Machine Learning – Assignment II

楊子萱
Institute of Data Science
National Cheng Kung University
Tainan, Taiwan
re6114056@gs.ncku.edu.tw

***Abstract*—利用 Car Insurance Claim Prediction 資料集去測試實作的 model 與 open source 的 model 在於準確性上的差別，並且透過 cross-validation 去驗證模型的穩健性，最終選出最適當的 model。從實驗的結果上得出，在於 f1-score 上最高皆可達成 0.91 的高分，在於不同演算法上沒有明顯差異。**

Keywords—machine learning, naive bayes, random forest, xgboost,

I. INTRODUCTION

利用 Car Insurance Claim Prediction 這個真實世界的資料集去測試實作的 naive bayes、random forest 與 sklearn 中 random forest，以及 open source 的 xgboost 在於準確性上的差別，並透過 Cross-validation 的方式去測試不同模型的穩健性，再設計融合不同 model 的結果去給定最終預測結果並比較與 Problem1 的差異，選出最適的 model。

II. DATA PREPROCESSING

A. *Train.csv*

本次的作業完全透過 trian.csv 做 train data、validation data 以及 test data 的拆分，is_claim 為 target feature，其餘的作為用於預測的 features。

扣除掉 target feature 總共有 43 個欄位，一開始先將 target feature 為缺失值的資料去除，再來將認為不相關的欄位去除，依序為 policy_id、max_torque、max_power、engine_type、turning_radius、length、width、height、gross_weigh。

數值型欄位做 Discretizer，因此將 policy_tenure、age_of_car、age_of_policyholder、population_density 轉化入三個區間中，值為[0,1,2]，Displacement 的部分是透過牌照稅的級距中的 1200，劃分為 1200 以上及 1200 以下，值為[0,1]。

類別型欄位因為是做 classifier，因此值的大小不影響分類，因此將該類別轉成數字去代表該類別。

III. METHOD OF PROBLEM1

A. *Naïve Bayer classifier*

單純貝氏分類器是假設特徵之間獨立下，運用貝氏定理為基礎的簡單機率分類器.

$$p(Y|x_1, x_2 \dots x_n) = \frac{p(Y)p(x_1,x_2 \dots x_n|Y)}{p(x_1,x_2 \dots x_n)} \quad (1)$$

透過上述的公式實作於 naive_bayes.py 中：

- 步驟一：計算出不同 class 發生的機率。
- 步驟二：計算出不同 feature 中該 feature value 發生的機率。
- 步驟三：計算出某一 class 時不同 feature 中該 feature value 發生的條件機率
- 步驟四：回傳所有透過 training set 得到的機率值
- 步驟五：用步驟四中的機率值預測 testing set 中各個 instance 為某一 class 的機率，並回傳機率值較大的類別

B. *Random forest*

- Tree 設計：

  使用 Cross-Entropy 公式去計算 Information Gain 以選出最好的分支，透過遞迴方式層層往下建構分類樹，直到僅屬於其中之一種類別或特徵僅剩下一種時，回傳該種類別或是數量較多的類別.

  $$Entropy(t) = -\sum p(j|t)\log(j|t) \quad (2)$$

  t為分支，j為 class，$Entropy(t)$為分支t中的亂度值，$p(j|t)$為分支t中 class 為j的機率

  $$GAIN_{split} = Entropy(p) - (\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)) \quad (3)$$

  p 為母分支，$i: 1-k$為分支p下的所有 feature value，n為分支p底下的數量，n_i為 feature value 等於 i 時的數量

  透過上述的方法實作於 tree.py

- Random forest 設計：

  步驟一：從所有的 feature 中 sample 出 $\frac{n}{2}$ 個特徵

  步驟二：從 training set 中 sample 出 $\frac{n}{10}$ 筆資料

  步驟三：將 sample 出來的資料與特徵傳入 tree 中建造出 forest，並回傳 forest

  步驟四：透過 forest 去預測資料並回傳每一棵樹的結果

  步驟五：將所有的結果做 bagging，得出最終最適結果並回傳

  上述的方法實作於 random_forest.py

C. *RandomForestClassifier by sklearn*

隨機森林是一種分類器，它在數據集的各個子樣本上擬合多個決策樹分類器，並使用平均來提高預測準確性和控制使得結果不會過度擬合某一顆決策數。

套用 sklearn 中已撰寫好的 RandomForestClassifier 套件做分類器，以此為對照組，比較自行撰寫的分類器效能.

