

Python 機器學習程式設計 2



code 放在 hub 中的 courses 內

- 為維護課程資料, courses 中的檔案皆為 read-only, 如需修改請 cp 至自身的環境中
- 打開 terminal, 輸入

```
cp -r courses/python_ML2 mypython_ML2
```

複製

目標資料夾

要存放的資料夾名稱
(自行命名)

- 今後的課程, 如果需要下載課程資料都會使用這樣的方式

session 1: intro

Python 機器學習程式設計

台灣人工智慧學校



各時段預計完成內容

時段	Session
09:30 - 10:30	決策樹與隨機森林
10:50 - 12:30	GBM 與 XGBoost
14:00 - 15:00	非監督式學習
15:20 - 17:00	Kaggle 實戰

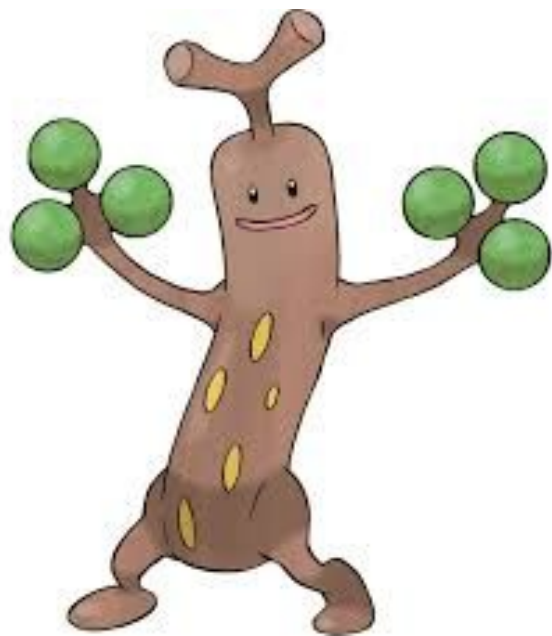
本次課程結束後你 (妳) 應該會什麼？

- **軟實力**

- 了解 tree-based 的模型，包含決策樹、隨機森林、GBM 與 XGBoost 的原理
- 了解 PCA, Hierarchical 等非監督式學習方法

- **硬底子**

- 如何用 Scikit-learn 使用上述的模型
- 若模型需要調整參數，有哪些是重要的參數及如何調整
- 如何用 Scikit-learn 中的 PCA, Hierarchical 等方法對資料進行降維
- 從 0 開始完成一場 Kaggle 實戰！



決策樹 Decision Tree

一顆可以幫你做決策的樹！



session 1: decision tree 1

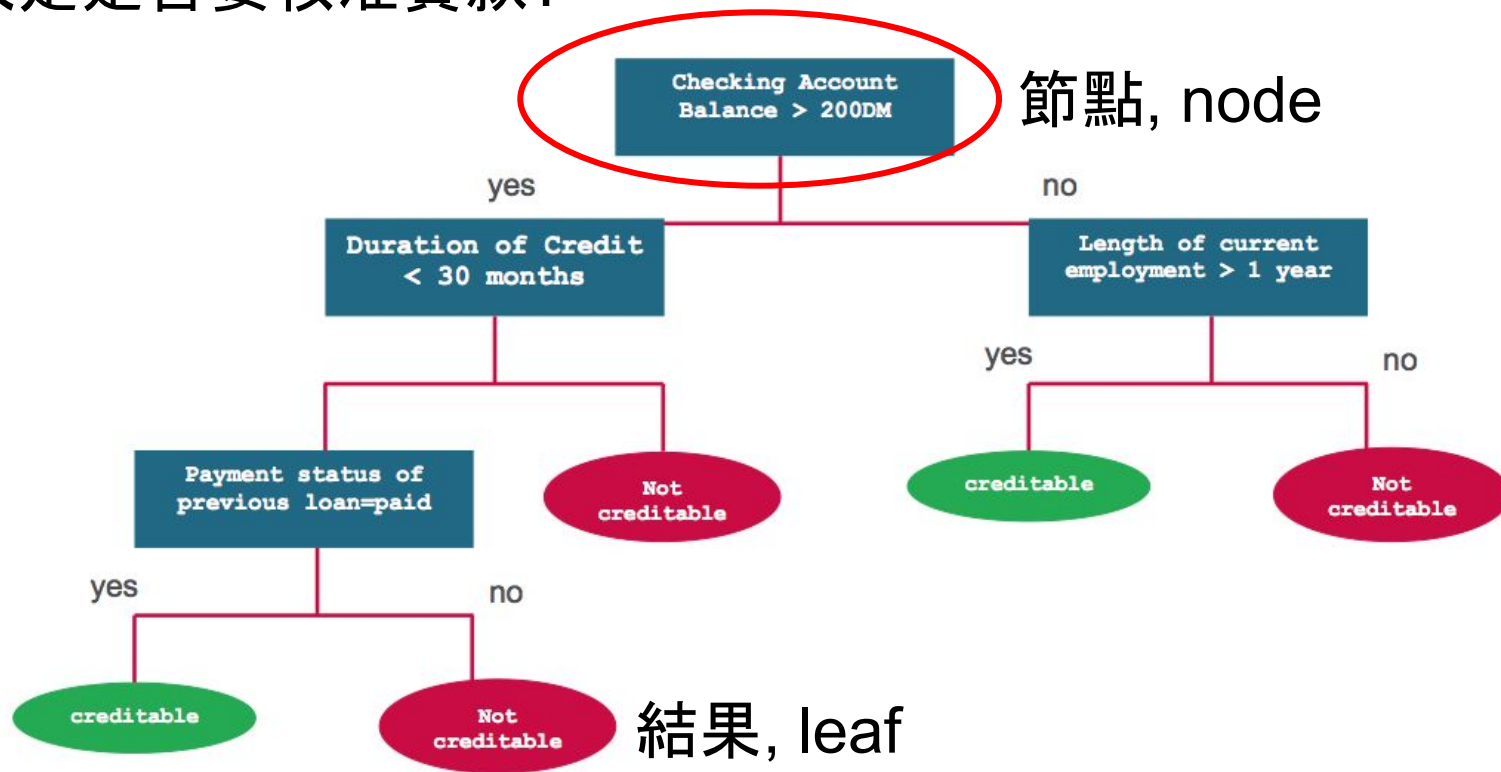
甚麼是決策樹?

