Decision Tree

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Today

- Decision Tree for Classification
- Quick Hands-on: Decision Tree on IRIS dataset
- Interpreting decision trees
- Splitting Attributes
- Splitting Criteria (Gini Impurity, Entropy)
- Finding class probabilities
- Training algorithm: CART algorithm
- Intro to Ensemble Learning (Random Forest)
- Homework

SVM Math (whiteboard)

Recap: Supervised Learning

Decision Tree

Neural Net

kNN

Logistic Regression

SVM

Classification

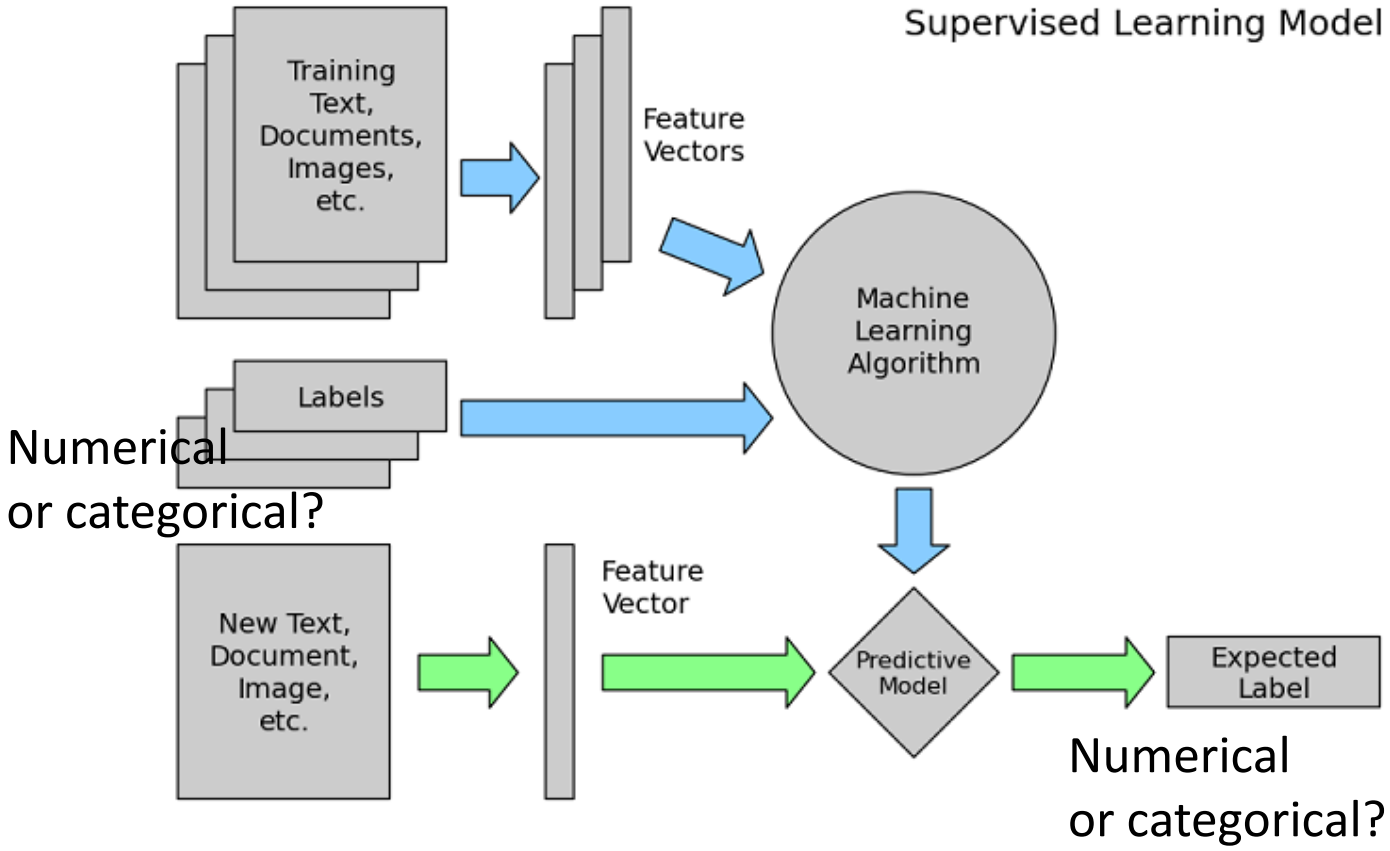
- Predicts class labels/categories
- ทำนายค่าที่เป็นหมวดหมู่ = จำแนกประเภท
- อาจมองเป็นการหา **boundary** ที่แบ่งข้อมูลในแต่ละหมวดหมู่ ออกจากกัน

Regression

- Predicts continuous values
- ทำนายค่าที่เป็นจำนวนจริง
- อาจมองเป็นการหา **hyperplane** ที่ **fit** กับข้อมูลที่มีมากที่สุด

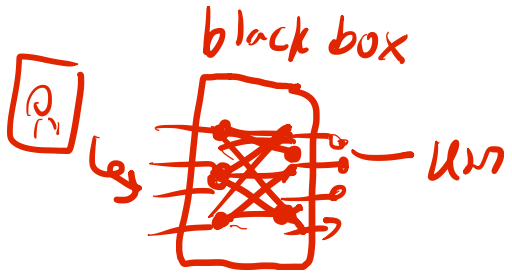
Linear Regression

k-NN
SVR
decision tree



Decision Tree

- เป็น supervised learning
  - ใช้ทำ classification เป็นหลัก แต่ทำ regression ก็ได้
  - เป็นวิธีแบบ non-parametric ไม่มีโมเดล/สมมติฐานเกี่ยวกับข้อมูล



- ข้อดี
  - เข้าใจง่าย – สามารถตีความและอธิบายที่มาที่ไปของผลการ predict ได้ (white box method) -- ต่างจาก black box model อย่าง Neural Network
  - ใช้กับข้อมูลที่เป็น numerical หรือ categorical ก็ได้
  - สามารถประมาณค่าความน่าจะเป็นที่จะเป็นแต่ละคลาส (class probability) ได้
  - สามารถ fit ข้อมูลที่มีความสัมพันธ์แบบ non-linear ที่มีความซับซ้อนมากได้

หมา 0.8
หมา 0.1
แก 0.1

- ข้อเสีย
  - อาจจะ overfit หาก decision tree มีความลึกมากไป
  - การ train decision tree เป็นปัญหา NP-complete ไม่สามารถหาคำตอบที่ดีที่สุดใน poly time ได้ จึงต้องใช้อัลกอริทึมแบบ heuristic เช่น greedy ในการหาคำตอบแบบ local optima

