Data Mining HW3
Walmart Recruiting: Trip Type Classification
Team 5

陳世運 R06725024 / 陸艾眉 R06725030 / 劉庭維 R06725031 / 陳信豪 R06725048

一、 作業概述 （Abstract）

HW3 的作業目標為運用不同分類模型去做 Data Mining 並進行比較。而此次我們選用 Walmart 所提供的訪客採購與旅程類別資料集。舉凡日常的消費、一週一次的大採購、節慶購物或換季購物等，都會分別被定義在不同的旅程類別。Walmart 已經根據其內部完整的資訊去做出旅程分類，而現在 Walmart 希望能夠透過更少的資訊來完成這項任務。因此在僅提供採購資訊和採購時間的條件下，我們將建立不同的模型來預測訪客該次的旅程類別，此外，我們會透過 Repeated Stratified K-Fold 去做 cross validation，並根據 fit-time、score-time、accuracy、log loss、f1-score、precision、recall 等數據來進行模型之間的效能比較。

二、 資料集 （Dataset）

Walmart Recruiting: Trip Type Classification

Walmart 於 2015 年時在 Kaggle 上提供了此份資料集。

Link: https://www.kaggle.com/c/walmart-recruiting-trip-type-classification

- 資料欄位說明

Column	Description	Example
TripType	旅程編號 (預測目標)	30
VisitNumber	訪客編號	7
Weekday	到訪星期	"Friday"
Upc	美國通用產品代碼	60538815980
ScanCount	商品購買數量 (負值代表退貨)	1
DepartmentDescription	商品所屬部門	"SHOES"
FinelineNumber	商品所屬類別	8931

三、 環境與套件 （Environment and Packages）

- 系統環境

Environment	Detail
作業系統	Windows10
處理器	Intel(R) Core i7-8700 CPU @ 3.20GHz 3.19 GHz
記憶體	16.0 GB
GPU	NVIDIA GeForce GTX 1070

- 語言與套件

Language/Package	Version
Python	Python 3.6.4 |Anaconda
pandas	0.22.0
scipy	1.0.1
sklearn	0.19.1
xgboost	0.7
lightgbm	2.1.0
matplotlib	2.1.2
ggplot	0.11.5

四、 特徵工程 （Feature Engineering）

A. VisitNumber

觀察 Training 和 Testing 中的 VisitNumber，我們發現它似乎是有時間代表性的，而我們猜測此欄位是根據結帳時間依序編碼。此外，如果上述成立的話，Training 所涵蓋的時間範圍和 Testing 所涵蓋的時間範圍將是彼此重疊的，意即並非取前一段時間當 Training，後一段時間當 Testing。因此，雖然 Walmart 沒有明確說明，我們仍將 VisitNumber 視為一項可用的 features。

Training VisitNumber: 5, 7, 8, 9, 10, 11, 12, 15, …

Testing VisitNumber: 1, 2, 3, 4, 6, 13, 14, …

B. Weekday

我們將星期一至星期日做 dummy，形成七個 features。

C. NMF on Visitor-Item-Count Matrix

觀察各項 features 的種類數量。

Trip	Visit	Weekday
38	191348	7

UPC	Department	Fineline
124694	69	5354

其中，UPC、Department 和 Fineline 的種類數量非常龐大，若與 VisitNumber 去建構 matrix 的話（值為 ScanCount），不但會形成 Sparse Matrix，記憶體也會很吃緊。為了解決此問題，我們決定將 Sparse Matrix 透過 NMF 進行降維。

另外，由於 ScanCount 有些是負值（退貨商品），NMF 無法處理，因此我們會將購買資訊和退貨資訊（將 ScanCount 取絕對值）分開去建 purchase-matrix 和 return-matrix。

最後得到六個 Matrix：

(i) Matrix1 (UPC-purchase)

(ii) Matrix2 (UPC-return)

(iii) Matrix3 (Dep-purchase)

(iv) Matrix4 (Dep-return)

