

Classification

Random forest = Decision tree үүрэг, өдөрнийн 12-ийн нийтийн (Forest)

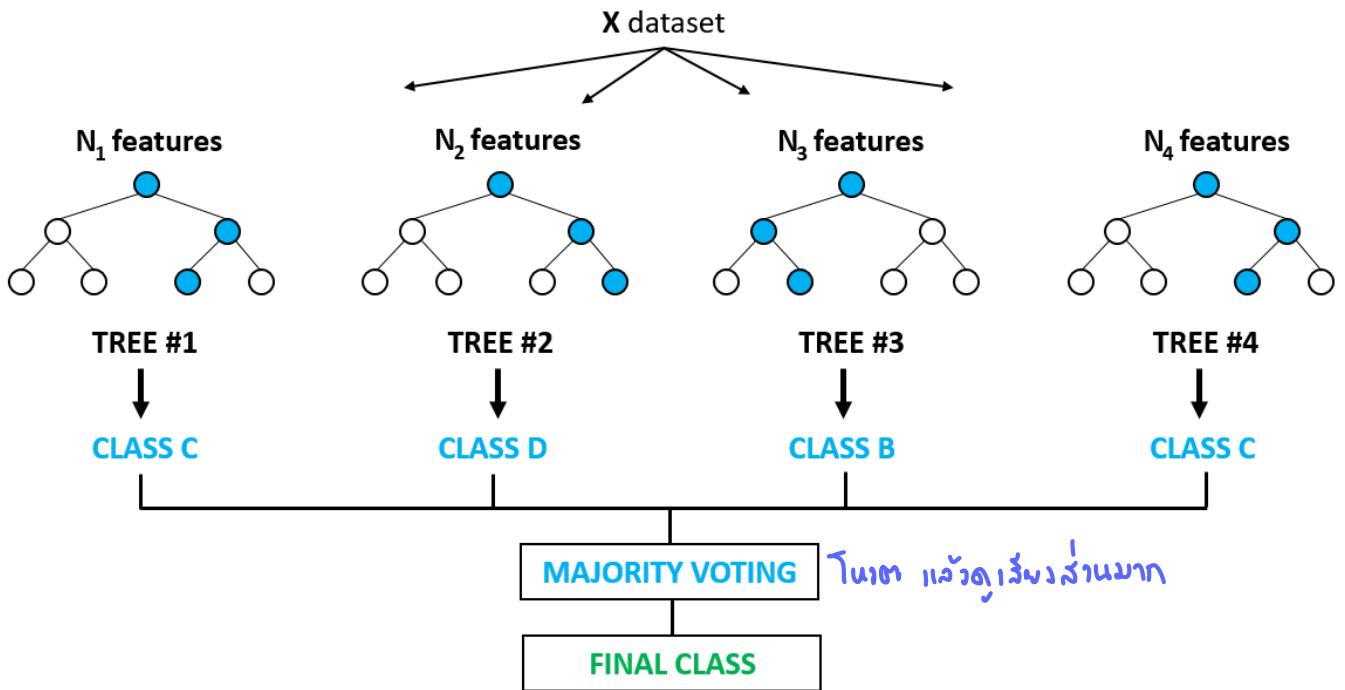


image from: [APPLYING RANDOM FOREST \(CLASSIFICATION\) – MACHINE LEARNING ALGORITHM FROM SCRATCH WITH REAL DATASETS](#)

- by  [scikit-learn](#) [scikit-learn \(RandomForestClassifier\)](#)

random forest օվյանձյալ

Ensemble Learning in Machine Learning

(ໝາຍະ: ດັ່ງລະກົມທີ່ ດີວັນຈີ
estimator e.g. Deci. tree
ໝາຍະ: ດີວັນຈີ ARF

Ensemble Learning เป็นเทคนิคที่มีประสิทธิภาพที่เกี่ยวข้องกับการรวมตัวแบบจำนวนมากเพื่อสร้างโมเดลที่มีประสิทธิภาพและทนทานมากขึ้น โดยการนำความสามารถของโมเดลต่าง ๆ มาใช้ร่วมกันและปรับปรุงจุดอ่อนของแต่ละโมเดล ทำให้ Ensemble Learning สามารถเพิ่มประสิทธิภาพในการทำนายได้อย่างมีนัยสำคัญ สถาปัตย์เก็งปีเพรีร์ model ที่ perfect 100% บนป้อมวัง ใจจริง model แหวน ทั่วโลก vote กัน

สำคัญ ใจอิցนี้เกิดปั้นเพริ่ง model ที่ perfect 100% วนไปเมื่อไร เวลาอีก model หาย ก็ตามก็จะ vote นั้