Quick Hands-on: Decision Tree on IRIS dataset

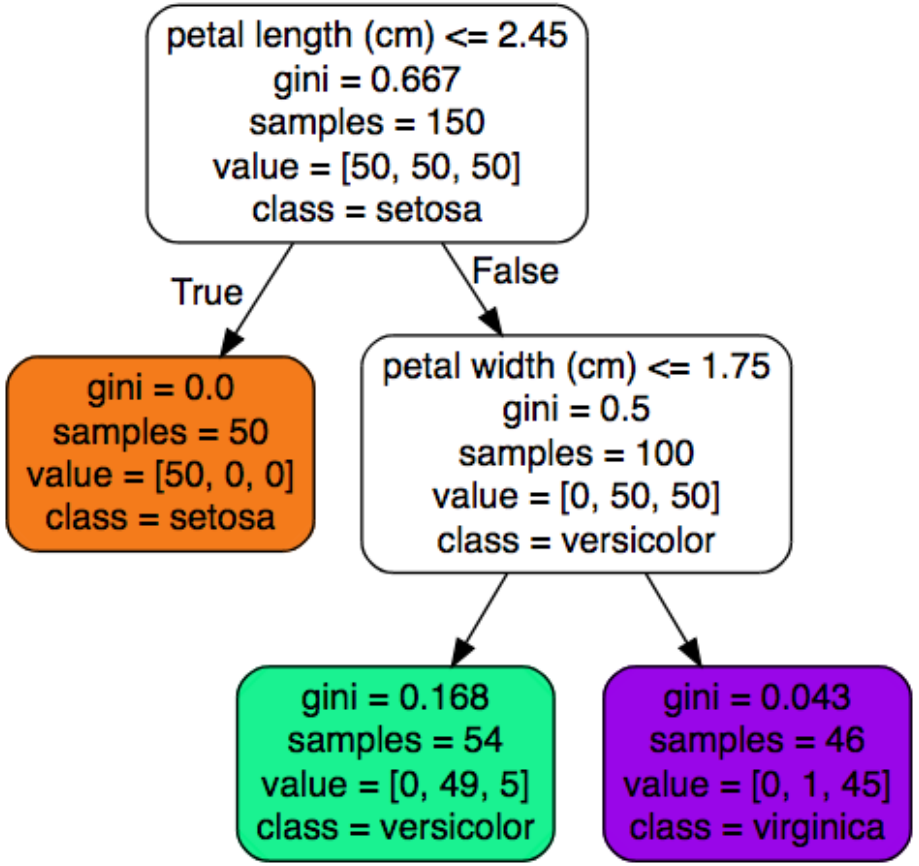
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
import numpy as np

iris = load_iris()
X = iris.data[:, 2:]        # feature: petal length/width
y = iris.target             # label: 0, 1, 2
np.random.seed(42)
tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
```

Visualizing the Decision Tree

```
from sklearn.tree import export_graphviz

tree_dot = export_graphviz(
    tree_clf,
    out_file=None,   # or out_file="iris_tree.dot"
    feature_names=iris.feature_names[2:],
    class_names=iris.target_names,
    rounded=True,
    filled=True
)
print(tree_dot)
```



จากนั้นนำผลลัพธ์ไปแปลงเป็นแผนภาพได้ที่ http://www.webgraphviz.com/

http://scikit-learn.org/stable/modules/tree.html

Alternative: using python-graphviz

Once trained, we can export the tree in Graphviz format using the `export_graphviz` exporter. If you use the conda package manager, the graphviz binaries and the python package can be installed with

```
conda install python-graphviz
```

Alternatively binaries for graphviz can be downloaded from the graphviz project homepage, and the Python wrapper installed from pypi with pip install graphviz.

Below is an example graphviz export of the above tree trained on the entire iris dataset; the results are saved in an output file iris.pdf:

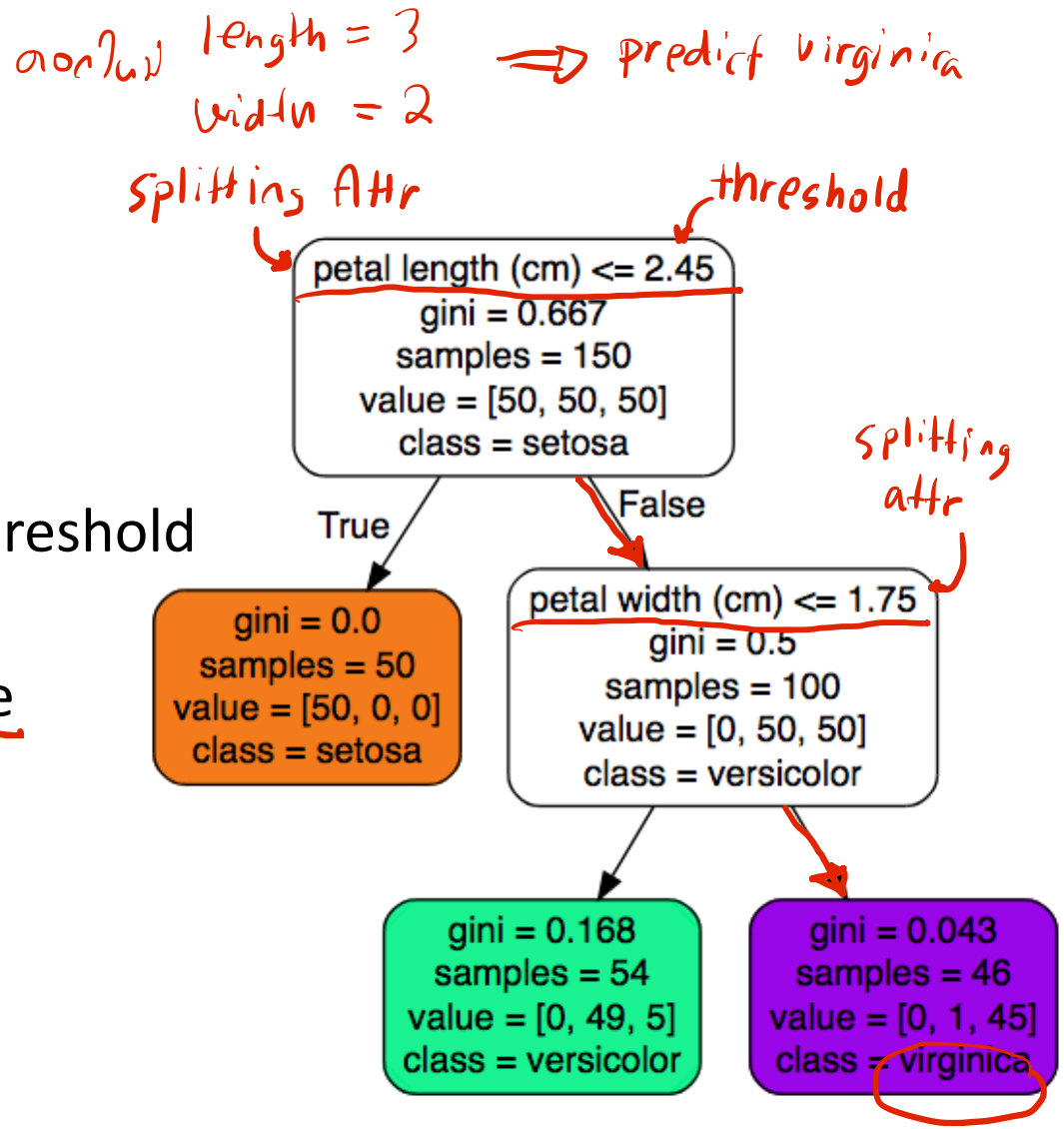
```
>>> import graphviz
>>> dot_data = tree.export_graphviz(clf, out_file=None)
>>> graph = graphviz.Source(dot_data)
>>> graph.render("iris")
```

The `export_graphviz` exporter also supports a variety of aesthetic options, including coloring nodes by their class (or value for regression) and using explicit variable and class names if desired. Jupyter notebooks also render these plots inline automatically:

```
>>> 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
```

การตีความภาพ decision tree

- ในการ **predict** จะเริ่มจาก **node** บนสุด **(root)** ลงล่าง
- **node** ขาว คือ การเช็คเงื่อนไข
  - เช็คว่า feature x_i ของดอกนั้นมีค่า **มาก** หรือ **น้อยกว่า** ค่า threshold
  - เช่น petal length <= 2.45 ?
  - ในที่นี้ เราเรียก petal length ว่าเป็น splitting attribute
- **node** สี คือ สรุปได้ว่าจะ **predict** เป็นคลาสใด
  - ส้ม = setosa
  - เขียว = versicolor
  - ม่วง = virginica

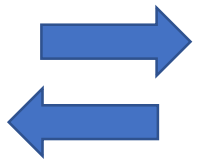


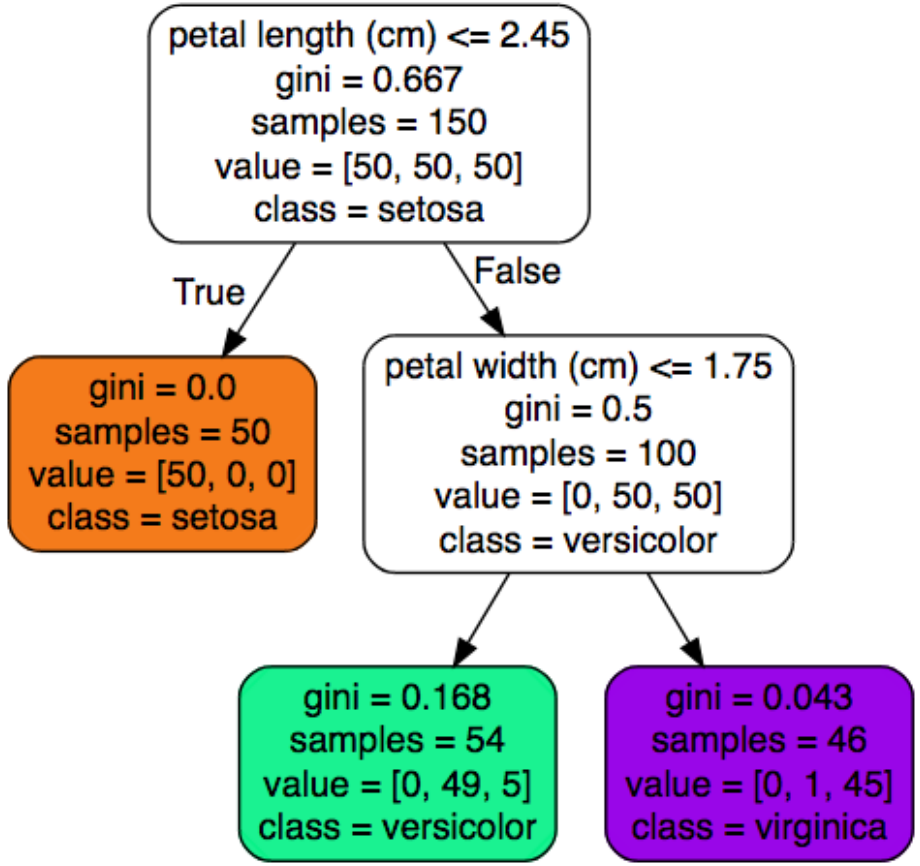
*value เก็บจำนวนข้อมูลชุดฝึกที่มีคลาสเฉลยเป็น [setosa, versicolor, virginica]

การตีความภาพ decision tree

- Decision tree เปรียบเสมือน flowchart ของกฎที่อยู่ในรูป if-else ดังนี้:

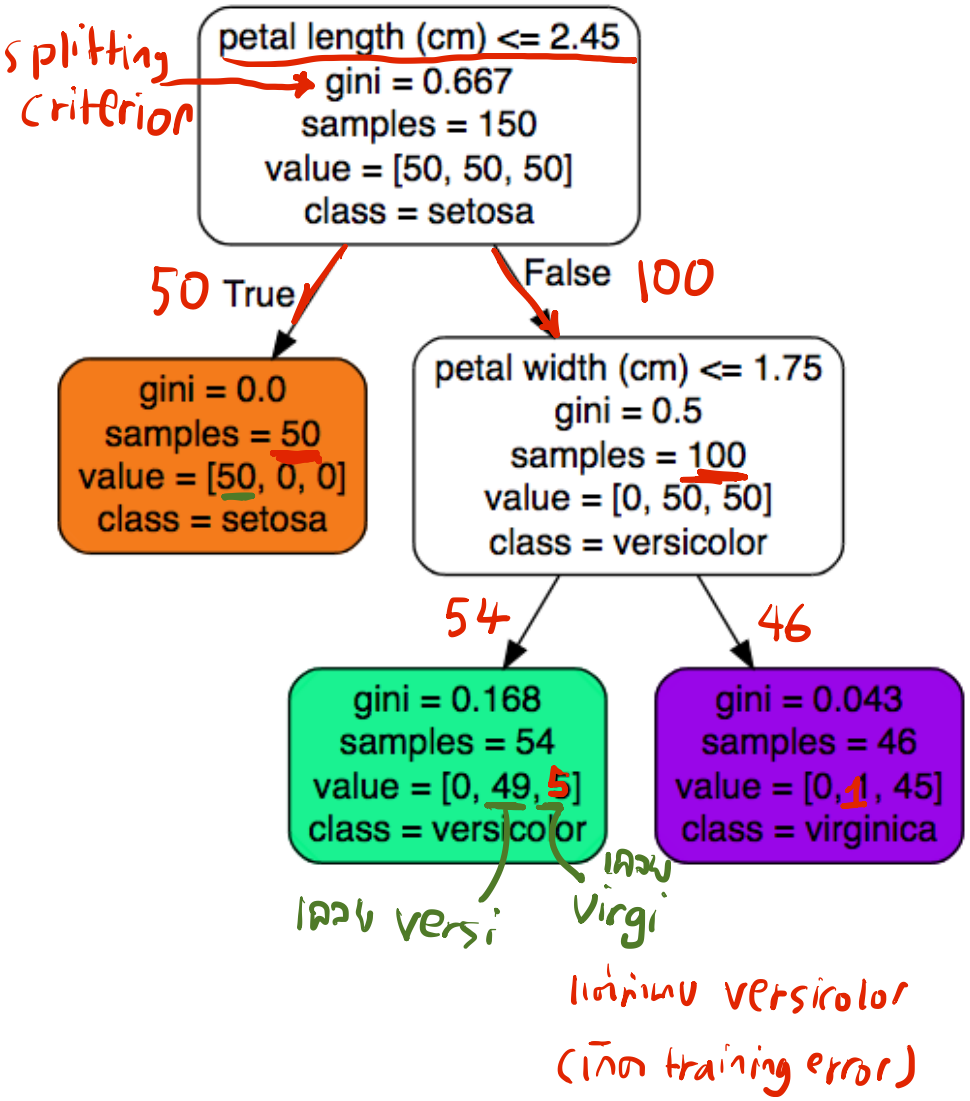
```
if petal_length <= 2.45:
    class = "setosa"
elif petal_width <= 1.75:
    class = "versicolor"
else:
    class = "virginica"
```





การตีความภาพ decision tree

