



# 機器學習基礎與演算法

## Chapter 7 決策樹系列 (Tree Based Model)



## 「版權聲明頁」

本投影片已經獲得作者授權台灣人工智慧學校得以使用於教學用途，如需取得重製權以及公開傳輸權需要透過台灣人工智慧學校取得著作人同意；如果需要修改本投影片著作，則需要取得改作權；另外，如果有需要以光碟或紙本等實體的方式傳播，則需要取得人工智慧學校散佈權。

# 課程內容

---

## 7. 決策樹系列 (Tree Based Model)

7-1[實作課程] 決策樹 (Decision Tree)

7-2[實作課程] 隨機森林 (Random Forest)

7-3[實作課程] 梯度提升機

(Gradient Boosting Machine)

7-4[實作課程] XGBoost

# Chapter 7 決策樹系列

## (Tree Based Model)

---

- 範例程式(example)的檔名會以藍色字體顯示且旁邊附上
- 練習(exercise)的檔案以紅色字體顯示且旁邊附上

# Section 7-1 [實作課程] 決策樹 (Decision Tree)

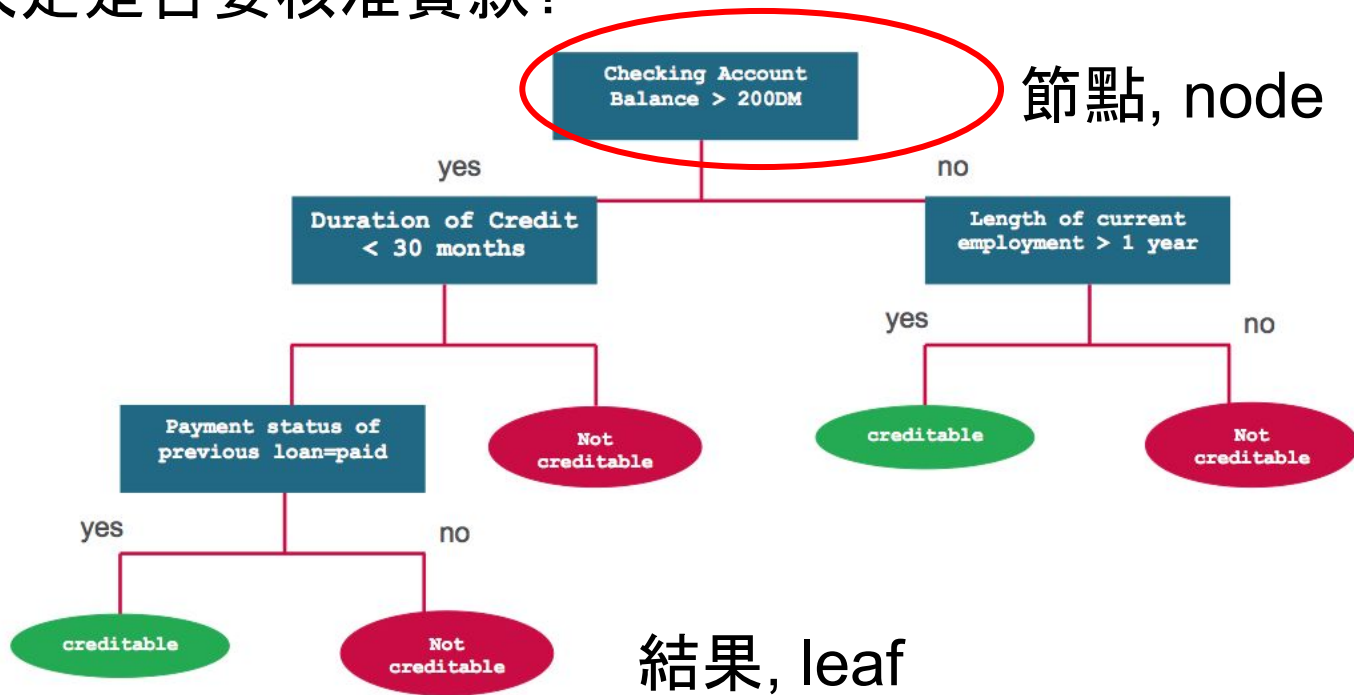
甚麼是決策樹？

- 決定是否要核准貸款？



# 甚麼是決策樹？

- 決定是否要核准貸款？



# 如何做決策？

---

- 該怎麼知道要用哪個 feature? 要用多少的值來做出我們的決策呢?
- 透過從訓練資料找出規則, 讓每一個決策能夠使**訊息增益** (Information gain) 最大化
- 如何衡量訊息增益?
  - 吉尼不純度, Gini impurity
  - 熵, Entropy



