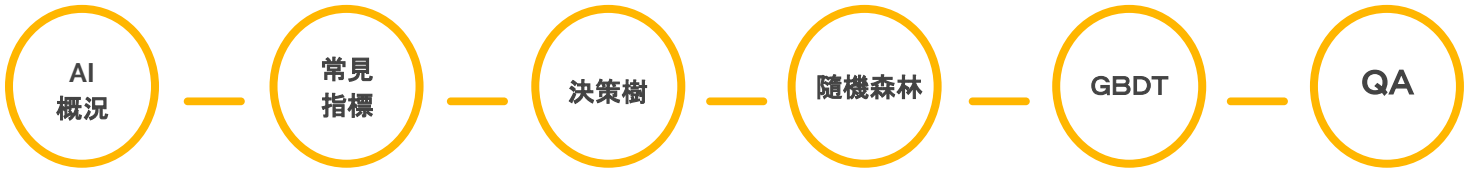
機器學習 - NTUDAC

James Yeh

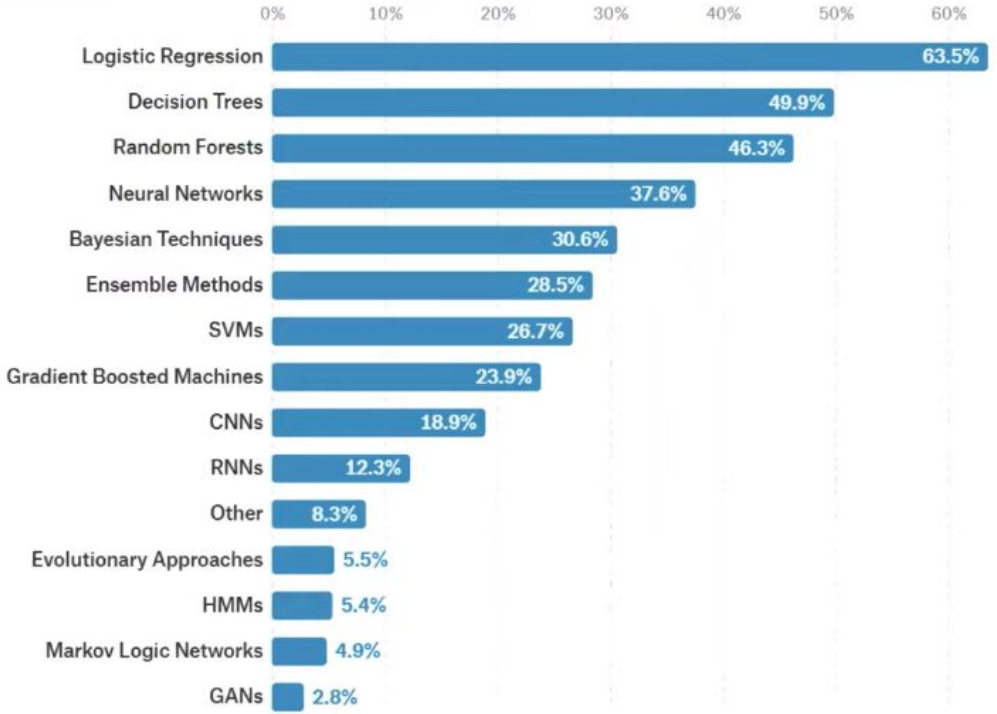
今天我們會



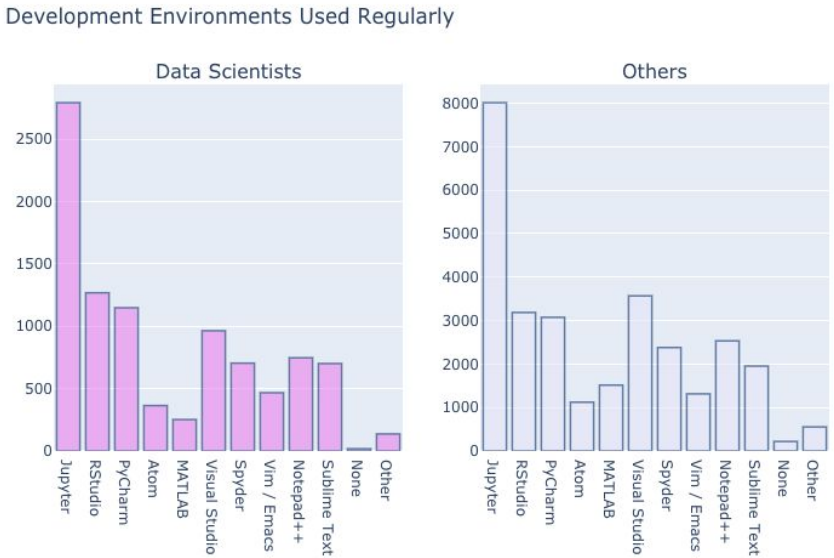
安裝套件(看決策樹長相)

pip install graphviz

工作中常見的模型(Kaggle統計)



Kaggle統計 人工智慧概況



https://www.kaggle.com/kaggle/kaggle-survey-2017/kernels
https://www.kaggle.com/kaggle/kaggle-survey-2018/kernels
https://www.kaggle.com/c/kaggle-survey-2019/notebooks

常見指標

- Accuracy
- Confusion Matric
- Precision
- Recall
- F1
- PR curve
- ROC curve
- RMSE

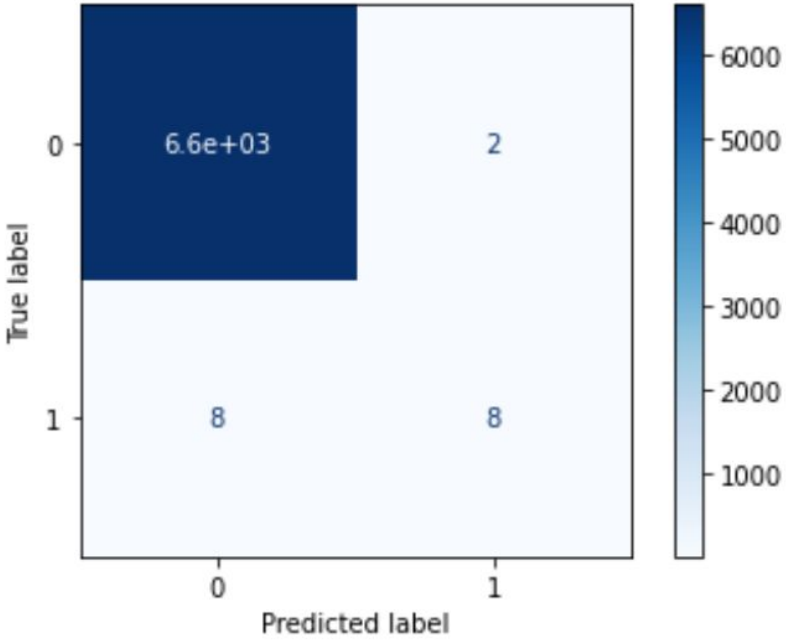
每種指標都只能片面的反應模型的部分性能

Accuracy

$$Accuracy = \frac{Ncorrect}{Ntotal}$$

Confusion matrix(混淆矩陣)

[6599, 2]
[8, 8]



Accuracy的問題

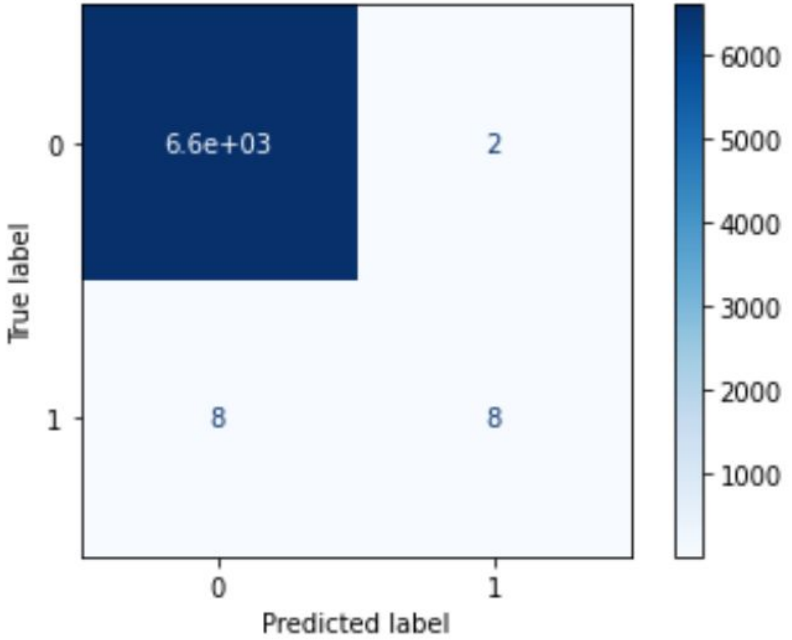
[6599, 2]
[8, 8]

