Decision Tree + Ensemble Learning

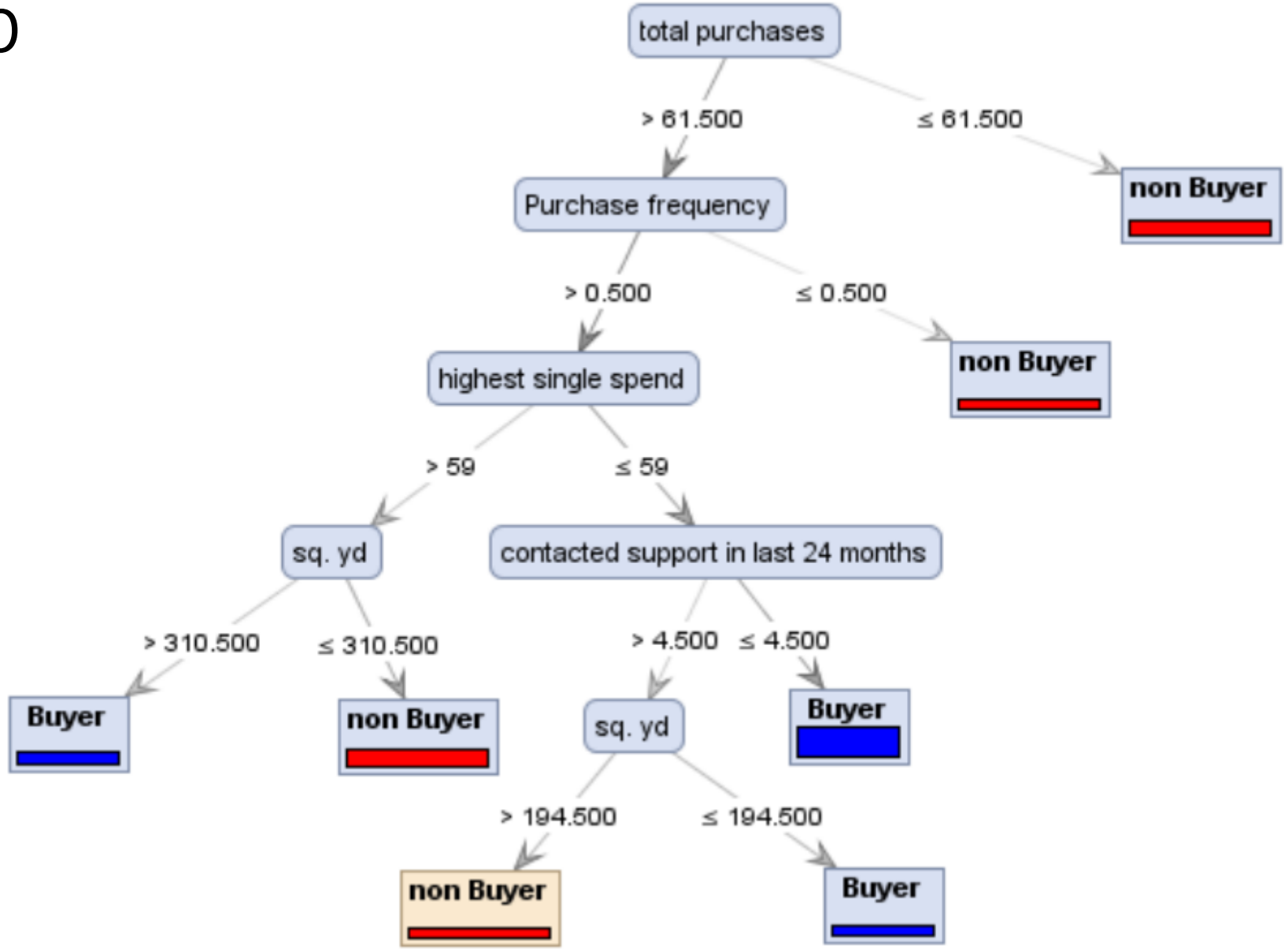
อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Today

- Recap
  - Decision Tree Classifier
  - Overfitting problem in Decision Tree
- Ensemble Learning
  - Bagging vs Boosting
- Random Forest
- Single Decision Tree vs Ensemble
- Decision Tree & Feature Importance
- Homework

Recap


total purchases
> 61.500
≤ 61.500
non Buyer
Purchase frequency
> 0.500
≤ 0.500
non Buyer
highest single spend
> 59
≤ 59
sq. yd
contacted support in last 24 months
> 310.500
≤ 310.500
> 4.500
≤ 4.500
Buyer
non Buyer
sq. yd
Buyer
> 194.500
≤ 194.500
non Buyer
Buyer

