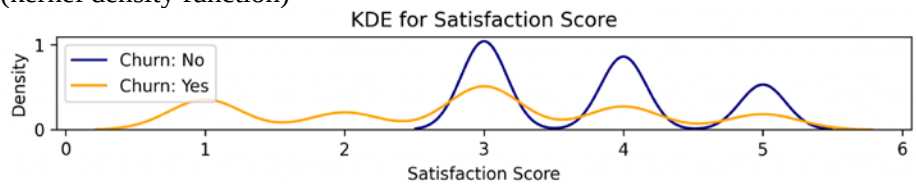
Machine Learning Final Project

R10725058 王佩晨　　R10725035 呂文楷　　R10725011 陳佑甄

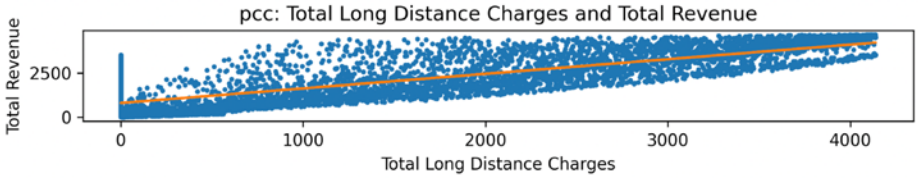
一. Exploratory Data Analysis

我們對資料進行探索式分析，藉由 KDE 觀察特徵之間的機率分布並用 PCC 觀察兩兩特徵之間的相關性。由於資料類別不平均，因此觀察分佈時是以是否 churn 來進行。

1. KDE (kernel density function)



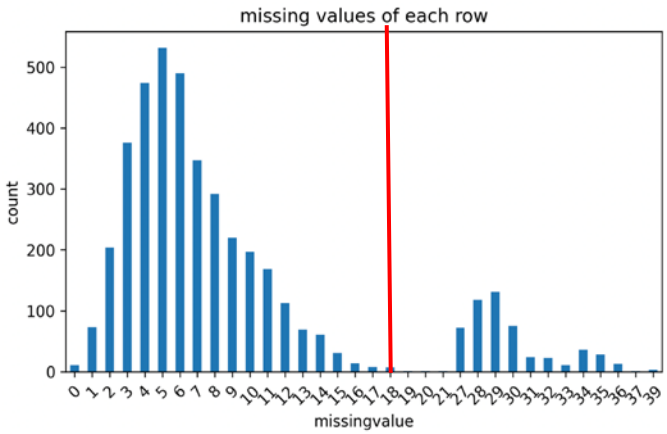
2. PCC (Pearson Correlation Coefficient)



二. Data Preprocessing

1. Data Cleaning

i. 將超過 18 個 features 為空值的 row 刪除，最後 training data 為 3681 筆



ii. Under 30, Senior Citizen / Age:

利用 Age 不為空值的欄位去補 Under 30 和 Senior Citizen 的為空值的欄位，最後再用 Under 30 和 Senior Citizen 去做互補。

iii. Dependents / Number of dependents

利用 Number of dependents 不為空值的欄位去補 Dependents。

iv. Referred a friend / Number of Referrals

利用 Number of Referrals 不為空值的欄位去補 Referred a friend。

v. Internet Service / Avg Monthly GB Download, Unlimited Data, Online Backup, Online Security

利用後面四個 features 的欄位將 Internet Service 補值。

vi. Phone Service / Avg Monthly Long Distance Charges

利用 Avg Monthly Long Distance Charges 不為 0 的欄位對 Phone Service 為空值的欄位補值。

vii. City / Zip Code

利用 uszipcode 套件將 Zip Code 不為空值的欄位進行轉換，對 City 補值。

viii. Latitude, Longitude / Lat Long

利用 Lat Long 來對 Latitude 與 Longitude 兩欄位進行補值，若 Lat Long 無缺值且兩欄位中有缺值就可以進行補值。

ix. Total Revenue

我們發現 Total Charges - Total Refunds + Total Extra Data Charges + Total Long Distance Charges = Total Revenue，故以這個規律來進行補值。補的方法為若五個變數中有只缺一個值的 row，就利用另外四個沒缺值的 features 取補值。

