資料探勘 Project 2 賴廷瑋 F44054045

資料集簡介

1. Dataset 由 kaggle 提供，內容爲咖啡豆紀錄，總共有 44 個欄位，1319 個 row
2. 已將部分資料前處理，包含移除缺失值，LabelEncoding，異常值修改等。
3. 總共從資料集篩選 21 個特徵作爲分類器輸入，且已將 Categorical 型態資料皆利用 One hot encoding 轉換。
4. 自行設定的預測結果爲 Binary 0, 1，1 表示購買該咖啡豆、0 則否，Column 名稱爲 ”Buy”。
5. 後續將整理完的資料以 8:2 的比例隨機分成 training set 及 testing set

Rules 規則

共挑選六個特徵作爲 Rule 條件，第一層判斷若總杯測分數爲 4，則購買咖啡豆(buy =1)，若沒有滿足該條件，再接續下層條件判斷，條件滿足則分類爲 1(買)，若非則繼續判斷，以此類推，若皆沒有滿足則分類爲 0(不買)

第二層 其中一個條件滿足就分類爲 1(買)

第三層 皆要滿足條件才分類爲 1(買)

	總杯測分數 (Total.Cup.Points)	品種 (Variety)	前處理法 (Processing. Method)	水份 (Moisture)	瑕疵豆數 (Category.Two.Defects)	平均種植海拔 (altitude_mean_meters)
第一層	=4					
第二層		=Bourbon	=Pulped natural / honey			
第三層				<=2	=1	>3
第四層	=3					

分類器模型

決策樹

未調任何參數的模型表現：

1. Training Accuracy：

```
model = clf.fit(train_x, train_y.values)
print('training accuracy: {}'.format(model.score(train_x, train_y)))
```

```
training accuracy: 1.0
```

2. Test performance：

```
precision for buy :0.9494 | for no buy: 0.9739
recall for buy :0.9615 | for no buy: 0.9655
f1 score for buy :0.9554 | for no buy: 0.9697
```

決策樹 - 討論

1. 此決策樹分類器在分類過程中的 root node 並不是我設定的第一層條件

出乎意料的是，決策樹第一層是篩選該咖啡豆是否酸度(Acidity)爲 1，這個特徵不在我自行設定的 Rules 中，且原預想第一層篩選的特徵應爲我設定的總杯測分數(判斷是否爲 4)，**爲釐清概念，我計算了各特徵的加權 Gini index，確認 Acidity_1 的確爲最佳的 Root 分類條件**。

```
gini_list = []

for idx, col in enumerate(train_x.columns):
#     print(idx)
    col_gini = get_weighted_gini(col)
    gini_list.append(col_gini)
# print(gini_list)
best_idx = np.argmin(np.array(gini_list))
print('best split column with lowest gini index: ',tra
print('lowest weighted gini index: {:.3f}'.format(gini
```

```
best split column with lowest gini index:  Acidity_1
lowest weighted gini index: 0.327
```

圖 1 計算各特徵的 weighted gini index

最後發現，我設定的第一層條件並非決策樹 Root Node 的原因在於，儘管在我的設定中，總杯測分數(Total_cup_points) 爲 4 的話全部都會被歸類爲”Buy”，Gini Index 爲 0，**但是若總杯測分數不爲 4，則分類結果中 Buy (1) 和 No Buy (0) 的比例相近，Gini Index 非常高，造成總杯測分數=4 此特徵的 weighted gini index 較酸度(Acidity_1)的 weighted gini index 高**，所以並不會被決策樹選作爲 Root node。

