Data Mining HW2 《P76111262林晨鈞》

Classification Problem:Who is good teacher?

Feature Information:

Age: 25-65 #年齡

Gender: M,F #性別

Height: 140-190 #身高

Weight: 40-100 #體重

Attitude: Serious,Free #教學態度(嚴肅/自由)

Prepare\_hours: 0-6(hours) #備課時間(單位:小時)

Class\_type: Online,Physical #授課方式(線上/實體)

Pass\_rate: Low,High #班上學生Pass課程比例(高/低)

Glasses: 0,1 #是否戴眼鏡

Classhw: Low,Middle,High #課堂作業(低/中/高)

Care: Yes,No #是否會關心學生(是/否)

Label: good,bad #好老師/壞老師

Absolute right rules:

Attitude: Serious #教學態度嚴肅

Prepare\_hours: 3-6 #備課時間達3-6小時

Pass\_rate: High #學生通過課程比例高

Care: Yes #會關心學生

Class\_type: Physical #實體授課

資料集說明:

此分類問題為”判斷該老師是否為一位好老師”，此資料集總共有6000筆資料，12個不同的features，其中”好老師”的資料數占3107筆，”壞老師”的資料數占2893筆。Absolute right rules也列在上面，只要完全符合上述5點，即為”好老師”。(此資料僅為想像，並無惡意詆毀之意)

生成資料檔案:data\_produce.py

資料集:teacher\_data.csv



一張含有 桌 的圖片

自動產生的描述

1. Decision Tree (Train\_set:0.75,Test\_set:0.25)

**Comparison :**

當樹的最大深度設為5時，可以發現無論是訓練集還是測試集的預測結果都是1.0，可以看到重要特徵都有被挑出來，且都是Absolutely right rules，分別是:

1. Attitude:0.28852774
2. Prepare\_hours:0.04443353
3. Class\_type:0.36432398
4. Pass\_rate:0.19989458
5. Care:0.10282016

PS: heatmap為測試集的預測結果

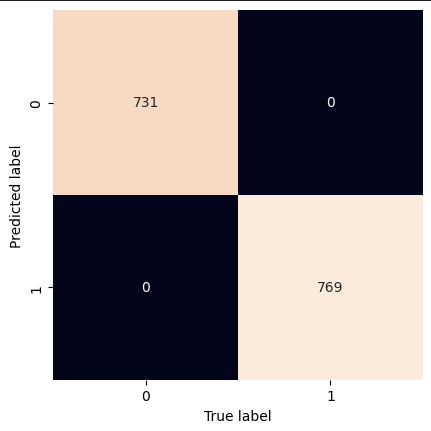
一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

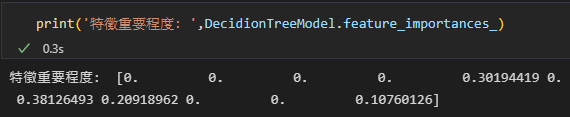
一張含有 文字, 標誌, 螢幕擷取畫面 的圖片

自動產生的描述 

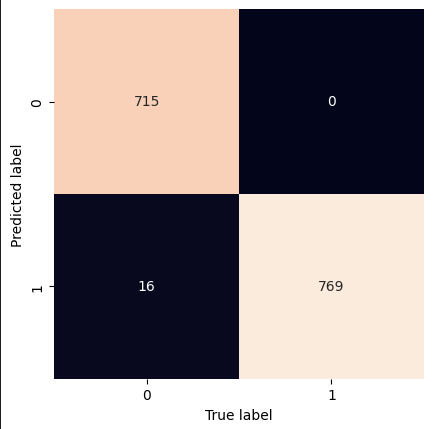
當樹的最大深度設為4時，發現訓練集的預測準確率為0.989333，測試集的預測準確率為0.988666，其中重要的特徵Prepare\_hours被省略了，但準確率還是不錯。

一張含有 文字 的圖片

自動產生的描述



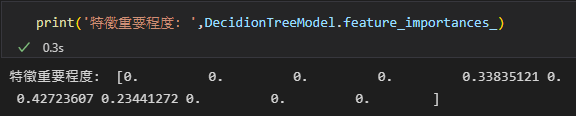
一張含有 文字, 標誌, 螢幕擷取畫面 的圖片

自動產生的描述 

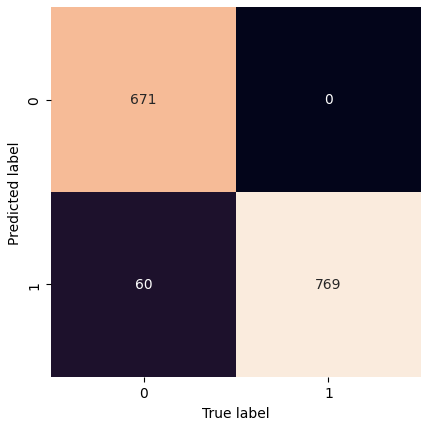
當樹的最大深度設為3時，發現訓練集的預測準確率為0.96，測試集的預測準確率為0.960444，其中重要的特徵Prepare\_hours, Care被省略了。