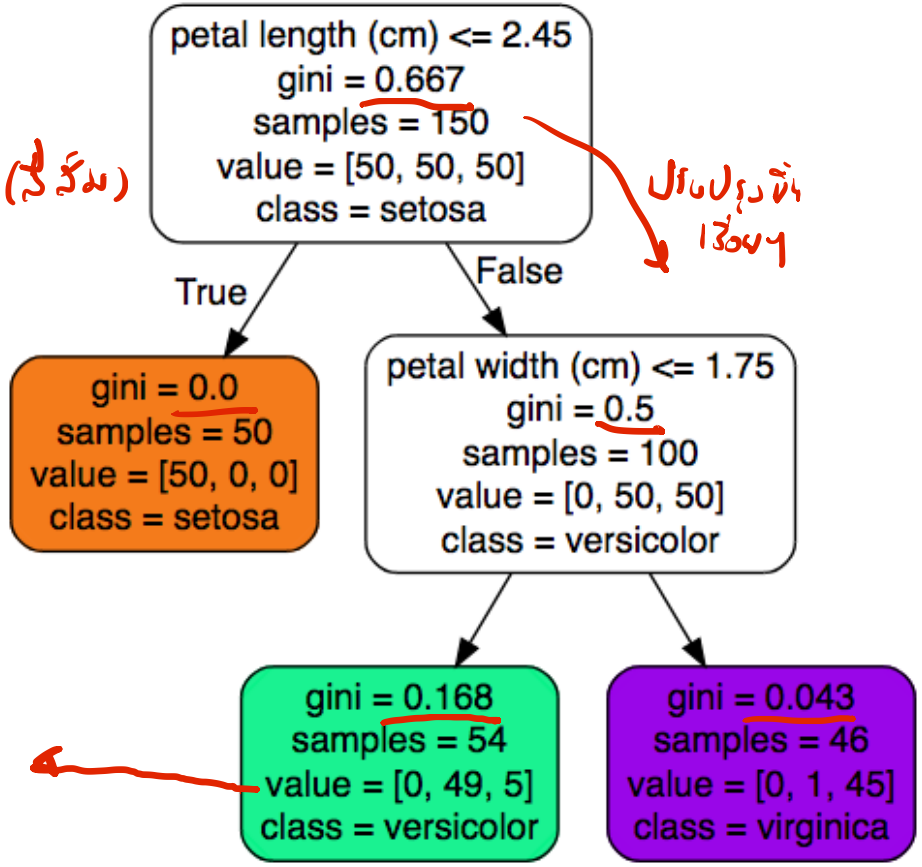
- สังเกตว่าในแต่ละ node จะบอก
  - samples คือ จำนวนข้อมูลชุด train ที่ถูกจำแนกถึงจุดนั้น
  - value คือ จำนวน samples แบ่งตามคลาส
    - [setosa, versicolor, virginica] = [50,50,50]
  - splitting criterion คือ ค่าที่ใช้เป็นเกณฑ์ในการดูว่า node นี้สามารถแบ่งแยกคลาสข้อมูลได้ดีแค่ไหน
    - ในตัวอย่างใช้ค่า Gini Impurity
    - ยิ่งน้อย ยิ่งแสดงว่าแบ่งแยกได้ดี



การตีความภาพ decision tree

Sample = 60
value = [20, 20, 20]
$gini = 1 - \left(\frac{20}{60}\right)^2 \times 3 = 1 - 3\left(\frac{1}{3}\right)^2 = 1 - \frac{1}{3} = 0.667$

- gini คือค่าความปะปน (Impurity) ของ node
  - gini = 0.0 แสดงว่าข้อมูล train ใน node นั้นเป็นคลาสเดียวกันหมด (มีสีเดียว)
  - gini มาก แสดงว่า node มีข้อมูลหลายคลาสปะปนกัน



จำนวนคลาส = 3

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

- โดยที่ $p_{i,k}$ คืออัตราส่วนของข้อมูลคลาส k ต่อข้อมูลทั้งหมด ณ node ที่ i

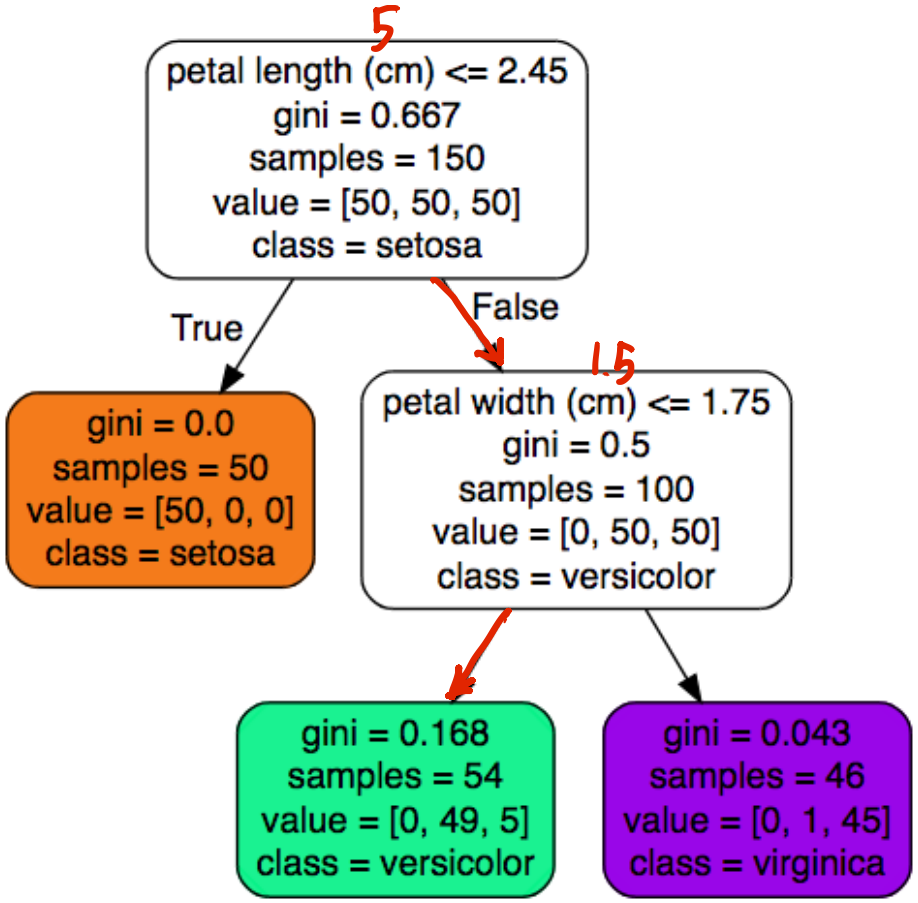