Recap

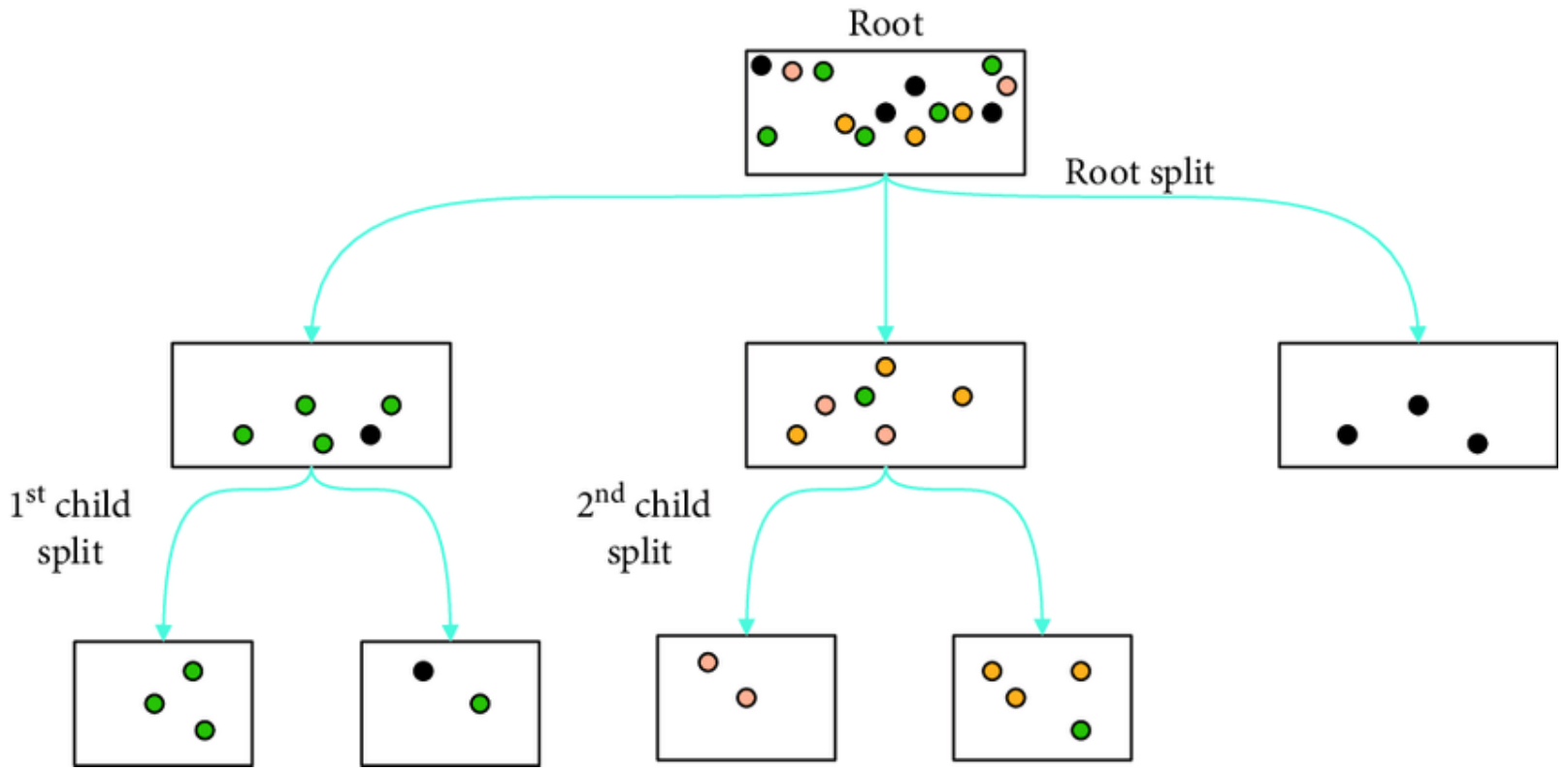
- Decision Tree ทำการ classify ข้อมูลโดยใช้คำถาม “feature <= ค่า threshold?” หลายๆ ครั้ง จนนำไปสู่คำตอบที่ leaf node

- ทุกคำถาม “feature <= ค่า threshold?” จะแบ่งข้อมูลออกเป็น 2 ฝั่ง
  - ฝั่งที่น้อยกว่า และ มากกว่า threshold
  - จึงเกิด decision boundary ที่ตั้งฉากกับแกน feature

- CART algorithm: ในการ train จะสร้าง node ทีละชั้นลงมา แต่ละชั้นจะต้องเลือกว่า
  - ใช้ feature อะไรมาเป็น splitting attribute
  - ใช้ค่า threshold เท่าไหร่

Recap

- ในการแบ่งแต่ละครั้ง จะพยายามให้ทั้ง 2 ฝั่งมีข้อมูลที่บริสุทธิ์/เป็นเอกฉันท์มากขึ้น
  - พยายามลดการปะปนของคลาส (= พยายามลด Gini Impurity หรือ Entropy)



