

# HTML 2021 Final Project




## Introduction

Team: teamname

Member: R10922132 吳峻銘 R10922110 謝賀淇 R10944016 劉怡萱

public score: 0.35543

private score: 0.28870

103	▼ 67	teamname	  	0.28870	33	2d
-----	------	----------	--	---------	----	----

## Data Preprocessing

觀察資料中發現很多 missing value 因此我們根據資料的性質和相依性去做處理。

### 刪除屬性

判斷此欄位跟最後顧客會不會決定續約的關聯性，如果沒有相關即把此屬性刪除，或是此屬性的性質可被其他屬性取代也直接刪除。

service.csv	location.csv	demographic.csv
Count, Quarter, Referred a Friend, Payment Method, Paperless Billing, Internet Type, Streaming TV, Streaming Movies, Streaming Music	Count, Country, State, Lat Long, City	Under 30, Count, Senior Citizen, Dependents

### 數值化

因為我們的模型輸入都是數值所以要把類別字串數值化，基本上只要 one-one mapping 就可以了，但在之後 SVM 和 MLP 的模型要考慮正歸化的操作，所以 mapping 的 range 要注意一些。

service.csv	location.csv	demographic.csv
Offer, Phone Service, Internet Service, Multiple Lines, Unlimited Data, Online Security, Online Backup, Device Protection Plan, Premium Tech Support	Zip Code	Gender, Married

### 平均數

把屬性資料切成 train、test、valid 並觀察 test 的資料分佈，如果它的 variance 值沒有很大就用平均數插入。

location.csv	demographic.csv	satisfaction.csv
Longitude, Latitude	Age, Number of Dependents, Gender, Married	Satisfaction Score

## 眾數

把屬性資料切成 train、test、valid 並觀察 test 的資料分佈，如果有個類別佔的比例高達 7, 80%，就用這個眾數插入。

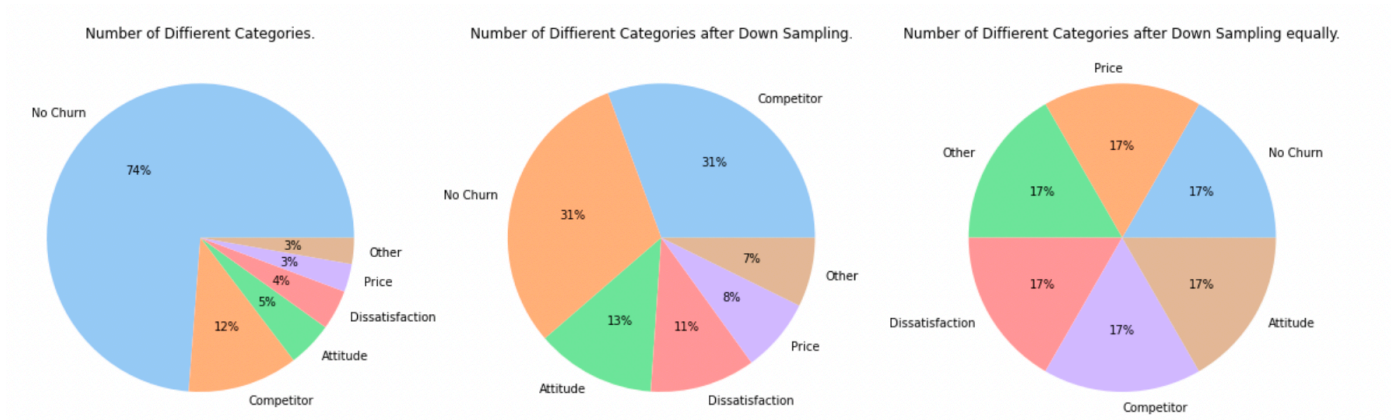
service.csv	location.csv
Phone Service, Internet Service, Multiple Lines, Contract, Online Security, Online Backup, Device Protection Plan, Premium Tech Support, Unlimited Data	City

## 線性回歸內插

把屬性資料切成 train、test、valid 並觀察 test 的資料分佈，如果有兩者的 correlation 數值高於0.7就可以用兩者進行線性回歸內插。

service.csv
(Tenure in Months, Total Revenue), (Total Long Distance Charges, Avg Monthly Long Distance Charges), (Monthly Charge, Total Charges), (Total Revenue, Total Charges)