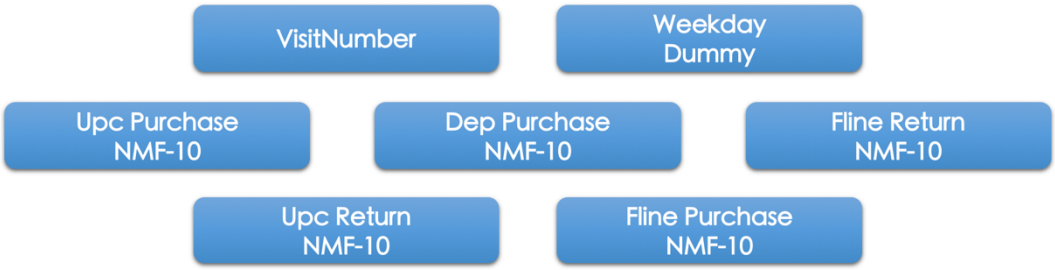
(v) Matrix5 (Fineline-purchase)

(vi) Matrix6 (Fineline-return)

其中 Matrix5 的值全為 0 而不採用，所以實際上為五個 Decomposed Matrix，而每個 Visitor 就會對應到五個 purchase/return vector。

最後，我們將壓縮維度設為 10，所以會得到 5 * 10 = 50 個 features。

D. All Features Taken



Features 總數：58

Target 總類別數：38 （即 Trip 的類別數）

Problem Type：multi-class Classification

五、 模型選用 （Model）

我們選用 LightGBM、RandomForest、ExtraTrees、AdaBoost、DecisionTree 和 NN 這六種不同類型的 Model 來進行比較。

- LightGBM、RandomForest、ExtraTrees、AdaBoost 為 meta-estimator，分類器數量統一設為 500 個。
- LightGBM、RandomForest、ExtraTrees、AdaBoost、DecisionTree 為 tree-based，max_depth 統一設置為展開至底。

以下為模型之介紹，和詳細參數設置。

A. LightGBM (LGB)

為微軟所開發的梯度 boosting 演算法（基於決策樹），其模型準確度堪比資料分析競賽常勝軍 XGBoost，並在速度上優於其好幾倍。

```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
        learning_rate=0.1, max_depth=-1, min_child_samples=20,
        min_child_weight=0.001, min_split_gain=0.0, n_estimators=500,
        n_jobs=-1, num_leaves=31, objective='multiclass',
        random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
        subsample=1.0, subsample_for_bin=200000, subsample_freq=1,
        verbose=0)
```

B. RandomForest (RF)

由多顆決策樹所構成，每一棵樹的訓練資料為 bootstrap 抽樣取得，最後將所有樹的預測結果進行投票。

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

C. ExtraTrees (EXT)

類似於 RandomForest，不過每一棵樹都是採用所有資料，而非做 bootstrap 抽樣。其每棵樹的差異來源在於分叉點是完全隨機決定的，而非像 RandomForest 去算出最佳值來決定佳分叉點。

```
ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
           max_depth=None, max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=-1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

D. AdaBoost (ADA)

AdaBoost 為將前一個分類器分錯的樣本加重權重以做為下個分類器輸入樣本的 boosting 演算法。

（base_estimator 之預設為 DecisionTree）

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=500, random_state=None)
```

E. DecisionTree (DT)

即單一顆決策樹（CART）。

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

F. MLPClassifier (MLP/NN)

即 Fully-Connected Feed-Forward Neural Network，在此我們設置四層 hidden layer，並切出 0.1 validation 來做 early stopping。

