

Classification rule

學生: 電機系大四 楊子右

學號: E24066250

## 1. 資料集生成

本次作業我打算以”判斷學生是否通過審查”的簡單二元分類來做為資料內容。我列出了 25 種特徵，也就是 25 門課程。亂數生成 1~7 的數字代表學生在該門課的表現程度，數字越大越好。

## 2. Absolute rule

在 Rule 選擇上面，我規劃了單純的**門檻限制**(程度必須大於特定數字)、**線性分類**(兩門課程的加總平均必須大於特定數字)以及**非線性分類**(Standard deviation 必須小於特定數字以篩選出在特定領域表現比較好的學生)。

```
#feature_list with 25 features
feature_list = ['al', 'la', 'os', 'ds', 'sw', 'dm', 'ca', 'st', 'tc', 'fi', 'ci',
                'en', 'em', 'fs', 'ch', 'ce', 'jp', 'aj', 'bs', 'ck', 'dv', 'ec', 'fj', 'ge', 'hy']

def generate_data(fea_list): #Generate the data with 25 attrs
    tmp = []
    ans_list = []
    for i in range(5000): #student data
        te = np.random.randint(1, 7, 25)
        tmp.append(list(te))
    stud_data = pd.DataFrame(np.row_stack(tmp), columns = feature_list)

    for ind in stud_data.index:
        if (stud_data.iloc[ind]['ds'] >= 4 and
            stud_data.iloc[ind]['la'] >= 4 and
            stud_data.iloc[ind]['fs'] >= 2 and
            stud_data.iloc[ind]['ch'] >= 1 and
            stat.stdev(stud_data.iloc[ind])<=1.6 and
            (stud_data.iloc[ind]['ds'] + stud_data.iloc[ind]['la'])/2 >= 4.5 and
            (stud_data.iloc[ind]['la'] > stud_data.iloc[ind]['dm']) and
            (np.mean(stud_data.iloc[ind]))>=3.5 ):
            ans_list.append(1)
        else:
            ans_list.append(0)
    return stud_data, ans_list
```

圖一:隨機生成 5000 位學生學習程度資料

- 門檻限制: DS, LA, FS, CH 必須分別大於(4, 4, 2, 1)。
- 線性分類:DS 與 LA 的平均必須大於 4.5，LA 表現需比 DM 好且總表現必須大於 3.5。
- 非線性:25 個 1~7 的數字分布變異數必須小於 1.6。

### 3. 分類器選擇

考量到資料限制的不同性質，我採用不同方式的分類器以觀察在不同情境下，是否會對分類準確率造成影響。實作方式，我皆採用 Sklearn 套件底下之 Function 進行套用實作。

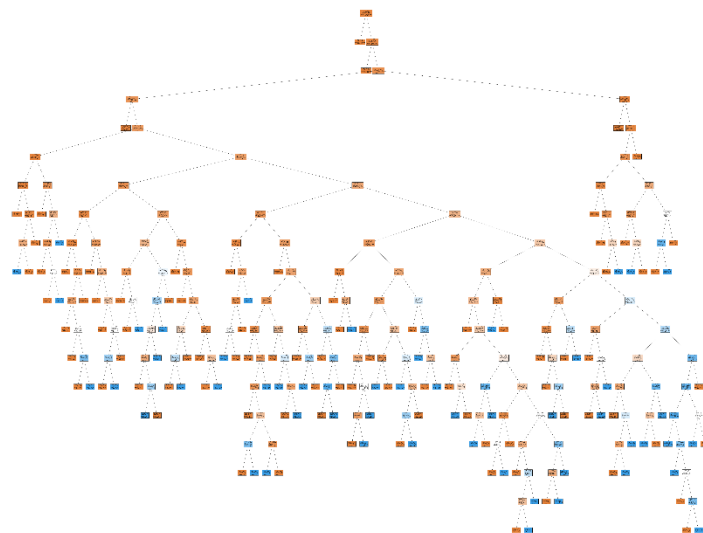
#### ● Decision Tree

```
def run_dec(tr_x, tr_y, te_x, te_y): #does not seperate the useless fea
    clf = DecisionTreeClassifier()
    gen_clf = clf.fit(tr_x, tr_y)
    text_representation = tree.export_text(gen_clf)
    print(text_representation)

    fig = plt.figure(figsize=(125,100))
    _ = tree.plot_tree(clf,
                       feature_names=feature_list,
                       class_names = ['P', 'F'],
                       filled=True)
    fig.savefig("decistion_tree.png")

    test_y_predicted = gen_clf.predict(te_x)
    accuracy = metrics.accuracy_score(te_y, test_y_predicted)
    print("dec_tree_Acc : ", float(accuracy*100), " % ")
```