# 吉尼不純度 (Gini impurity)

- 數字越大，代表序列中的資料越混亂

$$Gini = 1 - \sum_j p_j^2$$

	Parent
C0	6
C1	6
Gini = 0.5	

**Gini :**  
 $1 - (6/12)^2 - (6/12)^2$   
**= 0.5**





# 熵 (Entropy)

---

$$Entropy = - \sum_j p_j \log_2 p_j$$

- 如果序列中所有 sample 都是同一個類別

$$entropy = -1 \log_2 1 = 0$$

- 若序列中各有一半的 sample 分屬不同的類別

$$entropy = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$



# Gini vs. Entropy

---

- 都是在衡量一個序列中的混亂程度，越高越混亂
- 數值皆為 0 ~ 1 之間。0 代表序列都是同樣的值
- Scikit-learn 預設使用 Gini

$$Gini = 1 - \sum_j p_j^2$$

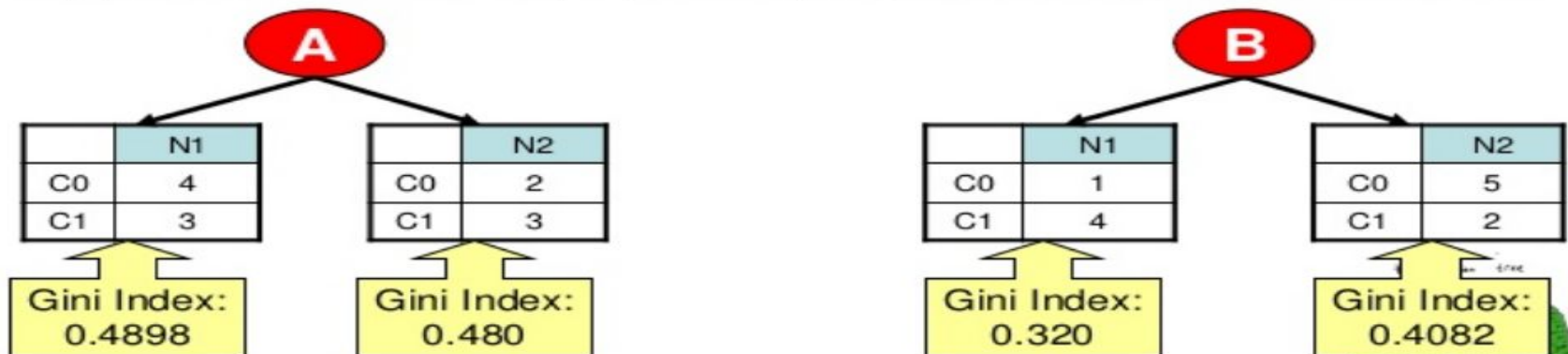
$$Entropy = - \sum_j p_j \log_2 p_j$$



# Information Gain 訊息增益

- 決策樹中，試著用 feature 將資料做切分，選取的 feature 必須能最大化訊息增益。而訊息增益則是由 Gini 或 Entropy 衡量，我們希望切分後的資料越純越好 (Gini=0)

Suppose there are two ways (A and B) to split the data into smaller subset.



Which one is a better split??

Compute the **weighted average of the Gini index** of both attribute



## 決策樹建立 (1/2)

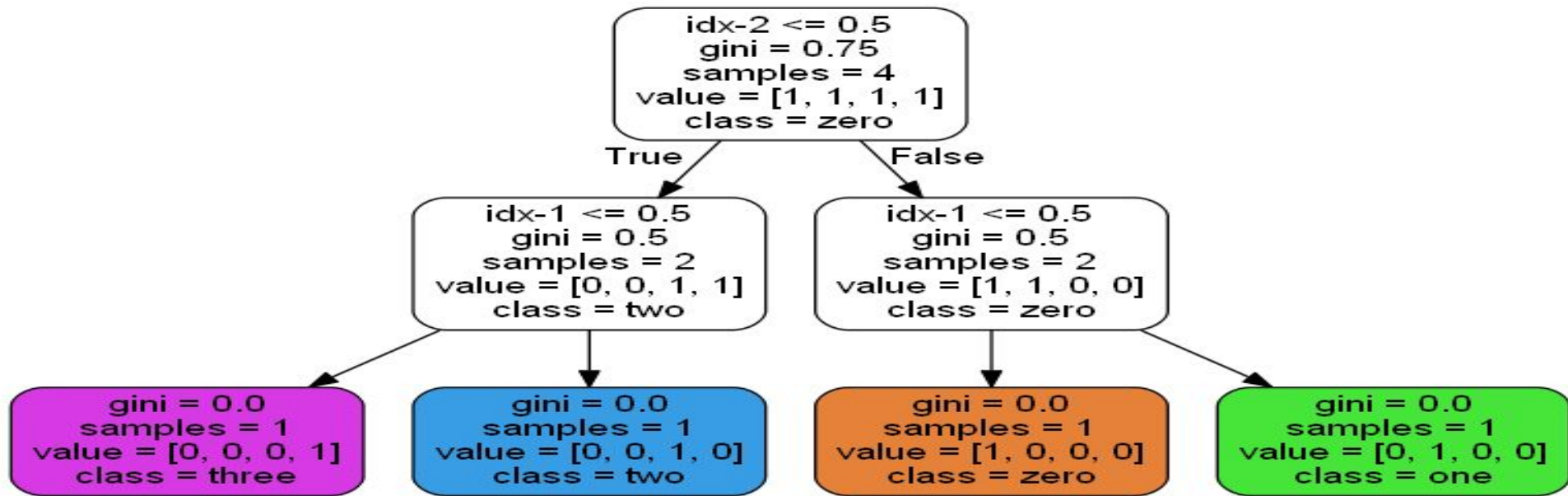
---

- 不斷尋找 feature 進行決策, 試著將資料切分為同一個類別 (minimize Gini)
  - 這樣會造成甚麼後果?



## 決策樹建立 (2/2)

- 當我們拿一批訓練資料給決策樹進行分類時，若沒有給定任何條件，決策樹會不斷進行分枝，直到所有 leaf 的資料都屬於同一個類別為止



# 決策樹 in Scikit-learn

---

- 兩行 code 建立決策樹

```
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
clf = DecisionTreeClassifier()
```



# 決策樹模型中的參數

---

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(
    criterion = 'gini',
    max_depth = None,
    min_samples_split = 2,
    min_samples_leaf = 1,
)
```



# Feature Importance

---

- 決策樹的另一優點是，我們可以從構建樹的過程中，透過 feature 被用來切分的次數，來得知哪些 features 是相對有用的
- 所有 feature importance 的總和會是 1
- 實務上，我們經常會用 feature importance 來排序 feature 的重要性以及選取要使用的 feature

```
# feature importance  
clf.feature_importances_
```





# 決策樹實戰

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
print(iris.data.shape, iris.target.shape)
(150, 4) (150, 1)

x_train, x_test, y_train, y_test = train_test_split(iris.data, iris.target)

print("shape of x_train: ", x_train.shape)
shape of x_train: (112, 4)

print("shape of x_test: ", x_test.shape)
shape of x_test: (38, 4)

clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                        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, presort=False, random_state=None,
                        splitter='best')

y_pred = clf.predict(x_test)
```