Accuracy
= 6607/6617 = 0.998
= 99.8%

準確度(Accuracy)

```
lr.score(X_test, y_test)
```

```
0.9984887411213541
```



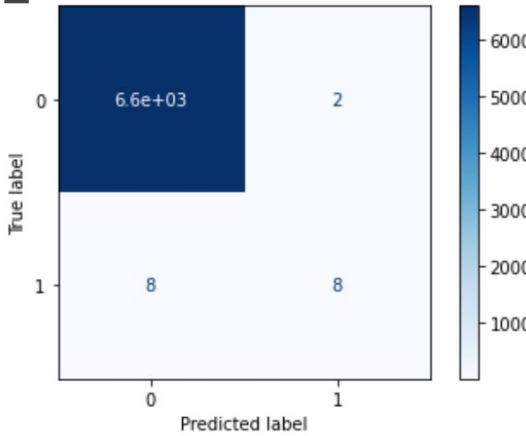
True Negative(TN) , 預測為0 實際為0

False Positive(FP) 預測為1 實際為0

False Negative(FN), 預測為0 實際為1

True Postive(TP) 預測為1 實際為1

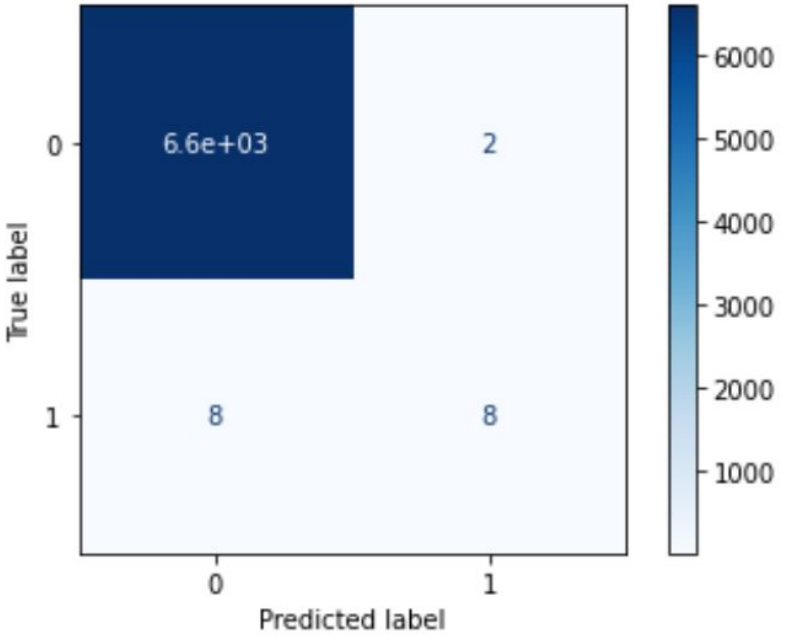
前面是形容詞 後面是名詞



Precision & Recall

$$Precision = \frac{TP}{TP + FP}$$

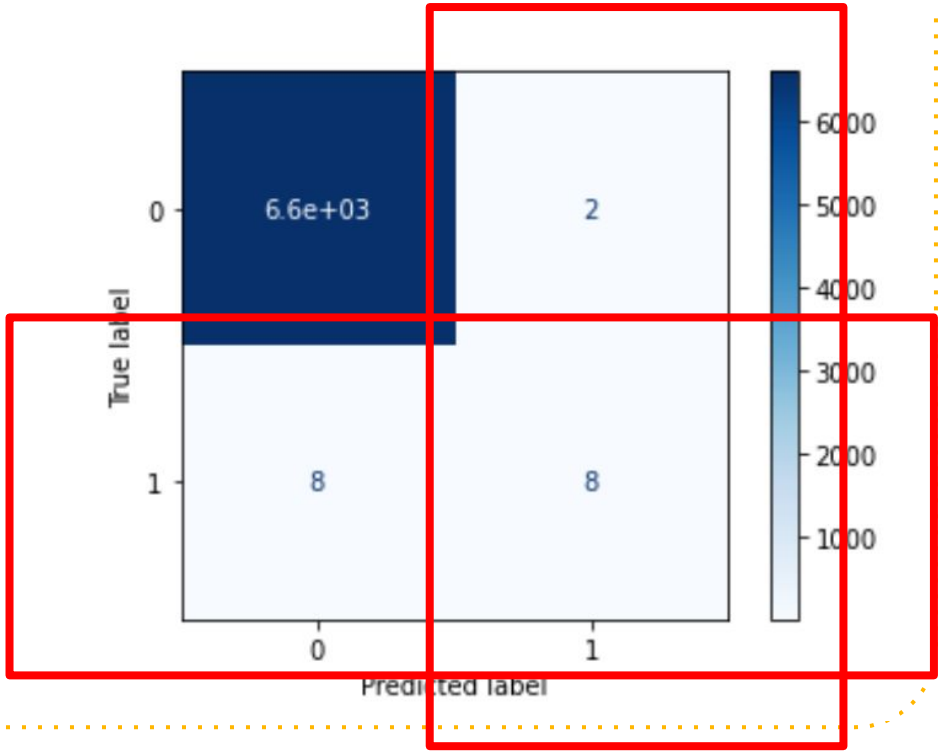
$$Recall = \frac{TP}{TP + FN}$$



Precision & Recall

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

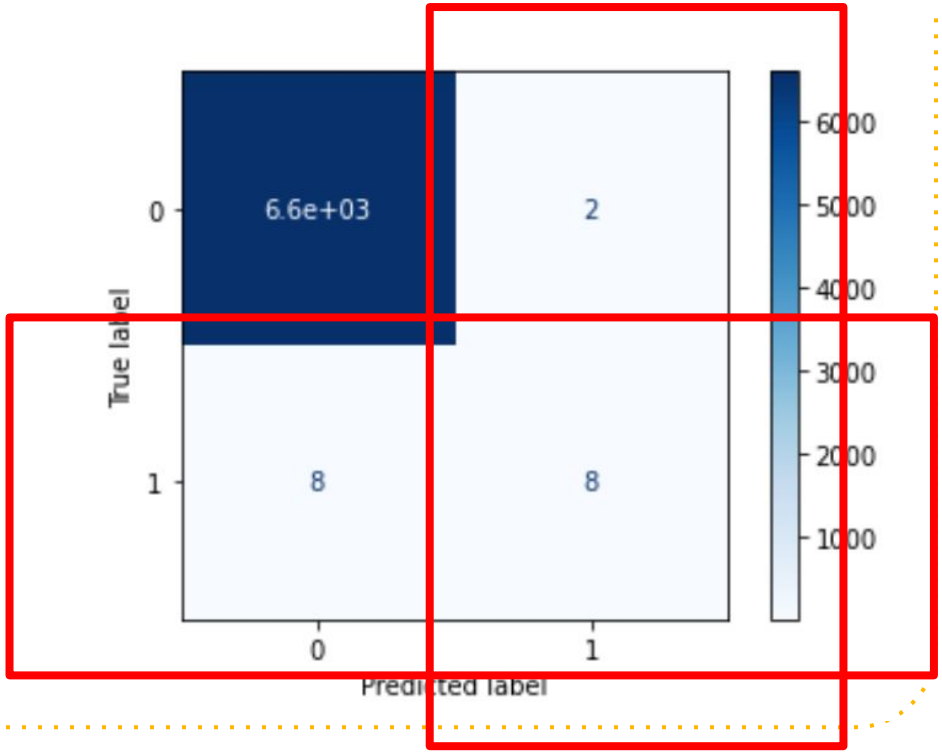


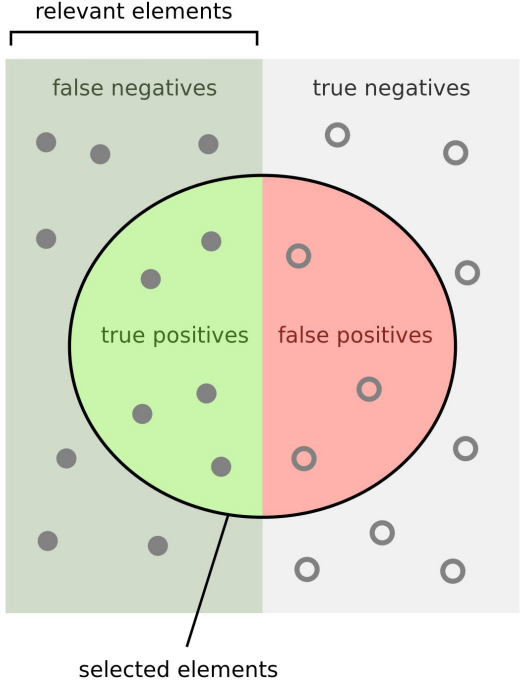
Precision & Recall

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

precision = 8/(8+2) = 0.8
recall = 8/(8+8) = 0.5





How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

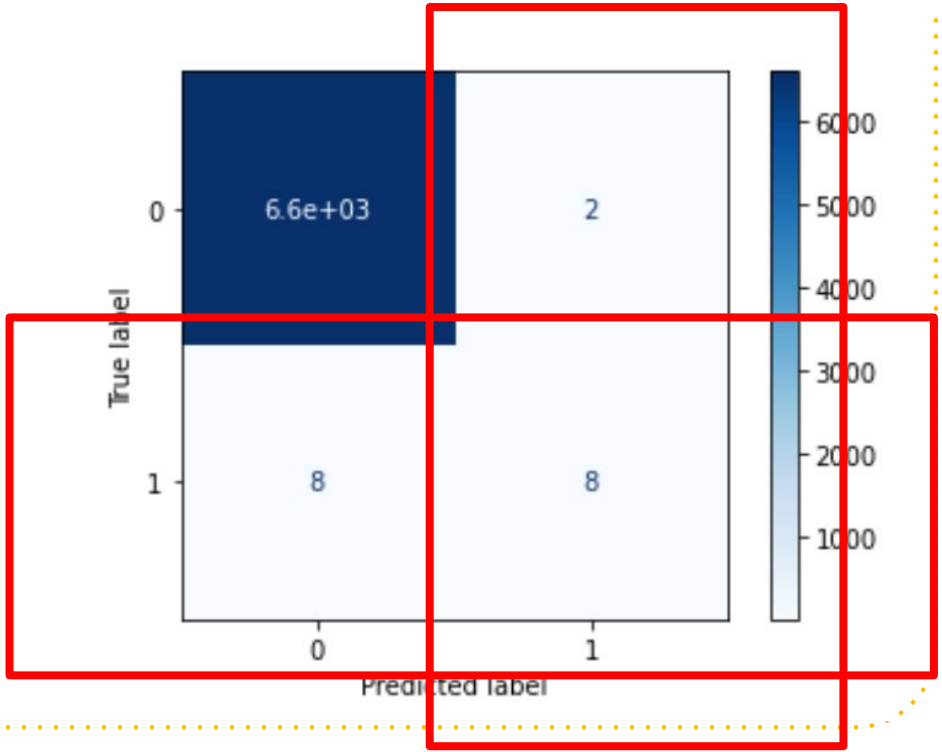
Precision & Recall

$$Precision = \frac{TP}{TP + FP}$$

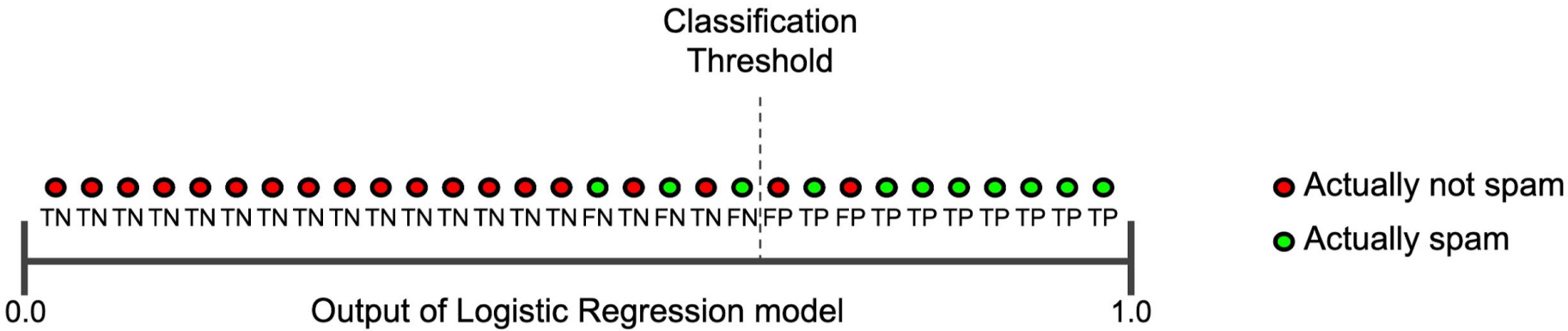
$$Recall = \frac{TP}{TP + FN}$$

precision = 8/(8+2) = 0.8
recall = 8/(8+8) = 0.5

想要提高precision, 越有把握的才預測為正樣本
想要提高recall, 只要有一點可疑就預測為正樣本



Precision & Recall 互為矛盾



Precision & Recall 實際問題

某商品搜索引擎返回的top 5 precision 非常高，但實際使用過程中，用戶常常找不到想到的結果，特別是一些冷門的商品，可能是哪個環節出了問題呢？

Precision & Recall 實際應用

搜索引擎 or **推薦系統**: TOP K precision & recall

表示法: precision@k & recall@k

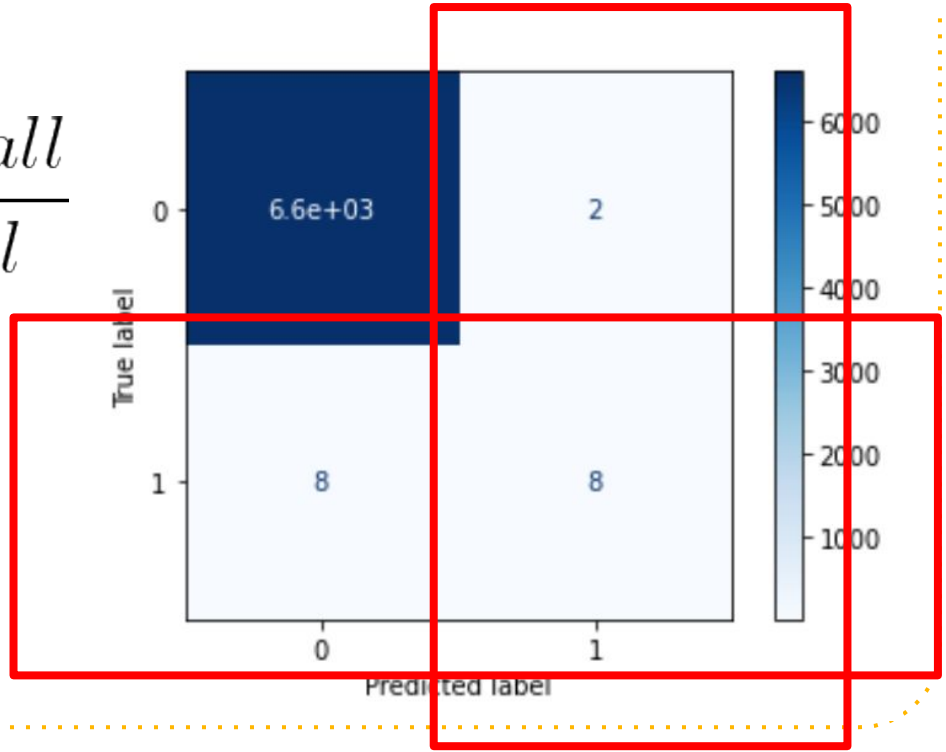
實際上要看不同TOP K**的**Precision Recall(K**越大** precision**越小** Recall**越大**）**以及**(PR**曲線**, F1, ROC)

F1 Score(綜合考慮precision+Recall)

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

precision = 8/(8+2) = 0.8
recall = 8/(8+8) = 0.5

F1 = (2*0.8*0.5) / 1.3 = 0.615



為什麼不直接Precision Recall 平均就好？

F1 結果範圍0~1之間
假設p和r一個是1.0一個是0.1,算術平均會接近0.5 而調和平均接近0.2,這說明調和平均會強調兩者的一致性，明顯不一致時傾向於小的值，這更符合人們的直觀感受。