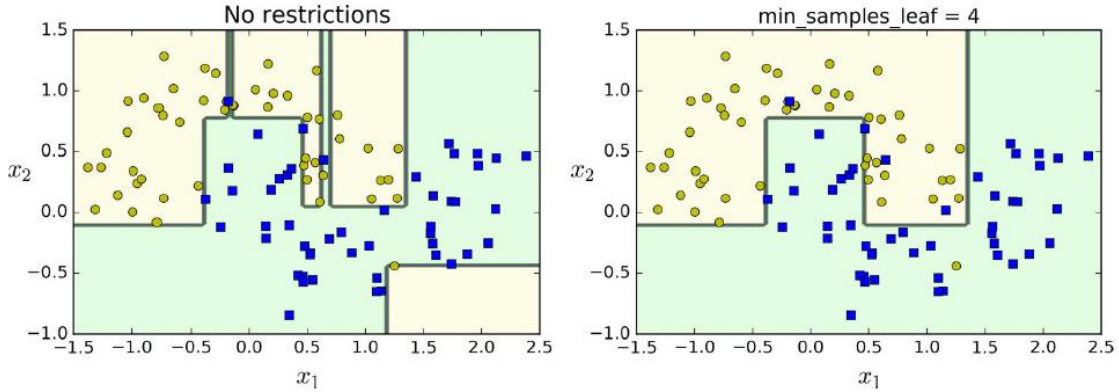
Hussain, et al. (2020). Detecting Congestive Heart Failure by Extracting Multimodal Features and Employing Machine Learning Techniques.

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

Recap: Overfitting Problem

- Decision tree มักจะ **overfit** ข้อมูลได้ง่าย ☹
  - เพราะสามารถสร้างต้นไม้ที่ลึกมากๆ เพื่อให้ตอบถูกเกือบ **100%** กับข้อมูลชุดฝึกได้
  - **Variance** สูง (ต้นไม้เปลี่ยนไปตามข้อมูลชุดฝึกมากไป)

- Solution:
  - ตั้งค่า **max_depth, max_leaf_nodes,...** เพื่อจำกัดขนาดต้นไม้
  - หรือใช้เทคนิค **Ensemble Learning (Bagging)**



Ensemble Learning

- คือการนำคำตอบจาก weak learner หลายๆ ตัวมารวมกัน (aggregate) เพื่อให้ได้คำตอบที่มีความแม่นยำขึ้น หรือเสถียรขึ้น และกลายเป็น strong learner
- ทำได้หลายวิธี เช่น
  - Voting (รวมผลหลายๆ อัลกอ เช่น SVM + kNN + Logistic Regression)
  - Bagging (ใช้อัลกอเดียว แต่ train กับข้อมูลชุดฝึกที่เลือกมาแบบสุ่มเปลี่ยนไปเรื่อยๆ)
  - Boosting (ใช้อัลกอเดียว วน train ซ้ำโดยให้น้ำหนักกับข้อมูลที่ทำนายผิดมากขึ้น)
  - Stacking (เหมือน Voting แต่เพิ่มการ train ในส่วนการรวมผล ให้รวมได้เก่งขึ้น)