圖二:決策樹選用



圖三:決策樹長相

```

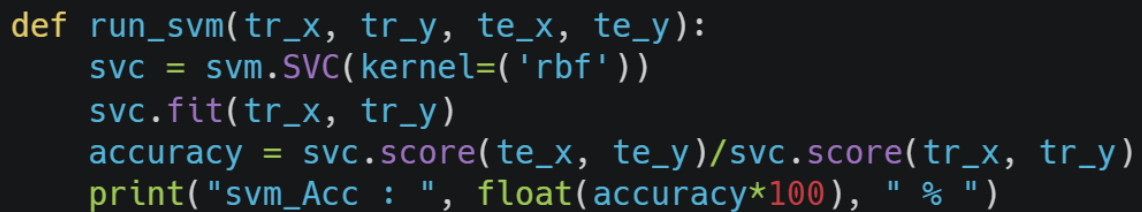
|--- feature_1 <= 4.50
|   |--- feature_1 <= 3.50
|   |   |--- class: 0
|   |--- feature_1 > 3.50
|   |   |--- feature_3 <= 4.50
|   |   |   |--- class: 0
|   |   |--- feature_3 > 4.50
|   |   |   |--- feature_5 <= 3.50
|   |   |   |   |--- feature_17 <= 2.50
|   |   |   |   |   |--- feature_19 <= 5.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_19 > 5.50
|   |   |   |   |   |   |--- feature_9 <= 5.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- feature_9 > 5.50
|   |   |   |   |   |   |   |--- feature_10 <= 3.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- feature_10 > 3.00
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- feature_17 > 2.50
|   |   |   |   |   |   |   |   |   |--- feature_6 <= 2.50
|   |   |   |   |   |   |   |   |   |--- feature_14 <= 5.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- feature_14 > 5.50
|   |   |   |   |   |   |   |   |   |   |--- feature_10 <= 5.50
|   |   |   |   |   |   |   |   |   |   |--- feature_4 <= 5.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |--- feature_4 > 5.50
|   |   |   |   |   |   |   |   |   |   |   |--- feature_24 <= 3.50
|   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |   |--- feature_24 > 3.50
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1

```

決策樹部分規則節錄

- **Support Vector Machine – Support Vector Classification**

因為資料集中具有非線性規則，我認為線性分類器極有可能無法辨別出該限制。因此我使用 SVC，希望其能透過 Kernel function 將資料映射至高維空間，藉此能夠更全面的了解資料分布情況。SVC 之 Kernel function 我選擇採用 rbf，其餘選擇預設值。

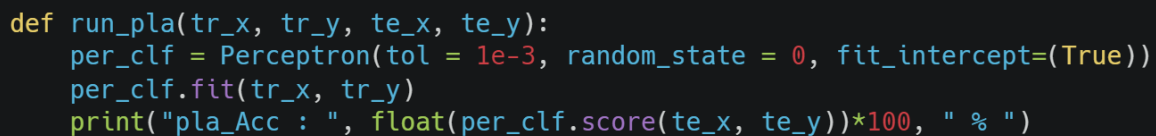


```
def run_svm(tr_x, tr_y, te_x, te_y):  
    svc = svm.SVC(kernel='rbf')  
    svc.fit(tr_x, tr_y)  
    accuracy = svc.score(te_x, te_y)/svc.score(tr_x, tr_y)  
    print("svm_Acc : ", float(accuracy*100), " % ")
```