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

precision = 8/(8+2) = 0.8
recall = 8/(8+8) = 0.5

F1 = (2*0.8*0.5) / 1.3 = 0.615

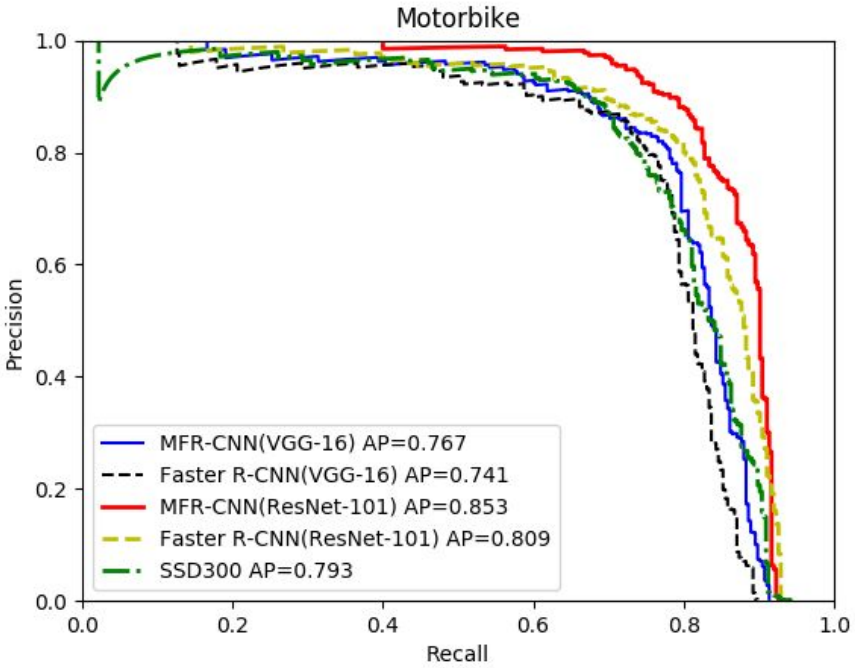
F1 score特性(希望平均都高)

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

兩個case:

1. precision =1 recall = 0.1 -> F1=0.18
2. precision =0.6 recall = 0.5 -> F1=0.55

PR 曲線



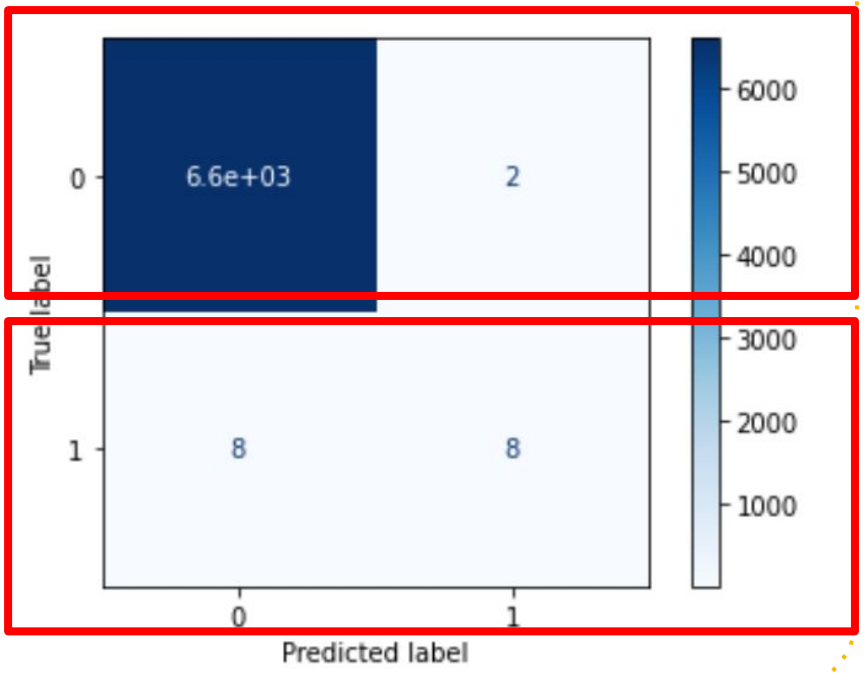
Threshold: 0~1之間

樣本1 predict probability:0.1
樣本2 predict probability:0.5
樣本3 predict probability:0.8

ROC(AUC) 曲線

$$FalsePositiveRate(FPR) = \frac{FP}{N}$$

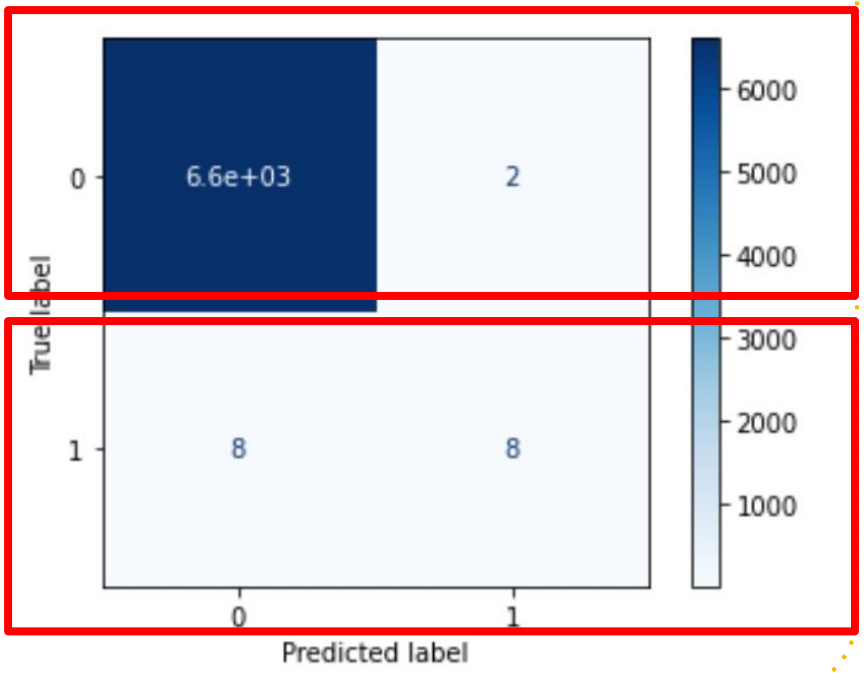
$$TruePositiveRate(TPR) = \frac{TP}{P}$$



ROC(AUC) 曲線

[6599, 2]
[8, 8]

Threshold = 0.5
FPR = 8/6599 = 0.001
TPR = 8/16 = 0.5



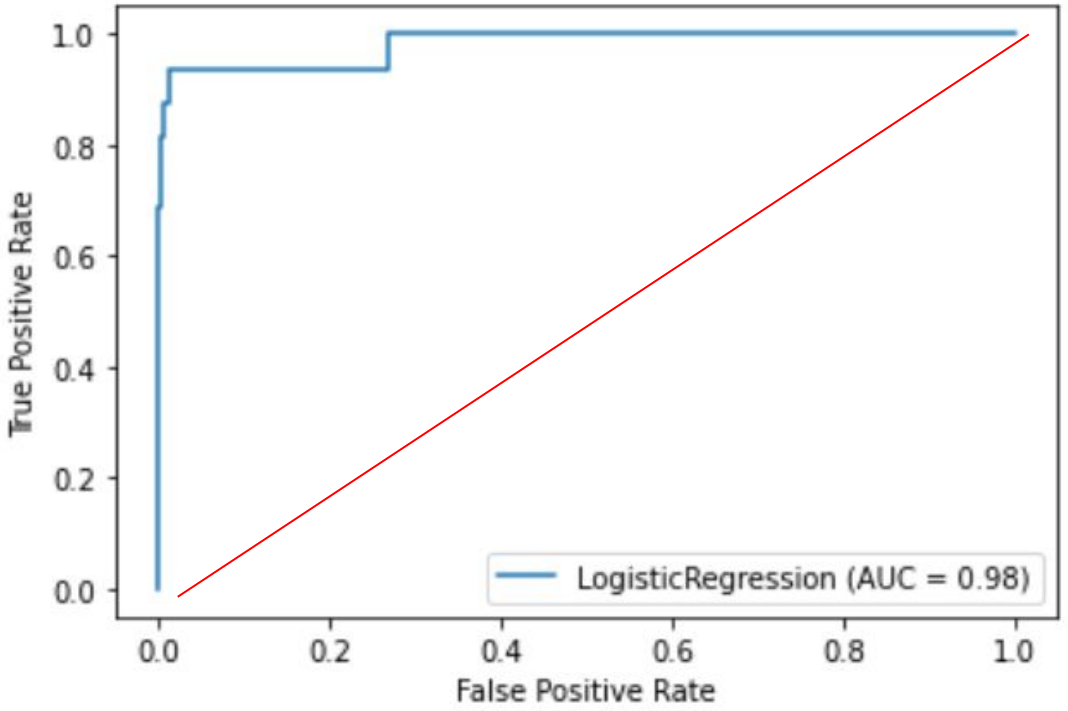
ROC(AUC) 曲線

```
fpr, tpr, thresholds = roc_curve(y_test, lr.predict_proba(X_test)[:,1], pos_label=1)
np.around(fpr,decimals=3)
np.around(tpr,decimals=3)
np.around(thresholds,decimals=3)
```