一張含有 文字 的圖片

自動產生的描述



一張含有 文字, 標誌, 螢幕擷取畫面 的圖片

自動產生的描述 

1. KNN

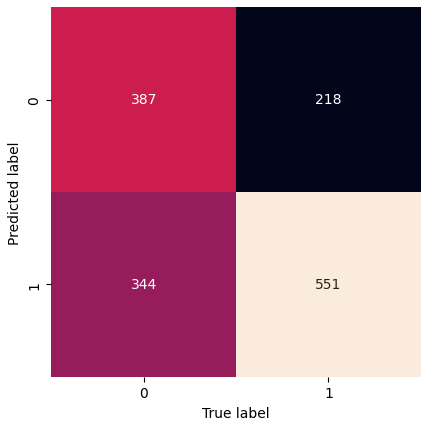
**Comparison :**

KNN相較於Decision Tree，預測的準確率偏低很多，將n\_neighbors設定為1會得到以下結果，訓練集的準確率為1.0，但測試集的準確率為0.625333

PS: heatmap為測試集的預測結果

一張含有 文字 的圖片

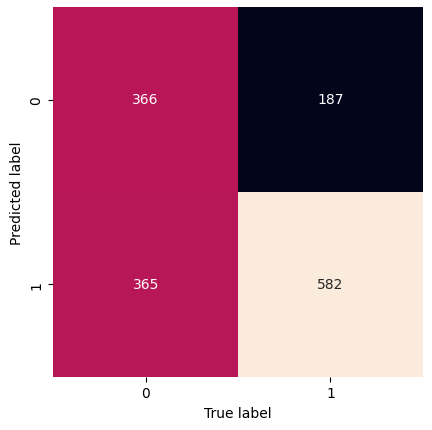
自動產生的描述



n\_neighbors設定為3會得到以下結果，訓練集的準確率為0.826，但測試集的準確率為0.632

一張含有 文字 的圖片

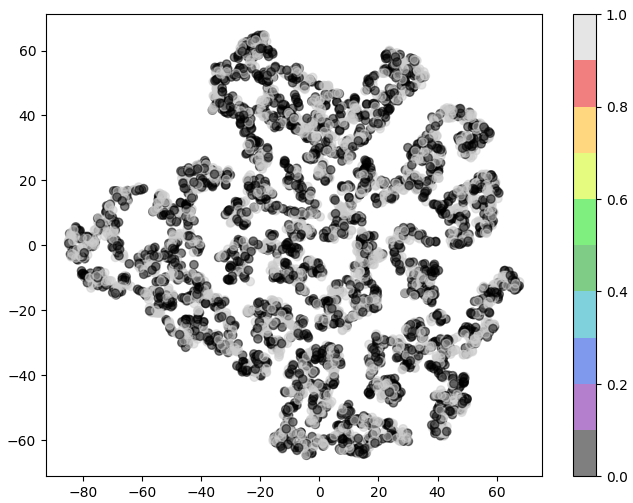
自動產生的描述



因為上述結果不盡理想，所以嘗試進行降維，我採用TSNE降維，並降成2維，以下第二張圖為降維後的分布圖，最後n\_neighbors設定為2，訓練集的準確率為0.80，但測試集的準確率為0.48，效果仍不盡理想，甚至更差。

一張含有 文字 的圖片

自動產生的描述



一張含有 文字 的圖片

自動產生的描述

1. NaiveBayes

使用Gaussian Naïve Bayes所得到的結果分別是:

訓練集準確率:0.9886666

測試集準確率:0.9893333

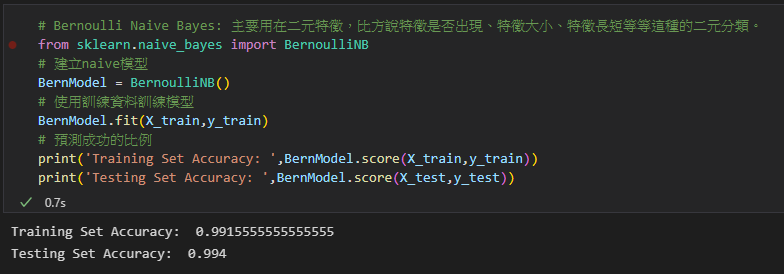
一張含有 文字 的圖片

自動產生的描述

使用Bernoulli Naïve Bayes所得到的結果分別是:

訓練集準確率:0.991555555555

測試集準確率:0.994



使用Multinomial Naïve Bayes所得到的結果分別是:

訓練集準確率:0.902666666666

測試集準確率:0.907333333333

一張含有 文字 的圖片

自動產生的描述

結論:

可看出Bernoulli Naïve Bayes所得到的結果較好，因為它主要適用於二元特徵，比如說特徵是否出現等等，跟我設計的資料集相符度甚高。

1. Random Forest

訓練集準確率:1.0 測試集準確率:1.0

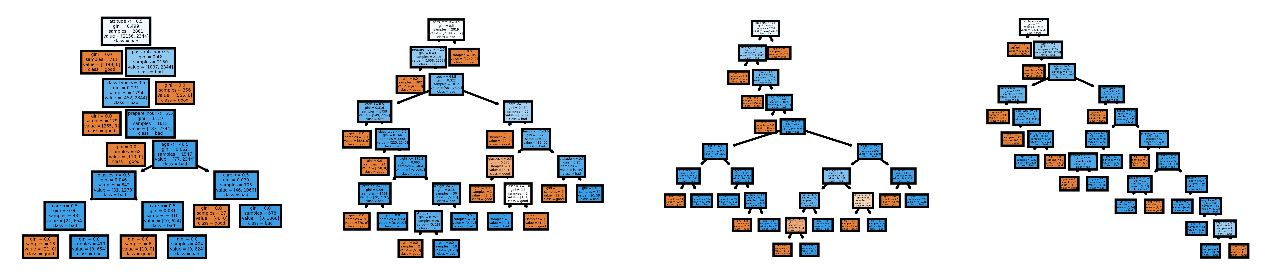
使用Random Forest可以發現，每個特徵都被列為參考對象，只是重要程度的區別，最重要的特徵是Class\_type:0.2495，再來我們的Absolute right rules也都有被篩選出來，依重要程度依序為:Attitude:0.21212888、Pass\_rate:0.21175923、Care:0.17424444、Prepare\_hours:0.13904489，最不重要的特徵是glasses:0.00058，最後一張圖呈現前4重要的樹。

一張含有 文字, 監視器, 螢幕, 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述



1. XGBoost

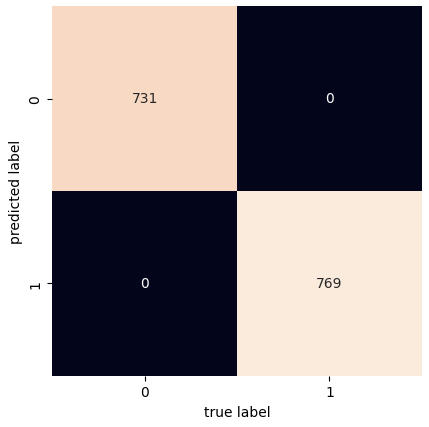
可看出Boosting方法跑出來的結果相當好，且觀察第二張圖，Absolute right rules都有被篩選出來，分別是陣列中第4,5,6,7,10個特徵，依序為: Attitude, Prepare\_hours, Class\_type, Pass\_rate,Care

一張含有 文字 的圖片

自動產生的描述

一張含有 桌 的圖片

自動產生的描述



Discussion:

Slightly alter the absolutely-right rules and generate another set of data; run the classifcation models on this set of data and include your observations in this section.

**原本的Absolutely-right rules:**

Attitude: Serious #教學態度嚴肅

Prepare\_hours: 3-6 #備課時間達3-6小時

Pass\_rate: High #學生通過課程比例高

Care: Yes #會關心學生

Class\_type: Physical #實體授課

**微改後的Absolutely-right rules:**

Attitude: Serious #教學態度嚴肅

Prepare\_hours: 2-6 #備課時間達2-6小時

Pass\_rate: High #學生通過課程比例高

Care: Yes #會關心學生

Class\_hw:Middle/High #回家作業比例(中/多)