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=True, epsilon=1e-08,
       hidden_layer_sizes=(256, 256, 128, 128), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=0, warm_start=False)
```

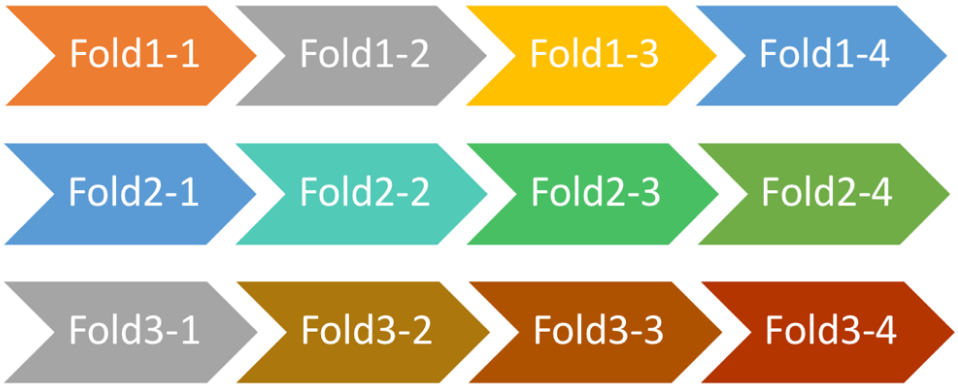
六、 驗證方法 （Evaluation Method）

此次的 Problem Type 為 multi-class Classification，而我們將會根據以下的幾個指標來進行 model 之間的比較。

Score	Detail
Fit Time	model training 時間
Score Time	validation set model predicting 時間
Accuracy	(TP + TN) / (TP + FP + TN + FN)
Log Loss	$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$
F1 (macro)	2 * (Precision * Recall) / (Precision + Recall)
Precision	TP / (TP + FP)
Recall	TP / (TP + FN)

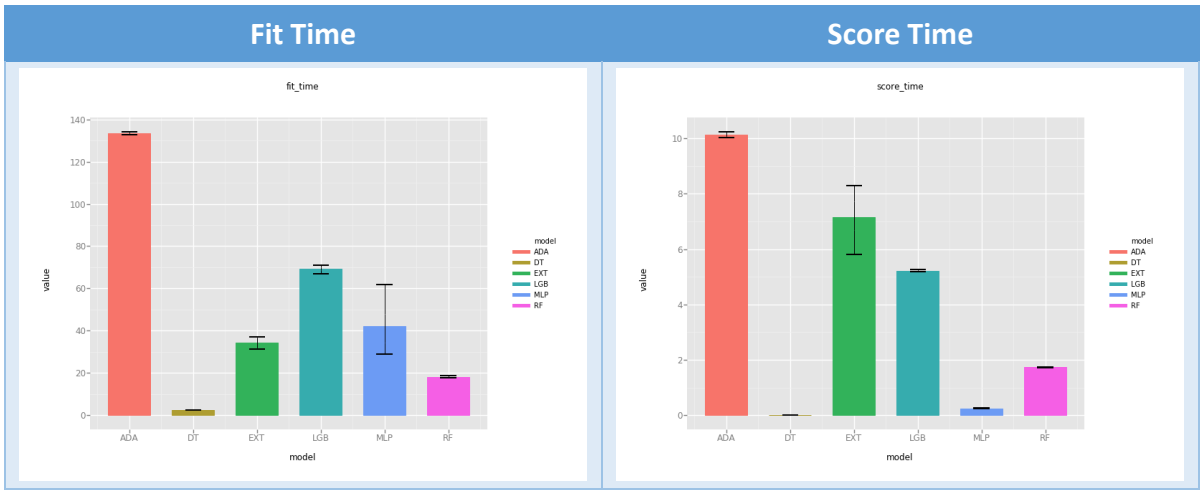
另外，為了提高驗證的可信度，我們會運用 Cross Validation，並採用 Stratified K-Fold 去做資料的切割。即每個類別會分開抽出樣本，讓類別的數量分佈在 Train Set 和 Valid Set 上都保持一致。