3. estimator univariatan. innovasiun = innovasiun univariatan estimator univariatan

1. Bagging (Bootstrap Aggregating):

→ 96 model ទាំង ៩៦, មានចំណាំ ៩៦ ក្នុង /model/ កែងក្រាំង នៃក្រុមការពារ (e.g. គុណភាពបាន row ០០១)

Bagging เป็นกระบวนการฝึกโมเดลหลายรูปของโมเดลเดียวกันนั่นชุดข้อมูลการฝึกแตกต่างกันและรวมคำทำนายของพากษาเพื่อทำนายสุดท้าย ตัวอย่างที่น่าสนใจของ Bagging คือ อัลกอริทึม Random Forest

Example: สมมติว่าคุณกำลังพยายามทำนายว่าลูกค้าจะซื้อผลิตภัณฑ์หรือไม่โดยอิงความเคลื่อนไหวของเว็บเบราว์เซอร์ ชิ้น อายุ และพฤติกรรมการซื้อ ก่อนหน้านี้ คุณสร้างโมเดลอิสระที่เป็นตัวเลือกสามแบบ:

- Model A: Decision Tree

- Model B: Logistic Regression
 - Model C: Support Vector Machine

model ຕ່າງກັນ ຂົບນູ້ເສຍົກໄ

สำหรับ Bagging คุณจะฝึกโมเดลแต่ละรูปแบบชุดข้อมูลการฝึกที่แตกต่างกัน ในการทำนายคุณสามารถใช้การโหวตเพื่อรวมคำทำนายของสามโมเดลเหล่านี้ เช่น ถ้าโมเดล A ทำนาย "ชื่อ" โมเดล B ทำนาย "ไม่ชื่อ" และโมเดล C ทำนาย "ชื่อ" คำทำนายสุดท้ายอาจเป็น "ชื่อ" จากการโหวตส่วนใหญ่

2. Boosting: = ອົບບົດຕົກລົງໂນໂລກ e.g. model sou1 ຕົກລົງ class A ໂພນນວກ ໂຕ C ຖ້າມາ
sou2 ດີເນີນເບີນຢູ່ທີ່ C ມາກີ່ນ ທັກທີ່ນ ແນວດ sou3 ໂອກຈະ NB

Boosting เน้นการปรับปรุงความแม่นยำของโมเดลที่อ่อนแอโดยการฝึกพวกรเข้าตามลำดับ โมเดลที่ตามมาจะเน้นการแก้ไขข้อผิดพลาดของโมเดลที่พบริบก่อนหน้านี้ ก็คือ model 3 แบบที่เน้น class A, B, C แล้วก็ทางลัด การทำนายสุดท้ายคือการผสานคำทำนายของโมเดลแต่ละตัวโดยใช้ค่าน้ำหนัก

Example: สมมติว่าคุณกำลังสร้างโมเดลเพื่อทำนายว่าผู้ป่วยมีโรคหรือไม่ โดยอิงตัวชี้วัดสุขภาพต่าง ๆ คุณเริ่มโดยการฝึกโมเดล A และตามด้วยการฝึกโมเดล B ด้วยข้อมูลที่โมเดล A ทำนายผิด และในลำดับต่อมาฝึกโมเดล C ด้วยข้อมูลที่โมเดล A และโมเดล B ทำนายผิด การทำนายสุดท้ายในกระบวนการ Boosting มักจะเป็นผลลัพธ์ที่ได้จากการรวมน้ำหนักของการทำนายของโมเดลทั้งสามรายการ

3. Stacking:

Stacking เป็นเทคนิคขั้นสูงของ Ensemble Learning ซึ่งนำเข้ามาใช้โนเดลเมต้าในการรวมการทำนายของโนเดลฐานหลัก ๆ รายการเข้าด้วยกัน โนเดลฐานจะทำนายผลบนข้อมูลนำเข้าและโนเดลเมต้าจะเรียนรู้วิธีกำหนดน้ำหนักและรวมการทำนายเหล่านี้ให้เหมาะสม ดังที่ classi 2 ระบุ

Example: สมมติว่าคุณกำลังพัฒนาระบบแนะนำสินค้าสำหรับเว็บค้าออนไลน์ คุณสร้างโมเดลฐานสามรายการ:

- Model A: Decision Tree ที่ใช้ประวัติการเรียกดู
 - Model B: Neural Network ที่ใช้ข้อมูลทางสถิติของผู้ใช้
 - Model C: Gradient Boosting ที่ใช้ประวัติการซื้อและรีวิว

} ans-model (1.98 Bagging und Boosting etc.)