- 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 (Information gain)?
 - 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

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

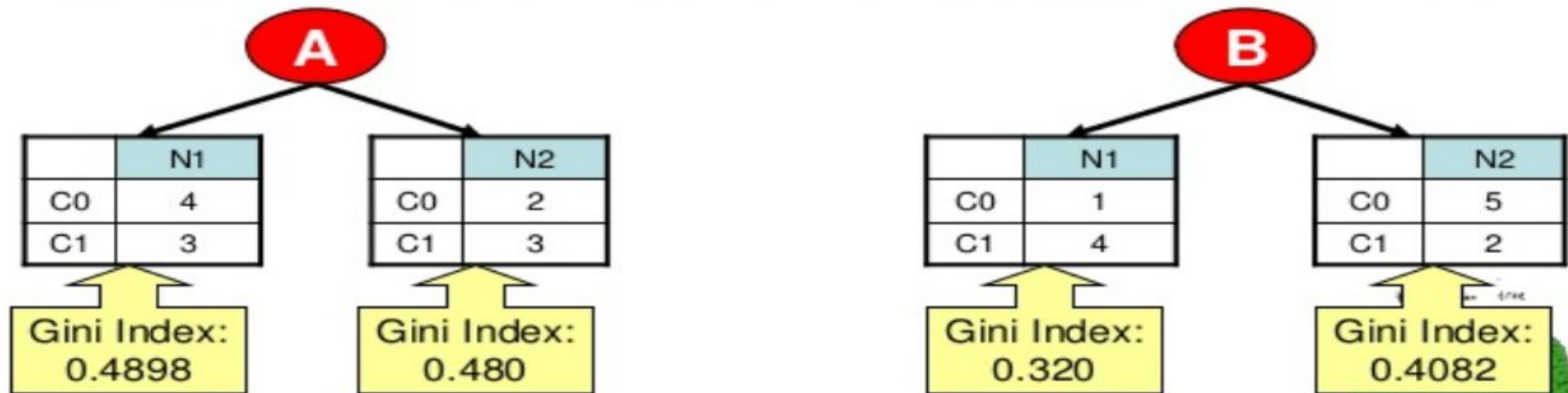
$$Entropy = - \sum_j p_j \log_2 p_j$$



Information Gain 資訊增益

- 決策樹中，試著用 feature 將資料做切分，選取的 feature 必須能最大化資訊增益。而資訊增益則是由 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

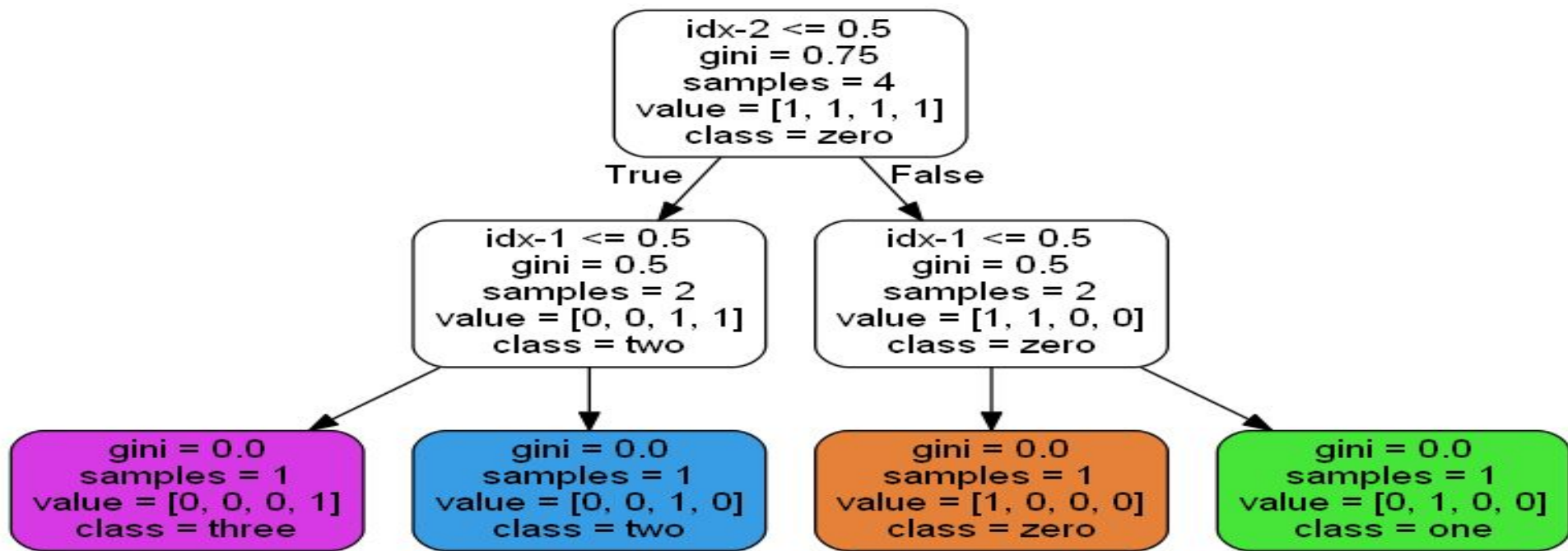
決策樹建立

- 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

- 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

影片中 code 有誤: **accuracy_score(y_test, y_pred)**

```
In [100]: 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

In [101]: iris = load_iris()

In [102]: print(iris.data.shape, iris.target.shape)
(150, 4) (150,)

In [103]: x_train, x_test, y_train, y_test = train_test_split(iris.data, iris.target)

In [104]: print('shape of x_train: ', x_train.shape)
shape of x_train: (112, 4)

In [105]: print('shape of x_test: ', x_test.shape)
shape of x_test: (38, 4)

In [106]: clf = DecisionTreeClassifier()

In [107]: clf.fit(x_train, y_train)

Out[107]: 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')

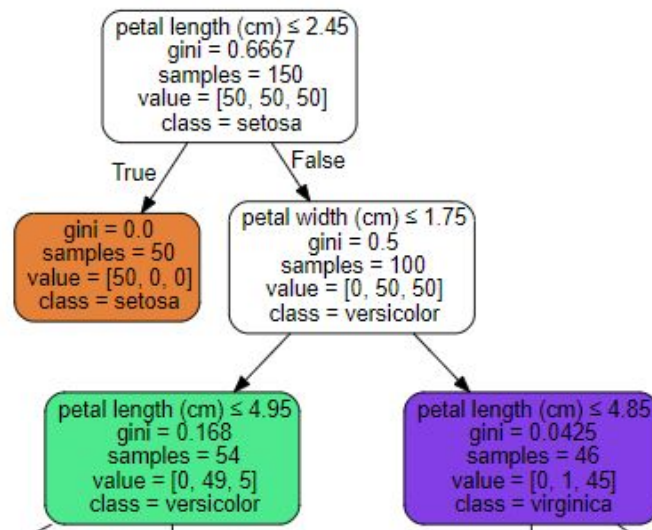
In [108]: y_pred = clf.predict(x_test)
```



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 DecisionTree 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 next to a decision tree diagram. The diagram illustrates a classification task for fruits based on color and diameter. A table of training data is shown at the top, followed by a decision node 'Is diameter ≥ 3 ?'. The 'False' branch leads to a leaf node predicting 'Grape' (100% confidence). The 'True' branch leads to a leaf node predicting 'Apple' (50%) and 'Lemon' (50%).

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

Is diameter ≥ 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？





隨機森林 (RandomForest, RF)

一棵樹不夠, 你有種第二顆嗎?