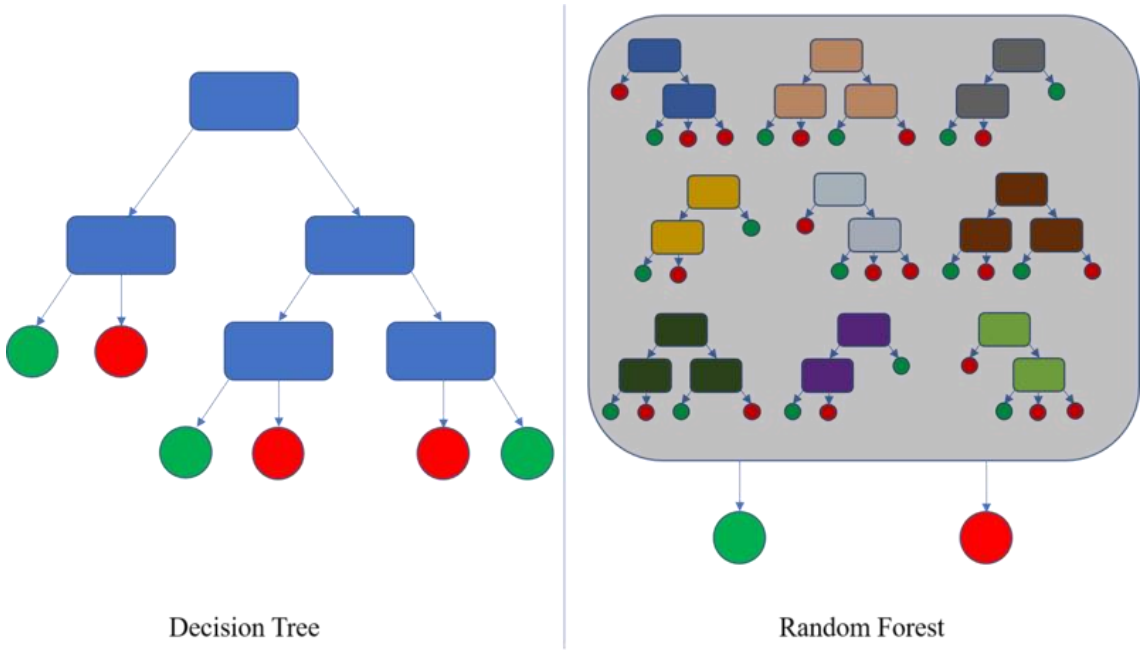
Bagging

- Bagging มาจากคำว่า Bootstrap Aggregating
- Step 1: Bootstrapping
  - คือการ train อัลกอเดียวกับข้อมูลชุดฝึกที่เลือกมาแบบสุ่มเปลี่ยนไปเรื่อยๆ และจึงทำให้เกิด predictor หลายตัว
  - สามารถ train แต่ละ predictor แบบ parallel ได้
- Step 2: Aggregating
  - คือการนำผลทำนายจาก predictor หลายตัวนั้นไปรวมเป็นคำตอบสุดท้าย
  - รวมโดยการ vote (classification) หรือหาค่าเฉลี่ย (regression)

Bagging

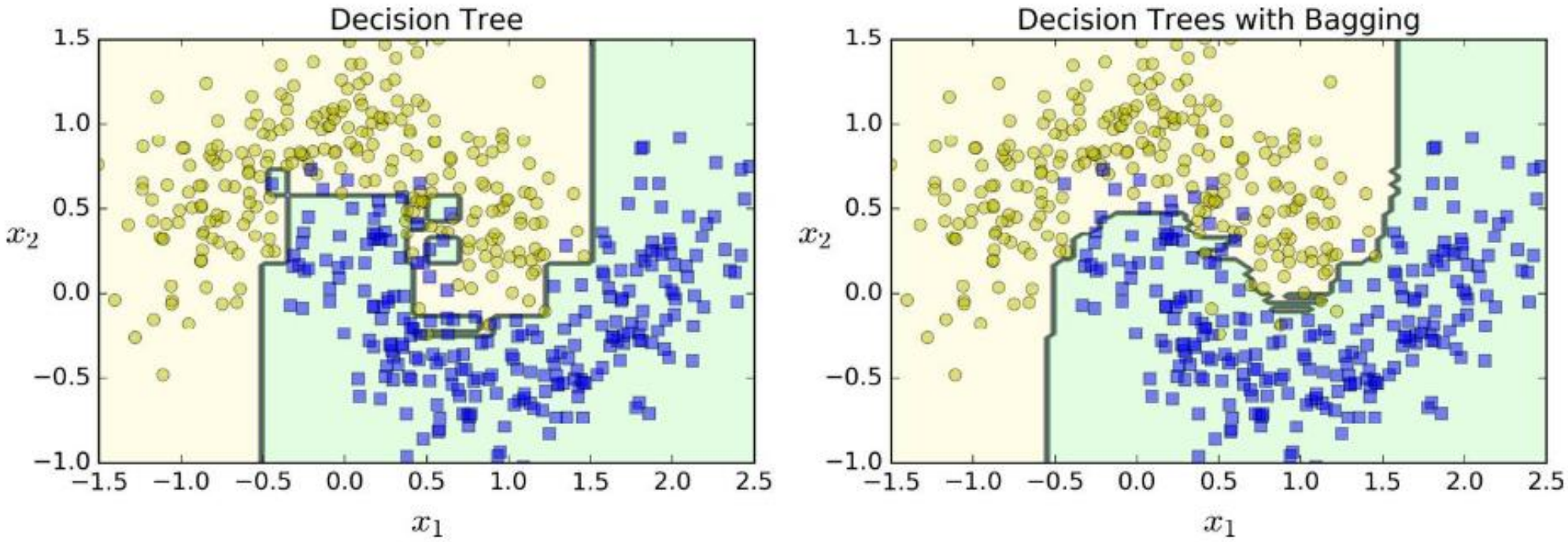
Random Forest

- **Random Forest** คือ การทำ **Bagging** โดยใช้ **Decision Tree** เป็น **base learner**
- และเพิ่มการสุ่มเลือกใช้ **feature** แค่บางตัว
- เกิด **tree** จำนวนมาก ที่มีความหลากหลาย มาช่วยกันตอบ