จากนั้นก็สร้างโมเดลเมต้า เช่น Random Forest Regression เพื่อรวมการทำนายของโมเดลฐานเหล่านี้ โมเดลเมต้าจะกำหนดน้ำหนักที่เหมาะสมให้กับการทำนายของแต่ละโมเดลฐาน ทำให้ได้ผลลัพธ์การแนะนำสินค้าสุดท้ายสำหรับผู้ใช้ แก้ตอบปัญหานี้ ปัจจุบันเรียกว่า **meta learning** หรือ **meta regression** แต่ก็มีความหมายเดียวกัน คือการใช้โมเดลเดียวกันเพื่อปรับแต่งโมเดลเดียวกันนั้น ให้ดีขึ้น แต่ก็มีความหมายเดียวกัน คือการใช้โมเดลเดียวกันเพื่อปรับแต่งโมเดลเดียวกันนั้น ให้ดีขึ้น

▼ Recap of Decision Trees:

Decision Trees เป็นอัลกอริทึมที่ได้รับความนิยมสำหรับการเรียนรู้ของเครื่องใช้ทั้งในงาน classification และ prediction โดยสร้างโมเดลที่เปรียบเสมือนต้นไม้ที่มีการตัดสินใจและผลที่เป็นไปได้ทั้งหมด โหนดภายในแต่ละโหนดแทนคุณสมบัติหรือแอดทริบิวต์ แต่ละกิ่งแทนกฎการตัดสินใจและโหนดในแทนผลลัพธ์

សរុប ~
Code Exercise: Building a Decision Tree Classifier

ใช้ข้อมูล Iris สร้าง Decision Tree โดยใช้ไลบรารี scikit-learn ของ Python

- Import ไลบรารีที่จำเป็น

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

- โหลด Iris dataset

```
iris = load_iris()
iris

{'data': array([[5.1, 3.5, 1.4, 0.2],
   [4.9, 3. , 1.4, 0.2],
   [4.7, 3.2, 1.3, 0.2],
   [4.6, 3.1, 1.5, 0.2],
   [5. , 3.6, 1.4, 0.2],
   [5.4, 3.9, 1.7, 0.4],
   [4.6, 3.4, 1.4, 0.3],
   [5. , 3.4, 1.5, 0.2],
   [4.4, 2.9, 1.4, 0.2],
   [4.9, 3.1, 1.5, 0.1],
   [5.4, 3.7, 1.5, 0.2],
   [4.8, 3.4, 1.6, 0.2],
   [4.8, 3. , 1.4, 0.1],
   [4.3, 3. , 1.1, 0.1],
   [5.8, 4. , 1.2, 0.2],
   [5.7, 4.4, 1.5, 0.4],
   [5.4, 3.9, 1.3, 0.4],
   [5.1, 3.5, 1.4, 0.3],
   [5.7, 3.8, 1.7, 0.3],
   [5.1, 3.8, 1.5, 0.3],
   [5.4, 3.4, 1.7, 0.2],
   [5.1, 3.7, 1.5, 0.4],
   [4.6, 3.6, 1. , 0.2],
   [5.1, 3.3, 1.7, 0.5],
   [4.8, 3.4, 1.9, 0.2],
   [5. , 3. , 1.6, 0.2],
   [5. , 3.4, 1.6, 0.4],
   [5.2, 3.5, 1.5, 0.2],
   [5.2, 3.4, 1.4, 0.2],
   [4.7, 3.2, 1.6, 0.2],
   [4.8, 3.1, 1.6, 0.2],
   [5.4, 3.4, 1.5, 0.4],
   [5.2, 4.1, 1.5, 0.1],
   [5.5, 4.2, 1.4, 0.2],
   [4.9, 3.1, 1.5, 0.2],
   [5. , 3.2, 1.2, 0.2],
   [5.5, 3.5, 1.3, 0.2],
   [4.9, 3.6, 1.4, 0.1],
   [4.4, 3. , 1.3, 0.2],
   [5.1, 3.4, 1.5, 0.2],
   [5. , 3.5, 1.3, 0.3],
   [4.5, 2.3, 1.3, 0.3],
   [4.4, 3.2, 1.3, 0.2],
```

```
[5. , 3.5, 1.6, 0.6],  
[5.1, 3.8, 1.9, 0.4],  
[4.8, 3. , 1.4, 0.3],  
[5.1, 3.8, 1.6, 0.2],  
[4.6, 3.2, 1.4, 0.2],  
[5.3, 3.7, 1.5, 0.2],  
[5. , 3.3, 1.4, 0.2],  
[7. , 3.2, 4.7, 1.4],  
[6.4, 3.2, 4.5, 1.5],  
[6.9, 3.1, 4.9, 1.5],  
[5.5, 2.3, 4. , 1.3],  
[6.5, 2.8, 4.6, 1.5],  
[5.7, 2.8, 4.5, 1.3],  
[6.3, 3.3, 4.7, 1.6],  
[4.9, 2.4, 3.3, 1. ],
```