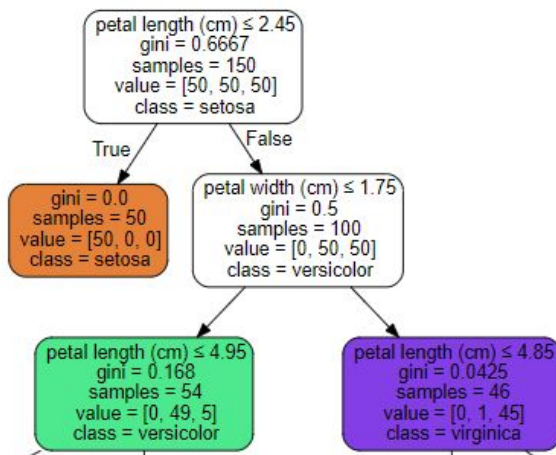
\*影片中 code 有誤: **accuracy\_score(y\_test, y\_pred)**



# 決策樹視覺化

- 生成好的樹，可以用額外的套件 graphviz，自動從 code 繪製成圖形，讓我們了解決策樹究竟學到了甚麼決策

```
>>> dot_data = tree.export_graphviz(clf, out_file=None,  
                                     feature_names=iris.feature_names,  
                                     class_names=iris.target_names,  
                                     filled=True, rounded=True,  
                                     special_characters=True)  
  
>>> graph = graphviz.Source(dot_data)  
>>> graph
```



# 決策樹小結

## 決策樹 summary

- 掃過所有 feature 與對應的值將資料做切分
- 希望資料盡可能分開, 透過切分後的資料純度 (Gini or Entropy) 來衡量
- 如果不對決策樹進行任何限制 (樹的深度、葉子至少要有多少樣本), 容易造成 Overfitting
- 透過 feature importance 來排序重要性



# 決策樹 Summary

---

- 掃過所有 feature 與對應的值將資料做切分
- 希望資料盡可能分開, 透過切分後的資料純度 (Gini or Entropy) 來衡量
- 如果不對決策樹進行任何限制 (樹的深度、葉子至少要有多少樣本), 容易造成 Overfitting
- 透過 feature importance 來排序重要性



# 決策樹進化! Ensemble

---

- 決策樹有著非常容易被理解的優點, 但是通常預測結果不會那麼準確
- 之後的學者想出方法, 把樹結合起來 (ensemble) 做改進
  - **Bagging (Bootstrap aggregating):** Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.
  - **Boosting:** Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.



## 練習 `decision_tree_example.ipynb`



- 請使用 Iris Dataset, 建立決策樹模型, 試著更改 Decision Tree 中的 **criterion**, **max\_depth**, **min\_samples\_split** 等參數, 並評估不同的參數是否會影響以下結果
  - training error / loss
  - testing error / loss
  - training speed (可用 `%%timeit` 計算 cell 執行的速度)



# Write a Decision Tree from Scratch (optional, but 推薦)



The image shows a man smiling on the left. To his right is a decision tree diagram. At the top is a table of fruit data. Below the table is a decision node: "Is diameter  $\geq 3$ ". The "False" branch leads to a leaf node with "R 1 Grape" and "R 1 Grape". The "True" branch leads to a leaf node with "G 3 Apple", "Y 3 Apple", and "Y 3 Lemon". A red banner across the bottom of the diagram area contains the text "{ML} Let's Write a Decision Tree from Scratch". Below the banner, there are two boxes: "Apple 100%" with a green leaf icon, and "Predict Apple 50% Lemon 50%" with a green leaf icon.

Color	Diam	Label
Green	3	Apple
Yellow	3	Apple
Red	1	Grape
Red	1	Grape
Yellow	3	Lemon

Is diameter  $\geq 3$ ?

False  
R 1 Grape  
R 1 Grape

True  
G 3 Apple  
Y 3 Apple  
Y 3 Lemon

**{ML}** Let's Write a Decision Tree from Scratch

Apple 100%

Predict  
Apple 50%  
Lemon 50%



## 補充閱讀

---

- 如果前面助教講的影片你都聽不懂，肯定是因為助教講的不夠清楚，只好幫各位找一些寫的不錯的文章，給大家參考
  - [決策樹 \(Decision Tree\)](#) - 中文
  - [how decision tree works](#) - 英文





## 思考問題

---

- 在分類問題中，若沒有任何限制，決策樹有辦法把訓練資料的 loss 完全降成 0 嗎？
- 決策樹做分類問題時，資料的不純度比較容易計算（是否屬於同一個類別）。那如果變成回歸問題，這時切分後的資料不純度該如何計算？樹建置完成後，又該如何進行預測呢？