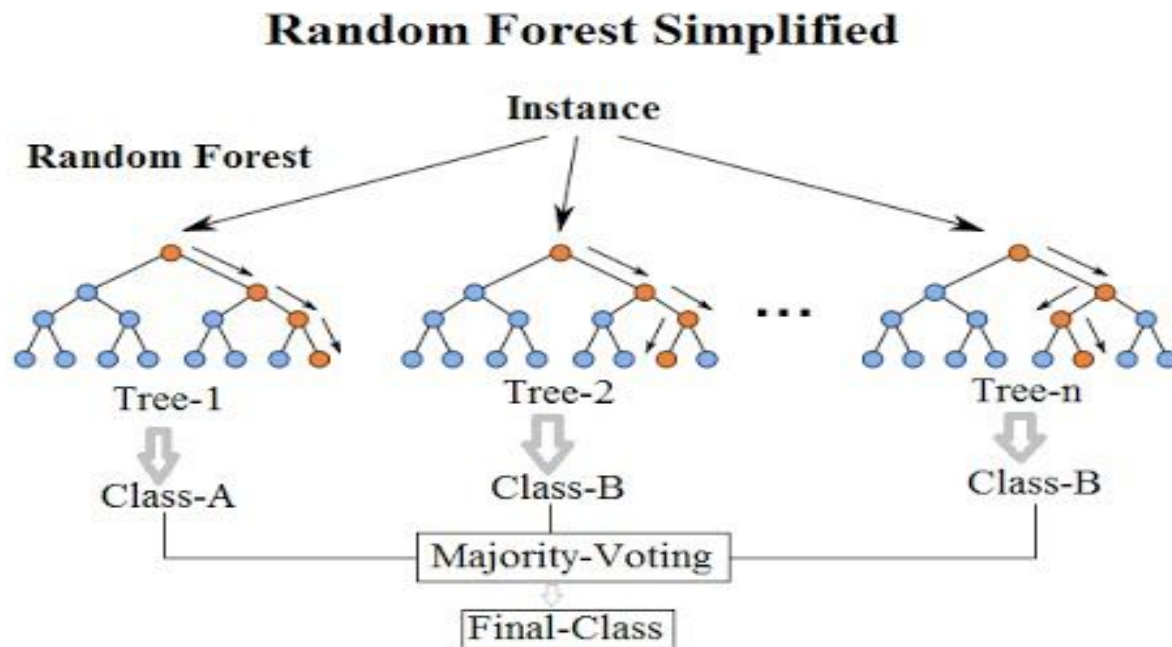
隨機森林 (RandomForest, RF)

一棵樹不夠。你有種第二顆嗎？



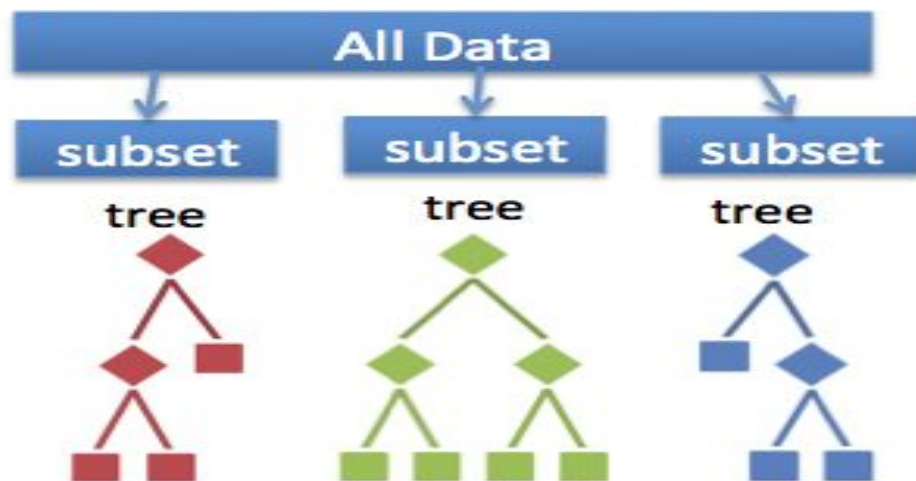
隨機森林 (Random forest, RF)

- 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!



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。

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，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
)
```



練習 random_forest_exercises.ipynb

- 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 Forests from Scratch (optional)

jupyter demo (AutoSaved)

File Edit View Insert Cell Kernel Help | Python 2.0

RANDOM FORESTS

```
graph TD
    Root[ ] -- yes --> Node1[Duration of Credit < 35 months]
    Root -- no --> Node2[Length of current employment > 1 year]
    Node1 -- yes --> Node3[Payment status of previous loan/paid]
    Node1 -- no --> Leaf1[not creditable]
    Node2 -- yes --> Leaf2[creditable]
    Node2 -- no --> Leaf3[not creditable]
    Node3 -- yes --> Leaf4[creditable]
    Node3 -- 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://www.kaggle.com/ulldragon/dataset/credit-risk>