- เช่น G(โหนดเขียว) $= 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 = 0.168$

k=1 setosa k=2 versicolor k=3 virginica

*note that value = [setosa, versicolor, virginica]

การจำแนกด้วย decision tree

- ตัวอย่าง: จงใช้ decision tree นี้ classify ดอกกล้วยไม้ที่มีความยาวกลีบ 5cm และมีความกว้างกลีบ 1.5cm

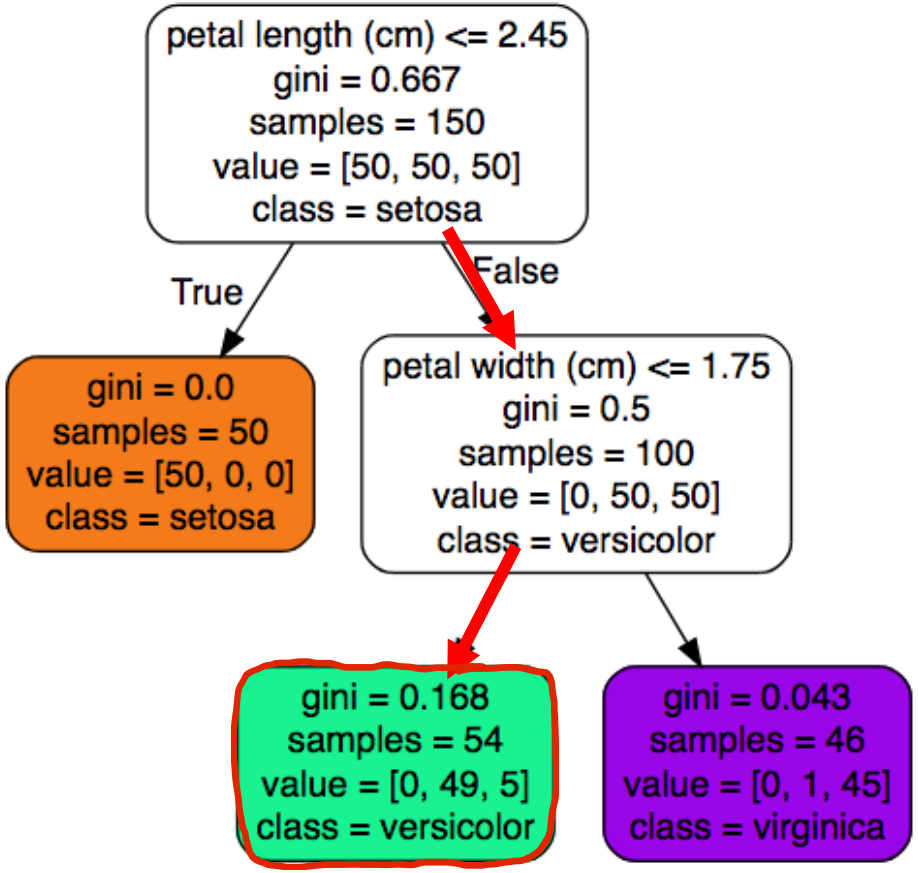


การจำแนกด้วย decision tree

- ตัวอย่าง: จงใช้ decision tree นี้ classify ดอกกล้วยไม้ที่มีความยาวกลีบ 5cm และมีความกว้างกลีบ 1.5cm

- คำตอบ: predict class="versicolor"

- Q: จงหา class probability ต่อไปนี้
  - ความน่าจะเป็นที่ดอกนี้จะเป็น setosa?
  - ความน่าจะเป็นที่ดอกนี้จะเป็น virginica?



การจำแนกด้วย decision tree

ทำให้ decision tree ทำได้เหมือน logistic regression

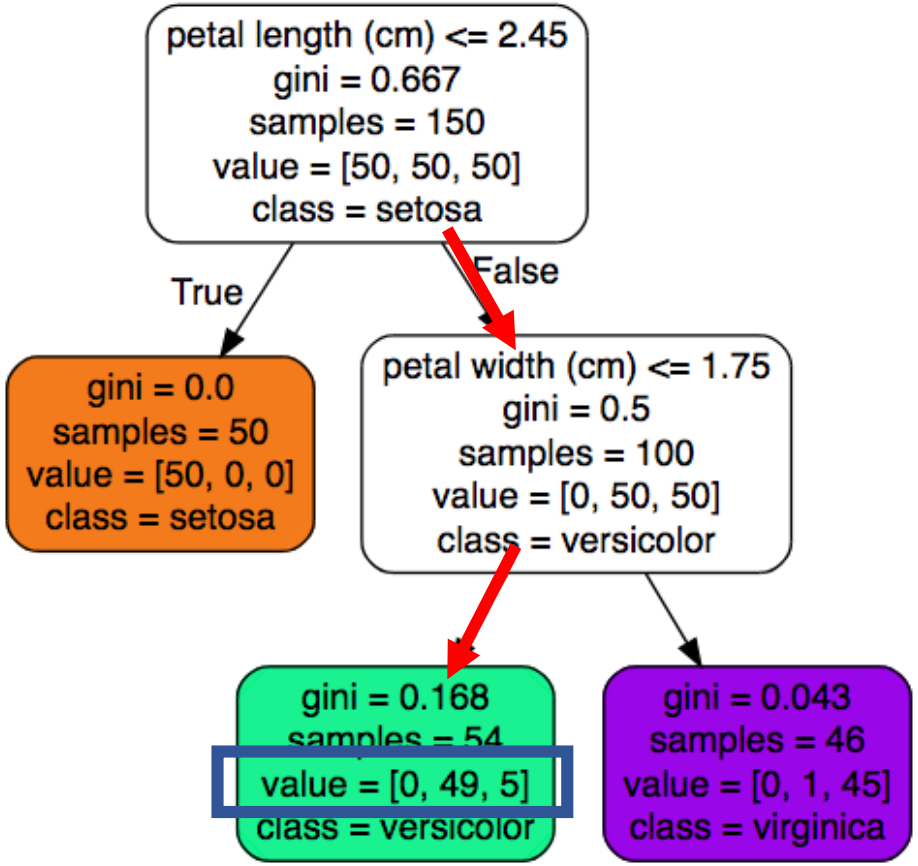
- ตัวอย่าง: จงใช้ decision tree นี้ classify ดอกกล้วยไม้ที่มีความยากลีบ 5cm และมีความกว้างกลีบ 1.5cm

- คำตอบ: predict class="versicolor"

- Q: จงหา class probability ต่อไปนี้
  - ความน่าจะเป็นที่ดอกนี้จะเป็น setosa?