## 7-2[實作課程] 隨機森林 (Random Forest)



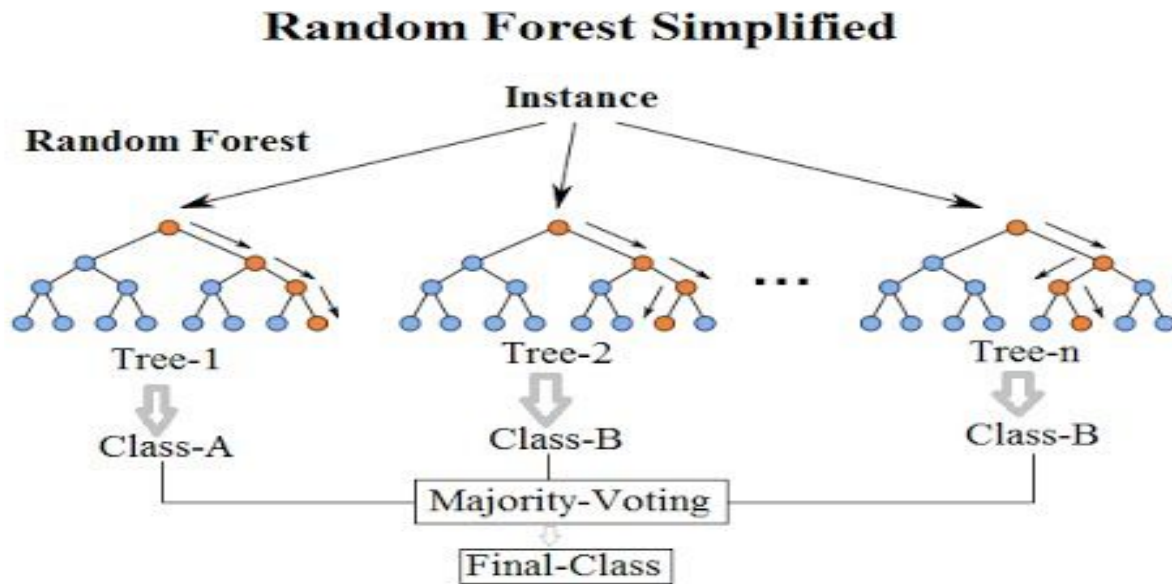
隨機森林 (RandomForest, RF)

一棵樹不夠。你有種第二顆嗎？



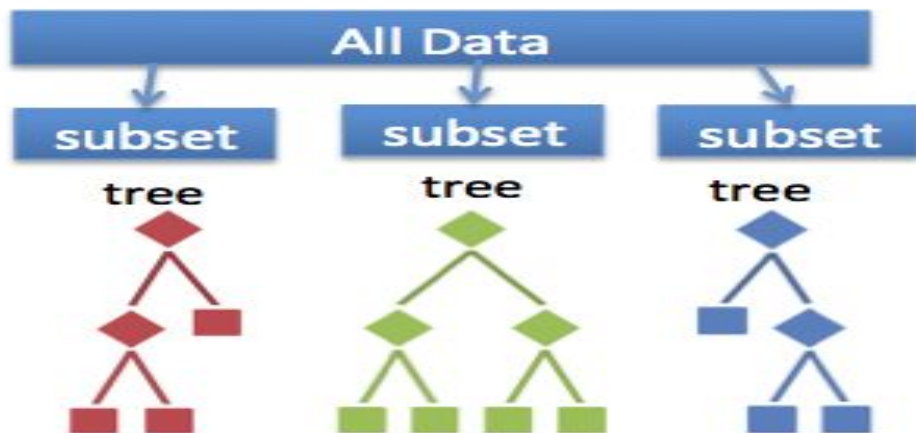
# 隨機森林 (Random Forest, RF)

- 決策樹非常容易 Overfitting (why?)
- 那如果多種幾棵樹，把樹變成森林會怎麼樣?...



# Why Random?

- 每一棵樹在生成過程中，都可能用到不同訓練資料及不同的 features
- 會用到哪些訓練資料及 features 則是隨機 (random) 決定!



# 決策樹系列：隨機森林 (Random Forest) (續)

## Why better than DecisionTree?

- Random forest 使用了我們稱作「Ensemble」的方式。從model的 import 就能看出

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

- 因為每棵樹可能是由不同樣本、不同 features 所生成，當使用集成方法，可將所有樹的結果做平均，使得預測結果更為穩定。

台灣人工智能學校



# Why better than Decision Tree?

---

- Random forest 使用了我們稱作「Ensemble」的方式。從model 的 import 就能看出

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

- 因為每棵樹可能是由不同樣本、不同 features 所生成，當使用集成方法，可將所有樹的結果做平均，緩解決策樹 Overfitting 的情形，使得預測結果更為穩定。



# 隨機森林 in Scikit-learn

---

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

```
from sklearn.ensemble import RandomForestRegressor
```

```
clf = RandomForestRegressor()
```



# 隨機森林中的常見參數

---

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

```
clf = RandomForestClassifier(  
    n_estimators=100, #number of trees  
    criterion="gini",  
    max_features="auto", #sqrt(features)  
    max_depth=10,  
    min_samples_split=2,  
    min_samples_leaf=1
```

)



- 請使用 Random forest 來執行 Iris dataset, 比較 Random forest 的模型結果是否比 Decision tree 來得好
- 請使用 digits dataset, 並比較如果樹的數量多寡 (`n_estimators`), 對結果是否會有改善?



# Write a Random forest from Scratch (optional)

The image is a composite. On the left, a Jupyter Notebook window titled 'jupyter demo' is shown. It contains a large, stylized title 'RANDOM FORESTS' in green and black. Below the title is a decision tree diagram. The tree starts with a root node 'Duration of Credit < 33 months'. If 'yes', it goes to a node 'Payment status of previous loan paid'. If 'no', it goes to a leaf node 'not creditable'. From 'Payment status of previous loan paid', if 'yes', it goes to a leaf node 'creditable', and if 'no', it goes to a leaf node 'not creditable'. If the root node is 'no', it goes to a node 'Length of current employment > 1 year'. If 'yes', it goes to a leaf node 'creditable', and if 'no', it goes to a leaf node 'not creditable'. Below the diagram, there is text explaining the task: 'We're going to learn about a machine learning model called a Random Forest. The task is to assess Credit Risk of someone using their financial history. Useful for insurance companies.' and a dataset link: 'Dataset: <https://github.com/joaquim/datasets/blob/master/credit/German%20Credit%20Data>'. Below this, it says 'This dataset classifies people described by a set of attributes as good or bad credit risks.' and 'What is a Random forest?'. On the right side of the image, a man with dark hair and a beard, wearing a white shirt, is pointing his fingers towards the viewer. He is in front of a background of a volcano with lava flowing.

```
graph TD
    Root[Duration of Credit < 33 months] -- yes --> Node1[Payment status of previous loan paid]
    Root -- no --> Leaf1(not creditable)
    Node1 -- yes --> Leaf2(creditable)
    Node1 -- no --> Leaf3(not creditable)
    Root -- no --> Node2[Length of current employment > 1 year]
    Node2 -- yes --> Leaf4(creditable)
    Node2 -- no --> Leaf5(not creditable)
```

We're going to learn about a machine learning model called a Random Forest. The task is to assess Credit Risk of someone using their financial history. Useful for insurance companies.