2. 決策樹分類的條件中出現不是我設定的 rules

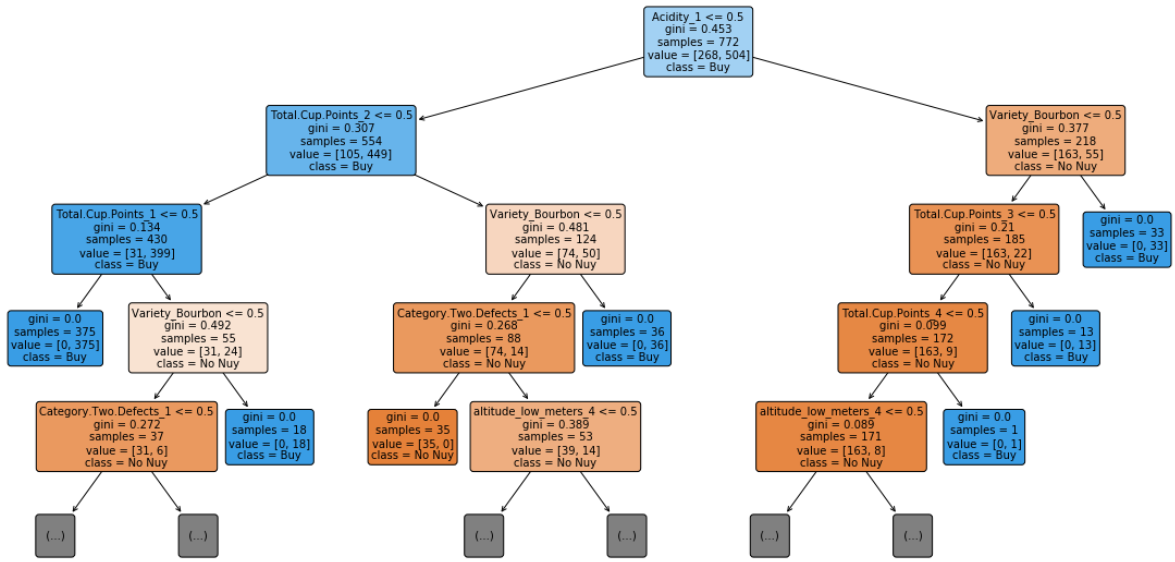

圖 2 未調參前決策樹分類規則

從決策樹分類條件中發現，除了 Root node 以外，在 Internal Node 的部分決策樹也會使用非 Rule 中內的特徵做爲 Split 條件，而經過分析發現，**若特徵之間的相關性過高，則決策樹很可能會因爲過高的相關性，而選擇並非在定義的規則(Rules)出現的特徵**。例如在我定義的規則裡，僅有出現 altitude_mean_meters (平均種植海拔) 的特徵，而決策樹在第四層就利 altitude_low_meters (最低種植海拔) 這個非規則中的特徵來分類。

3. 爲何某些 Test data 會被分類錯誤？

找出在測試資料集中，index = 3 的該筆資料的 Ground Truth 爲 0 (不買)，但分類器預測結果爲 1 (買)，所以針對該筆資料深入探討，發現以下結論

對於 unseen data，該分類器較容易預測錯誤

會得出這樣的結論，是發現依照該筆資料的特徵，在此決策樹的決策過程順序如下：

酸度不爲 1 → 杯測分數爲 2 → 品種不爲 Bourbon → two defects 爲 1 → 最低海拔爲 4 → 濕度(Moisture) 不爲 3 → 買

在此決策過程中，判斷 two defects，海拔高度，及濕度的確是自行設定的分類規則，規則中規定 two defects 需爲 1，且海拔高度爲 4，且濕度等於 1 或 2 的話，則該咖啡豆會被分類爲 1 (買)。但 test set 的這筆 data 的確不符合我們的設定規則(這筆 data 中濕度爲 4)，故 Ground Truth 爲 0(不買)，但決策樹分類器仍然分類爲 1(買)，原因在於，在 training data 中所有符合上述的決策樹分類規則的咖啡豆，濕度皆爲 1 或 2，所以會成功被分類爲 1(買)，但 test data 中出現了 training set 沒有的資料，該筆資料的濕度爲 4，所以決策樹判斷失準，照成分類錯誤。

簡單貝氏分類器(Bernouli NB classifier)

因爲有將所有特徵皆做 one hot encoding，參考 source code 規定後，決定以伯努力貝氏分類器作爲分類模型。

模型表現:

1. Training accuracy :

```
B_model = B_cls.fit(train_x, train_y)
print('Training Accuracy : {}'.format(B_model.score(train_x, train_y)))
```

```
Training Accuracy : 0.8989637305699482
```

2. Test accuracy:

```
ts = precision_recall_fscore_support(B_pred_y,
('precision for buy :{0[0][0]:.4f} | for no buy
('recall for buy :{0[1][0]:.4f} | for no buy: {
('f1 score for buy :{0[2][0]:.4f} | for no buy:
```

```
precision for buy :0.9494 | for no buy: 0.8870
recall for buy :0.8523 | for no buy: 0.9623
f1 score for buy :0.8982 | for no buy: 0.9231
```

簡單貝氏分類器 - 討論

爲何有些 Test data 會被錯誤分類？

同樣的找出在測試資料集中第一筆被分類錯誤的 data，找出 Index = 3 的該筆資料的 ground truth 爲 0 (不買)，但預測結果爲 1 (買)。細部探討結論如下：

*如果貝氏分類器的輸入有**多個高度相關的特徵**，且其中大多數的特徵皆沒有出現在設定的 rule 條件中，則分類器可能會因爲在貝氏定理的計算中，因爲連乘的關係造成 P(data|某 class)的機率過大或過小，進而造成分類錯誤。*

同樣的，爲了深入探討錯誤分類的原因，我計算了該筆資料的 P(data|class= 0) 及 P(data| class= 1)，以探討那些特徵爲造成分類錯誤的主要原因。針對 index =3 的這筆被錯誤分類的咖啡豆，作了以下計算：

```
for Number.of.Bags col, cond_prob = 0.30
for Bag.Weight col, cond_prob = 0.44
for Variety col, cond_prob = 0.21
for Processing.Method col, cond_prob = 0.72
for Aroma col, cond_prob = 0.59
for Flavor col, cond_prob = 0.03
for Aftertaste col, cond_prob = 0.47
for Acidity col, cond_prob = 0.02
for Body col, cond_prob = 0.05
for Balance col, cond_prob = 0.04
for Uniformity col, cond_prob = 1.00
for Clean.Cup col, cond_prob = 1.00
for Sweetness col, cond_prob = 1.00
for Total.Cup.Points col, cond_prob = 0.47
for Moisture col, cond_prob = 0.30
for Category.One.Defects col, cond_prob = 1.00
for Quakers col, cond_prob = 1.00
for Category.Two.Defects col, cond_prob = 0.46
for altitude_low_meters col, cond_prob = 0.21
for altitude_high_meters col, cond_prob = 0.28
for altitude_mean_meters col, cond_prob = 0.26

given no buy prod: 2.4105098987189464e-12
```