然而，我們發現有一 TripType 的類別僅有四筆資料，也就是說如果做 Stratified K-Fold 我們頂多只能做到 4-Fold。為了解決這個問題，我們採用了 Repeated Stratified K-Fold，即做 R 次的 K-fold，而這 R 次每次切出來的 Fold 都會不同，最後我們選擇 3 x 4 Repeated Stratified K-Fold，如下：



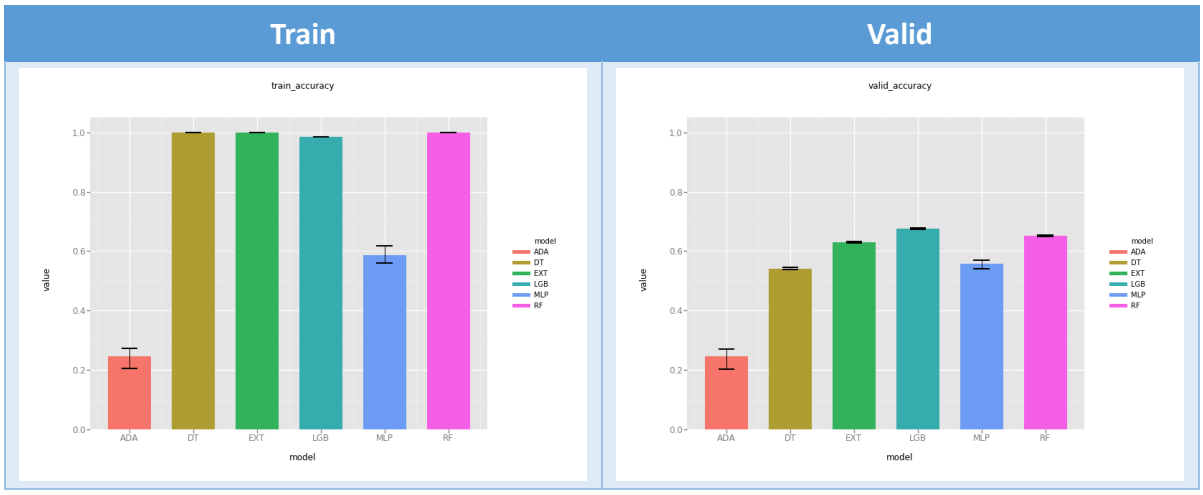
七、 數據結果 （Analytical Result）

A. Fit Time and Score Time



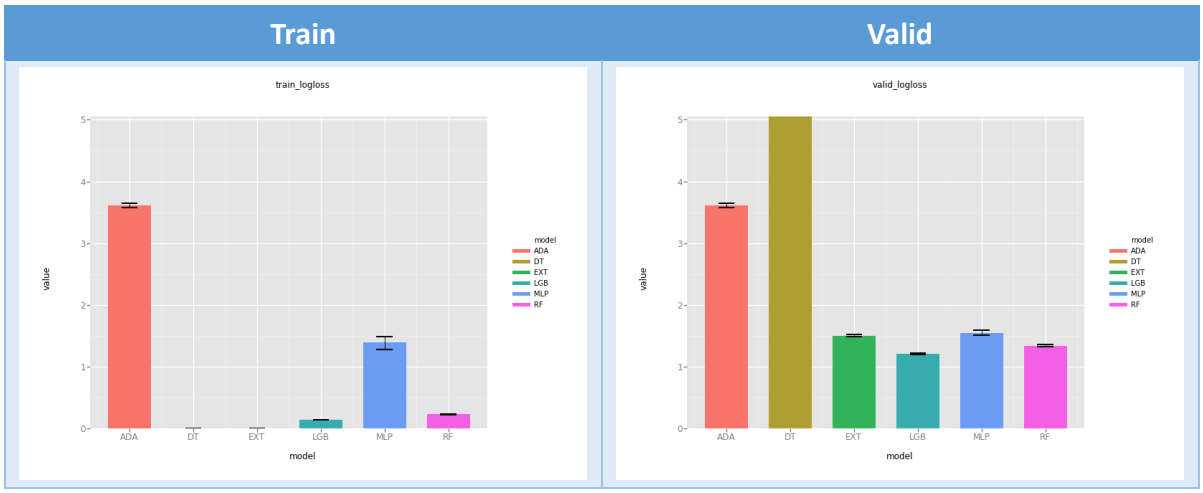
整體而言，最花時間的是 AdaBoost 演算法，無論是 fit time 還是 score time 都花了最久的時間。而 DecisionTree 這一最簡單的演算法則是在兩項指標上都耗費最少的時間完成。另外，可以特別觀察到 MLP/NN 演算法在 fit time 上雖然時間較長，但在 score time 上時間卻是極短的。最後根據 error bar，可以發現 fit time 上 MLP/NN 所花的時間較不穩定，score time 上 ExtraTrees 較不穩定。

B. Accuracy



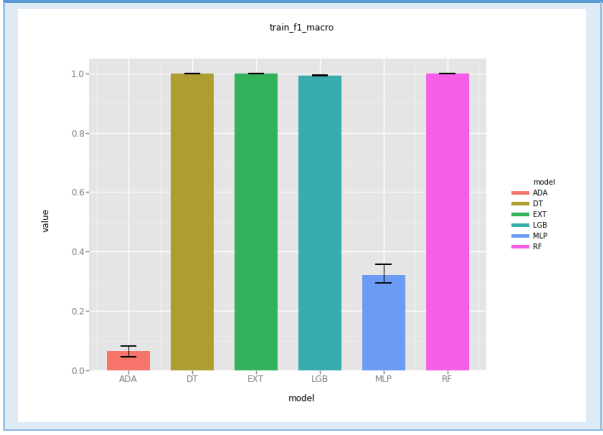
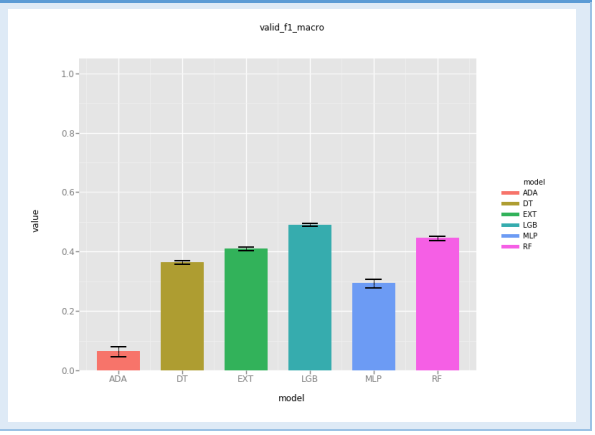