Dataset: <https://github.com/joaquim/datasets/blob/master/credit/German%20Credit%20Data>

This dataset classifies people described by a set of attributes as good or bad credit risks.

What is a Random forest?



## 補充閱讀

---

- 如果前面助教講的影片你都聽不懂，肯定是因為助教講的不夠清楚，只好幫各位找一些寫的不錯的文章，給大家參考
  - [隨機森林 \(random forest\)](#) - 中文
  - [how random forest works](#) - 英文



# 思考問題

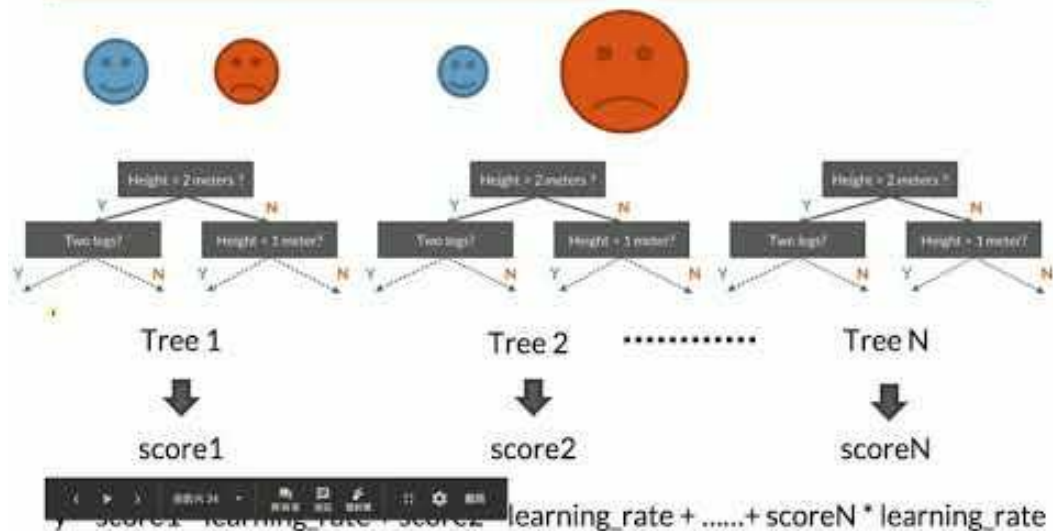
---

- Random Forest 中的每一棵樹，是希望能夠
  1. 盡量的生長 (讓樹生成很深，比較複雜)
  2. 不要過度生長，避免 Overfitting ?
- 假設資料總共有  $N$  筆 samples ( $N$  is large), 每棵樹用**取後放回**的方式抽了總共  $N$  筆資料來生成一棵樹，請問這棵樹大約使用了多少 % 的 unique 原資料生成 (不重複)?
  - hint: google 0.632 bootstrap



## 7-3[實作課程] 梯度提升機 (Gradient Boosting Machine)

GBM 的概念



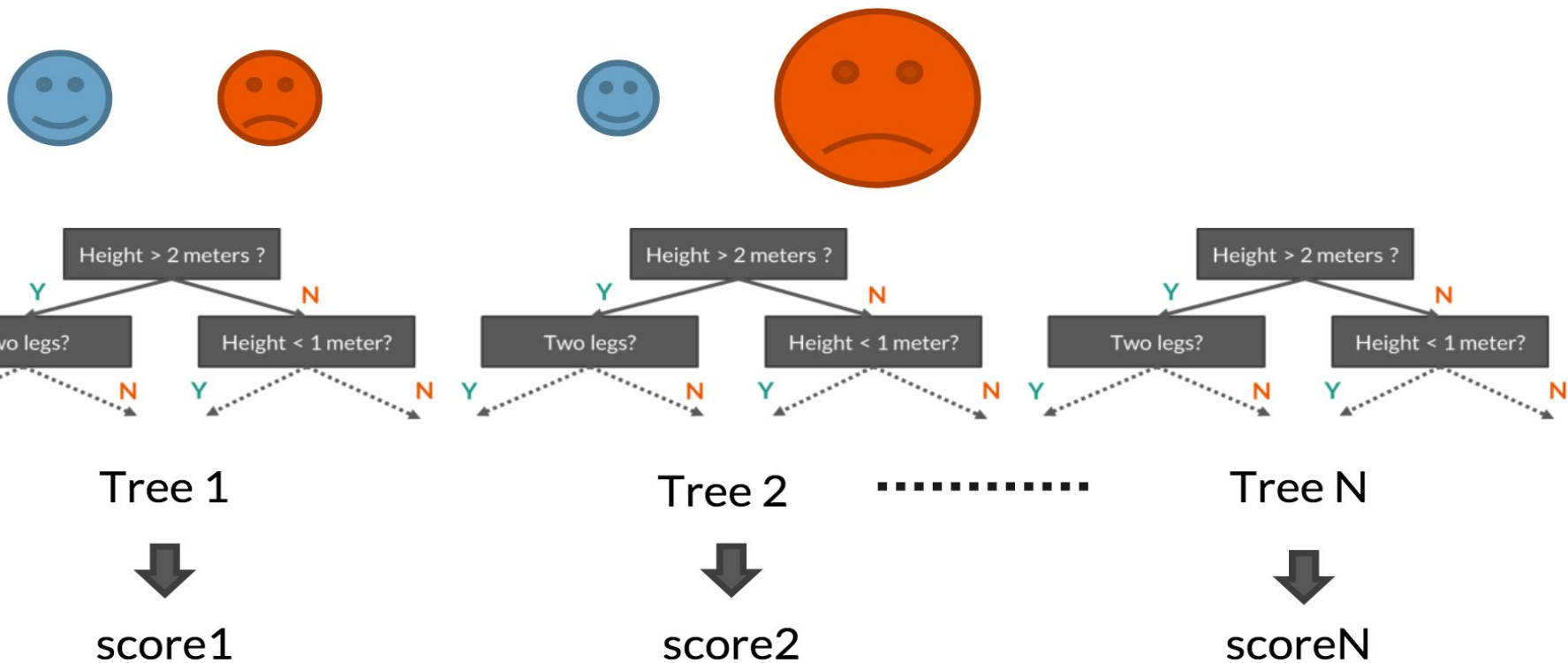
# Boosting? Gradient?