Train Feature:	Before		After	Test Feature:	Before		After
Senior Citizen	839	->	493	Senior Citizen	330	->	196
Dependents	862	->	493	Dependents	321	->	202
Referred a Friend	449	->	55	Referred a Friend	321	->	202
Internet Service	441	->	50	Internet Service	339	->	27
Phone Service	482	->	112	Phone Service	324	->	198
City	867	->	518	City	351	->	221
Latitude	882	->	512	Latitude	341	->	212
Longitude	879	->	503	Longitude	331	->	218
Total Charges	465	->	187	Total Charges	338	->	254
Total Refunds	471	->	197	Total Refunds	326	->	237
Total Extra Data Charges	482	->	209	Total Extra Data Charges	339	->	244
Total Long Distance Charges	446	->	175	Total Long Distance Charges	355	->	259
Total Revenue	453	->	195	Total Revenue	340	->	247

2. Encoding

由於資料中有些欄位是屬於 category 的資料，故在丟進模型訓練前要先對其進行 encoding，我們嘗試了三種 encoding 的方法如下：

i. Label Encoding：

為把每個類別 mapping 到某個整數，不會增加新欄位。適合用在 Ordinal Data ，也就是類別有階層相關的欄位。

ii. Hash Encoding

適合用在 Nominal Data ，也就是類別間無階層關係的欄位。可以使用 n_component 參數來固定轉換後的維數。像是 City 欄位中有多個類別，若利用 One-hot encoding 會產生 sparse 的問題，故此類欄位可以利用 Hash encoding 設定轉換後的維度進行 encoding。

iii. Binary Encoding

適合用在 Nominal Data，也就是類別間無階層關係的欄位。為 Hash encoding 與 One-hot encoding 的結合。轉換步驟為分類特徵 => 數值 => 二進制數 => 將二進制數拆分為不同的 columns，也適合用在有大量類別的時候。

3. Imputating

i. Simple Imputation

將資料分成兩種類型，float 類型的資料使用中位數補值，
category 類型的資料使用眾數補值。

ii. KNN Imputation

將所有資料放入 KNNImputer 並令 n_neighbors=5。

iii. Simple + KNN Imputation

4. Scaling

i. Min-Max Scale

讓每個特徵中的最小值變成 0，最大值變成 1。讓資料維持大小關係但縮放到[0,1]之間。

ii. Robust Scale

可以有效的縮放帶有 outlier 的數據，透過 Robust 如果資料中含有異常值在縮放中會捨去。

三. Feature Engineer

1. Feature Augmentation

i. City_avg_income

透過 uszipcode 套件取得該城市家庭所得的中位數(median_household_income) ，並新增為一個 feature。

	City	City_avg_income
1	Rosamond	59142.0
2	Huntington Beach	86043.0
3	San Jose	104705.0

ii. stream_total

將 Streaming Movies, Streaming TV, Streaming Music 三個資料合併計算總共使用的 streaming 相關服務總數。

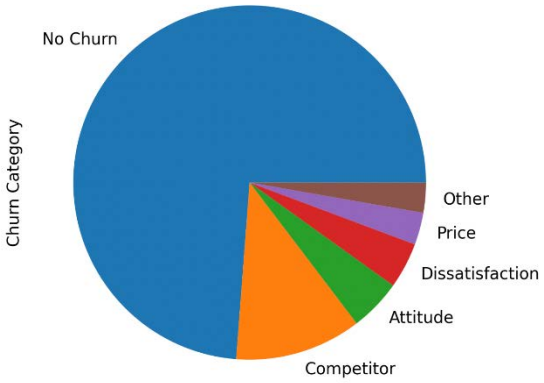
2. Feature Selection

使用十八種 Feature Selection Methods 進行各種組合的實驗，經過多次觀察後，最後挑出以下兩種效果較佳的方法進行 Feature Selection。