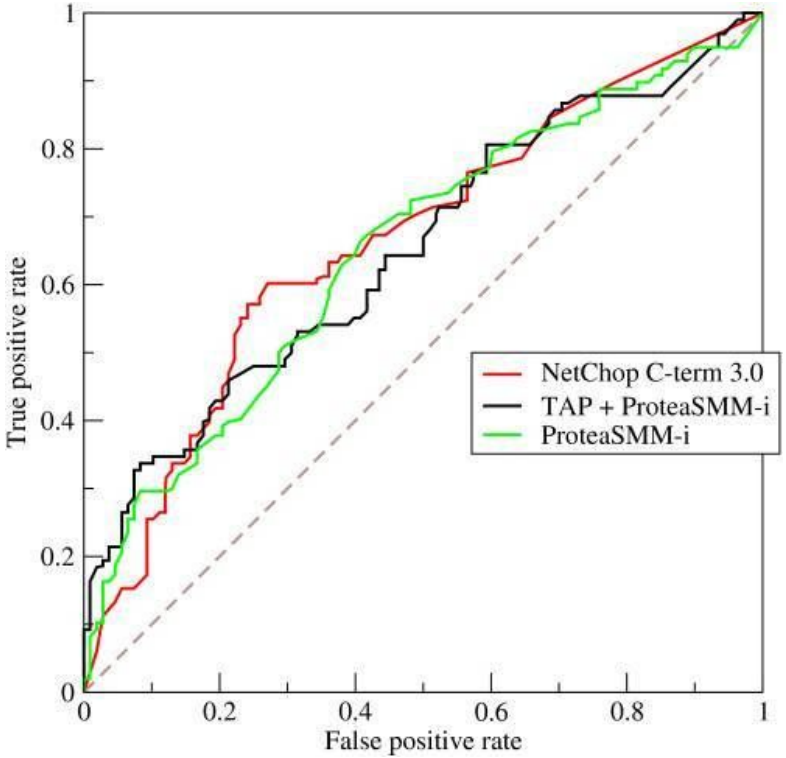
```
array([0.   , 0.   , 0.   , 0.   , 0.   , 0.001, 0.001, 0.002, 0.002,
       0.003, 0.003, 0.006, 0.006, 0.012, 0.012, 0.268, 0.268, 1.   ])

array([0.   , 0.062, 0.5  , 0.5  , 0.625, 0.625, 0.688, 0.688, 0.75 ,
       0.75 , 0.812, 0.812, 0.875, 0.875, 0.938, 0.938, 1.   , 1.   ])

array([1.998, 0.998, 0.611, 0.504, 0.313, 0.212, 0.183, 0.106, 0.098,
       0.065, 0.06 , 0.03 , 0.03 , 0.012, 0.012, 0.   , 0.   , 0.   ])
```

ROC(AUC) 曲線

ROC值在0.5~1之間
（越大越好）



ROC(AUC) 曲線


True positive rate
False positive rate
0
0.2
0.4
0.6
0.8
1
NetChop C-term 3.0
TAP + ProteaSMM-i
ProteaSMM-i