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

- RadomForest 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

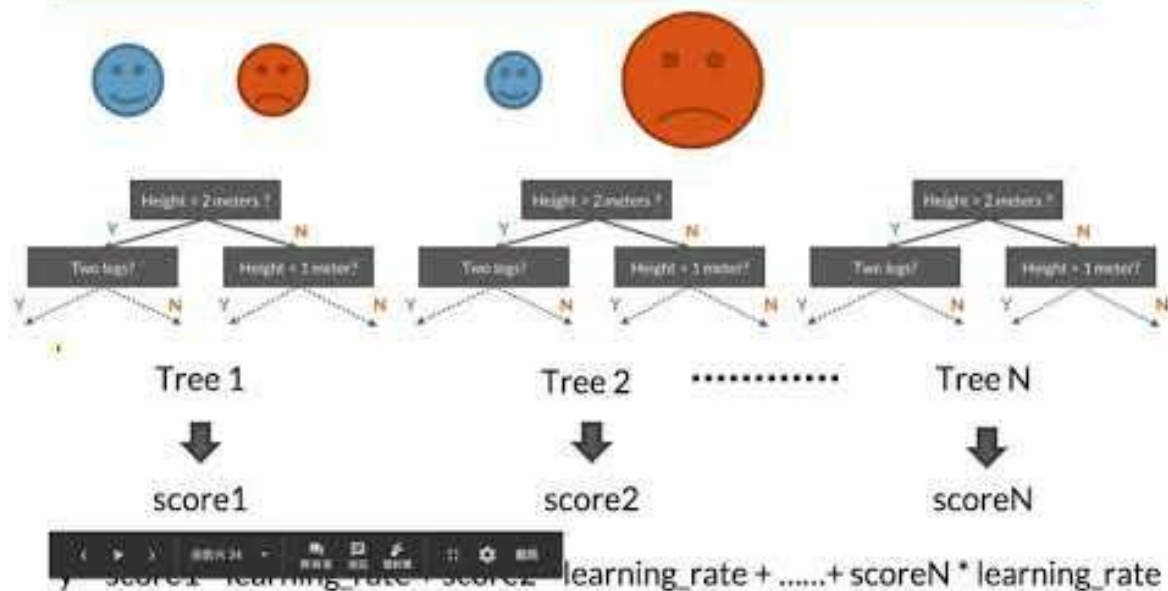




梯度提升機 Gradient Boosting Machine, GBM



GBM 的概念

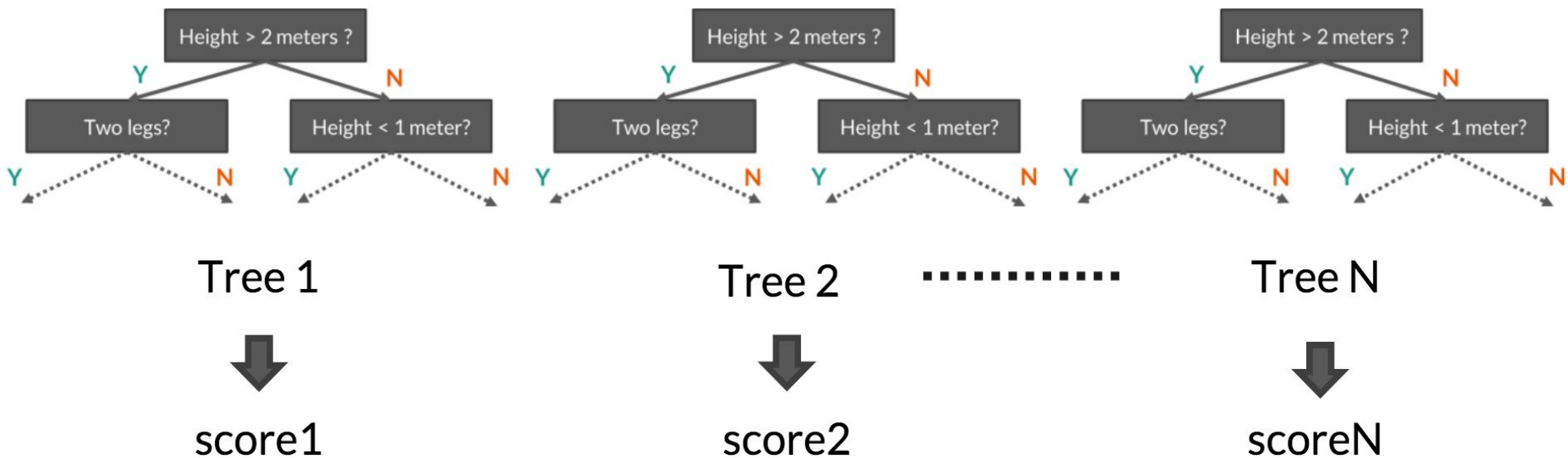


Boosting? Gradient?

- 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)

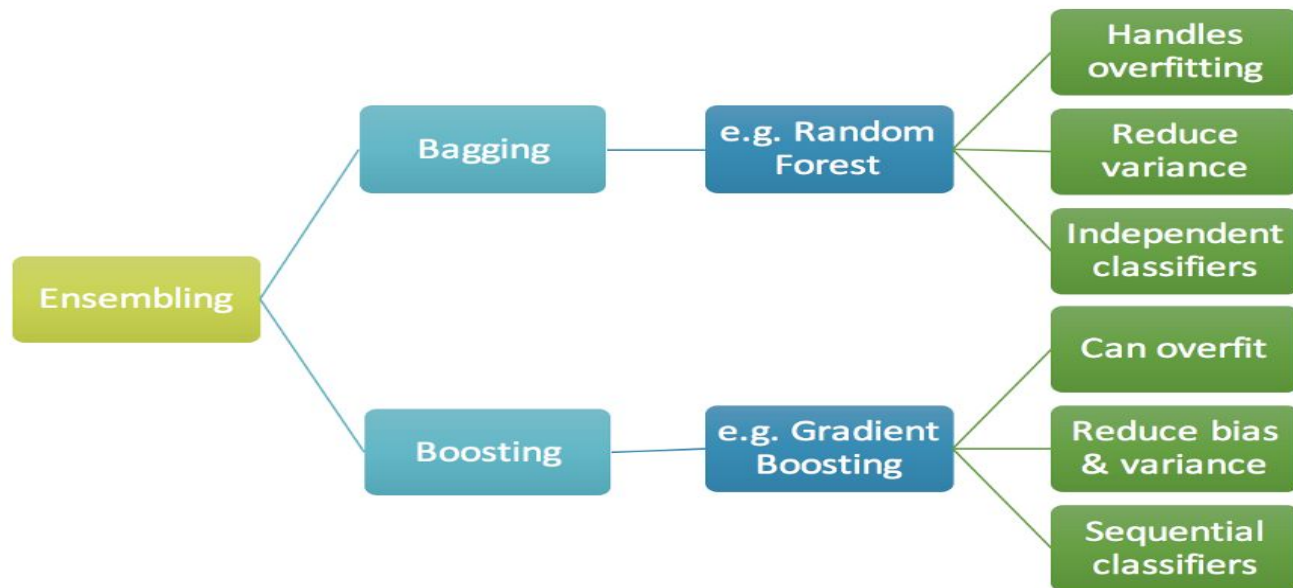


GBM 的概念



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

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 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





終極大殺器 - XGBoost



What's XGBoost?

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



XGBoost

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?

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



Linear Regression



Gradient Boosting



XGBoost

What's XGBoost?

- Additive model (與 GBM 類似)
- Features sampling (與 Random forest 類似)
- Add regularization in objective function
- Use 1st and 2nd 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, gblinear)



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

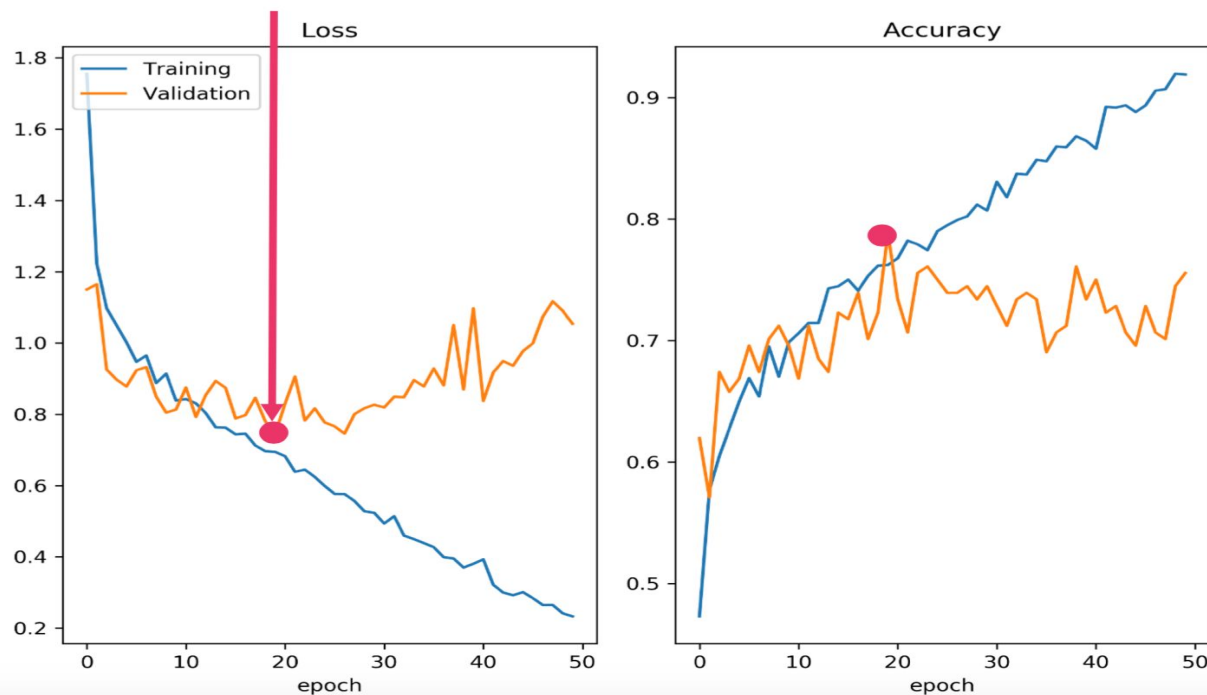
lambda [0]: L1 loss regularization, too small → overfitting