X = iris.data *feature նշեցած*
X

```
array([[5.1, 3.5, 1.4, 0.2],  
       [4.9, 3. , 1.4, 0.2],  
       [4.7, 3.2, 1.3, 0.2],  
       [4.6, 3.1, 1.5, 0.2],  
       [5. , 3.6, 1.4, 0.2],  
       [5.4, 3.9, 1.7, 0.4],  
       [4.6, 3.4, 1.4, 0.3],  
       [5. , 3.4, 1.5, 0.2],  
       [4.4, 2.9, 1.4, 0.2],  
       [4.9, 3.1, 1.5, 0.1],  
       [5.4, 3.7, 1.5, 0.2],  
       [4.8, 3.4, 1.6, 0.2],  
       [4.8, 3. , 1.4, 0.1],  
       [4.3, 3. , 1.1, 0.1],  
       [5.8, 4. , 1.2, 0.2],  
       [5.7, 4.4, 1.5, 0.4],  
       [5.4, 3.9, 1.3, 0.4],  
       [5.1, 3.5, 1.4, 0.3],  
       [5.7, 3.8, 1.7, 0.3],  
       [5.1, 3.8, 1.5, 0.3],  
       [5.4, 3.4, 1.7, 0.2],  
       [5.1, 3.7, 1.5, 0.4],  
       [4.6, 3.6, 1. , 0.2],  
       [5.1, 3.3, 1.7, 0.5],  
       [4.8, 3.4, 1.9, 0.2],  
       [5. , 3. , 1.6, 0.2],  
       [5. , 3.4, 1.6, 0.4],  
       [5.2, 3.5, 1.5, 0.2],  
       [5.2, 3.4, 1.4, 0.2],  
       [4.7, 3.2, 1.6, 0.2],  
       [4.8, 3.1, 1.6, 0.2],  
       [5.4, 3.4, 1.5, 0.4],  
       [5.2, 4.1, 1.5, 0.1],  
       [5.5, 4.2, 1.4, 0.2],  
       [4.9, 3.1, 1.5, 0.2],  
       [5. , 3.2, 1.2, 0.2],  
       [5.5, 3.5, 1.3, 0.2],  
       [4.9, 3.6, 1.4, 0.1],  
       [4.4, 3. , 1.3, 0.2],  
       [5.1, 3.4, 1.5, 0.2],  
       [5. , 3.5, 1.3, 0.3],
```

```
[4.5, 2.3, 1.3, 0.3],  
[4.4, 3.2, 1.3, 0.2],  
[5. , 3.5, 1.6, 0.6],  
[5.1, 3.8, 1.9, 0.4],  
[4.8, 3. , 1.4, 0.3],  
[5.1, 3.8, 1.6, 0.2],  
[4.6, 3.2, 1.4, 0.2],  
[5.3, 3.7, 1.5, 0.2],  
[5. , 3.3, 1.4, 0.2],  
[7. , 3.2, 4.7, 1.4],  
[6.4, 3.2, 4.5, 1.5],  
[6.9, 3.1, 4.9, 1.5],  
[5.5, 2.3, 4. , 1.3],  
[6.5, 2.8, 4.6, 1.5],  
[5.7, 2.8, 4.5, 1.3],  
[6.3, 3.3, 4.7, 1.6],  
[4.9, 2.4, 3.3, 1. ],
```

y = iris.target **عن class**
y

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,  
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,  
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

len(y)

150

- แบ่งชุดข้อมูลเป็นชุดฝึกและชุดทดสอบ

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size = 0.3, random_state=555  
)
```

- สร้าง Decision Tree