  - P(setosa) = 0/54 = 0%

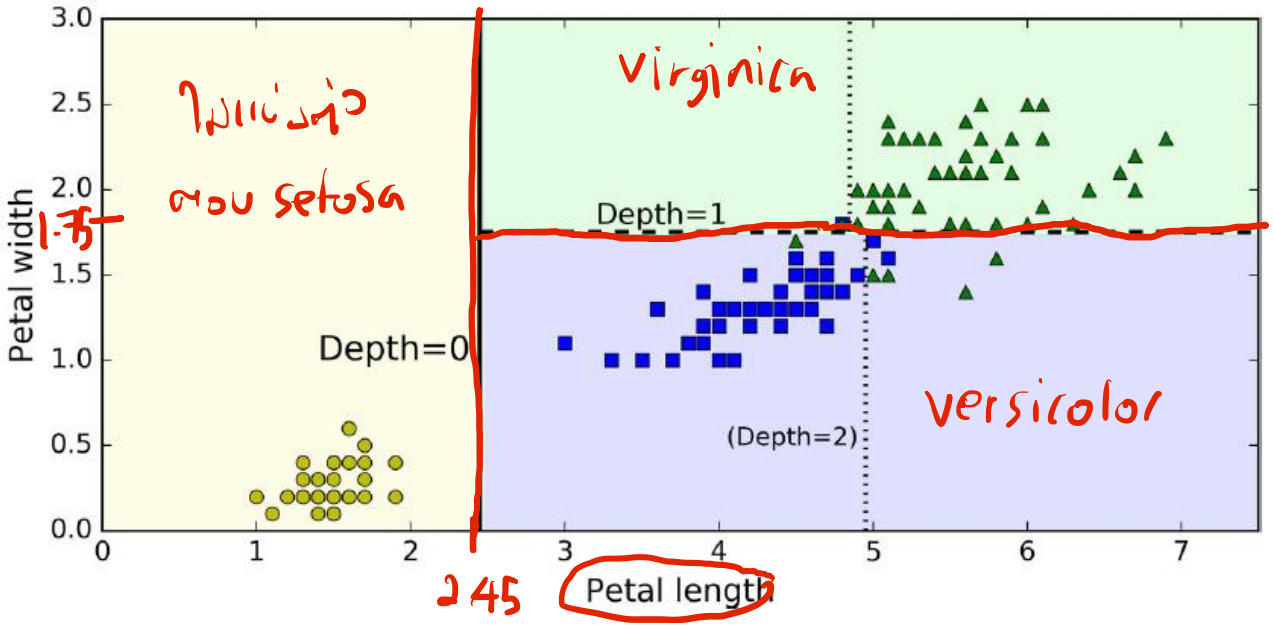
  - ความน่าจะเป็นที่ดอกนี้จะเป็น virginica?
  - P(virginica) = 5/54 = 9.3%

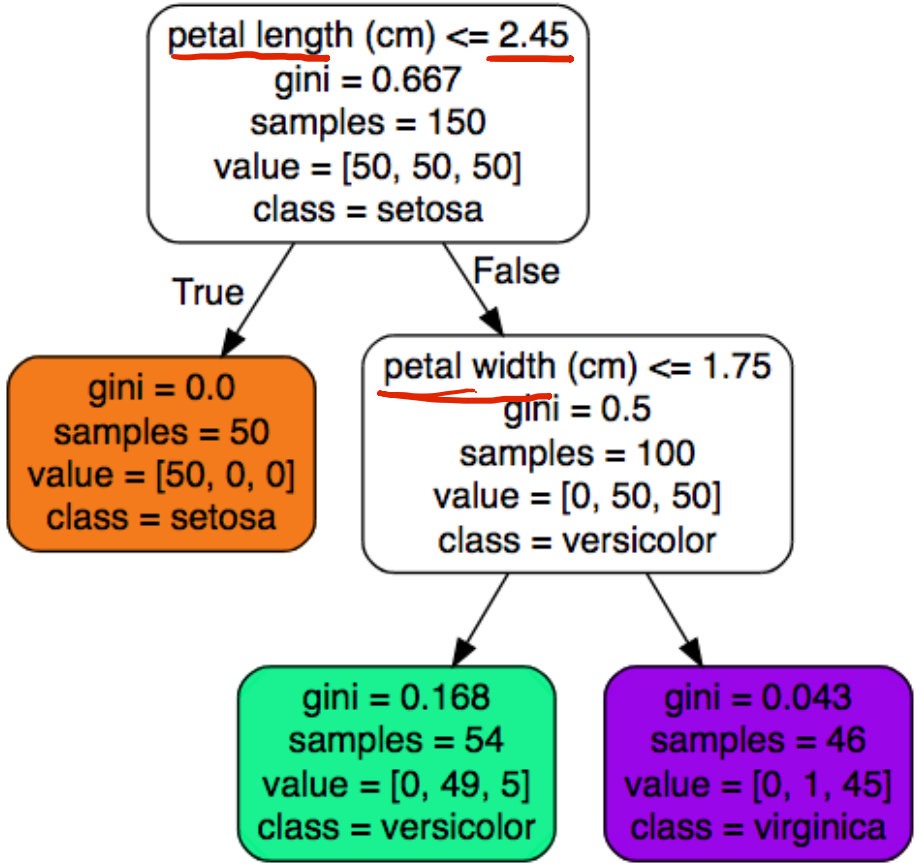
  - P(versicolor) = 49/54 = 90.7%



Decision Boundary

- เราสามารถมองว่าแต่ละ **level/depth** ในต้นไม้นี้ เป็นการสร้าง **decision boundary** ที่ตัดแบ่ง **space** ของข้อมูลชุด **train** ออกเป็น 2 ส่วน ด้วยเส้นตั้งฉากดังสมการ
  - splitting attribute = threshold value



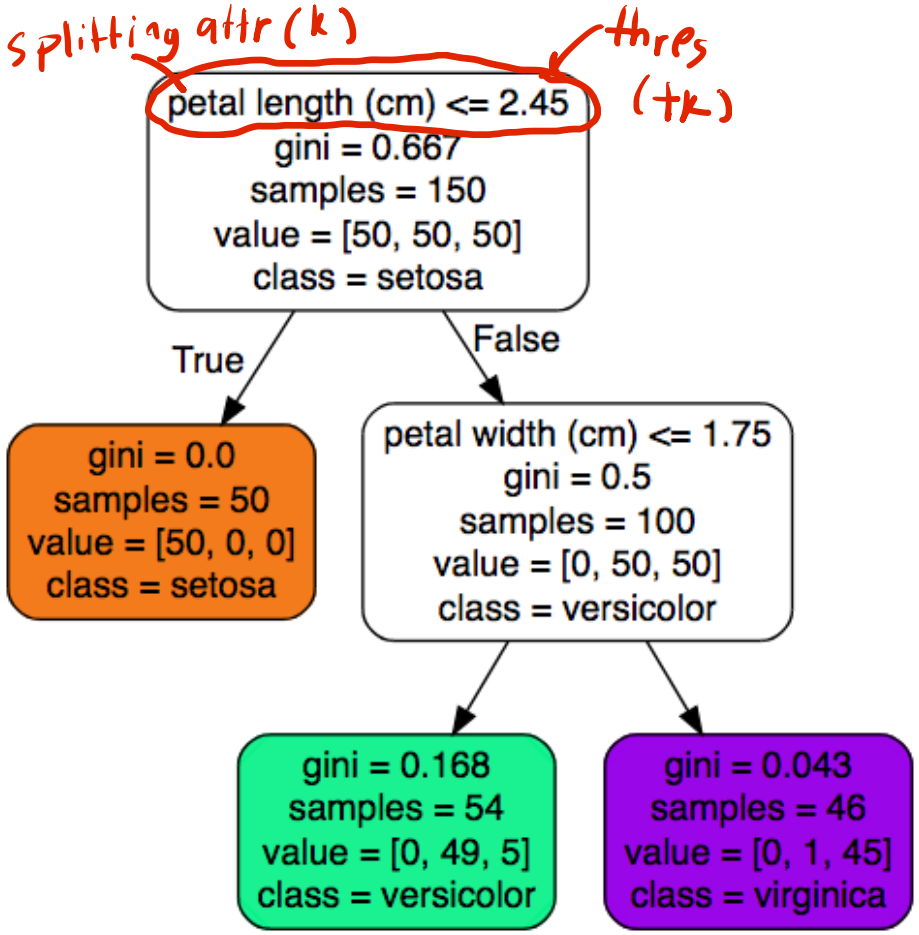


[Hands-On Machine Learning with Scikit-Learn and TensorFlow, Géron]

CART Algorithm สำหรับสร้าง decision tree

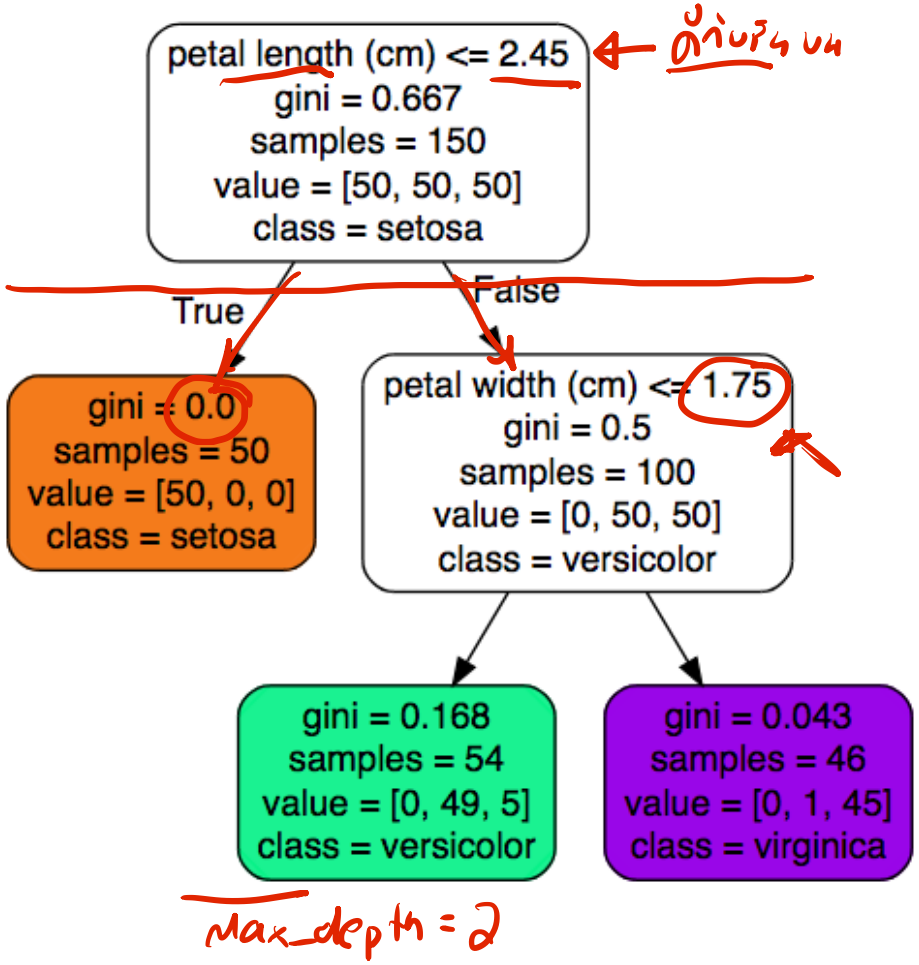
- การสร้าง **decision tree** จะเริ่มจาก **root node**
- ในแต่ละ **level** เราจะเลือก **(k, tk)** ที่ดีที่สุดมาเป็นตัว **split**
  - attribute k
  - threshold tk
  - กฎ: k <= tk
- จะเลือก **(k, tk)** ที่ดีที่สุดได้อย่างไร?

(petal length, 2.45)
(petal width, 0.8)