---

- 前面我們學到的方法稱為 Bagging (Bootstrap aggregating), 用抽樣的資料與 features 生成每一棵樹, 最後再取平均
- Boosting 則是希望能夠由後面生成的樹, 來修正前面樹學的不好的地方
- 要怎麼修正前面學錯的地方呢? 計算 Gradient! (先想像 Gradient 就是一個能教我們修正錯誤的東西)

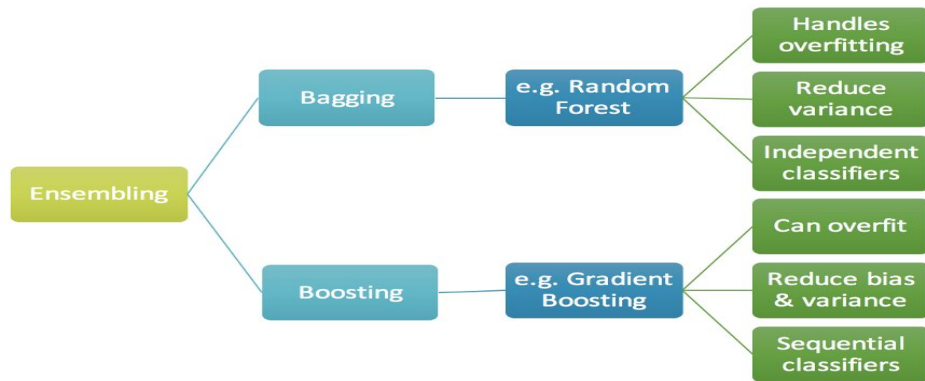


# GBM 的概念



# Bagging vs. Boosting

- Bagging: 透過抽樣的方式生成樹，每棵樹彼此獨立💬
- Boosting: 透過序列 (additive) 的方式生成樹，後面生成的樹會與前面的樹有關
- 一般來說，Boosting 的模型會比 Bagging 來的準確





# Kaggle 大師帶你理解 Gradient boosting (連結)

If linear regression was a Toyota Camry, then gradient boosting would be a UH-60 Blackhawk Helicopter. A particular implementation of gradient boosting, [XGBoost](#), is consistently used to win machine learning competitions on [Kaggle](#). Unfortunately many practitioners (including my former self) use it as a black box. It's also been butchered to death by a host of drive-by data scientists' blogs. As such, the purpose of this article is to lay the groundwork for classical gradient boosting, intuitively and comprehensively.



Linear Regression



Gradient Boosting



# 決策樹系列：梯度提升機 (Gradient Boosting Machine) (續)

GBM 常見參數設定

```
from sklearn.ensemble import GradientBoostingClassifier

clf = GradientBoostingClassifier(
    n_estimators=100, #number of trees
    learning_rate=0.1, # shrinkage of prediction
    max_features="None",
    max_depth=3
)
```

台灣人工智慧學校



# GBM in Scikit-learn

---

```
from sklearn.ensemble import GradientBoostingClassifier  
from sklearn.ensemble import GradientBoostingRegressor  
  
clf = GradientBoostingClassifier()
```



# GBM 常見參數設定

---


```
from sklearn.ensemble import GradientBoostingClassifier

clf = GradientBoostingClassifier(
    n_estimators=100, #number of trees
    learning_rate=0.1, # shrinkage of prediction
    max_features="None",
    max_depth=3
)
```



## 練習

---

- 請改用 **Gradient boosting (gradient\_boosting\_example .ipynb)**  **的模型來執行 Iris / digits dataset**，並試著增加樹的數量 (n\_estimators)，比較是否會影響結果
- 如果單純增加樹的數量，沒有對 learning\_rate 做調整，是否會影響結果？



# 這麼難的模型還是想要手刻?! (optional)

---

- 沒問題! 單純用 Python 實現 Gradient Boosting Machine



## 補充閱讀

---

- 如果前面助教講的影片你都聽不懂，肯定是因為助教講的不夠清楚，只好幫各位找一些寫的不錯的文章，給大家參考
  - [GBM 簡介](#) - 中文
  - [intro to gradient boosting](#) - 英文



## 7-4[實作課程] XGBoost

What's XGBoost?

- 全名為 eXtreme Gradient Boosting
- XGBoost is an implementation of gradient boosted machine but add some features





# 一段 Kaggle 冠軍的訪談

---

Interview from Kaggle winner (What have you taken away from this competition?)