多元分類指標

概念跟二元分類一樣,
改成當下這個類別vs 其他類別做比較

```
: print(classification_report(lr.predict(X_test), y_test))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6607
1	0.50	0.80	0.62	10
accuracy			1.00	6617
macro avg	0.75	0.90	0.81	6617
weighted avg	1.00	1.00	1.00	6617

回歸類模型指標

MAE（mean absolute error）

$$MAE = \frac{1}{n}\sum_{k=1}^{n}|(actual_1 - predicted_1)| + \cdots + |(actual_n - predicted_n)|$$

RMSE（root mean squared error）

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n}(actual_1 - predicted_1)^2 + \cdots + (actual_n - predicted_n)^2}{n}}$$

回歸類模型指標

MAE（mean absolute error）

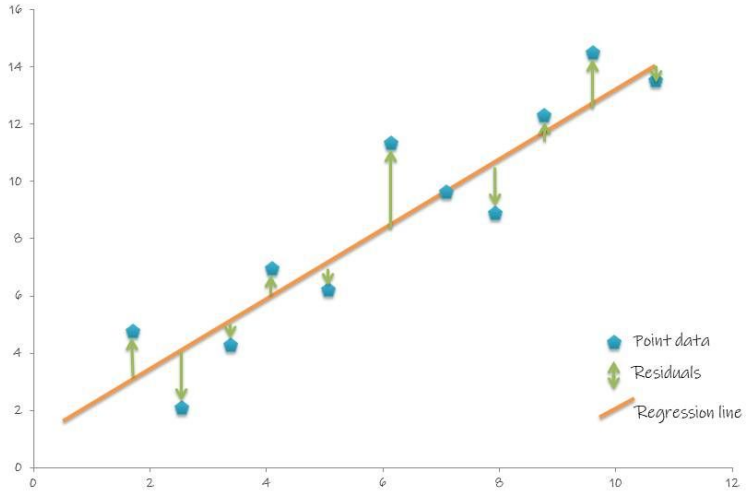
$$MAE = \frac{1}{n}\sum_{k=1}^{n}|(actual_1 - predicted_1)| + \cdots + |(actual_n - predicted_n)|$$

RMSE（root mean squared error）

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n}(actual_1 - predicted_1)^2 + \cdots + (actual_n - predicted_n)^2}{n}}$$

有少數差異很大的outlier會造成RMSE變大，誤差會被平方放大

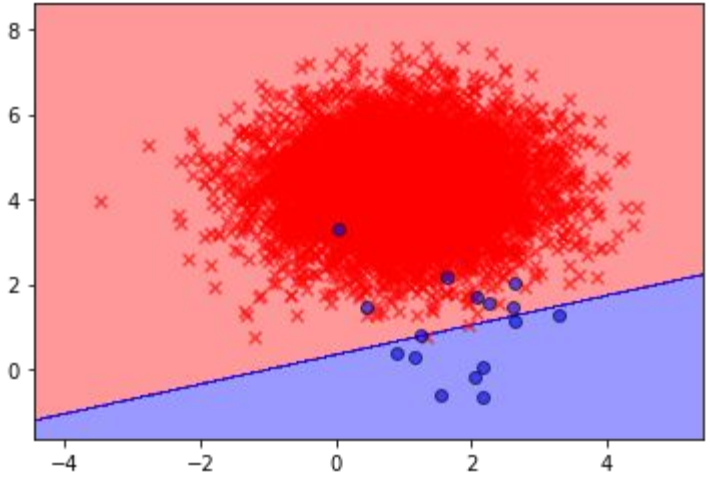
回歸類模型指標


16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12
Point data
Residuals
Regression line

樹模型

可解釋性的最佳模型

解釋性



模型的決策邊界: -100 + 6*溫度 + 3*濕度 = 0

-100 + 6*溫度 + 3*濕度 > 0 預測是一個美味的披薩
-100 + 6*溫度 + 3*濕度 < 0 預測是一個難吃的披薩

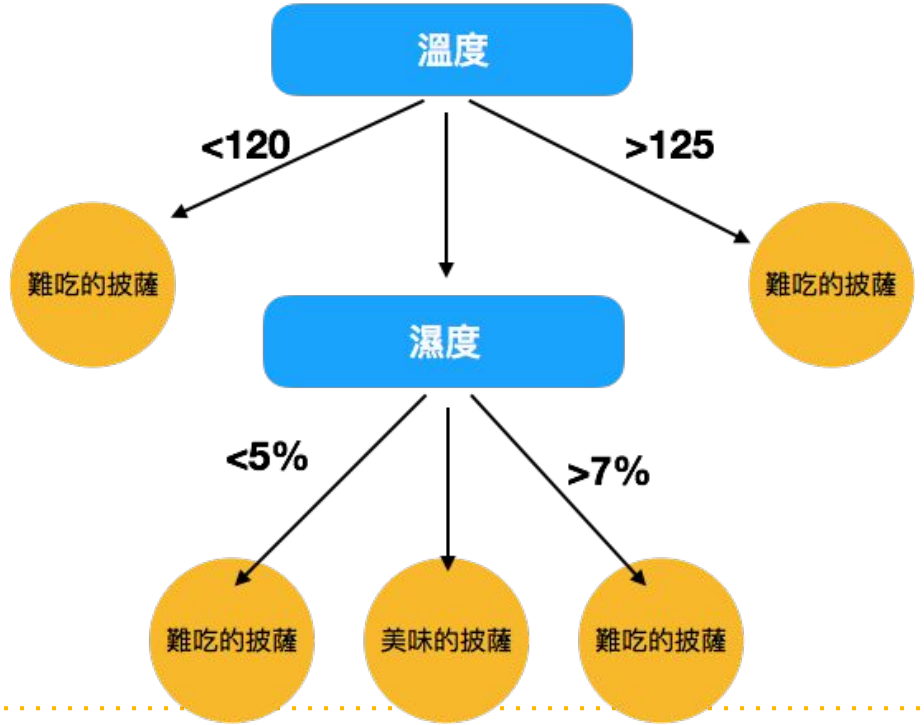
解釋性

這只是一個最簡單的模型，實際上的模型會複雜超級多，像是：

$$-100 + 6*溫度^2 + 3*濕度^3 + 20*溫度*濕度^2 + -70*溫度*濕度^2*氣壓 + \ldots.$$

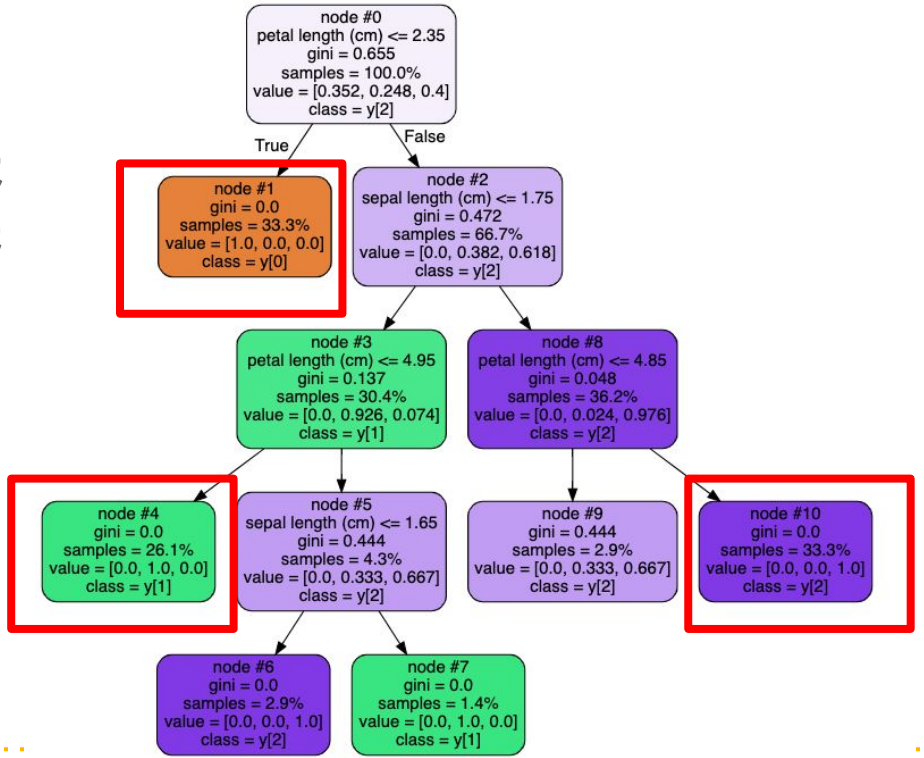
解釋性

決策樹(抽象概念)



決策樹(實際長相)

- 分支通常是2
- 根據Gini, Entropy(結果基本一樣)來做分類問題的分支決策



決策樹切割原理

資訊增益

獲得的資訊量　原本的資訊量　經由分割後的資訊量

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

二元分類資訊增益

獲得的資訊量　原本的資訊量　分割後左邊資訊量　分割後右邊資訊量

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

細節請參考: [資料分析&機器學習] 第3.5講：決策樹(Decision Tree)以及隨機森林(Random Forest)介紹

決策樹切割原理(Entropy)

Entropy(熵) 熱力學第二定律 熵增原理:在孤立系統中，體系與環境沒有能量交換，體系總是自發的向混亂度增大的方向變化，使整個系統的 熵值越來越大。
簡單來說就是:描述亂度

Source:
https://www.zhihu.com/question/24053383

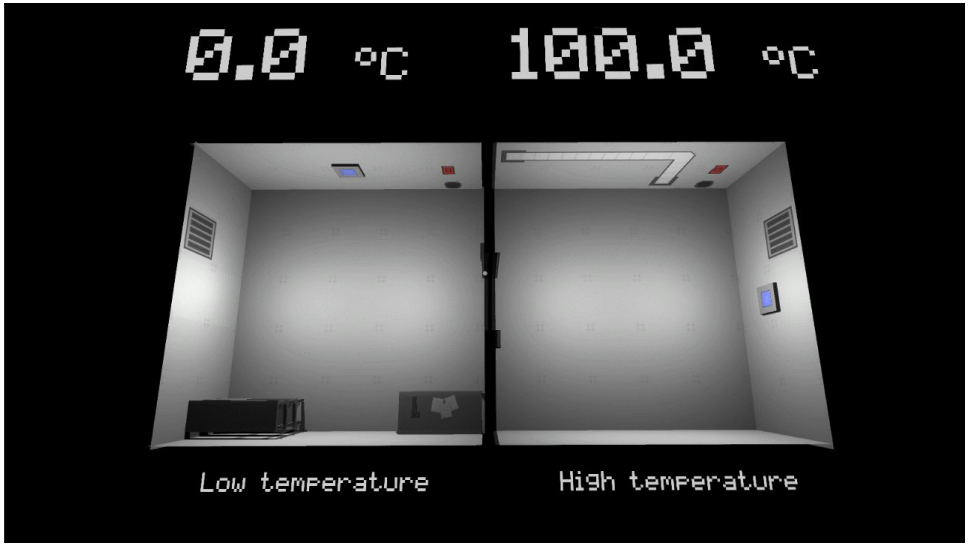