CART Algorithm สำหรับสร้าง decision tree

- จะเลือก (k, tk) ที่ดีที่สุดได้อย่างไร?
- เป้าหมาย: ต้องการ (k, tk) ที่ทำให้ subset ข้อมูลหลังการ split บริสุทธิ์ (pure) ที่สุด
  - = Gini ต่ำที่สุด
- ดังนั้น Cost function ที่เราต้องการ minimize คือ:
  - $J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$
- ในแต่ละชั้นต้นไม้ อัลกอริทึม CART จะหา (k, tk) ที่ทำให้ J ต่ำสุด



*ปัญหาการหา optimal tree เป็น NP-complete ไม่สามารถหาใน poly time ได้
จึงต้องใช้วิธีแบบ greedy คือหา (k, tk) ที่ทำให้ cost น้อยสุด ทีละชั้น แม้ว่าลงไปชั้นล่างๆ แล้วอาจทำให้ cost กลับมาสูงขึ้นได้ก็ตาม

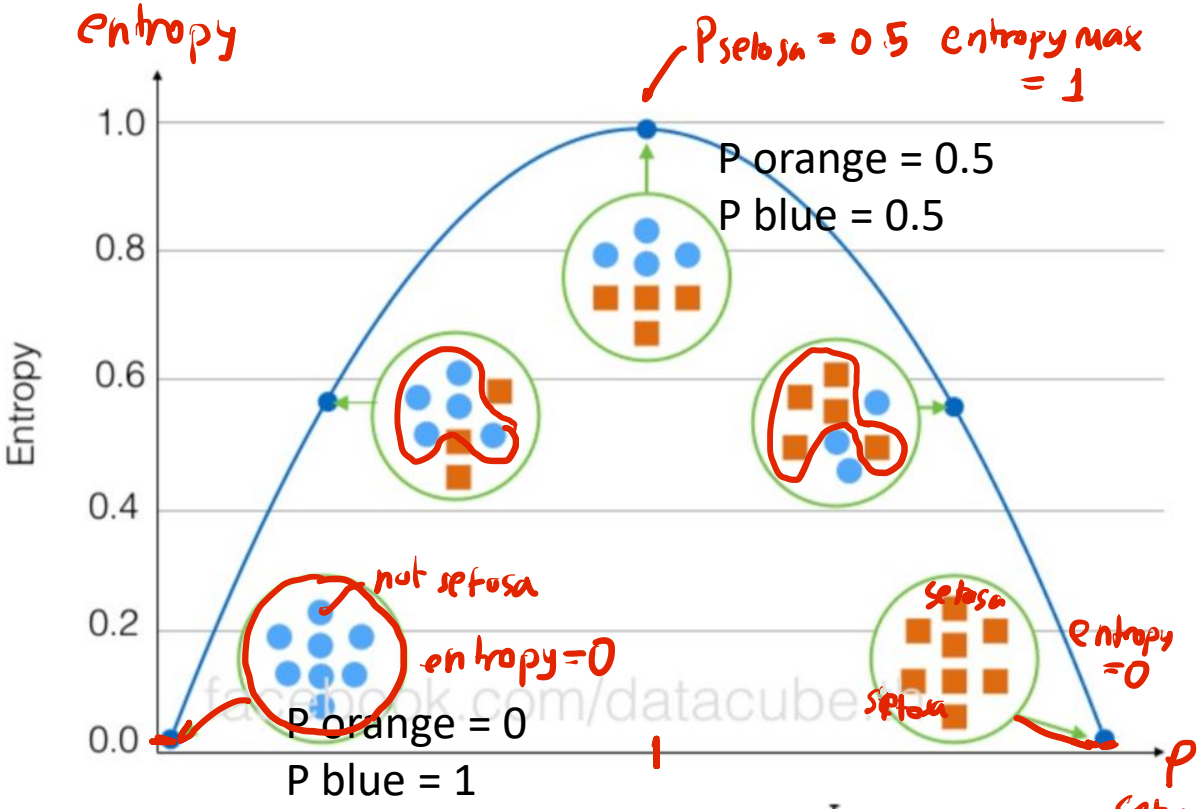
Choosing Splitting Criteria

$gini = 1 - \Sigma p$

$entropy = -\Sigma p \log p$

- นอกจาก **Gini Impurity** แล้ว ยังสามารถใช้ **Entropy/Information Gain** เป็นเกณฑ์ในการวัดประสิทธิภาพของ **splitting attribute** ได้ด้วย
  - ID3 algorithm

- **splitting criterion** ถือเป็น **hyperparameter** หนึ่งของ **decision tree**