scale_pos_weight [1]: use for imbalance data



Earllystop in XGBoost

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



Earllystop in XGBoost

```
# 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.
```

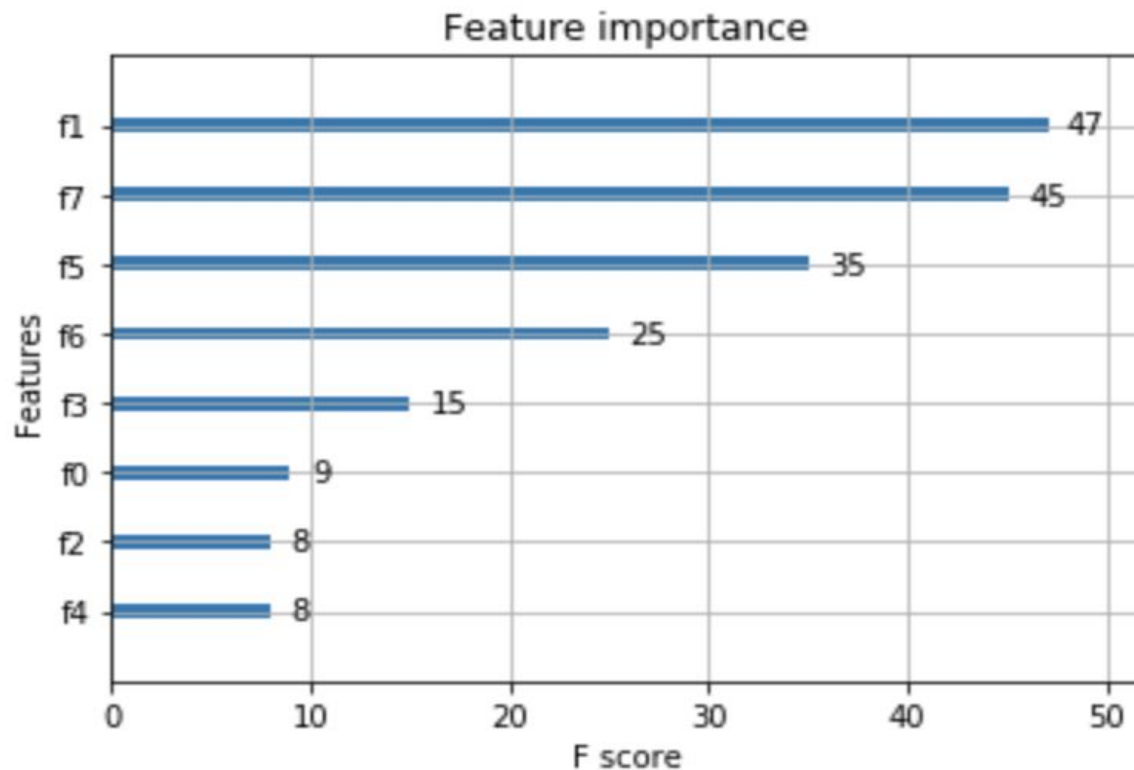
- 將 testing data 放進 eval_set, 如果 validation 的結果 10 次沒有進步, 就提前結束 training
- 也可以改放 training data, 觀看 training loss 下降的感覺 (文字一樣會顯示 validation_0)



feature importance in XGBoost

XGBoost 內建功能

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



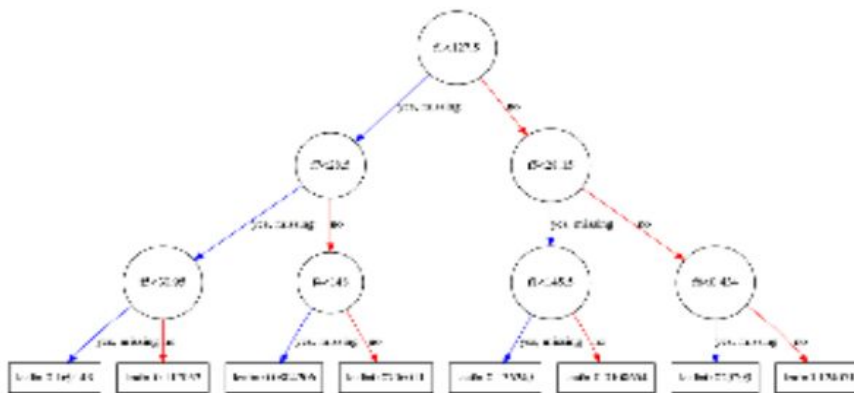
XGBoost 視覺化

- 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.pylab 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 DecisionTree, RandomForest, 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 `earlystop`, 把 `n_estimators` 調高 (500~1000), 就可以體驗看看甚麼叫做 Overfitting





理解 XGBoost 原理與算法

(optional but 建議)



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



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 RandomForest 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?