以下兩種 Feature Selection Methods 都分別實驗過 f_classif 與 mutual_info_classif 的演算法來計算特徵相關性。其他方法詳見第八點附錄。

i. SelectKBest
選擇 K 個最好的特徵，這邊 K=20。
ii. SelectFwe
對每個特徵使用常見的單變量統計檢驗，選擇與 Family-wise error rate 對應的 p 值。

3. Smote (Synthetic Minority Oversampling Technique)
由於觀察到 label 中 No Churn 的數量遠大於其他總數量的加總，為了避免模型訓練或預測時數量多的類別容易被預測到，因此對其他 label 進行 oversampling，使此資料集成為平衡的資料集。



四. Machine Learning Algorithm

實驗七種演算法，經過與各種前處理方法的交叉實驗後挑出以下三種表現較好的模型，其他方法詳見第八點附錄。

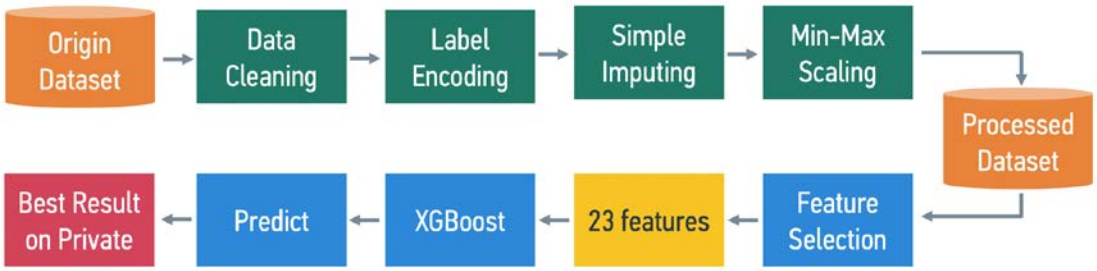
1. Bagging
Bagging 的優點在於原始訓練樣本中有 noise 資料，透過 Bagging 抽樣就有機會不讓有 noise 資料被訓練到，所以可以降低模型的不穩定性。

2. XGBoost
結合 Bagging 和 Boosting 的優點。保有 Gradient Boosting 的做法，每一棵樹互相關聯，目標希望後面生成的樹能夠修正前面一棵樹犯錯的地方。XGboost 採用特徵隨機採樣，在生成每一棵樹的時候隨機抽取特徵，因此在每棵樹生成時不會每次都拿全部的特徵參與決策。且 XGboost 在目標函數添加了標準化，防止 overfitting 的情況產生。

3. LightGBM
LightGBM 採用 Leaf-wise tree growth 的方法進行分裂，這樣節省了大量分裂節點的資源。Leaf-wise 的分裂方法能產生比 XGBoost 使用 level-wise 分裂方法更復雜的樹，能使得模型得到更高準確率。

五. Experiment

1. Final best private score (檔名: 2022-01-15-203221-xgboost-mannual impute)



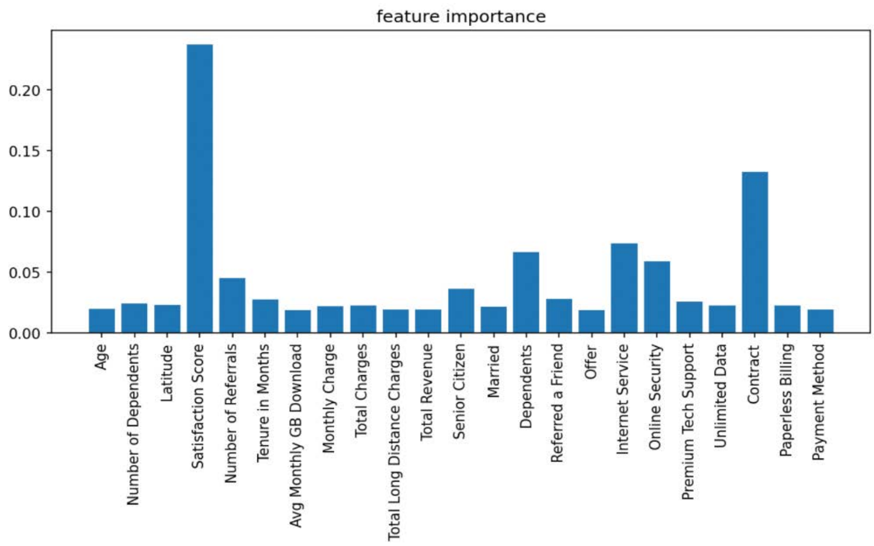
i. SelectFwe by f_classif score
The highest uncorrected p-value for features to keep is 0.1.