https://commons.wikimedia.org/wiki/File:Decision_Tree_vs._Random_Forest.png

Decision Tree with Bagging

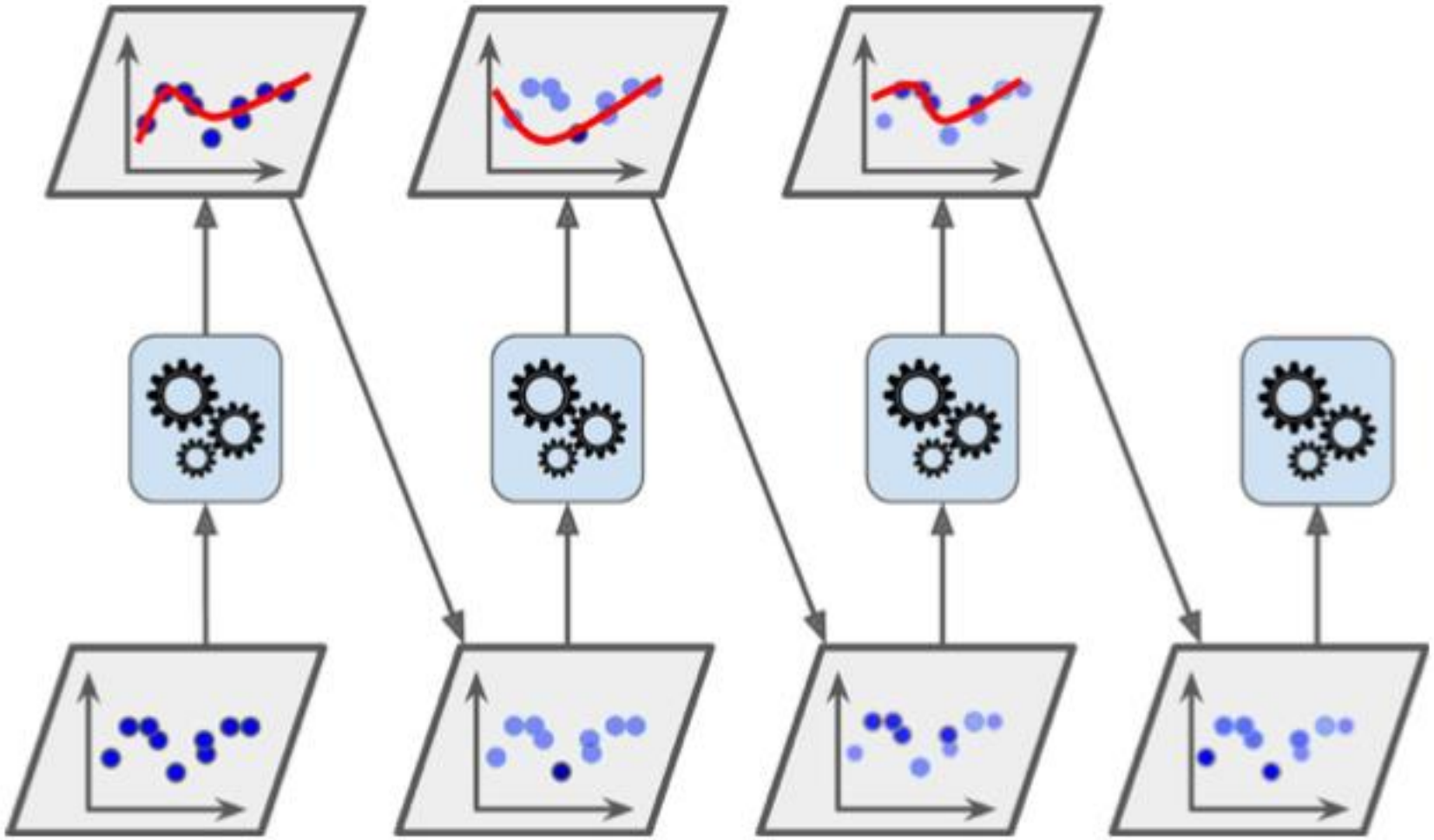


[Hands-On Machine Learning with Scikit-Learn and TensorFlow, Géron]