Class\_type: Physical #實體授課

共1000筆資料，其中”好老師”資料佔513筆，”壞老師”資料佔487筆

生成資料檔案:data\_produce\_alter.py

資料集:teacher\_data\_alter.csv



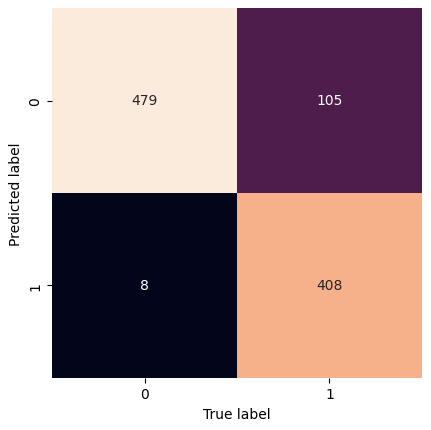
一張含有 桌 的圖片

自動產生的描述

1. Decision Tree

準確率:0.887

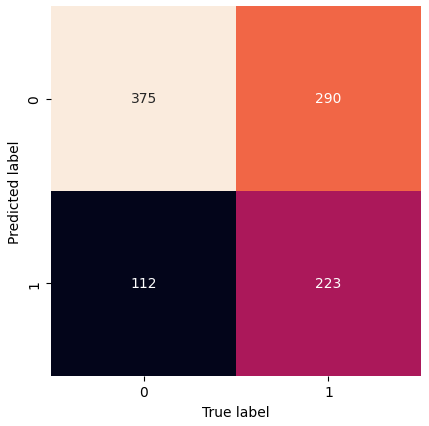




1. KNN

準確率:0.598

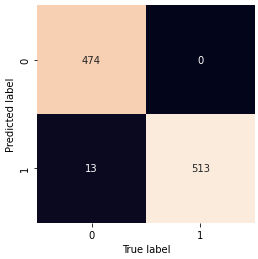




1. Naïve Bayes (Bernoulli Naïve Bayes)

準確率:0.987

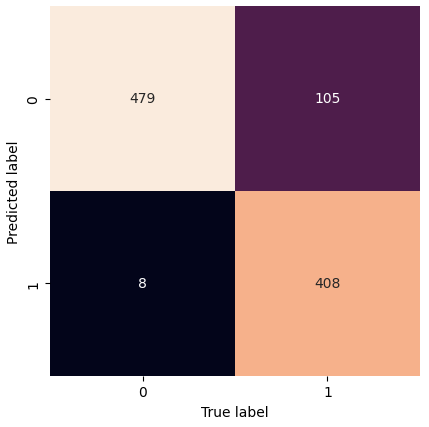




1. Random Forest

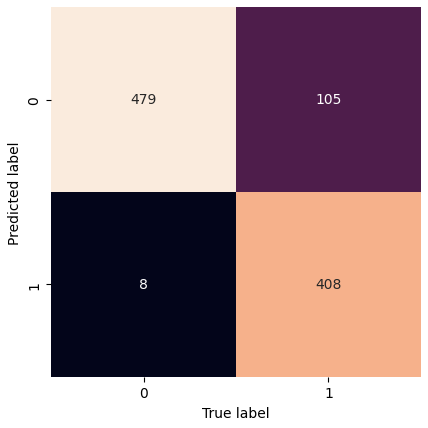
準確率:0.887





1. XGBoost

準確率:0.887

結論:

我微調了Absolutely-right rules並生成對應資料集，再各自丟到了不同的classify model做分類，原本預期XGBoost會是5個裡面最好的，【因為 Bagging 透過隨機抽樣的方式生成每一棵樹，每棵樹彼此獨立並無關聯。隨機森林就是 Bagging 的實例。而 Boosting 則是透過序列的方式生成樹，後面所生成的樹會與前一棵樹相關。XGBoost 就是 Boosting 方法的其中一種實例。正是每棵樹的生成都改善了上一棵樹學習不好的地方，因此 Boosting 的模型通常會比 Bagging 還來的精準。】，但是可以看到Decision Tree、Random Forest、XGBoost這3個分類模型的準確率和heatmap跑出來的結果都一模一樣，我個人覺得他們學到的特徵和最後建出來的樹應該都是相差不遠的，造成此結果的原因可能是我本身設計的資料集的feature就不夠複雜，大部分影響結果的重要的feature都只有2-3類，所以即便我微調了Absolutely-right rules，結果也是一樣的。另外，不難看出KNN因為微調了Absolutely-right rules，所以資料被混得更亂了，導致準確率很低。最讓我意外的是Naïve Bayes準確率高達98%，應該是運氣好，為此，我另外設計了一個Absolutely-right rules(這邊沒有呈現出來)，Naïve Bayes的準確率就沒有比其他來的好。