$$H(T) = I_E(p_1, p_2, \ldots, p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

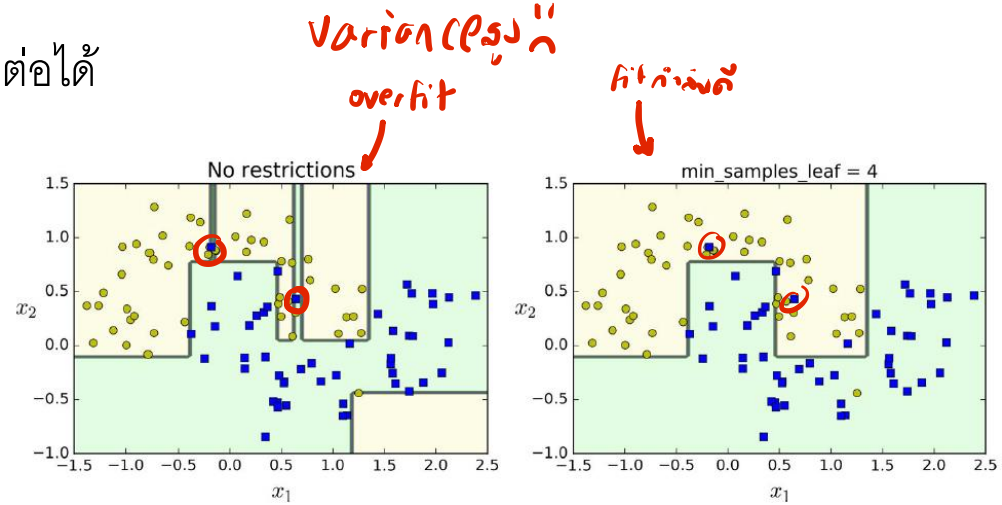
$$\overbrace{IG(T,a)}^{\text{Information Gain}} = \overbrace{H(T)}^{\text{Entropy(parent)}} - \overbrace{H(T|a)}^{\text{Weighted Sum of Entropy(Children)}}$$

[Hands-On Machine Learning with Scikit-Learn and TensorFlow, Géron]

Overfitting Problem

- Decision tree เป็นวิธีแบบ non-parametric (ต่างจาก logistic regression, SVM)
- ไม่มี model หรือ assumption เกี่ยวกับข้อมูล ทำให้มีความอิสระ/ยืดหยุ่นสูงมาก
- **Problem:** หากปล่อยให้ต้นไม้ใหญ่เกิน จะทำให้ overfit ข้อมูลชุดนั้นได้
- **Solution:** regularize โดยการตั้งค่า hyperparameter เพื่อจำกัด
  - max_depth ความสูงต้นไม้ห้ามเกิน
  - max_leaf_nodes จำนวน leaf node ห้ามเกิน
  - min_samples_split จำนวน sample อย่างต่ำ ถึงจะ split ต่อได้
  - min_samples_leaf จำนวน sample ขั้นต่ำที่ leaf ต้องมี

gridsearchCV หา max_depth ... ที่ดีที่สุด
test error ต่ำสุด



This week’s lab

- Ensemble Learning & Random Forest