- With a good computer, R can process “big data” too
- Always write data processing code with scalability in mind
- **When in doubt, use XGBoost**



# What's XGBoost? (1/2)

---

- 全名為 eXtreme Gradient Boosting
- XGBoost is an implementation of gradient boosting machine but add some features



Linear Regression



Gradient Boosting



XGBoost



## What's XGBoost? (2/2)


---

- Additive model (與 GBM 類似)
- Features sampling (與 Random forest 類似)
- Add regularization in objective function
- Use 1<sup>st</sup> and 2<sup>nd</sup> derivative to help training



## XGBoost vs. GBM

---

- 阿里巴巴的面試題目：請問 XGBoost 與 GBM (Gradient boosting machine) 有什麼差異？
  - objective function 加上 regularization, 避免 Overfitting 
  - 用上一階及二階導數來生成下一棵樹
  - feature / data sampling。與 RF 相同, 每棵樹生長時用到不同的資料與 features



# XGBoost 安裝

---

- XGBoost 是由華盛頓大學博士班學生陳天奇所開發，是目前 Kaggle 比賽中最常見到的算法！
- Hub 環境上已經幫各位安裝完成

```
from xgb import XGBClassifier, XGBRegressor
```

- 若希望在自己的本機上安裝，請參考
  - Windows: [install XGBoost on windows](#)
  - Mac / linux: `pip install XGBoost`



# XGBoost model

---

```
from xgb import XGBClassifier, XGBRegressor
```

```
clf = XGBClassifier()
```

```
clf.fit()
```

```
...
```



# XGBoost 常見參數設定

---

- XGBoost 需設定的參數大概是目前我們學習到所有模型中最多的
- 要學會如何設定參數，需要先瞭解參數的意義
  - booster [default=gbtrees]: (gbtree, gblinier)



# XGBoost 常見參數設定 - 樹參數設定

---

**n\_estimators** [100]: number of trees

**learning\_rate** [0.1]: shrinkage

**max\_depth** [3]: too large → overfitting

**gamma** [0]: L2 loss regularization, too small → overfitting 

**lambda** [0]: L1 loss regularization, too small → overfitting

**scale\_pos\_weight** [1]: use for imbalance data 

\*方括 [ ] 內為該參數預設值





## Early stop in XGBoost (1/2)

- XGBoost model 非常強大, 但也容易 Overfitting, Early stop 幫助我們在 Overfitting 前提早停下來



## Early stop in XGBoost (2/2)

```
# eval_metrics = rmse, logloss, error, auc, merror, mlogloss, custom
eval_set = [(X_test, y_test)]
model = XGBClassifier()
model.fit(X_train, y_train, early_stopping_rounds=10, eval_metric="auc",
          eval_set=eval_set, verbose=True)
```

```
[0]    validation_0-auc:0.817834
```

```
Will train until validation_0-auc hasn't improved in 10 rounds.
```

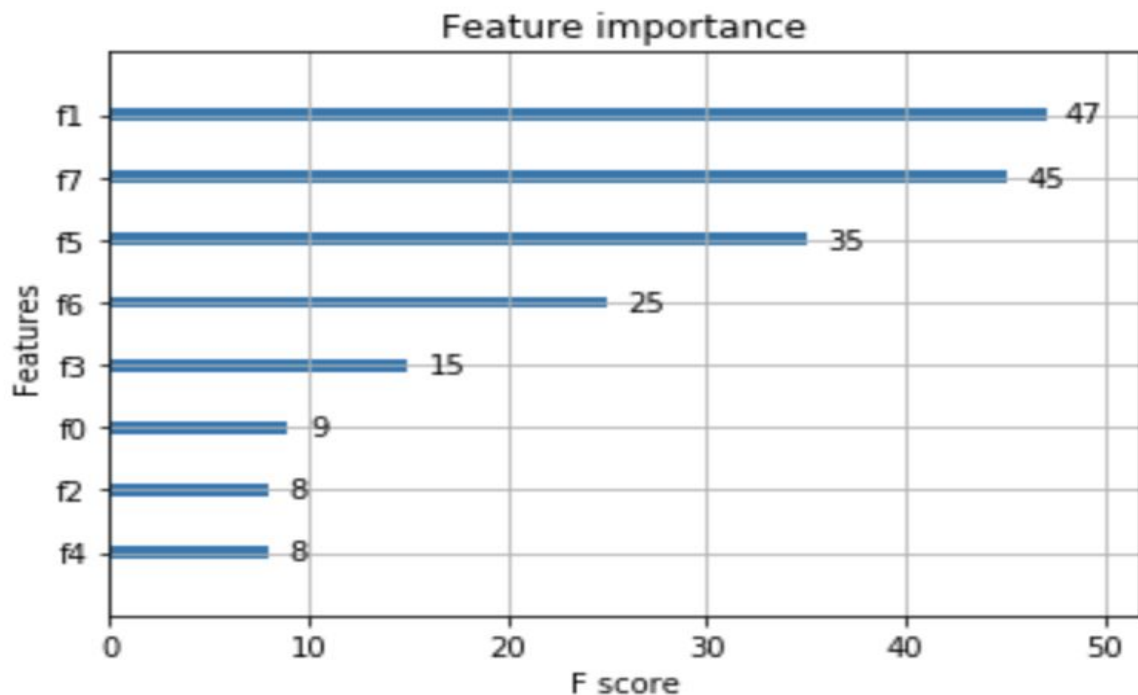
- 將 testing data 放進 eval\_set, 如果 validation 的結果 10 次沒有進步, 就提前結束 training
- 也可以改放 training data, 觀看 training loss 下降的感覺  
(文字一樣會顯示 validation\_0)



# Feature Importance in XGBoost

## XGBoost 內建功能

```
from xgboost import plot_importance
plot_importance(model)
plt.show()
```



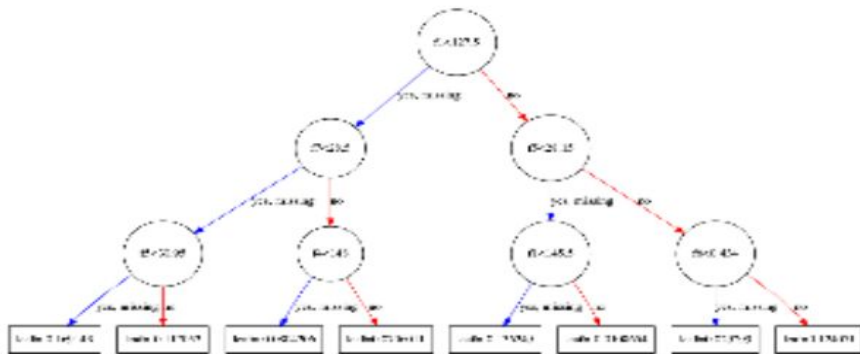
# XGBoost 視覺化

- 若執行這段 code 有 error, 代表環境還沒有安裝好 graphviz, 請重開 Server


```
In [31]: from xgboost import plot_tree
          from matplotlib.pyplot import rcParams

          plot_tree(model, num_trees=1)
          # plt.title("max_depth = 100, with gamma = 10")
          # plt.savefig("tree_with_max_depth_gamma", dpi = 700)
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f416570b6a0>
```