## Data Imbalance



在探勘資料的過程中會發現最後的 label 會很不平均，其中 "No Churn" 這個類別特別多佔了 74%，這個 data imbalance 的狀況可能會造成最後分類器訓練有誤差，對於 data imbalance 的解決發法主要可以分成 upsample、downsample、調整 loss function，然而我們所採用的模型不太需要調整 loss function，資料集額外的資訊也太少做 upsample 的效果有限，因此我們考慮將多數的類別做 downsample 的動作，詳細的辦法就是把多數的類別隨機 sample 到一定的值，將所有類別平衡以後對每個類別抓取的特徵值才會平均。

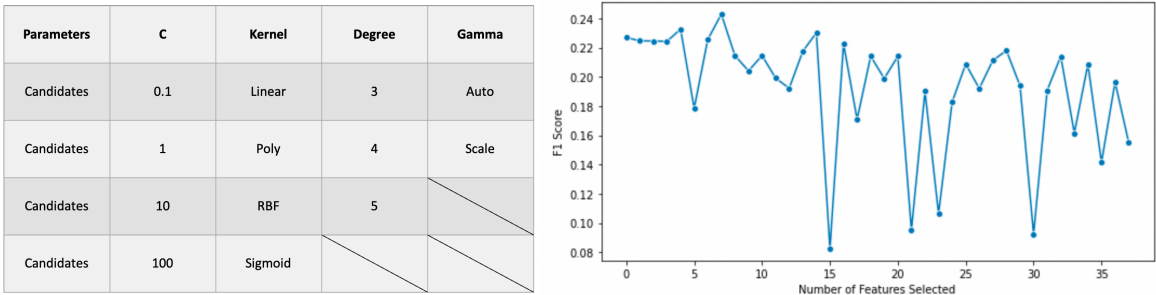
## Approach

### SVM

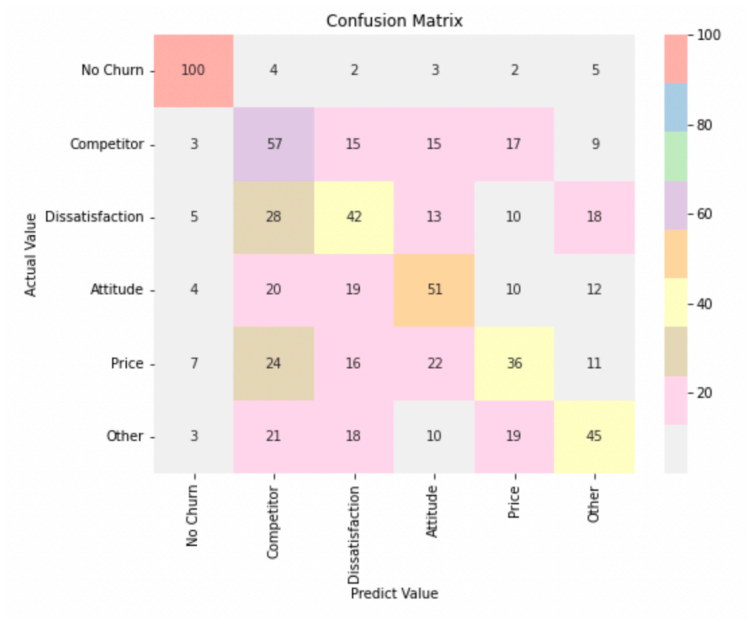
首先，由於無法保證應當取哪些 Features 可以使得 Performance 為最佳，因此將對每一 Feature 都執行一次訓練藉以得到相對應的結果，而在此的特徵挑選將以計算 Target Class 彼此之間的資訊量來得到 K 個最佳特徵。

另外，對於 SVM 而言當中最為重要的參數莫過於 C、kernel 以及 gamma，但如同課堂所述這些參數無法直接給予一個最佳的值，皆需藉由資料資訊以及訓練之後才能得知，於是這裡會先以 Grid Search 的方式找出每個 Feature Selection 之下的一組最為適合的參數。