```
clf = DecisionTreeClassifier()
```

- เรียนรู้ข้อมูล

```
clf.fit(X_train, y_train)
```

```
• DecisionTreeClassifier  
DecisionTreeClassifier()
```

- สร้างต้นไม้เรียบง่ายที่ใช้ตัดสินใจ

- Decision tree ของ scikit-learn คือ binary classifier
(คุณภาพ 2 ทิศ (yes/no))

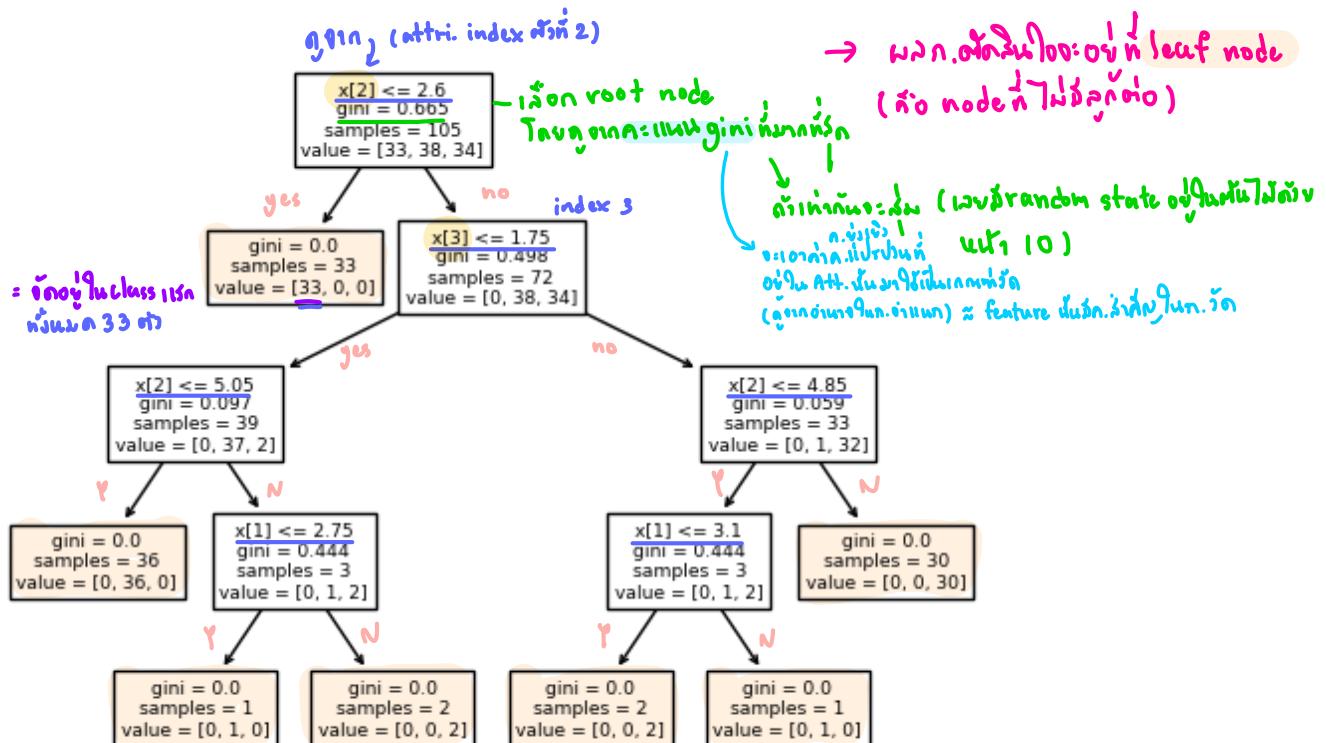
- ทำนายผลด้วยข้อมูลทดสอบ

```
y_pred = clf.predict(X_test)  
accuracy_score(y_test, y_pred)  
0.9333333333333333
```

- แสดง Tree

```
import matplotlib.pyplot as plt  
from sklearn.tree import plot_tree
```

```
plot_tree(clf)  
plt.show()
```



→ print หาตัวที่มี index ที่ต้องการ

```
print(export_text(clf, feature_names=iris.feature_names))
```

```
|--- petal length (cm) <= 2.60
|   |--- class: 0
|--- petal length (cm) >  2.60
|   |--- petal width (cm) <= 1.75
|       |--- petal length (cm) <= 5.05
|           |--- class: 1
|           |--- petal length (cm) >  5.05
|               |--- sepal width (cm) <= 2.75
|                   |--- class: 1
|                   |--- sepal width (cm) >  2.75
|                       |--- class: 2
|--- petal width (cm) >  1.75
|   |--- petal length (cm) <= 4.85
```

```

    |   |   |   --- sepal width (cm) <= 3.10
    |   |   |   |--- class: 2
    |   |   |   --- sepal width (cm) > 3.10
    |   |   |   |--- class: 1
    |   |   |   --- petal length (cm) > 4.85
    |   |   |   |--- class: 2