ii. XGBoost parameters

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=6,
              num_parallel_tree=1, objective='multi:softprob', predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
              subsample=1, tree_method='exact', use_label_encoder=False,
              validate_parameters=1, verbosity=0)
```

iii. 5 CV by StratifiedShuffleSplit on test size 0.3 with randomseed = 1

iv. Feature importance



2. Compare three approaches

	Baseline	Approach 1	Approach 2	Approach 3
Short description	Comparing only	Best Private Score	Best Public Score	Expected best model by us
Encoding	Label Encoder	Label Encoder	Label Encoder	Binary Encoder
Imputating	Simple Impute	Simple Impute	Simple impute + KNN impute	Simple impute + KNN impute
Scaling	No	Min-Max Scaling	No	Min-Max Scaling
Feature Selection	No	SelectFwe by f_classif score	SelectKBest by f_classif score	SelectFwe by f_classif score
Training shape	(3681, 42)	(3681, 23)	(3681, 24)	(3681, 33)
Smote	False	False	True	True
Model	Bagging	XGBoost	LGBM	XGBoost
Time	0.38 sec*	**0.14 sec***	2.44 sec*	4.95 sec*
Public Score	0.28733	0.33763	**0.34983**	0.31444
Private Score	0.29285	**0.36041**	0.32161	0.31837

*Average runtime for one training iteration

六. Discussion

1. Efficiency

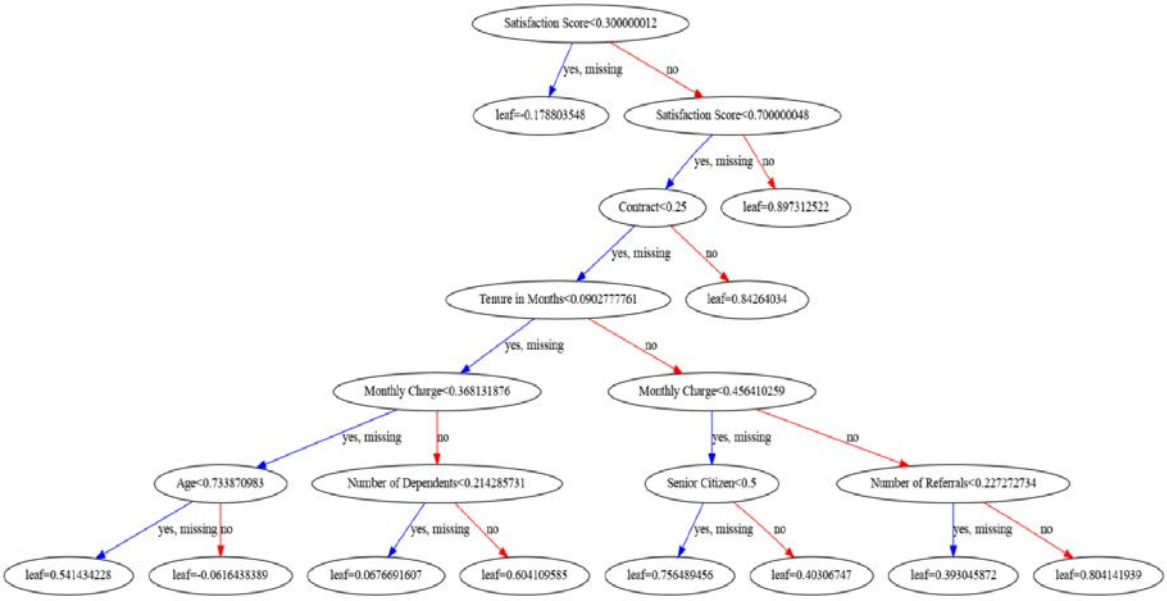
綜合所有實驗結果，我們發現有無加上 smote 會嚴重影響到 Time Efficiency ，由於資料量不多因此影響層面不大，但若未來有較大的資料量就考慮不使用 smote，改為使用 under sampling 來平衡 Efficiency 與 Performance 的 trade-off。除此之外，Min Max Scaling 和 Feature Selection 也會些微影響到模型效率，有進行縮放或選擇較少參數皆會提升訓練速度。

2. Scalability

針對 Scalability，我們最終挑選的 XGBoost 和 LightGBM 架構在大量資料下皆有可擴充性[9][10]，在實驗的過程中，我們也有妥善利用這個特性，在一台電腦上進行平行執行緒運算，如下圖。當資料量非常大的時候，兩個模型皆可以拓展到更多台機器或電腦叢集進行分散式模型訓練。

3. Interpretability

Tree-based model 可以解釋每個 feature 的重要性(如下圖)，因此利用此類型的 model 可以針對重要的 feature 去做重點調整。以此實驗為例，在得知 satisfaction score 的重要性大後，可以針對此 feature 去做更精確的補值，或是於 production 階段中加強該資料在搜集時的精確度。



