 的空間複雜度會隨資料量成長；HGBR、DNR 可自由定義模型架構和參數量**，記憶體需求可視為固定常數，RFR 和 DNR 記憶體量分別為約略 1.09 GB、247 MB；關於時間複雜度，我們著重訓練過程，RFR 之於 HGBR 可平行訓練不同決策樹，然而 HGBR 的每一棵樹是基於上一次疊代的殘差項做樹展開，在每個維度將 N 筆資料依 histogram 分成 h 堆，需切從 N 降為常數 h，大幅降低時間複雜度，因此 **HGBR 在我們訓練時間是遠小於 RFR 的；而 DNR 是矩陣運算，可平行化程度極高**，雖然訓練時間僅取決於何時停止，實務上訓練一個已收斂且表現可接受的模型僅約略半小時內，其顯著缺點是對 GPU 平行設備的需求。在具備 GPU 條件下，對該 final project，我們認為 efficiency 的排行是 HGBR > DNR >> RFR。

**Scalability 針對資料量級、維度、可適用問題和資料型態多寡進行比較：**

**RFR 複雜度會隨資料的維度和數量爆增，而 HGBR 和 DNR 的參數量和訓練時間不受限於此。**此外，**類神經網路的架構能依據資料的結構性「客制化」不同架構**，譬如以天為單位的不同數量訂單資料作為 structural input，撇除人為設計 feature，tree-based 算法無法處理這類問題，反之類神經網路能利用 CNN、RNN 和 self-attention layer 處理這類結構性的資料，且其目標函數可以自由定義，我們對 scalability 排行是 DNR >> HGBR > RFR。

## 2. Cancel model masking

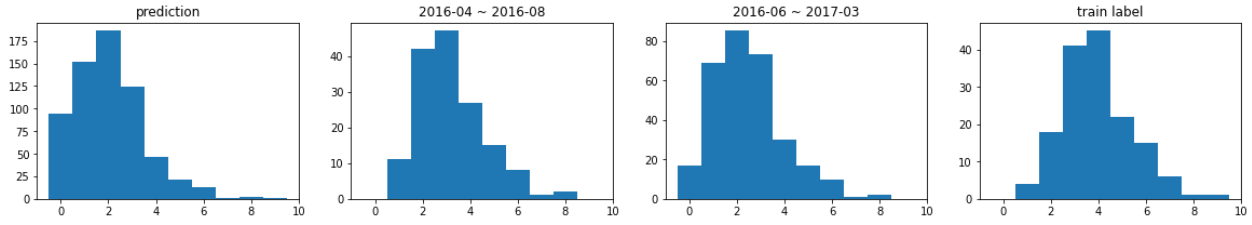
使用 Random Forest Classifier(RFC)，該步驟為將預測 cancel 的訂單設為 0。我們認為將沒有 cancel 的判為 cancel 的代價相對大很多，在 cost 的權重比例設為 1:50 且 criterion 設 entropy 時在表現 label MAE 表現最好。

## 3. Label-level Aggregation

**有鑑於 validation 的 revenue 預測結果未必能反映 label MAE，我們以一天 revenue 總和得到的 label prediction(floating value) 作為 blending 的 feature**，第一部分為增加 hypothesis 的多樣性，我們採取了 **re-samping scheme**；第二部分為基於前述多種算法以及 re-sampling 的技巧，比較 **linear blending** 和 **conditional blending** 兩種 model aggregation 的策略：

**(1) re-sampling scheme**

單筆算法在 testing data 的 public score 上得到最好表現是 0.315789，有鑑於 validation 和 public score 的落差，加上 testing 為跨年分的資料，我們推測 training 和 testing 分佈有一定程度的差異，假設在 testing 上的 prediction 和真實 label 已經有足夠的相似度，那我們**應該可以從 predition 回推和 testing data 的 label 分布較為相似的訓練資料，duplicate 分佈可能相似的訓練資料使 hypothesis 的預測更可能貼近 testing 的分布**，我們嘗試 duplicate 兩部分和測試集 label histogram 較為相近的資料作出基於各學習算法作出兩個新模型，分別是 2016 年的四月至八月份以及 2016 年的六月至 2017 年的三月，label histogram 如下圖所示，可以看到 train 和 prediction 的分佈是有偏差的，而兩種 re-sampling 的分布可以使該偏差問題減緩：