Boosting

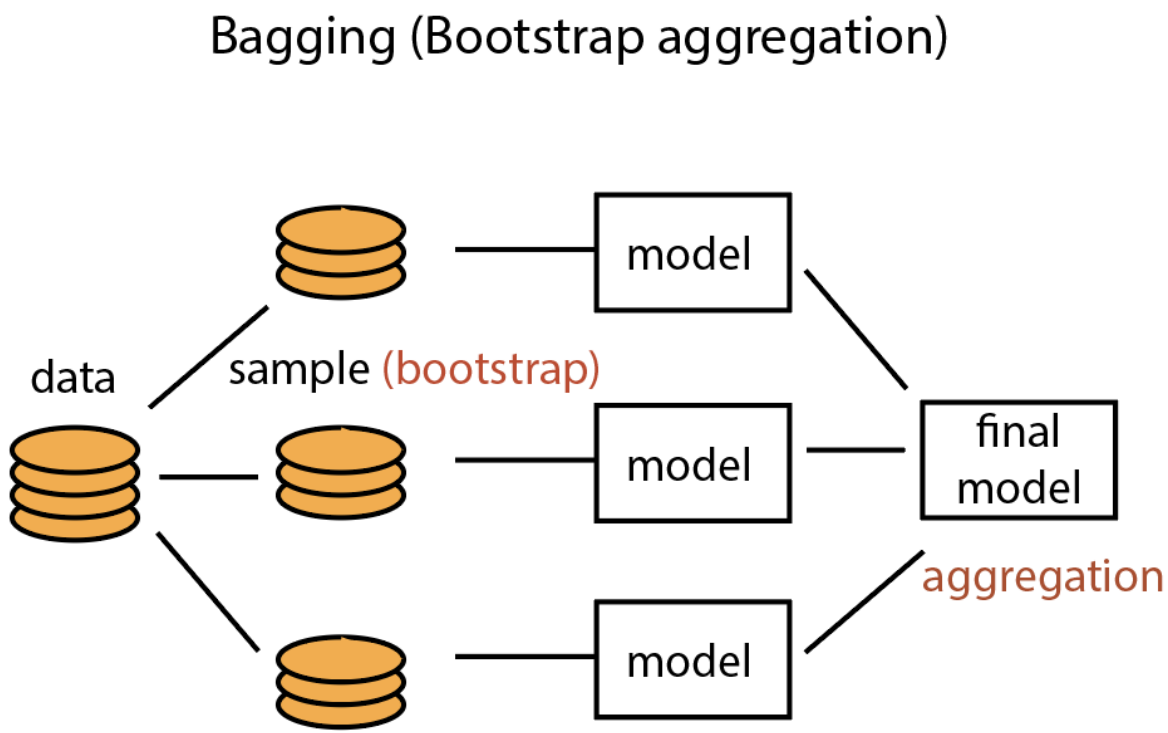
- คือการ train predictor ซ้ำๆ
  - แต่ละครั้งจะให้น้ำหนัก (weight) หรือตัวคูณกับจุดข้อมูลที่ทำนายพลาดมากขึ้น
  - เพื่อให้ train แล้วเกิด predictor ใหม่ที่แก้ไขจุดอ่อนของ predictor เก่า
  - นำผลของ predictor ทุกตัว มาเฉลี่ยกัน (โดย weight ตามความแม่นยำของ predictor)
- เป็นการ train แบบ sequential ไม่สามารถทำขนานได้
- อัลกอริทึมที่นิยมใช้
  - AdaBoost
  - Gradient Boosting (XGBoost, LightGBM)

