Machine Learning Final Project

FinalProjecTeam

B07902037 蔡沛勳 B07902072 陳光裕 B07902128 蘇聖祐

一、Data Preprocessing

Preprocess主要分為以下三個階段,

1. Csv file preprocess:

   將 csv 檔讀進來後轉換為數字, 其中若是二分類的 feature (Yes / No) 則轉為 (1 / -1), 而若是多分類的 feature (eg. city) 則用 One-hot 的方式處理。然後再對具有相關性的或從屬關係的 features 各自調整, 最後刪除單分類及被包含關係的feature。具體如下:

   Demographics:

   若 Dependents 為 NaN 或 0 且 Number of Dependents 為 NaN 則將 Number of Dependents 設為 0。

   Drop:Count、Under 30、Senior Citizen、Dependents

   Satisfaction:

   直接使用 Satisfaction Score。

   Location & Population:

   由於 Country、State 都只有一種, 而 City 若用 One-hot 方式處理會有 1101 種分類, 太過細碎, 故三者都不使用。而 Latitude 或 Longitude 為 NaN 實則由 Lat Long 項取值。Zip Code 項則從 population.csv 中找出對應的 Population 並替換。

   Drop:Count、Country、City、State、Lat Long

   Service:

   service.csv 中包含 12 個二分類 features及 4 個多分類 features 依前面提到的方式處理, 而若 Referred a Friend 為 No 且 Number of Referrals 為 NaN 則將 Number of Referrals 設為 0。

   Drop:Count、Quarter、Referred a Friend

   Status:

   依 kaggle 競賽格式, 將No Churn、Competitor、Dissatisfaction、Attitude、Price 及 Other 轉為 0、1、2、3、4、5。

2. Merge all data and fill NaN:

   將前一部分五個 csv 檔的處理結果依照 Train_IDs 及 Test_IDs 對齊 Customer ID merge 成兩組 data, 再統一對 NaN 值去做處理。首先我們可分分為 X 中的 feature 及 y 中的 NaN, y 為 NaN 的 ID 屬於 train data 則丟棄該筆資料, 若屬於 test data 則將其填上數量最多的種類。若是 X 中的 features 為 NaN, 則先決定是否丟棄該筆資料, 若否, 對於前文中的二分類 feature NaN 值填 0，其他則分為四種填值方式，分別為：

   Fill 0：將 NaN 填上 0。

   Fill mean：將 NaN 填上該 feature 不含 NaN 的平均數。

   Fill forward：將 NaN 填上該筆資料往前第一筆非 NaN 的值。

   Fill backward：將 NaN 填上該筆資料往後第一筆非 NaN 的值。

3. Scaler

   為了使 SVM 等需要運算到距離的模型能夠正常運作，我們測試了兩種 scaler 對資料做最後一步處理, 分別是

   MinMaxScaler：(x - x.min) / (x.max - x.min), 將資料壓到 [0, 1]。

   StandardScaler：(x - x.mean) / x.var, 使資料 mean = 0、var = 1。

二、Machine Learning Models

我們選擇了八種模型來做測試, 分別為一個純手刻的用 Logistic Regression 實作的 One versus one model。Sklearn 函式庫中用 SVC 實作的 OneVsOne 及 OneVsRest 的 Classifier, 用 DecisionTreeClassifier 實作的 AdaBoost 及 Bagging 的 Classifier, 其餘還有GradientBoosting Classifier 及 RandomForest Classifier, 和整合上述六個 model 的 Voting Classifier。我們以 GridSearchCV 的方式調整模型參數, 得到結果如下：

1. Self-implemented One Vs One Logistic Regression model：

   由於是手刻出來的 model, 速度緩慢, 跑完四次 V_folds(5) 的時間為 21:35, Train F1 mean 為 0.216821, Validation F1 mean 為 0.214689, 兩者的 variance 都在 1e-7 左右, 分數不高但表現相當穩定且不會 overfitting。

2. Sklearn One Vs One Classifier：

GridSearchCV target：SVC(C、kernel、degree、gamma、max_iter)
OVO = OneVsOneClassifier(SVC(C = 59, kernel = ‘poly’, degree = 4, gamma = ‘scale’, max_iter = -1))

3. Sklearn One Vs Rest Classifier：

GridSearchCV target：SVC(C、kernel、degree、gamma、max_iter)
OVR = OneVsRestClassifier(SVC(C = 11, kernel = ‘poly’, degree = 4, gamma = ‘scale’, max_iter = -1))

4. Sklearn AdaBoost Classifier：

GridSearchCV target：DecisionForest(max_depth)、learning_rate、n_estimators
ABC = AdaBoostClassifier(DecisionTreeClassifier(max_depth = 4), n_estimators = 50, learning_rate = 1, algorithm = "SAMME")

5. Sklearn Bagging Classifier：

GridSearchCV target：DecisionForest(max_depth)、n_estimators、bootstrap_features、max_features、max_samples、n_jobs
BC = BaggingClassifier(DecisionTreeClassifier(max_depth = 7), n_estimators = 4, bootstrap_features = False, max_features = 0.9, max_samples = 0.3, n_jobs = -1)