因此，可以由下圖的參數組合關係得出不同數量的特徵選擇下之結果。並且由過程可知最好的情況為選擇 5 個特徵 (Satisfaction Score, Contract, Offer, Total Long Distance Charges, Total Charges) 時使用 C: 10, Kernel: RBF, Gamma: Scale 之參數，並且可以由 Filter Selection 的圖中得知過多的 Feature 對於實驗結果而言未必會有更好的結果。



並且在 Macro F1 Score 評測之下的 Performance 最高為 0.25，而相對應的 Confusion Matrix 如下圖所示；可以從該圖該模型對於 Class No Churn 的能力最佳，其餘的預測能力則仍然非常混亂。



然而當在 Training 時最高可以得到 0.235 的 F1 macro score，但是將此模型預測 Test 並放上 Kaggle 時，可以得到的 Public score 大約座落在 0.23~0.25 之間，並且其對應的 Private score 可以比 Public 再多個 0.05 即為 0.235 ~ 0.255 左右。

## Gradient Boost

如同上課所提到的，Boosting 會在每一次訓練時試著去更新資料的權重，提升原本錯誤的資料，降低原本已經做對的資料，讓模型可以在每一次訓練的時候做得更好。而 Gradient Boosting 可以看成是 AdaBoost 的延伸，每一輪可以使用定義的 error function 做最佳化，使 model 越來越好。

本次 final 使用 sklearn 的 GradientBoostingClassifier 當作最後的 kaggle 比賽的模型，error function 使用預設的 'friedman\_mse'，我們針對不同的參數用 GridSearch 的方式依序找出最佳的參數做訓練，以下為不同的參數做實驗比較得到的結果：

## n\_estimators

此參數表示弱分類器的數量（如上課講的 weak base learning algorithm），我們比較了不同弱分類器數量的影響，由下表可以發現分類器數量越多，training 和 public 的分數就越高，但是 private 的成績卻是往下的，可以推論數量超過 100 很有可能已經造成 overfitting。

n_estimators	500	300	100	50
training score	0.78733	0.76427	0.70071	0.65371
public score	0.35014	0.32955	0.33248	0.33645
private score	0.29375	0.29970	0.31412	0.31173

## learning rate

由下表可以看到較小的 learning rate 通常 training 分數較低但private 分數較好，推測可能因為 learning rate 小使得 model 對一些特定的特徵可以學得更好，但通常 learning rate 越小所需要的分類器就要越多。

learning rate	0.8	0.5	0.1	0.05
training score	0.71560	0.68206	0.55906	0.52233
public score	0.34674	0.32620	0.32094	0.30120
private score	0.30040	0.30686	0.31079	0.28882

## max\_depth

max\_depth 決定了分類器的深度，深度越大表示 model 越複雜，可以讓訓練集的資料更擬合，但同時就可能會讓測試集的資料越差。

max_depth	1	2	4	8
training score	0.55906	0.72259	0.93961	1.0
public score	0.32094	0.32390	0.34304	0.33334
private score	0.31079	0.26859	0.29895	0.28355

總結來說，在調參數的過程中發現，training score 的分數越高會因為 overfitting 而在 public / private 的成績就越不理想，我們平均每個 private 成績都會比 public 成績掉 0.02 左右，且跌落的幅度很大，因此很難去用 training 和 validation 的分數來推測最後的走向。

## MLP

MLP 是多個 perceptron 疊在一起的模型，可以把它看成最基本的類神經網路架構，就我們所知MLP的層度越深可以學習的特徵表象越多，然而太深也會讓模型太複雜很難收斂，於是我們要實驗一下資料集中在MLP中各深度的表現並用GridSearch找出最佳的參數解。

## hidden layer

可以發現 hidden layer 在只有一層表現最好，在層數增加後有遞減的趨勢，猜測是特徵在前處理後沒有太複雜所以一層就可以用了。

hidden layer	1	2	3	4
training score	0.21867	0.14596	0.15880	0.16427
public score	0.18356	0.17735	0.14520	0.15785
private score	0.17927	0.16900	0.14567	0.16564