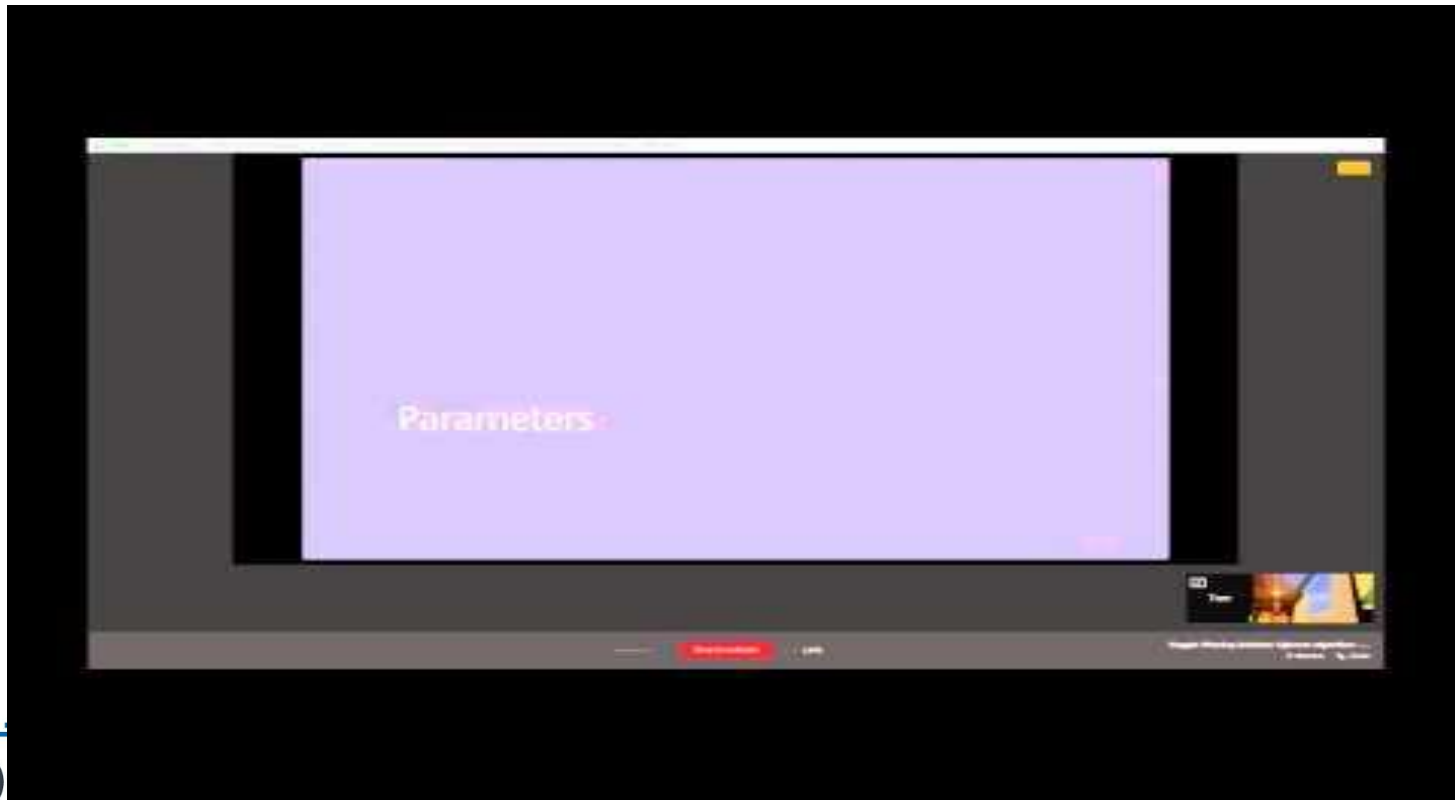
## 練習

- 請使用 example 中的 diabetes dataset, **使用 XGBoost 進行訓練**  , 試著更改如 n\_estimators、max\_depth 甚至是 scale\_pos\_weight (平衡 imbalance data, 如果 **類別 0 數量 : 類別 1 數量 = 5 : 1**, 則可設置 5)
- 與 Decision Tree, Random Forest, Gradient Boosting Machine 進行比較, XGBoost 真的有比較厲害? (記得使用同一份 testing set)
- 不設定 early stop, 把 n\_estimators 調高 (500~1000), 就可以體驗看看甚麼叫做 Overfitting



# XGBoost 作者講解並推導原理

---



## 補充閱讀

---

- 如果前面助教講的影片你都聽不懂，肯定是因為助教講的不夠清楚，只好幫各位找一些寫的不錯的文章，給大家參考
  - [XGBoost 詳解](#) - 中文
  - [XGBoost parameter tuning](#) - 英文
  - [slides from XGBoost author](#) - 英文



## 思考問題

---

- 同樣的 dataset 若存在兩個完全一模一樣的 feature (feature1, feature2), 這兩個 feature 的 importance, 在 XGBoost 與 Random Forest 的模型結果中, 會一樣嗎?
- XGBoost 中, row\_sample 代表對資料筆樹抽樣 (row), col\_sample 代表對 features 抽樣, 若這兩個都設置成 1 (代表不抽樣, 全部使用), 每次訓練後的樹會長的一模一樣嗎?