決策樹切割原理(Entropy)

房間越來越亂是熱力學第二定律之下的必然結果

Source:
https://www.zhihu.com/question/24053383

決策樹切割原理(Entropy)

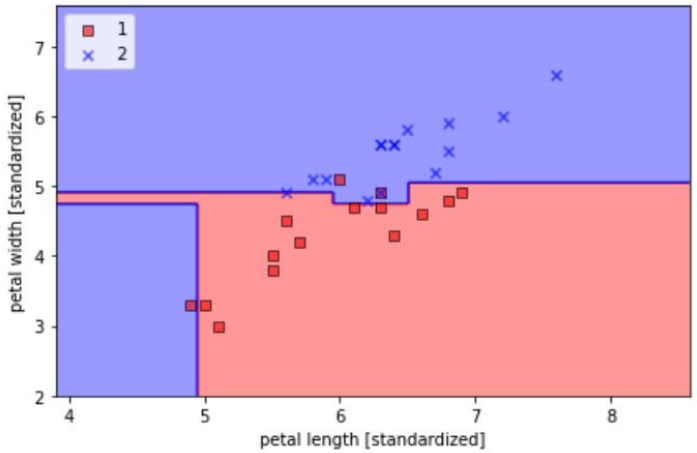
兩個房間溫度0, 100 Entropy最小
兩個房間溫度一樣Entropy最大(越自然的狀態熵越大)

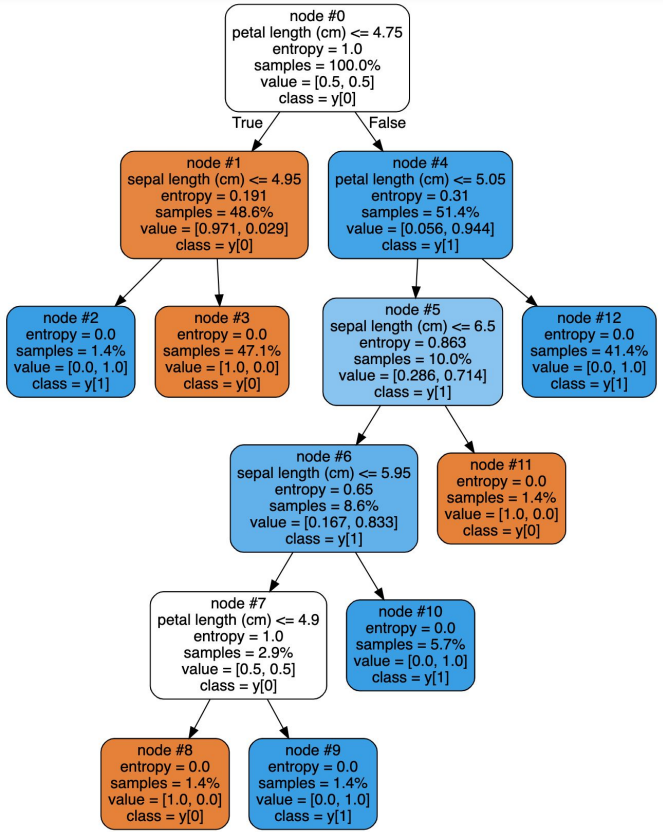


Source:
https://www.zhihu.com/question/24053383

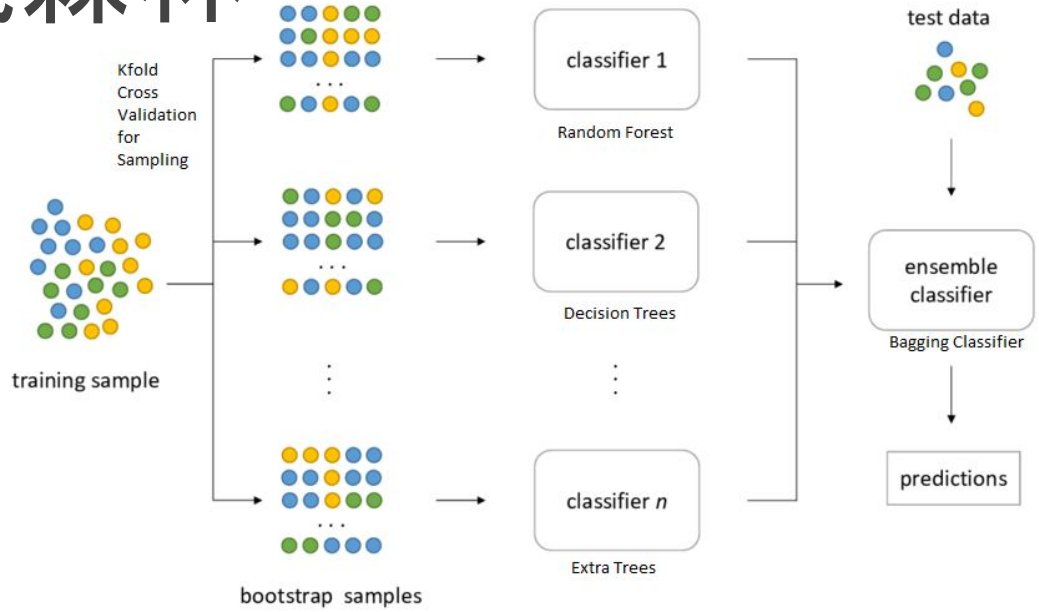
決策樹切割原理(Entropy)

決策樹Demo


1
2
petal width [standardized]
petal length [standardized]
2
3
4
5
6
7
4
5
6
7
8


node #0
petal length (cm) <= 4.75
entropy = 1.0
samples = 100.0%
value = [0.5, 0.5]
class = y[0]
True
False
node #1
sepal length (cm) <= 4.95
entropy = 0.191
samples = 48.6%
value = [0.971, 0.029]
class = y[0]
node #4
petal length (cm) <= 5.05
entropy = 0.31
samples = 51.4%
value = [0.056, 0.944]
class = y[1]
node #2
entropy = 0.0
samples = 1.4%
value = [0.0, 1.0]
class = y[1]
node #3
entropy = 0.0
samples = 47.1%
value = [1.0, 0.0]
class = y[0]
node #5
sepal length (cm) <= 6.5
entropy = 0.863
samples = 10.0%
value = [0.286, 0.714]
class = y[1]
node #12
entropy = 0.0
samples = 41.4%
value = [0.0, 1.0]
class = y[1]
node #6
sepal length (cm) <= 5.95
entropy = 0.65
samples = 8.6%
value = [0.167, 0.833]
class = y[1]
node #11
entropy = 0.0
samples = 1.4%
value = [1.0, 0.0]
class = y[0]
node #7
petal length (cm) <= 4.9
entropy = 1.0
samples = 2.9%
value = [0.5, 0.5]
class = y[0]
node #10
entropy = 0.0
samples = 5.7%
value = [0.0, 1.0]
class = y[1]
node #8
entropy = 0.0
samples = 1.4%
value = [1.0, 0.0]
class = y[0]
node #9
entropy = 0.0
samples = 1.4%
value = [0.0, 1.0]
class = y[1]

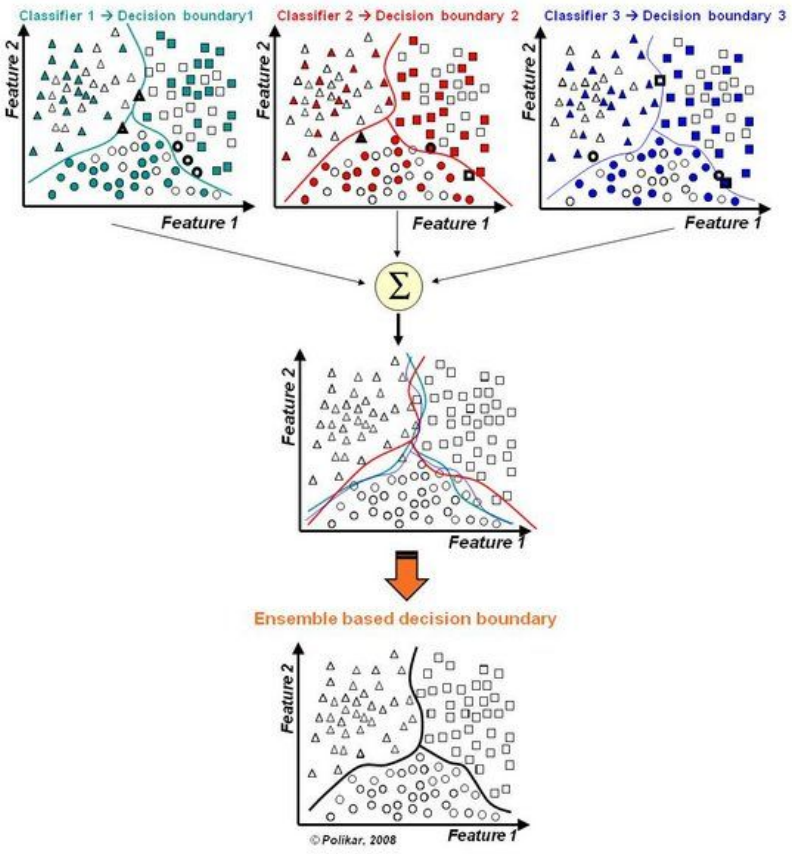
隨機森林



Bagging Classifier Process Flow

每棵樹只用到部分資料、Features

隨機森林


Classifier 1 → Decision boundary1
Classifier 2 → Decision boundary 2
Classifier 3 → Decision boundary 3
Feature 2
Feature 1
Σ
Ensemble based decision boundary
© Polikar, 2008

隨機森林 Demo

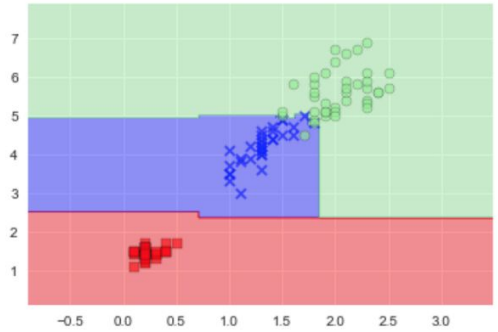
```
from sklearn.ensemble import RandomForestClassifier
```