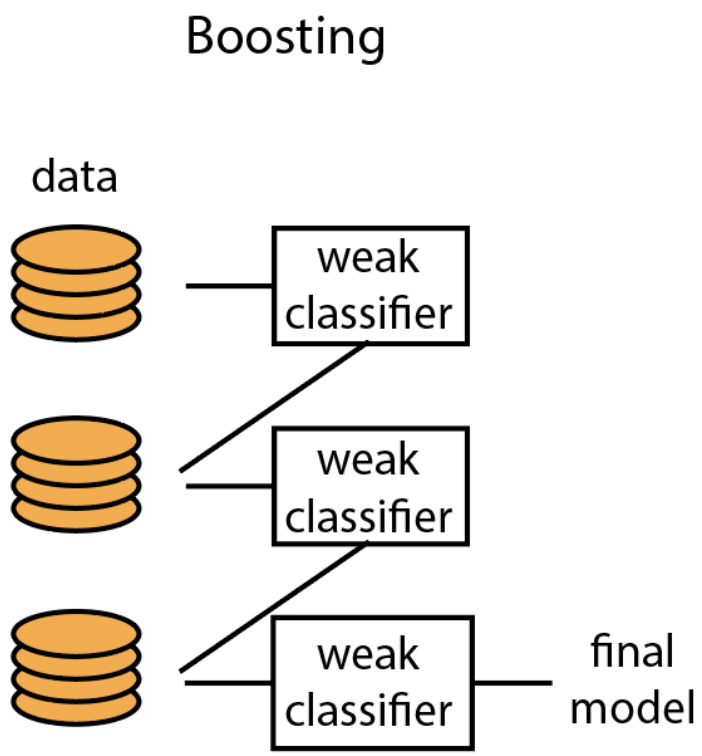
Boosting



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

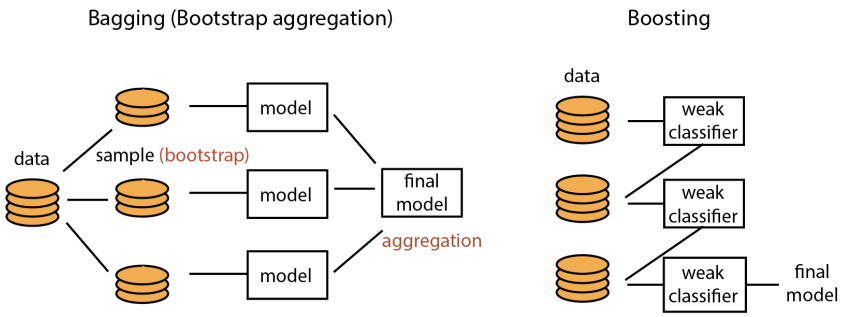
Bagging vs Boosting





https://tupleblog.github.io/bagging-boosting/

Bagging vs Boosting

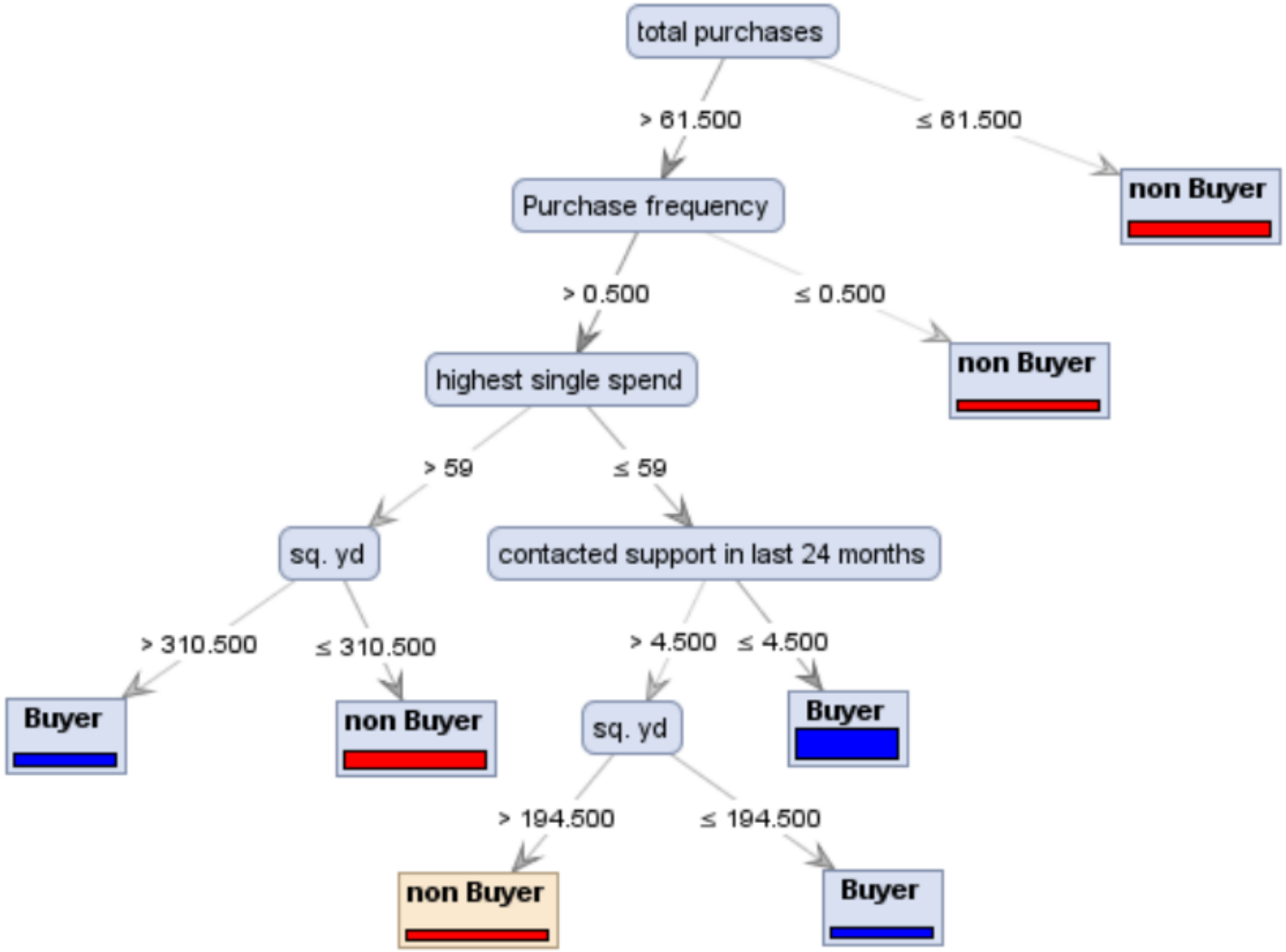


	Bagging	Boosting
ทำเพื่ออะไร	ลด Variance ลดการ Overfit	ลด Bias ลดการ Underfit
ข้อมูลที่ใช้ train แต่ละ predictor	ข้อมูลที่หยิบจาก training set โดยสุ่ม	ข้อมูลที่ถูกปรับถ่วงน้ำหนักตามการทำนายผิดพลาดในครั้งก่อน
ลักษณะการ train	Parallel = เร็วกว่า	Sequential = ช้ากว่า
เหมาะกับ base predictor แบบไหน	deep decision tree (ซึ่งมักจะ overfit)	shallow decision tree (ซึ่ง train ได้เร็ว)
ตัวอย่างอัลกอริทึม	Random Forest	XGBoost
ความแม่นยำ	มักจะต่ำกว่า	มักจะสูงกว่า

Single Decision Tree vs Ensemble

- การทำ ensemble (random forest, gradient boosted tree) ช่วยทำให้ผลการทำนายจาก decision tree แม่นยำและเสถียรขึ้น
  - เหมาะกับนำไปใช้แข่งขัน เช่น Kaggle
- Drawback: ทำให้อธิบายผลลัพธ์ได้ยากขึ้น ☹ (จาก white-box กลายเป็น black-box)
- การเลือกใช้ขึ้นกับว่าต้องการ performance หรือ interpretability
  - ทำนายแม่น เลือก ensemble
  - ตีความง่าย เลือก single decision tree

Decision Tree & Feature Importance



Homework – Decision Tree

จง apply decision tree กับชุดข้อมูล student_major.csv เพื่อทำนาย major ของนักศึกษา

1. แบ่งข้อมูลออกเป็นชุด train, test ในอัตราส่วน 75:25 โดยใช้ train_test_split (ตัวอย่างโค้ดในสไลด์ถัดไป)
2. ใช้ข้อมูลชุด train (X_train, y_train) ในการ train/validate โมเดล decision tree
   - ใช้ GridSearchCV เพื่อค้นหาค่า hyperparameter ของ model (เช่น max_depth, criterion) ที่ให้ค่า accuracy สูงที่สุด