非監督式學習- PCA

主成分分析



主成份分析 (Principal Component Analysis, PCA)

- 實務上我們經常遇到資料有非常多的 features, 有些 features 可能高度相關, 有什麼方法能夠把高相關的 features 去除?
- PCA 透過計算 eigen value, eigen vector, 可以將原本的 features 降維至特定的維度
 - 原本 Data 有 100 個 features, 透過 PCA, 可以將這 100 個 features 降成 2 個 features
 - 新 features 為舊 features 的線性組合



新 features 彼此不相關

The original variables is noted as x_1, x_2, \dots, x_n , and the new variables can be represented as

$$\left. \begin{aligned} z_1 &= a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ z_2 &= a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ &\vdots \\ z_n &= a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n \end{aligned} \right\} \text{Uncorrelated}$$



PCA in Scikit-learn

```
from sklearn.decomposition import PCA
```

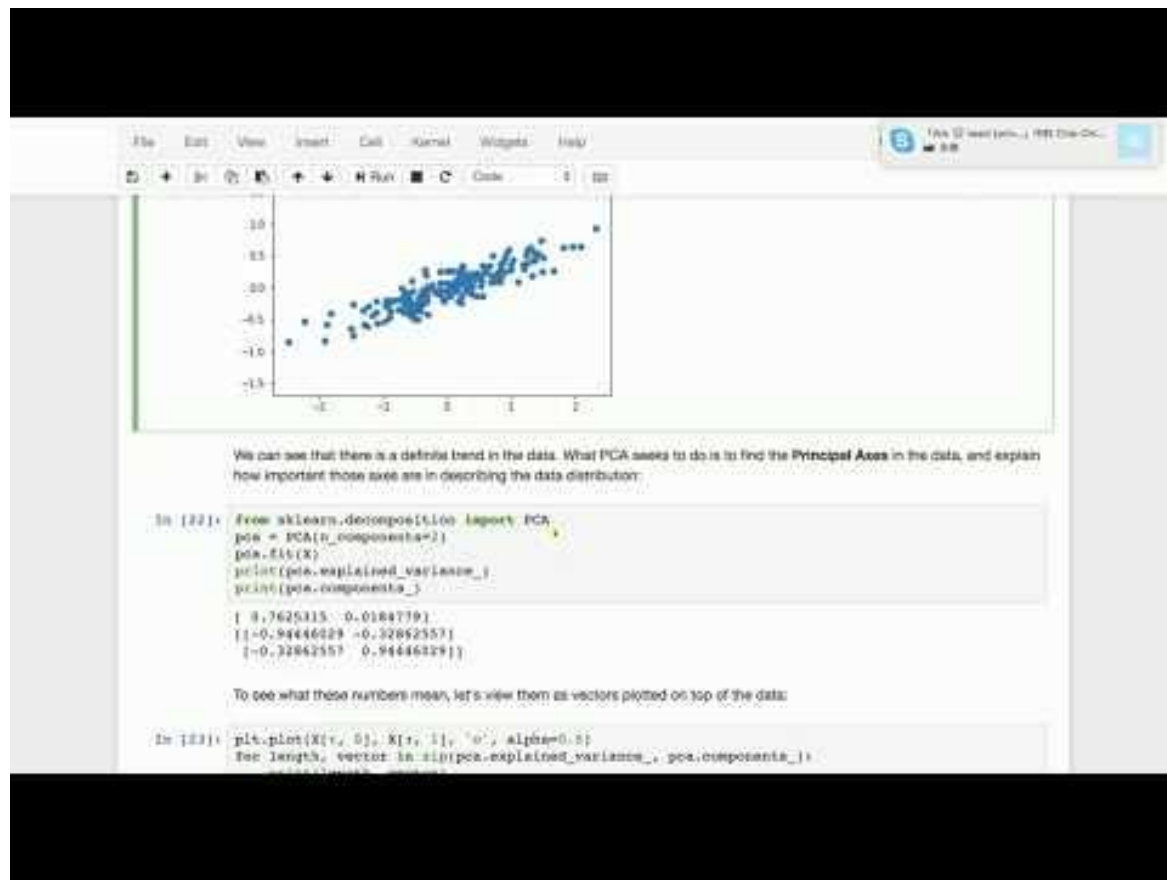
```
pca = PCA(n_components=2)
```

```
X_reduct = pca.fit_transform(X) #X.shape=(200, 64)
```

```
print(X_reduct.shape) #(200, 2)
```



PCA 實戰



練習

- 使用 digits dataset, 比較如果將資料降維之後再訓練模型, 準確度是否會提升





非監督式學習- Hierarchical clustering

階層式分析



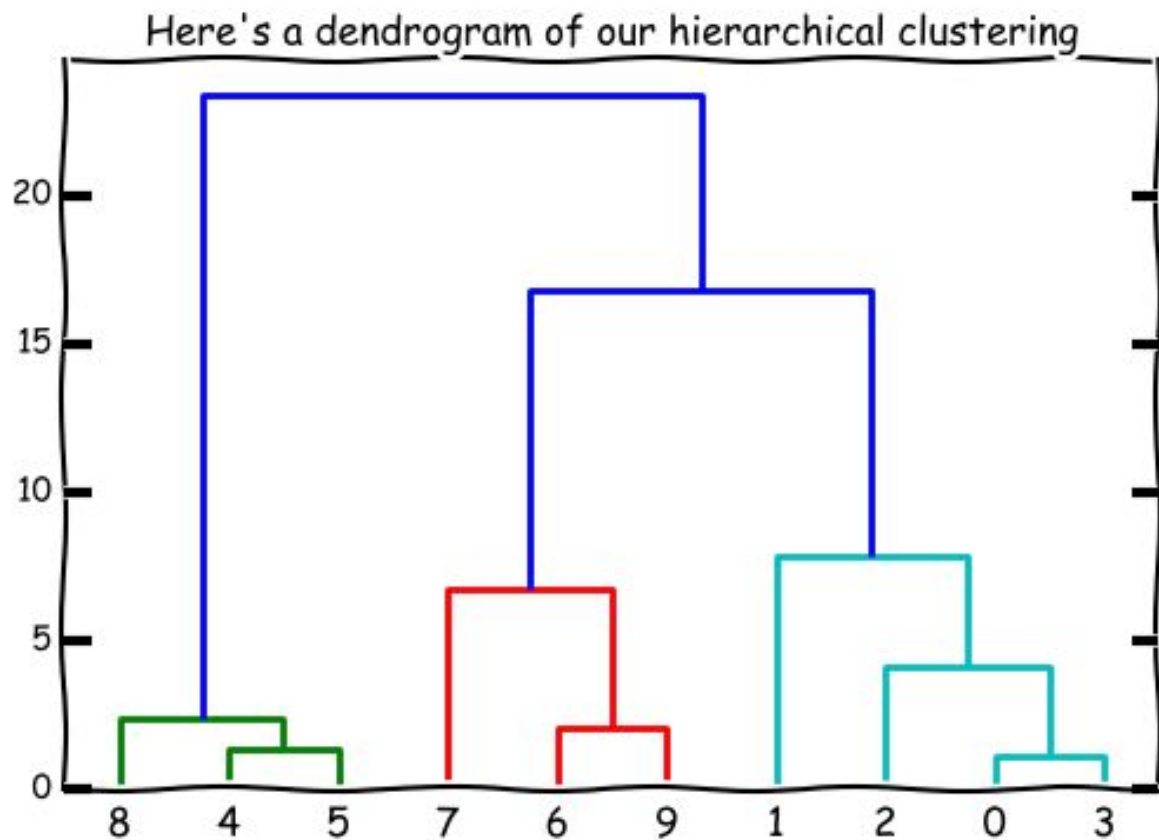
階層式分析

- 不需指定分群的數量
 1. 每筆資料視為獨立一群
 2. 計算每兩群之間的**距離**
 3. 將最近的兩群合併成一群
 4. 重複 2,3 直到所有資料合併為同一群為止
- 計算距離的方式有
 - **'complete'**: cluster 中, 最遠兩點的距離
 - **'single'**: cluster 中, 最近兩點的距離
 - **'average'**: cluster 中, 所有點的距離平均