圖三:SVC 選用

- **Perceptron Learning Algorithm**

因為資料集為線性可分，我認為十分符合 PLA 的使用情境因此選用其來進行測試。在參數方面，因為學生間彼此沒有關連性，所以 random\_state 設為零，意思是不對資料進行隨機處理。fit\_intercept 設為 True，因為資料在訓練前並未做過任何中心化處理。



```
def run_pla(tr_x, tr_y, te_x, te_y):  
    per_clf = Perceptron(tol = 1e-3, random_state = 0, fit_intercept=True)  
    per_clf.fit(tr_x, tr_y)  
    print("pla_Acc : ", float(per_clf.score(te_x, te_y))*100, " % ")
```

圖四:PLA 選用

- **Random Forest**

因為資料集中只有少數幾項特徵(科目)在規則內，許多特徵對整體規則影響力不如特定特徵。因此我選擇基於 Bagging 概念的 Random forest method，希望藉由此方法，可以更好提煉出最符合實際規則之分類方法。參數方面，n\_estimators 設為 100 顆樹，max\_depth 設為 5，因為特徵量不少，我希望讓子樹縱深提高，最後 random\_state 設為 0。

```
def run_ran(tr_x, tr_y, te_x, te_y):
    ran_clf = RandomForestClassifier(n_estimators = 100, max_depth=(4))
    ran_clf.fit(tr_x, tr_y)
    print_decision_rules(ran_clf)
    pred_y = ran_clf.predict(te_x)
    accuracy = metrics.accuracy_score(te_y, pred_y)
    print("ran_tree_Acc : ", float(accuracy*100), " % ")
```

圖五:RF 選用

#### 4. 分類結果

以 10000 筆學生資料並且重複生成 5 次後，可以發現分類器的表現都有 90% 以上的優秀準確度。

分類器名稱	準確率			
Decision tree	95.86	96.5	96.5	95.7
Random forest	98	97.93	97.83	97.36
SVC	99.8	100	97.7	99.5
PLA	98.03	97.8	97.8	93.5

表一:準確率

#### 5. 實作後心得想法:

從上表可以得知，Decision tree 的表現相較與其他分類器的表現略遜一籌。我認為這跟其特性很有關係，他只能在特徵值上進行選擇，但 Absolute rule 中有許多是需要多個特徵間的交互關係才能產生(Hidden rule)。因此藉由 Decision tree 所建立的 rule 沒辦法很好反映出實際的 Absolute rule。再者，我實作多次 Decision tree 並觀察每次其實際模型，發現到每次的決策都具有有一些差異，例如特徵”jp”對於分類的幫助不大，但在樹的結構中卻不停出現，代表 Decision tree 沒辦法很好顯示出有非線性關係資料集的 Absolute rule。

而隨機森林(Random forest)，下稱 RF。RF 為實現 Bagging 概念的一種分類方法，由許多弱分類器來投票決定最後結果。我認為 RF 表現不差的原因在於，資料集的 Absolute rule 只與特定幾個特徵有相關性，在選擇弱分類器的時候可以將表現不好的分類器或特徵忽略，這樣會更符合實際的 rule。此概念與 Machine learning 中的 feature selection and extraction。

而支持向量機(Support Vector Machine)，下稱 SVM。SVM 的表現是四種分類器中表現最好的，由於其有非線性轉換的特性，我們無法圖像化並了解其分類過程。不過從結果可以推論 SVM 有了分別非線性特徵的功能後，表現的確勝過其他分類器。

6. 心得:

在實作完這些分類器後，我發現到一直以來對於 Machine learning 的想法有些偏差。過於追求準確率的提升而忽視一些基本概念。即使能夠達成不錯的準確率，但如果對於資料與結果之間的關聯性描述有錯，那模型的穩定性便會受到懷疑。總的來說，分類器不具有大金剛法則，每個都是很好工具但是要先對資料有充分的了解後再進行使用，如此以來才能夠準確地描述行為。

詳細 Code 內容: [Zxiro/NCKU\\_CSIE\\_Data\\_Mining\\_Class\\_rule \(github.com\)](https://github.com/Zxiro/NCKU_CSIE_Data_Mining_Class_rule)