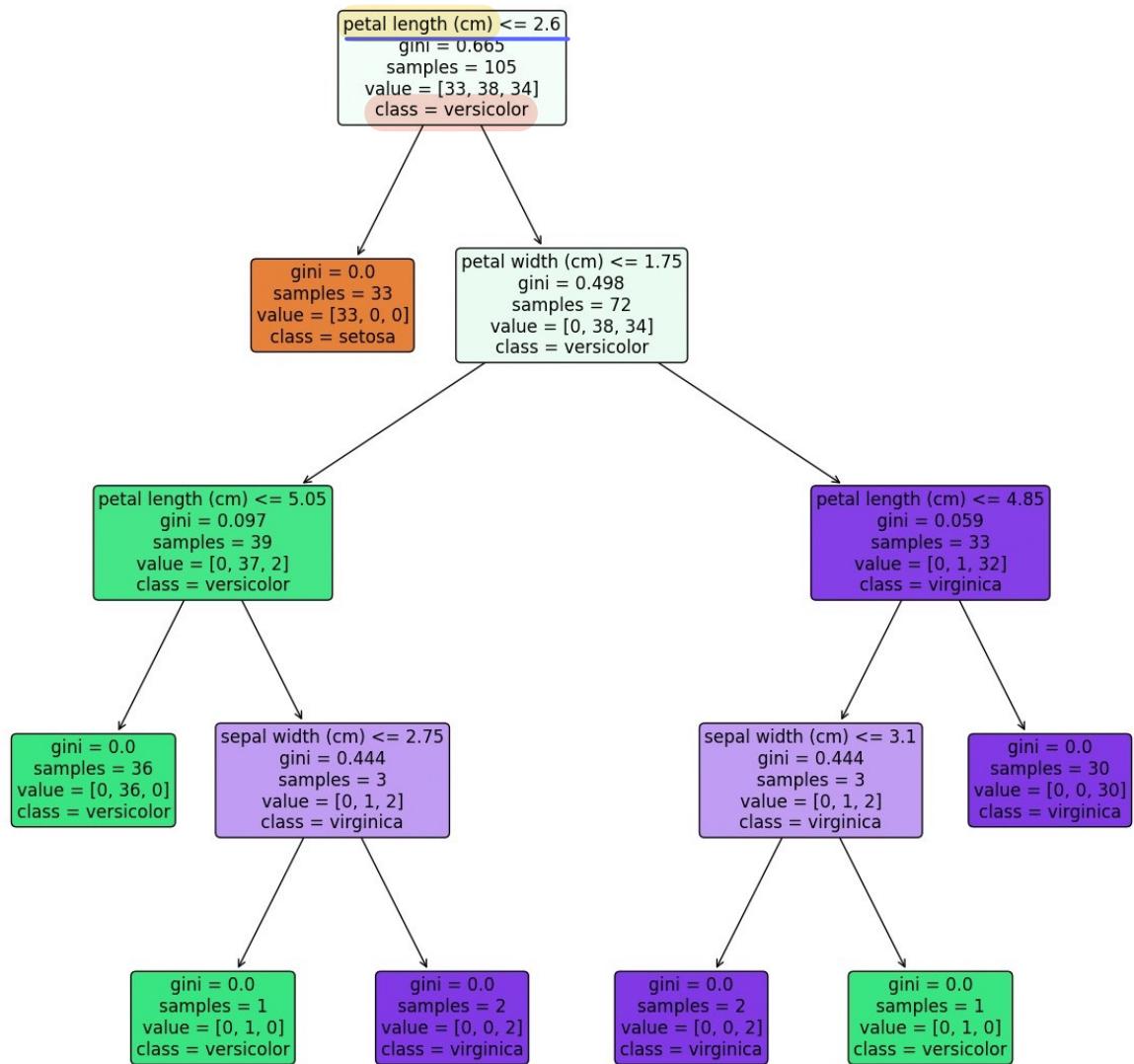
```

→ Plot 9 ប៉ុណ្ណោះឱ្យដាក់ទូទាត់ និង index នៅលើលេខរៀងទៅលក្ខណៈទាំងអស់

```

plt.figure(figsize=(15, 15))
plot_tree(clf, feature_names=iris.feature_names,
           class_names=iris.target_names, filled=True, rounded=True)
plt.show()

```



Random Forest Algorithm

The Random Forest algorithm is an ensemble learning technique that constructs multiple decision trees and aggregates their predictions. It introduces randomness in both data and feature selection during tree construction, leading to improved generalization and reduced overfitting. The final prediction is determined through a majority vote (for classification) or an average (for regression) of individual tree predictions.

▼ สร้าง DataFrame

- ใช้ dataset จากไฟล์ winequality-red (INTERM DS).csv
- อัปโหลดขึ้น sample_data
- เรียกใช้ pandas
- อ่านไฟล์ winequality-red (INTERM DS).csv จาก sample_data โดยคลิกขวาที่ชื่อไฟล์ และเลือก Copy path