階層分析後的樹狀圖 (dendrogram)

- 可定義 4, 5 是一群, 或 8, 4, 5 是一群, 端看距離怎麼衡量



練習

- 請參考 session3 中的 hierarchical_clustering_example, 試著理解 code



補充閱讀

- [PCA](#)
- [Hierarchical](#)





Kaggle 實戰



台灣人工智慧學校



Kaggle 實戰



台灣人工智慧學校



台灣人工智慧學校

Kaggle 介紹

- Kaggle 為一個全球性的資料科學競賽網站，任何人都可以上傳數據資料來舉辦比賽，很多公司會發布一些接近真實業務的問題，並提供獎金來吸引愛好數據科學的人來一起解決問題
- 有些公司甚至會用 Kaggle 上的排名來評估應徵者



請完成 scikit-learn-practice 比賽

1. scikit-learn-practice

- 這是單純讓大家熟悉 Scikit-learn 的比賽。總共有一千筆訓練資料、40個 features, 簡單的二元分類問題
- 資料並沒有提供 features 的意義, 純粹是讓大家練習 features scaling、建模、調參數等步驟
- 每天最多上傳 10 次結果
- 請在 private / public leaderboard 上取得 0.87 以上的準確度。達標代表你對 Scikit-learn 的操作有一定水準囉！






請完成 House Prices 比賽



2. House Prices: Advanced Regression Techniques

- 總共有 76 個 features, features 的說明[請點此](#)
- 每天最多上傳 5 次結果
- 請將自己最佳的結果截圖, 上傳至[登錄表單](#), 上傳期限為 2/9 (五) 17:40
- 本次比賽不會列入成績也不影響證書領取, 請大家踴躍練習, 互相交流機器學習技巧:)



資料已經幫各位整理好

 ..	seconds ago
<input type="checkbox"/>  data-science-london-scikit-learn	2 hours ago
<input type="checkbox"/>  house_prices_advanced_regression_techniques	2 minutes ago

 ..
<input type="checkbox"/>  test.csv
<input type="checkbox"/>  train.csv
<input type="checkbox"/>  trainLabels.csv



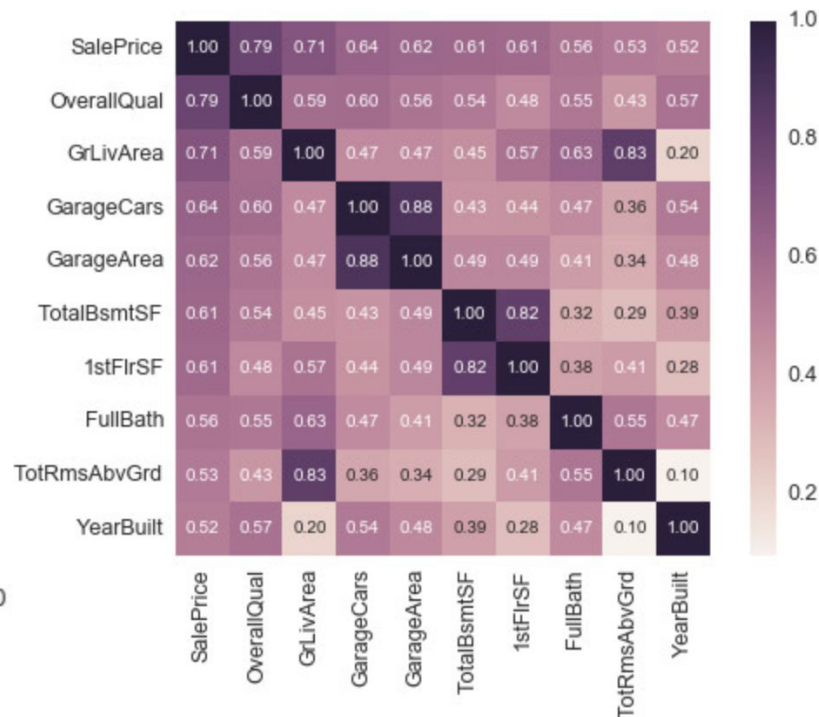
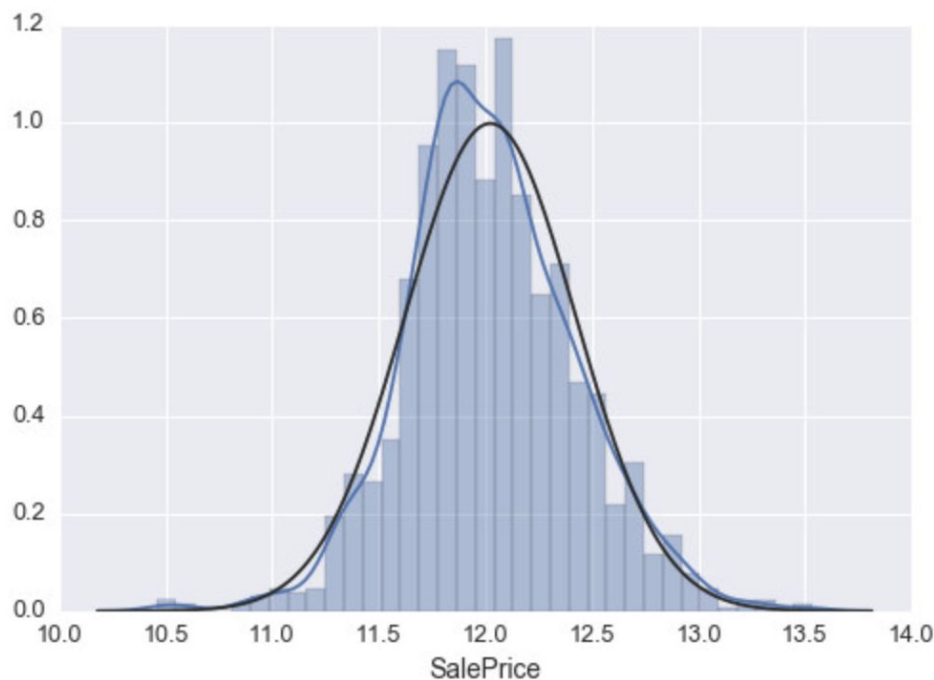
資料清洗與轉換

- 注意是否有遺失值 (missing value), 該用什麼方式填補
- 注意是否有 outliers。實務上依照不同 task, outliers 的定義不同, 須自行判斷
- 一些右偏分佈的 feature, 可透過取 log 將其轉為常態分佈
- 類別的變數可透過 one-hot encoding 轉為數字
- 類別太多可試著將相近的類別歸為一類
- 如果會用到一些算法像是 PCA, regression, 要記得將資料進行標準化 (normalization)



探索式資料分析 (EDA)

- 繪製變數的分佈圖 (hist)、盒鬚圖 (box)、相關係數圖等等



特徵工程 (feature engineering)

- 特徵建立是一門藝術，依靠創意＋經驗＋專家領域知識，沒有標準答案。能找到最關鍵的 feature，即使用普通的模型，也能榜上有名
- 常見的做法：feature 之間的交互關係，例如相乘、相除、取 log、平方、三次方等等



在開始比賽前...