6. Sklearn Gradient Boosting Classifier：

GridSearchCV target：n_estimators、learning_rate、loss
GBC = GradientBoostingClassifier(n_estimators = 50, learning_rate = 1, loss = "deviance")

7. Sklearn Random Forest Classifier：

GridSearchCV target：DecisionForest(max_depth)、learning_rate、n_estimators
RFC = RandomForestClassifier(n_estimators = 1000, min_samples_split = 4, criterion = 'entropy', max_depth = None, max_features = None, bootstrap = False, n_jobs = -1)

8. Sklearn Voting Classifier：

GridSearchCV target：DecisionForest(max_depth)、learning_rate、n_estimators
VC = VotingClassifier(estimators = [('AdaBoost', ABC), ('GradientBoosting', GBC), ('Bagging', BC), ('RandomForest', RFC),], voting = 'soft', n_jobs = -1)

三、**Experiments**

實驗分為兩個部分，分別針對 data preprocessing 及 machine learning model 做進一步的測試，並比較各自的數據來找出最好的 preprocessed data 及 machine learning model，最終將兩者結合再放上 kaggle 做測試。

Data Preprocessing：

本部分的測試使用 Sklearn 的用 SVC 實作的 One Vs One Classifier 來做比較測試，測試方式為以V_fold 分為五份做 train 及 validation 並將 F1 Score 收集起來，重複 20 次一共 100 筆

1. 比較兩種 Scaler (MinMaxScaler 和 StandardScaler) 、是否移除有 NaN 的資料、四種 fill NaN的方式，共 2 x (4 + 1) = 10 種組合的訓練結果：

MinMaxScaler：

	Fill 0	Fill mean	Fill forward	Fill backward
Save NaN	0.241589	0.287337	0.249973	0.254839
Drop NaN	0.231332			

StandardScaler:

	Fill 0	Fill mean	Fill forward	Fill backward
Save NaN	0.18575	0.20637	0.18961	0.19095
Drop NaN	0.194105			

2. 用結果最好的組合 (MinMaxScaler x Save NaN x Fill mean)，實驗發現移除 X_{35} 可以有較高的 F1 score, 以此為基礎接著再依序移除 X_{24}、X_{33}、X_{22}、X_3

35:Internet type DSL → 24:total refunds → 33:Offer E
→ 22: monthly charges → 3:married (Final validation F1 = 0.29315)

Machine Learning Model:

本部分的測試在 meow1 工作站上進行, 對不同 model 進行比較:

	Train F1 mean	Train F1 variance	Validation F1 mean	Validation F1 variance	Computing Time (Total)
AdaBoost	0.594921	0.000124	0.314446	0.000005	2’49’’
Gradient Boosting	0.998900	4.5*10^-7	0.335014	7*10^-5	18’27’’
Random Forest	0.938392	1.17*10^-5	0.312251	3.94*10^-5	14’14’’
Bagging	0.432289	0.000130	0.294688	0.000113	36’’
One Vs One	0.951935	4.29*10^-7	0.297713	5.21*10^-5	5’10’’
One Vs Rest	0.944285	4.06*10^-7	0.283582	4.48*10^-5	5’18’’
Voting	0.994210	6.88*10^-7	0.323433	0.000190	31’13’’

四、**Conclusion**

結合 Data Preprocessing 兩項實驗的結果, 再從 Machine Learning Model 中選取三個較具代表性的 model, 我們選了以下三個 model:

	Validation F1 Score	Public Score	Private Score	Computing Time per Round
Gradient Boosting	0.33518	0.38402	0.32751	10 s
Bagging	0.29254	0.28451	0.31858	0.25 s
Voting	0.32256	0.30762	0.32695	17.07 s

選取 Gradient Boosting 的原因為他是 Validation F1 Score mean 最高者, 此外也擁有很低的 Validation F1 Score variance, 代表模型足夠穩定。測出來的結果如預期中是我們所有 model 中分數最高的, Public Score 甚至比本機分數還高不少。而其餘兩個 model 分別為運算速度最快及預期最穩定的 model, 兩者也都有不錯表現。

其中比較特殊的點為我們原本預期 Gradient Boosting 的 Public 與 Private 的差距要最小, 最終結果卻是相反。而 Voting 就如預期般穩定不少, 而在時間、運算資源有限的情況下, Bagging 雖然 Private Score 低於其他 model, 但運算時間遠勝其他 model。

五、Division of Works

蔡沛勳
B07902037
	1. Code Implementation of
	  a. Train
	  b. Test
	2. Experiments designing
	3. Models' parameters fine-tuning
	4. Report writer

陳光裕 B07902072	1. Data preprocessing experiments - StandardScaler 2. Models' parameters fine-tuning 3. Experiments executing 4. Report writer

蘇聖祐
B07902128
	1. Code Implementation of data preprocess
	2. Data preprocessing experiments executing
	  a. MinMaxScaler
	  b. delete 5 columns to achieve highest F1 Score
	3. Report writer