```
data = pd.read_csv('/content/sample_data/winequality-red (INTERM DS).csv')
data
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	0.00	0.00	0.9970	3.5	0.54	11.0
1	7.8	0.880	0.00	2.6	0.098	0.00	0.00	0.9970	3.5	0.54	13.9
2	7.8	0.760	0.04	2.3	0.092	0.00	0.00	0.9970	3.5	0.54	14.5
3	11.2	0.280	0.56	1.9	0.075	0.00	0.00	0.9970	3.5	0.54	15.2
4	7.4	0.700	0.00	1.9	0.076	0.00	0.00	0.9970	3.5	0.54	15.5
...
1594	6.2	0.600	0.08	2.0	0.090	0.00	0.00	0.9970	3.5	0.54	15.5
1595	5.9	0.550	0.10	2.2	0.062	0.00	0.00	0.9970	3.5	0.54	15.5
1596	6.3	0.510	0.13	2.3	0.076	0.00	0.00	0.9970	3.5	0.54	15.5
1597	5.9	0.645	0.12	2.0	0.075	0.00	0.00	0.9970	3.5	0.54	15.5
1598	6.0	0.310	0.47	3.6	0.067	0.00	0.00	0.9970	3.5	0.54	15.5

1599 rows × 12 columns

```
data['quality'].value_counts()
```

```
1      855  
0      744  
Name: quality, dtype: int64
```

- ในที่นี้ต้องการทำนาย quality (y)

```
X = data.drop(['quality'], axis=1)  
y = data['quality']
```

- แบ่งข้อมูลออกเป็นชุดการเรียนรู้ และชุดทดสอบ โดยใช้ `train_test_split`

- แบ่งข้อมูลสำหรับเรียน 80% สำหรับทดสอบ 20%
 - ผลลัพธ์ที่ได้จะอยู่ในรูปของ

`X_train, X_test, y_train, y_test`

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, train_size=0.8, random_state=555  
)
```

```
print(len(x_train), len(x_test))
```

1279 320

▼ สร้าง RandomForest Classifier Model

- เรียกใช้งาน RandomForestClassifier library

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

- สร้าง model

```
rf = RandomForestClassifier(n_estimators=82, random_state=555)
```

- เรียนรู้จากชุดข้อมูล

```
rf.fit(X_train, y_train)
```

RandomForestClassifier

```
RandomForestClassifier(n_estimators=82, random_state=555)
```

- 82 tree ย่อยที่อยู่ในชุด estimators

```
rf.estimators_
```

```
[DecisionTreeClassifier(max_features='sqrt', random_state=935590298),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1299452590),  
 DecisionTreeClassifier(max_features='sqrt', random_state=205536233),  
 DecisionTreeClassifier(max_features='sqrt', random_state=707945505),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1908992580),  
 DecisionTreeClassifier(max_features='sqrt', random_state=2114428285),  
 DecisionTreeClassifier(max_features='sqrt', random_state=800496166),  
 DecisionTreeClassifier(max_features='sqrt', random_state=2027234710),  
 DecisionTreeClassifier(max_features='sqrt', random_state=348782825),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1684888573),  
 DecisionTreeClassifier(max_features='sqrt', random_state=612799984),  
 DecisionTreeClassifier(max_features='sqrt', random_state=762371381),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1915208288),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1459143764),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1460248470),  
 DecisionTreeClassifier(max_features='sqrt', random_state=337997398),  
 DecisionTreeClassifier(max_features='sqrt', random_state=517266357),  
 DecisionTreeClassifier(max_features='sqrt', random_state=282005066),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1945446869),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1507071003),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1029793244),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1319647290),  
 DecisionTreeClassifier(max_features='sqrt', random_state=148196080),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1283649540),  
 DecisionTreeClassifier(max_features='sqrt', random_state=638202561),  
 DecisionTreeClassifier(max_features='sqrt', random_state=97385597),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1335160042),  
 DecisionTreeClassifier(max_features='sqrt', random_state=631648762),  
 DecisionTreeClassifier(max_features='sqrt', random_state=657582569),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1154413460),  
 DecisionTreeClassifier(max_features='sqrt', random_state=190991539),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1686341197),  
 DecisionTreeClassifier(max_features='sqrt', random_state=176930169),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1394802003),  
 DecisionTreeClassifier(max_features='sqrt', random_state=2010758723),  
 DecisionTreeClassifier(max_features='sqrt', random_state=270441747),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1669654500),  
 DecisionTreeClassifier(max_features='sqrt', random_state=886201687),  
 DecisionTreeClassifier(max_features='sqrt', random_state=2045514936),  
 DecisionTreeClassifier(max_features='sqrt', random_state=81674227),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1771626705),  
 DecisionTreeClassifier(max_features='sqrt', random_state=915474692),  
 DecisionTreeClassifier(max_features='sqrt', random_state=498673769),  
 DecisionTreeClassifier(max_features='sqrt', random_state=879330060),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1380473585),  
 DecisionTreeClassifier(max_features='sqrt', random_state=594732481),  
 DecisionTreeClassifier(max_features='sqrt', random_state=347330824),  
 DecisionTreeClassifier(max_features='sqrt', random_state=942612500),  
 DecisionTreeClassifier(max_features='sqrt', random_state=1703444379),  
 DecisionTreeClassifier(max_features='sqrt', random_state=987903645),
```