```
forest = RandomForestClassifier(criterion='entropy', n_estimators=10,random_state=3,n_jobs=2)
```

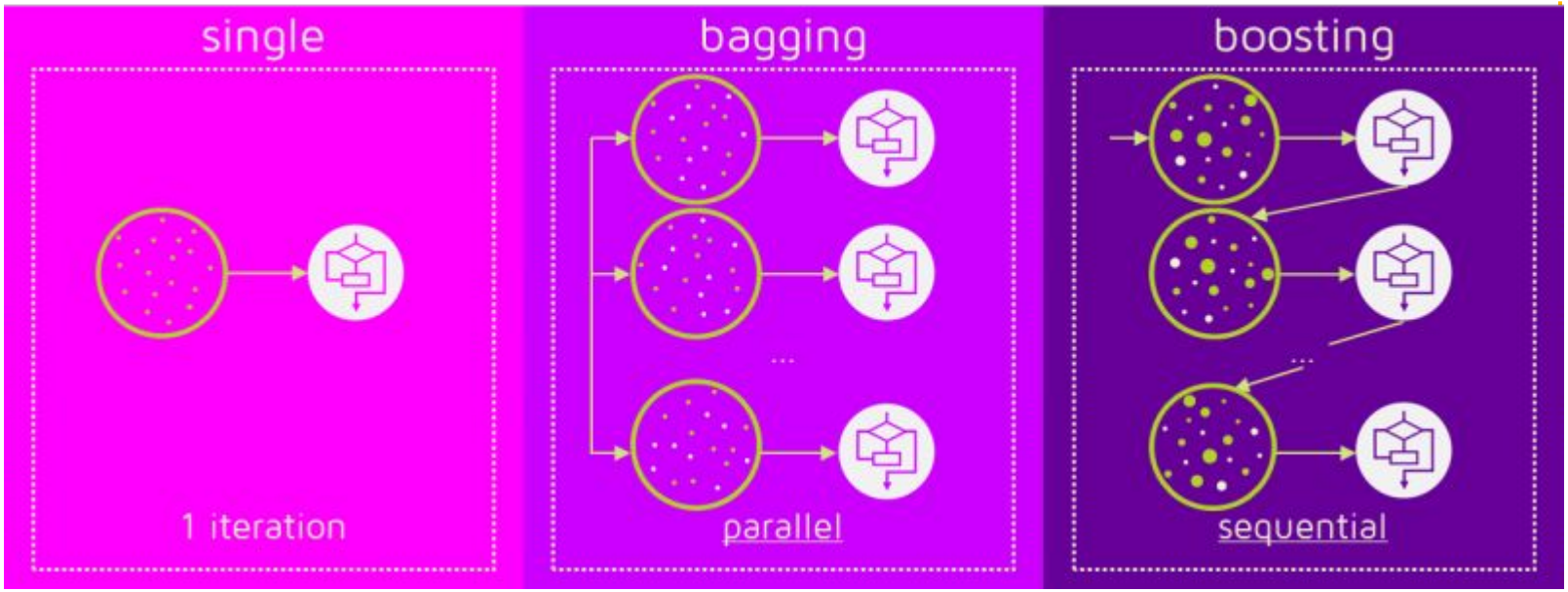
```
forest.fit(X_train,y_train['target'].values)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=10, n_jobs=2, oob_score=False, random_state=3,
            verbose=0, warm_start=False)
```