- 以下約略是整個資料分析競賽的流程
 - 資料清洗與轉換
 - 探索式資料分析 (EDA)
 - 特徵工程 (feature engineering)
 - 建立模型
 - 調整參數
 - 上傳結果
- 以下替各位做簡單的整理



建立模型

- 每個模型都可以嘗試看看，但建議先從簡單的模型開始！
e.g. linear regression，並把結果當成 baseline 參考
- 後續的模型如果結果有比 baseline 還差，就要注意是否參數有問題，甚至 code 是否寫錯



調整參數

- 強大的模型伴隨著許多參數要調控，但如果一直使用固定的資料來進行調參，就有可能發生 Overfitting 的情形
- 善用 cross-validation + grid search 來尋找最好的參數，再使用這個參數，重新訓練你的模型，細節請參考[正確調參數的方式](#)



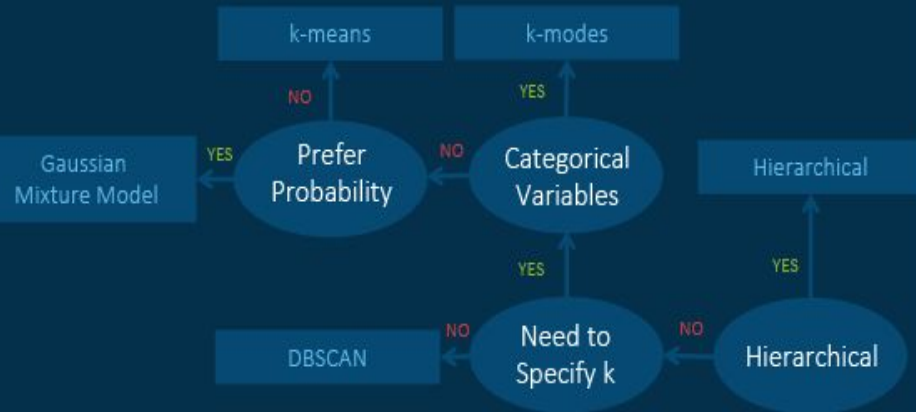
常見問題

- 別人的模型比較好？那是因為別人用了『Ensemble』！把 RadomForest + XGBoost + GradientBoosting 的結果全部合併起來，通常會再提昇一些些 performance
- 若是模型在自己切的 testing data 表現很好，但是上傳後 publicboard 的分數卻很低，那很有可能是 Overfitting
- Data 數量較少時，請務必使用 cross-validation 評估結果
- 分類問題如果遇到 data imbalance，可嘗試使用 oversampling 或 undersampling 的方式改善，可參考[連結](#)



Machine Learning Algorithms Cheat Sheet

Unsupervised Learning: Clustering

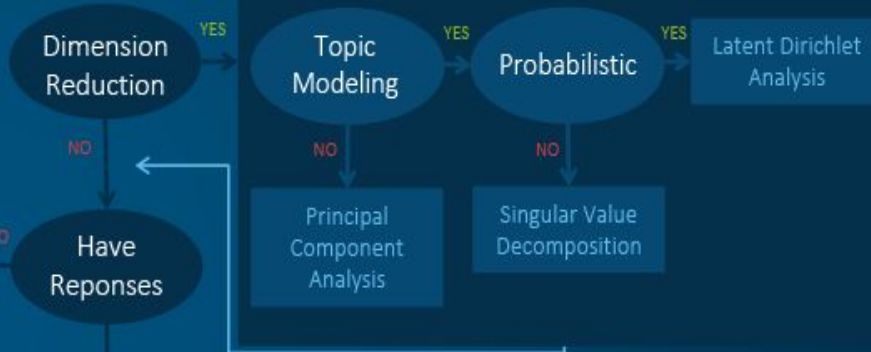


Supervised Learning: Classification



START

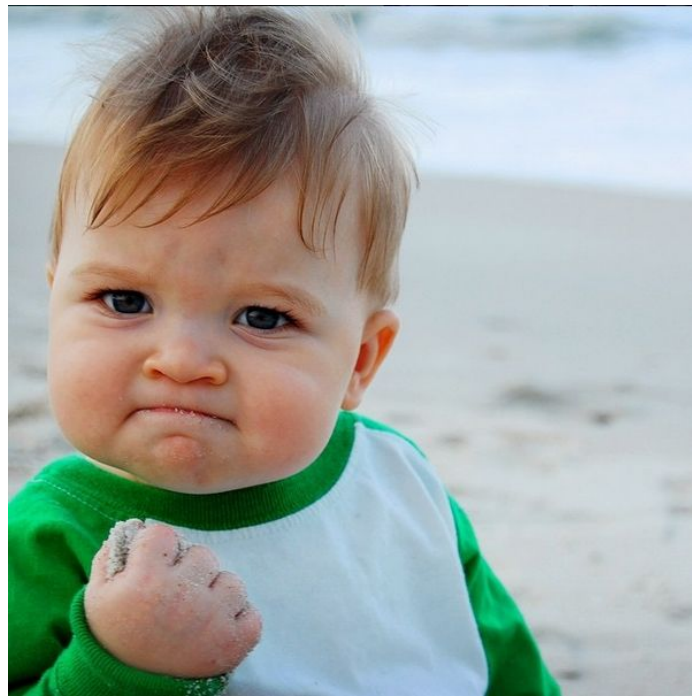
Unsupervised Learning: Dimension Reduction



Supervised Learning: Regression



祝各位 Kaggle 愉快!



小故事: Kaggle 3百萬獎金的比賽

- 預測病人待在醫院的天數。第一名與第二名的誤差只差了0.001

<https://www.kaggle.com/c/hhp#milestone-winners>



AIA_TA_TEAM in 趨勢 T-brain 競賽

- 僅上傳 3 次就得到 0.939, 排名 7 / 198
- 徵求高手與我們一同參加比賽, 為 AI 學校的尊嚴而戰



#	隊伍名稱	成員	提交次數	分數	上傳時間
1	Northern Light	1	24	0.956503	2/3/2018 12:48:46 AM
2	test	1	12	0.953544	2/7/2018 10:14:08 AM
3	NorwegianWood	2	27	0.946333	2/7/2018 10:07:01 PM
4	SunsetKiwi	2	9	0.945860	2/8/2018 12:01:35 AM
5	Vito	4	12	0.943091	2/1/2018 9:42:55 AM
6	BLJ	2	10	0.940768	1/26/2018 1:19:29 PM
7	AIA_TA_TEAM	4	3	0.939923	2/8/2018 2:23:44 AM