圖 3 在不買的情況下，該筆資料發生的機率

```
for Number.of.Bags col, cond_prob = 0.40
for Bag.Weight col, cond_prob = 0.22
for Variety col, cond_prob = 0.26
for Processing.Method col, cond_prob = 0.72
for Aroma col, cond_prob = 0.14
for Flavor col, cond_prob = 0.34
for Aftertaste col, cond_prob = 0.11
for Acidity col, cond_prob = 0.36
for Body col, cond_prob = 0.28
for Balance col, cond_prob = 0.34
for Uniformity col, cond_prob = 1.00
for Clean.Cup col, cond_prob = 1.00
for Sweetness col, cond_prob = 1.00
for Total.Cup.Points col, cond_prob = 0.12
for Moisture col, cond_prob = 0.21
for Category.One.Defects col, cond_prob = 1.00
for Quakers col, cond_prob = 1.00
for Category.Two.Defects col, cond_prob = 0.61
for altitude_low_meters col, cond_prob = 0.24
for altitude_high_meters col, cond_prob = 0.26
for altitude_mean_meters col, cond_prob = 0.26

given buy prod: 4.5830142837180764e-10
```

圖 4 在買的情況下，該筆資料發生的機率

針對該筆被錯誤分類的資料進行計算後，發現模型預測該筆爲 1(買)的主要原因在於，在兩個不同 class 情況下，Flavor, Acidity, Body, Balance 這四個特徵出現的機率相乘相差非常多倍，但驚訝是，這四個特徵皆不是規則設定中的特徵，但依照決策樹的結果來看，酸度 Acidity 可能恰巧與我設定的 ground truth 有很強的鑑別度，但是卻因爲其他三個特徵與酸度這個特徵的相關性太高，造成整體條件機率過小，而導致模型錯誤預測爲 1(買)。

隨機森林

參數:：僅將參數手動改爲 n_estimators = 200

模型表現：

```
print('rf training accu: {}'.format(rf.score(train_x, train_y)))
```

```
rf training accu: 1.0
```

```
results = precision_recall_fscore_support(rf_pred_y, test_y)
print('precision for buy :{0[0][0]:.4f} | for no buy: {0[0][1]:.4f}
print('recall for buy :{0[1][0]:.4f} | for no buy: {0[1][1]:.4f}'.fo
print('f1 score for buy :{0[2][0]:.4f} | for no buy: {0[2][1]:.4f}'
```

```
precision for buy :0.9873 | for no buy: 0.9739
recall for buy :0.9630 | for no buy: 0.9912
f1 score for buy :0.9750 | for no buy: 0.9825
```

隨機森林模型 討論

可以發現增加隨機森林中的決策樹數量至 200 後測試集表現有明顯地比單一決策樹模型還好，這部分的原因是原資料集都將以取出不放回的方式隨機抽樣 N 筆當作 input 隨機給森林中的單一決策樹，由於資料可能會被重複抽取，所以每顆樹都在給定更少的資訊下訓練，而最後將 200 個樹 ensemble 後利用多數決決定最後預測結果，這個方式也驗證了能有效解決 overfitting 的問題。

有趣的是，找出前 10 個隨機森林地 feature importance 最高的特徵後，也將前十個單一決策樹劃出其結構後發現，在泛化的過程中，隨機森林雖有將設定的規則中的 Variety = Burbon 這個特徵作爲條件，但是規則中許多特徵(如濕度, category_two_defects, altitudes) 等皆未出現在前十名的重要特中，而 10 個單一決策樹的 root node 也都在這前 10 名的 feature，並非我設定的第一層篩選條件。最後我的結論是，可能其他特徵(如酸度, body 等)，剛好在我設定的 rule 下，更好的將 target 分類，所以決策樹和隨機森林分類的方式其實與 ground truth 會有一段落差，主要決定於 rules 的設定方式和資料的分布。

```
ft_importance[:10]
```

```
[('Variety_Bourbon', 0.1158),
 ('Total.Cup.Points_3', 0.0741),
 ('Total.Cup.Points_2', 0.0698),
 ('Acidity_1', 0.0616),
 ('Flavor_1', 0.0478),
 ('Total.Cup.Points_4', 0.0466),
 ('Balance_1', 0.0436),
 ('Total.Cup.Points_1', 0.0392),
 ('Aroma_1', 0.0341),
 ('Body_1', 0.0309)]
```