```
plot_decision_regions(X_train.values, y_train['target'].values, classifier=forest)
```



梯度提升決策樹(GBDT)



隨機森林 vs 梯度提升決策樹(GBDT)

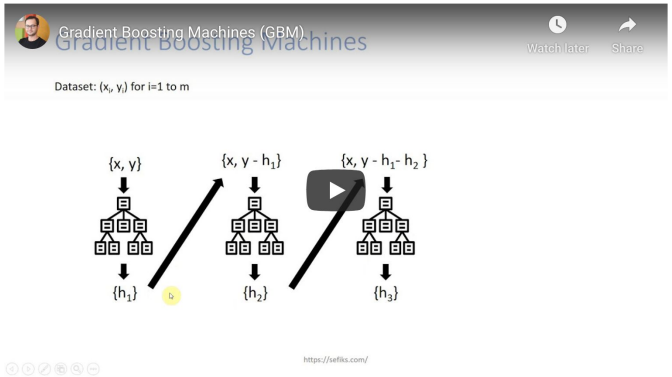
期末考時(可以分組考試的情況下):

1. 隨機森林:每人分配部分章節同時去唸
2. 梯度提升:每人輪流念全部內容，並把不懂或是做錯的部分標記起來給下個人

梯度提升決策樹(GBDT)

Objective

The following video covers the idea behind GBM. Then, we will mention GBM with a step by step example.



We actually apply boosting in real world!

This is very similar to baby step giant step method. We initially create a decision tree for the raw data set. That would be the giant step. Then, it is time to tune and boost. We will create new decision tree based on previous tree's error. We will apply this approach several times. These would be baby steps. Terence Parr described this process wonderfully in golf playing scenario as illustrated below.

詳細數學推導過程

梯度提升決策樹(GBDT)

Demo

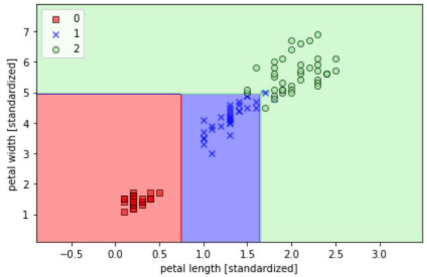
```
from sklearn.ensemble import GradientBoostingClassifier
```

```
gbdt = GradientBoostingClassifier(n_estimators=10,random_state=3)
```

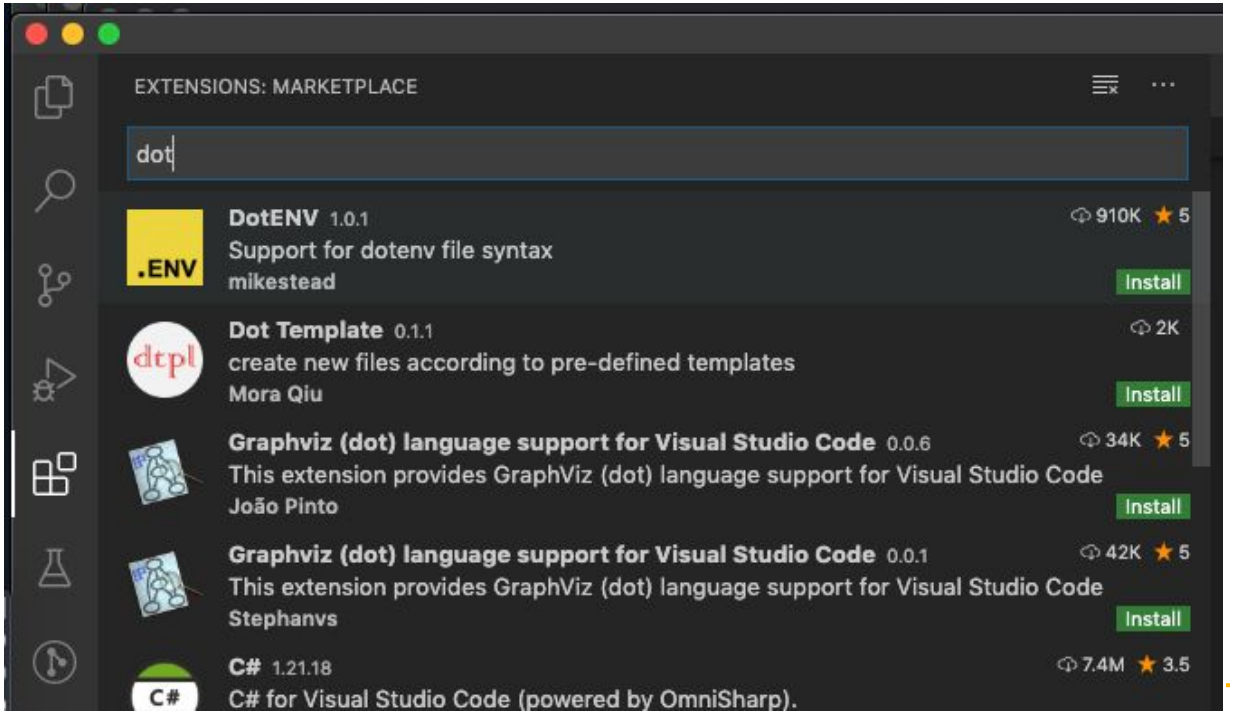
```
gbdt.fit(X_train,y_train['target'].values)
```

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=10,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=3, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

```
plot_decision_regions(X_train.values, y_train['target'].values, classifier=gbdt)
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



使用vscode看樹的結構



使用vscode看樹的結構