## activation function

經過實驗可以發現此資料集在 relu 和 identity 的表現會比較好，故之後再增大模型時可以參考這個實驗採用相同的 activation function

activation function	relu	identity	logistic	tanh
training score	0.21867	0.19211	0.14596	0.14596
public score	0.18356	0.19338	0.13761	0.13761
private score	0.17927	0.18467	0.14055	0.14055

整體來看 MLP 在這個資料集上會有點沒有辦法收斂的趨勢，train 和 valid 都在 0.2 上下飄，推測可能是因為訓練資料只有幾千筆的關係，在訓練資料變多情況下有機會提升模型的表現。

## Comparison

### Effectiveness

*GradientBoost > SVM > MLP*

根據我們前面實驗三個模型在不同參數下的最佳解可以發現在 F1 score 中表現最好的是 Gradient Boost，其次是 SVM 再來才是 MLP。

### Efficiency

*SVM > MLP > GradientBoost*

在 Efficiency 的比較上，我們計算了實際操作在這次競賽資料的時間，如下表，我們以每個方法中最好的 model 來衡量訓練時間，可以發現 Gradient Boosting 的時間最長，SVM 計算的時間最短，前者考慮到分類器的數量多寡以及深度會影響到訓練的時間，而後者沒有這樣的問題所以時間相對來講縮短許多。

另外，我們也參考了[1]的文章，計算了三種方法的訓練時間複雜度，其中  $M$  表示分類器的數量， $n$  表示資料量， $d$  表示模型的複雜度。

	Gradient Boost	SVM	MLP
Time(s)	44.97	2.66	7.11
Time Complexity[1]	$O(Mn\log(nd))$	$O(n^2d^2)$	$O(nd)$

## Scalability

$MLP > GradientBoost > SVM$

由上表的時間複雜度來看，當  $n$  越大，Gradient Boosting 的時間複雜度就越高，計算的成本也就越大，而 MLP 的時間複雜度最低。因此可以發現若數據量規模大增，Gradient Boosting 和 SVM 的方法相對於 MLP 就不是那麼適合應用在大規模數據的資料上。

另外，我們發現傳統的 Gradient Boosting 在訓練時間上就會變得沒有效率，因此在 2016 年有人提出了 XGBoost[2]，他是 Gradient Boosting 的改良版，考慮了 scalable 的問題，也提供平行化處理的方式，可以大幅提升效率和時間複雜度。我們也有額外利用 XGBoost 進行實驗，發現與 GradientBoostingClassifier 的結果很相近，推測因為此次競賽資料量不多，因此使用 XGBoost 沒有太顯著的表現。

## Interpretability

$GradientBoost > SVM \models MLP$

先從 SVM 說起，SVM 之所以具有較差的可解釋性能力是因為其過程中的 Kernel 轉換，並且由於在此最佳的 SVM 模型並非只是使用較為單純的 Linear Kernel，因此對於這種轉換之後在超平面進行劃分後分類的行為當 Feature 數的挑選一高時則其解釋性能力就越差。至於 MLP 則是如同常見的 NN 一樣具有偏向黑盒子一般的缺點，然而由於我們的 MLP 只使用一層 Hidden Layer，因此我們判斷其表現應該會與 SVM 來的相近。

相較之下以建立 Additive tree 的 Gradient Boost 之可解釋性能力就來的比其他兩個明顯傑出許多，因為我們可以很直觀地透過檢查所有建立的 Tree 來判斷模型的分類是如何建構的，換句話說從中就可以了解到哪些 Feature 的重要性或 Sample 是如何被模型給判定的。

## Conclusion

根據前面比較三者在不同模型的比較，我們認為 MLP 是最適合的模型，因為他在 scalability 上有最好的表現，在擁有最低時間複雜度下可以應對公司日益俱增的顧客人數，以系統穩定性的觀點是最好的選擇，即使在現有的資料集的 effectiveness 不是最好，但在資料集變多的狀況下可以藉由調整 hyperparameter 讓表現變好。

## Contribution

吳峻銘: MLP, data preprocessing

謝賀淇: SVM, data preprocessing

劉怡萱: GradientBoost, data preprocessing

## Reference

[1] <https://medium.com/analytics-vidhya/computational-complexity-of-ml-algorithms-1bdc88af1c7a>

[2] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.