4. Overfitting

綜合實驗結果可以發現 XGBoost 在 Private Score 的平均表現優於 LightGBM。由於我們的資料集小於 10000 筆，LightGBM 可能會產生 overfitting 的情況，雖然 LightGBM 是個強大的模型並讓我們在 public score 上表現最好，考慮 overfitting 的情況，我們最後選擇繳交的 final model 為 XGBoost。

七. Conclusion

綜合以上實驗，我們最終推薦 Approach 1 (Best Private Score)為最佳的解決方案。在 Approach 1 中我們在各階段使用不同的方法，每個方法各其有優缺點，我們經過多次實驗從眾多組合中選擇此種組合，以在各項指標達到平衡。以下為針對 Approach 1 的優缺點：

- Pros
  1) 進行 Feature Selection 來縮短訓練時間、降低雜訊特徵並提高準確度。
  2) Simple Imputation 雖然無法確保補值的準確性，但可以確保不會因為補值而產生偏差。
  3) 使用 XGBoost 結合 Bagging 與 Boosting 優點，使平均表現佳。
  4) 使用 XGBoost 在未來若資料量大時具備可擴充性，可進行分散式模型訓練也可結合 GPU 進行訓練。
- Cons
  1) 相較於 linear model，較有可能產生 overfitting。
  2) Simple Impute 會直接將缺失值以單一方法補值，如果能有更多訓練資料，可以針對不同類型的參數去設計獨特的補值方法。
  3) XGBoost 有眾多參數，較難進行參數調整。

八. Reference

1. 初學 Python 手記#3-資料前處理(Label encoding、 One hot encoding)
2. Here's All you Need to Know About Encoding Categorical Data (with Python code)
3. Simple Methods to deal with Categorical Variables in Predictive Modeling
4. Here's All you Need to Know About Encoding Categorical Data (with Python code)
5. Handling "Missing Data" Like a Pro — Part 2: Imputation Methods
6. Imputation under non-normal distributions
7. xgboost: eXtreme Gradient Boosting
8. LightGBM: A Highly Efficient Gradient Boosting Decision Tree

九. Appendix

本實驗所有使用的 Feature Selection 和 Machine Learning Algorithm

Feature Section Method	Machine Learning Algorithm
1. GenericUnivariateSelect 2. SelectPercentile 3. SelectKBest 4. SelectFpr 5. SelectFdr 6. SelectFromModel 7. SelectFwe 8. SequentialFeatureSelector 9. RFE 10. RFECV 11. VarianceThreshold	1. Random Forest 2. Decision Tree 3. LinearSVM 4. KNN 5. Bagging 6. XGBoost 7. LightGBM

十. Members Workload

王佩晨(33%): Preprocessing, Feature Engineering, Model Experiment, Paper work

呂文楷(33%): EDA, Preprocessing, Model Experiment, Paper work

陳佑甄(33%): Preprocessing, Model Experiment, Paper work