3. นำ model ที่ประสิทธิภาพสูงที่สุดไปใช้ทำนาย major ของนักศึกษากับข้อมูลชุดทดสอบ (X_test, y_test) แสดงผลการทำนายทั้งหมด
4. ประเมินประสิทธิภาพของโมเดลที่ได้โดยใช้ metric ต่อไปนี้
   - accuracy score
   - confusion matrix (optional)

*ใน Jupyter พยายามให้ code ทั้งหมดอยู่ใน cell เดียว (หากจำเป็น สามารถใช้ได้ 4 cell max)

Homework – Decision Tree

วิธีการทำ train_test_split

```
from sklearn.model_selection import train_test_split

X = ...
y = ...
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=33)
```

จากนั้นสามารถนำ X_train, y_train ไปใช้ในการฝึก และนำ X_test, y_test ไปใช้ในการทำนาย เพื่อวัดประสิทธิภาพได้

Homework – Decision Tree

5. จากการทดลองในข้อ 4. ให้
   - แสดง line plot ของ accuracy v.s. max_depth จากการทดลอง ในกรณีที่ใช้ Gini criterion
   - แสดง line plot ของ accuracy v.s. max_depth จากการทดลอง ในกรณีที่ใช้ entropy criterion

   โดยที่ค่า accuracy ของแต่ละค่า hyperparameter จะถูกเก็บอยู่ใน GridSearchCV.cv_results_[“mean_test_score”]
6. แสดงแผนผัง decision tree ของโมเดลที่ดีที่สุด
   - นำผลจาก export_graphviz ของโมเดลที่ดีที่สุดไปแสดงบนเว็บ webgraphviz
   - capture screen ภาพ decision tree แล้วแนบส่งมาใน Teams พร้อมกับโค้ด

Homework – Decision Tree

วิธีส่งงาน

- ใน Jupyter Notebook ให้เขียนโค้ดทำ decision tree โดยพยายามให้ทั้งหมดอยู่ใน cell เดียว (หากจำเป็น สามารถใช้ได้ 4 cell max)
- ทำการ export html โดยไปที่ File -> Download as -> HTML
- rename ชื่อไฟล์เป็น ชื่อ สกุล ภาษาไทย ไม่มีคำนำหน้า
- ส่งไฟล์ html และภาพจาก webgraphviz ที่ MS Teams

Extra

- ลองเปลี่ยนไปใช้ Decision Tree + Bagging, Random Forest หรือ XGBoost
  - https://scikit-learn.org/stable/modules/ensemble.html#bagging
- เปรียบเทียบความแม่นยำกับข้อมูลชุดทดสอบ