比對 Train 和 Valid Accuracy 的結果我們可以發現到，Decision Tree、ExtraTrees、LightGBM 和 RandomForest 有 overfitting 的現象。而 AdaBoost 和 MLP/NN （因為有做 early stopping）則無。而觀察 Valid Accuracy 的結果可以發現到 LightGBM 有最好的成績。

C. Log Loss



從 Log Loss 也可以發現到 Decision Tree、ExtraTrees、LightGBM 和 RandomForest 有 overfitting 的現象。而 DecisionTree 本身無法做到 soft classification，所以在 valid logloss 上有爆表的情形。以 Valid Log Loss 而言，LightGBM 也是有最好的成績。

D. F1 (macro)

Train	Valid
	

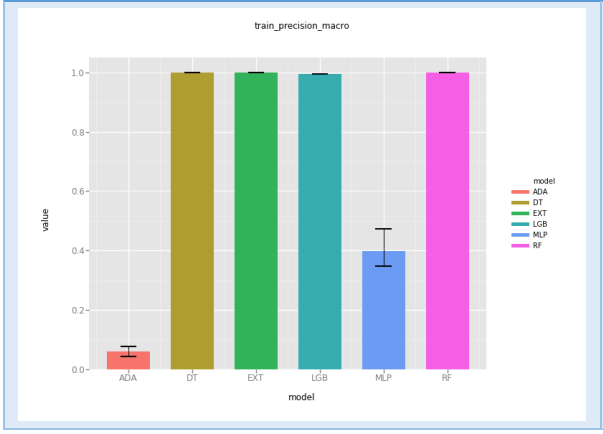
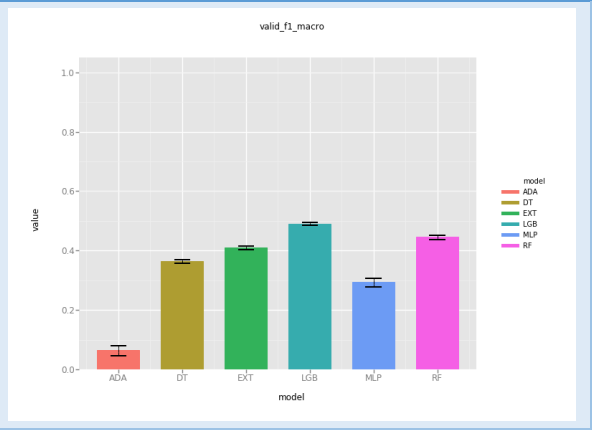
同 Accuracy

Overfitting: DT、EXT、LGB、RF

Best Valid Score: LGB

Worst Valid Score: ADA

E. Precision (macro)

Train	Valid
	

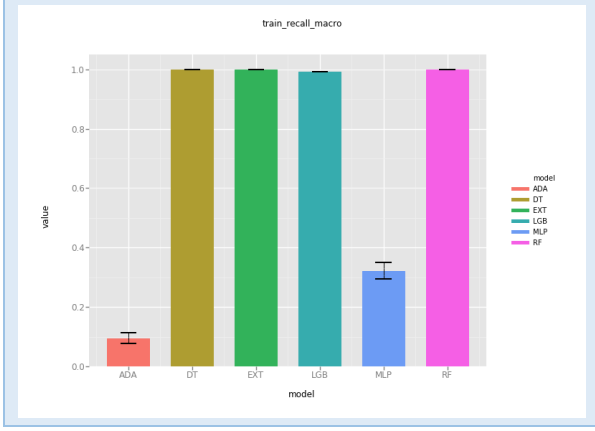
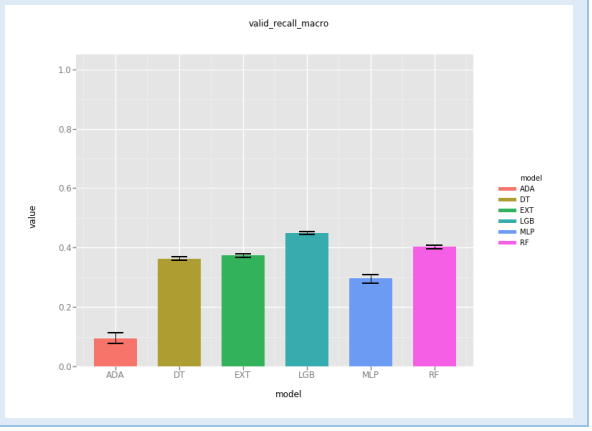
同 Accuracy

Overfitting: DT、EXT、LGB、RF

Best Valid Score: LGB

Worst Valid Score: ADA