網址：https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

D. *Xgboost*

XGboost 與 Random forest 相同都會採用特徵隨機採樣的技巧，使得在生成每一棵樹的時候隨機抽取特徵，因此在每棵樹的生成中並不會每一次都拿全部的特徵參與決策。此外為了讓模型過於複雜，XGboost 在目標函數添加了標準化。因為模型在訓練時為了擬合訓練資料

套用 open data 中已撰寫好的 XGBClassifier 套件做分類器，以此為對照組，比較自行撰寫的分類器效能.

網址：https://xgboost.readthedocs.io/en/stable/index.html

IV. EXPERIMENT

資料集中 is_claim 為 0.0 的有 20801 筆，為 1.0 的有 1427 筆，可以看出該資料集非常的 imbalance，因此較適用透過 Precision 與 Recall 組合計算而成的 F1-score。

	Naïve bayes	random_forest	sklearn random_forest	xgboost
F1-score	0.91	0.91	0.91	0.91

由結果中可以得出在此資料集上，不管是實作的 Naïve bayes、random_forest，還是開源的程式碼應用上，效能上都可達成 F1-score 為 0.91 的高分。

V. METHOD OF PROBLEM2

A. *Use k=3,5,10, and make some discussions of your observation*

	k = 3	k = 5	k = 10
Naïve bayes	0.90	0.901	0.902
random_forest	0.90	0.91	0.90
sklearn random_forest	0.90	0.901	0.902
xgboost	0.90	0.90	0.902

由上表的結果中可以看出，在此資料集的分類問題中上，當 k 等於 5 時的驗證效果最佳，皆為 0.91，且不論是實作的 model，或是開源的 model 再於測試上結果均可達成高穩健性。

B. *Please design an algorithm that can merge/aggregate the predicted results from k classifiers in k-fold cross-validation.*

- 步驟一：將 testing data 預測於 Problem1 中的 4 個分類器
- 步驟二：將各個分類的預測的結果存入 df 當中
- 步驟三：透過 bagging 的方式選出最佳的分類結果

	k = 3	k = 5	k = 10
Merge model	0.905	0.905	0.905

最終的預測 F1-score 為 0.905，與前面 Problem1 的預測結果分數差不多，最有可能的原因為前面的各個分類器預測結果皆已經足夠好，因此將結果做簡單的 bagging 對於整體的預測效果不會有顯著的提升。

複雜度的部分，由於前面皆為單一種分類器，random forest 的時間複雜度為$O(f * nlog(n))$，f為 feature 的數量，n為樣本數， naïve bayes 的時間複雜度為$O(c * f)$，c為 class 的數量，f為 feature 的數量，因此若只是單純的堆疊演算法，那其實時間複雜度則會直接的相加，但效果不見得顯著。

C. *How do we know the performance of one model is really better than another one?*

有多種方法可以去評估機器學習模型的性能，而最常見的方式是將數據分成訓練集以及測試集，然後在訓練集上訓練模型，並透過測試集去評估模型的性能。

而在於此題中是去評估 5-fold cross-validation 和 the result of Problem 1，在於 5-fold cross-validation 中 model 的最佳 f1-score 皆是 0.91，而 Problem1 的結果也皆是 0.91，因此在此處無法單看結果去評估模型的性能，可能需要做進一步的細化 criterion，或者是轉化前處理的方式，去觀察其中的差異。

VI. Conclusion

在於本次實作結果上，與現存的分類方法準確性部分並無明顯差異，推測可能的原因會在於前處理或資料本身的特性，由於沒有過多的去轉化資料本身的特徵，因此較多的特徵值，在於分類時數量雖然少但也能夠準確分入某一類別，因此整體 f1-score 皆達到很高的水準。

期望未來能夠再將此分類器，去實際使用於其他資料集中在去細部觀察其中的差異性，或者是優化模型增加更多可以調控的參數。

本次實作所遇到的困難：

Naïve Bayes：

1. 若是 testing data 出現 trianing data 該特徵中沒有出現過的特徵值，則無法得知該特徵值出現的機率，去做最終的預測。
2. 當資料集中特徵過多時，或者是特徵中特徵值數量過多時，計算出來的機率值很小，在最終加總起來的機率值會極度小，增加運算上的負擔。

Random Forest：

1. 當特徵中特徵值數量過多時，容易分散出很多分支，但多數分支在後續分類中資料集為空，因此容易出現錯誤，可能會需要在前處理時多合併數量較少的變項，或者說針對數量少於一定值時，做一個剪枝回傳的設定
2. Sampling Data 或 Feature 的方式不確定以何種 sample 方式最為適當，因此未來可能可以設計多種 sample 方式可供選擇。

GitHub 網址：https://github.com/tenyang1999/ML-assignment-2/