Algorithm

Logistic Regression

Introduction :

於訓練之初，我們採用邏輯回歸演算法 (logistic regression) 搭配一對多 (OVA)、一對一 (OVO) 概念進行分類，試圖找出特徵 (features) 與類別 (label) 之間是否存在簡單的線性關係。

Result :

有鑒於部分顧客資料缺人口資料表 (demographics) 或 服務資料表 (services) 中的資料，我們嘗試使用三種特徵 (features) 組合進行訓練，分別為人口資料表 (demographics) 之特徵, 服務資料表 (services) 之特徵, 所有資料集之特徵，然而效果不彰，皆只有約 0.2 之準確度，顯示出特徵與類別之關係並不全然線性可分或是存在雜質。

SVM

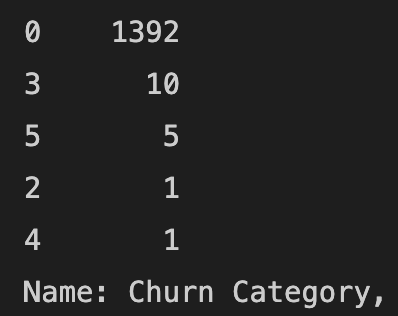
Introduction :

根據邏輯回歸演算法 (logistic regression) 所得出之推論：資料可能含不少雜質或是不容易線性分割。

且資料特徵數較多因此嘗試採用支撐向量分類演算法 (SVM)，搭配逕向基核函數核 (Radial basis kernel function) 用意在於提高維度，試著求出較佳之分線。再者，為了避免過度擬合 (overfitting)，C 值設為 0.1，gamma 值則交由套件自動算出。且欲預測之類別為多類別之分類，分類上選擇一對多 (OVA) 產生 N\*(N-1)/2 共 10 種分類器選擇出最適合的分類。

Result :

E\_in = 0.257,E\_validate = 0.168  
乍看之下似乎獲得不錯結果，Kaggle 上之 Fscore 卻只有 0.14，且實際預設 TestID 類別如下:



顯示出訓練出之支撐向量分類 (SVM) 模型在類別 0 以外之分類並沒有如預期中學習。

AdaBoost-Decision-Tree

Introduction :

由於支撐向量分類演算法 (SVM) 之分類方式或許過於強大，以致於無法學習到類別數量較少之分類，我們嘗試採用具條件式判斷的樹狀結構搭配自適應增強演算法 (AdaBoost) 方法聚合一些較弱之分類。

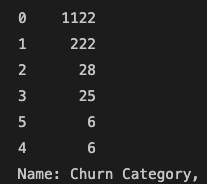
透過自適應增強演算法 (AdaBoost) 中的機制取出樣本 (sample) ，並使用決策樹 (decision tree) 訓練後再交由自適應增強演算法 (AdaBoost) 中的機制更新資料權重，迭代數輪候算出所需之數值 (alpha)。其中決策樹 (decision tree) 長得越高會導致Ein值等於0的情況，也代表著εt = 0 => α=∞

故須限制決策樹 (decision tree) 高度，讓樹 (tree) 變成弱決策樹 (weak decision tree)，這些弱決策樹 (weak decision tree) 將會以投票 (voting) 方式來決定預測的結果。

Result :

E\_in = 0.145 ; E\_validate = 0.275

而 Fscore 上升至 0.29，TestID 類別預測如下 ：



預測分布上比起前述兩種演算法似乎更靠近了原訓練資料之分布狀況，雖準確度提昇但仍不盡理想，惟自適應增強演算法 (AdaBoost) 對雜質 (noise) 影響極為敏感，所以推測資料預處理需要更完善或是須改善演算法對雜質的容忍度。

XGBoost

Introduction :

極度梯度增強演算法 (eXtreme Gradient Boost) 為一種梯度增強樹狀結構算法（Gradient Boost Decision tree）的實現，不同於自適應增強演算法 (AdaBoost) 在訓練新一輪的樹時調整樣本錯誤權重，極度梯度增強演算法 (XGBoost) 目標函式中加入前 (t - 1) 輪樹枝預測結果以保證新一輪的樹將更擬合資料集，此外，其在訓練上引入 Bagging 概念並在目標函式上加入了正則項以改善自適應增強演算法 (AdaBoost) 對雜質 (noise) 過度敏感的問題。

Result :

在實驗結果上，只須三層樹即可達到非常優異的  E\_in = 0  和  E\_val = 0.001 。Fscore也略好於上述自適應增強演算法 (AdaBoost)。

Self-Defined Desicion Tree

Introduction :

此名詞是我們定義出的一套概念，底層實作方法可抽換自適應增強演算法 (AdaBoost) 或極度梯度增強演算法 (XGBoost) 等演算法，概念說明如下：

在訓練各種演算法的模型時，我們採用所有表格的欄位一起進行訓練，因此在預測TestID的類別時理當需要這些特徵欄位輸入模型中進行運算，然而每筆 TestID 不一定皆存在於所有表格中，代表 TestID 並不含所有欄位資訊，因此將會面臨 TestID 並無足夠特徵數輸入由所有表格欄位訓練出之模型的狀況。  
Solution ：  
我們採三層樹的架構以處理上述之狀況。第一層樹由所有表格的欄位訓練出之模型進行預測，如TestID有缺失任一欄位無法進行預測，則進入第二層樹，以「services」資料表進行訓練所得出的模型進行預測，如有缺失「services」表格中任一欄位資料則進入下一層樹，由「demographics」表格所訓練出之模型進行預測。用意在於，避免TestID 缺乏特徵資訊而無法進行預測導致只能隨機猜測其類別或是固定猜測其類別。