F. Recall (macro)

Train	Valid
	

同 Accuracy

Overfitting: DT、EXT、LGB、RF

Best Valid Score: LGB

Worst Valid Score: ADA

八、 結論 （Conclusion）

從此次作業，我們可以發現運用不同 model 在問題上，會有非常不同的效能與結果。整體而言，根據 Repeated Stratified K-Fold Cross Validation 不同面向的成績，我們認為 LightGBM 表現最優、RandomForest 與 ExtraTrees 次之，再來是 MLP/NN，最後則是 AdaBoost 與 DecisionTree。另外，我們發現 AdaBoost 除了在 Log Loss 上優於 DecisionTree 外，其表現都較差，而這是一個令人頗訝異的結果，因為 AdaBoost 是 ensemble 多個決策樹的演算法，其表現理應較佳。後來，我們分析了一下背後的原因可能是因為此次的多類別分佈極為不平均，而導致 AdaBoost 中的每個分類器的預測都會傾向給予權重較大的幾個類別，並將上一個分類器分對的資料再度分錯，而無法達到越來越好的效果。而最後在進行加權投票時，分類器彼此之間不但沒有「合作」，反而是相互「抵觸」了，而導致最後的結果變糟。最後，此次作業中，礙於時間的緣故，我們對不同 model 的參數並無做太多的調整，只有讓特定一些影響較重的參數保持一致（如 n_estimators 和 max_depth）。若要改進的話，理應針對不同 model 去做參數的 tuning，並在將各自最好的結果拿來進行最後的比較，而這將是我們在日後可以改進的方向。