我們兩種 re-sampling 測試在單體 RFR 模型的結果如下表格：

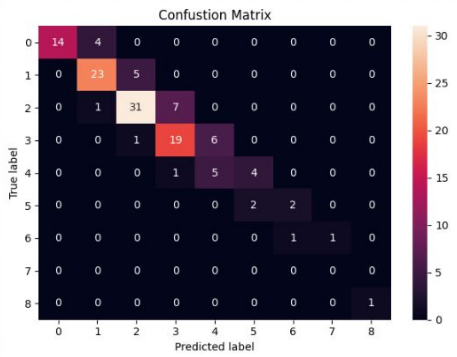
| | w/o res | w/l res1 | w/i res2 |
|---|---|---|---|
| public | 0.315789 | 0.315789 | 0.315789 |
| private | 0.38961 | 0.363636 | 0.363636 |

**(2) linear blending strategy**

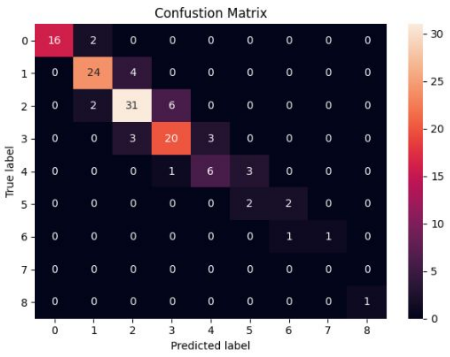
不同模型權重為參考 validation 和 public score 的分數為來制定，權重總合為 1。

**(3) conditional blending strategy**

**RFR**




**DNR**




上圖為 ordinal label 的 confution matrix，我們透過 validation 的觀察發現 RFR 在部分 label 會有高估傾向，DNR 則較為輕緩，**因為單體 RFR 模型在 public score 已經比 linear blending 高，在 RFR 的 public score 能夠反映真正表現的假設下，針對 RFR 容易犯錯部分用其他模型條件式的 refinement 為一較精準的策略**，refinement 的策略有二，其一為**拉高 revenue 總和映射成 label 的門檻，在原高於邊界的 label prediction 若低於門檻則會降一級**；其二為**鎖定在 RFR 預測常錯的 label，若 DNR 預測 label 小於 RFR 則會將兩模型在該條件的 label prediction 作 interpolation**，結合兩種 refinement 做法，public score 從 0.315789 進步到 0.263158。

### (4) Blending strategy comparison

model aggregation 包含所有單體算法在 efficiency 的缺點，但同時優點是能補足各模型表現上的缺點，**linear blending 能被應用在任何問題上，相較 conditional blending 更具備 scalability 和 populatiry，而 conditional blending 為基於後天人們觀察制定，表現與否直接對應在 interpretability，因此 conditional blending 解釋性較高。**

## C. Performance Comparison

| RFR | HGBR | DNR | lin blnd | cond blnd |
|---|---|---|---|---|
| 0.315789 | 0.355263 | 0.381579 | 0.328947 | 0.263158 |
| 0.38961 | 0.350649 | 0.402597 | 0.350649 | 0.376623 |

上圖可以看到 **RFR 在單體模型上和 HGBR 的表現是具可比性的**，雖然 GBR 會針對模型多樣性進行抽樣，但其方法同時會對離群值敏感，如果選擇不恰當的損失函數，誤差會被放大，或是訓練和測試資料分佈差異過大優勢就不會太大。而 RFR 每一棵樹只會被局部小區域離群值影響，最後被多棵樹的取樣平均掉，在取捨之下從而使 RFR 和 HGBR 表現持平。

可以看到 linear blending 在 private 上的表現是高於 conditional blending 的，而**使用 conditional blending 的時候，我們有兩點強烈假設，其一為主體 RFR 模型的 generalization 能力夠好，其二為 public score 的表現和 private score 相似，但這次提供的資料量級很低，結果隨機性偏高，相較之下 linear blending 的權重設置得當的話，則不會過於依賴單體模型的決策，每個模型決策錯誤的風險較容易分擔。**

我們認為在一次 final project 中，單體模型除了 peformance 外，scalability 、 efficeincy 為選擇單體算法模型中很重要的選項，訓練快速能夠在短期內大量實驗，增加自己找到表現好的設定的機會，雖然 **RFR 在效率上可能遠不如 HGBR，但其表現以及運算時間仍屬可接受範圍，因此在單體模型上，RFR 為我們的 best recommended approach**；對於 aggregation model，我們認為在這個 final project 裡面，除了表現之外

generalization 的穩定度也相當重要，雖然最好的 conditional blending 在 public 和 private 的總體表現最高，但其因為測試資料量級太低，導致變異性非常高，反之 **linear blending 的上傳結果非常穩定維持在一定水平，因此我們認為 linear blending 為 aggregation model 中的 best recommended approach。**