```
DecisionTreeClassifier(max_features='sqrt', random_state=86291744),  
DecisionTreeClassifier(max_features='sqrt', random_state=1355320689),  
DecisionTreeClassifier(max_features='sqrt', random_state=1852831279),  
DecisionTreeClassifier(max_features='sqrt', random_state=1209559812),  
DecisionTreeClassifier(max_features='sqrt', random_state=1766281748),  
DecisionTreeClassifier(max_features='sqrt', random_state=322039019),  
DecisionTreeClassifier(max_features='sqrt', random_state=554315984),  
DecisionTreeClassifier(max_features='sqrt', random_state=1889898831).
```

tree = rf.estimators_[0] กู้หุ้นล้วน
tree

```
DecisionTreeClassifier  
DecisionTreeClassifier(max_features='sqrt', random_state=935590298)
```

- เรียกใช้เครื่องมือเพื่อช่วยแสดง tree

```
from sklearn.tree import export_text
```

- แสดง tree แบบข้อความ (กู้หุ้นล้วน) → ถู๊ดักทั้งหมด นี่ = ถู๊ดักทั้งหมด

```
print(export_text(tree, feature_names=list(X.columns)))
```

```
--- alcohol <= 9.95  
    --- sulphates <= 0.57  
        --- density <= 1.00  
            --- sulphates <= 0.53  
                --- alcohol <= 9.55  
                    --- volatile acidity <= 0.33  
                        --- free sulfur dioxide <= 33.00  
                            --- class: 1.0  
                            --- free sulfur dioxide > 33.00  
                                --- class: 0.0  
                            --- volatile acidity > 0.33  
                                --- class: 0.0  
    --- alcohol > 9.55  
        --- chlorides <= 0.08  
            --- density <= 1.00  
                --- chlorides <= 0.08  
                    --- class: 0.0  
                --- chlorides > 0.08  
                    --- residual sugar <= 1.80  
                        --- class: 0.0  
                    --- residual sugar > 1.80  
                        --- class: 1.0  
            --- density > 1.00  
                --- class: 1.0  
        --- chlorides > 0.08  
            --- residual sugar <= 1.90  
                --- density <= 1.00  
                    --- class: 1.0  
                --- density > 1.00  
                    --- class: 0.0  
            --- residual sugar > 1.90
```

```

    |   |   |   |   |   |--- class: 0.0
    |--- sulphates > 0.53
    |--- chlorides <= 0.07
    |   |--- class: 0.0
    |--- chlorides > 0.07
    |   |--- pH <= 3.16
    |   |   |--- class: 0.0
    |--- pH > 3.16
    |   |--- pH <= 3.34
    |   |   |--- total sulfur dioxide <= 85.00
    |   |   |   |--- class: 1.0
    |   |   |--- total sulfur dioxide > 85.00
    |   |   |   |--- alcohol <= 9.60
    |   |   |   |   |--- class: 0.0
    |   |   |   |--- alcohol > 9.60
    |   |   |   |   |--- class: 1.0
    |--- pH > 3.34
    |   |--- residual sugar <= 1.95
    |   |   |--- class: 0.0
    |   |--- residual sugar > 1.95
    |   |   |--- alcohol <= 9.65
    |   |   |   |--- class: 1.0
    |   |   |   |--- alcohol > 9.65
    |   |   |   |   |--- class: 0.0
    |--- density > 1.00
    |   |--- chlorides <= 0.07
    |   |   |--- fixed acidity <= 12.05

```

- ในกรณีนี้ class อยู่ในรูปแบบของตัวเลข

- 0 - Bad Wine
- 1 - Good Wine

tree.classes_

```
array([0., 1.])
```

- จึงสร้างเป็น tuple ของ class name ตาม label ที่มี

```
index: 0          1
wine_classes = ('Bad Wine', 'Good Wine')
```

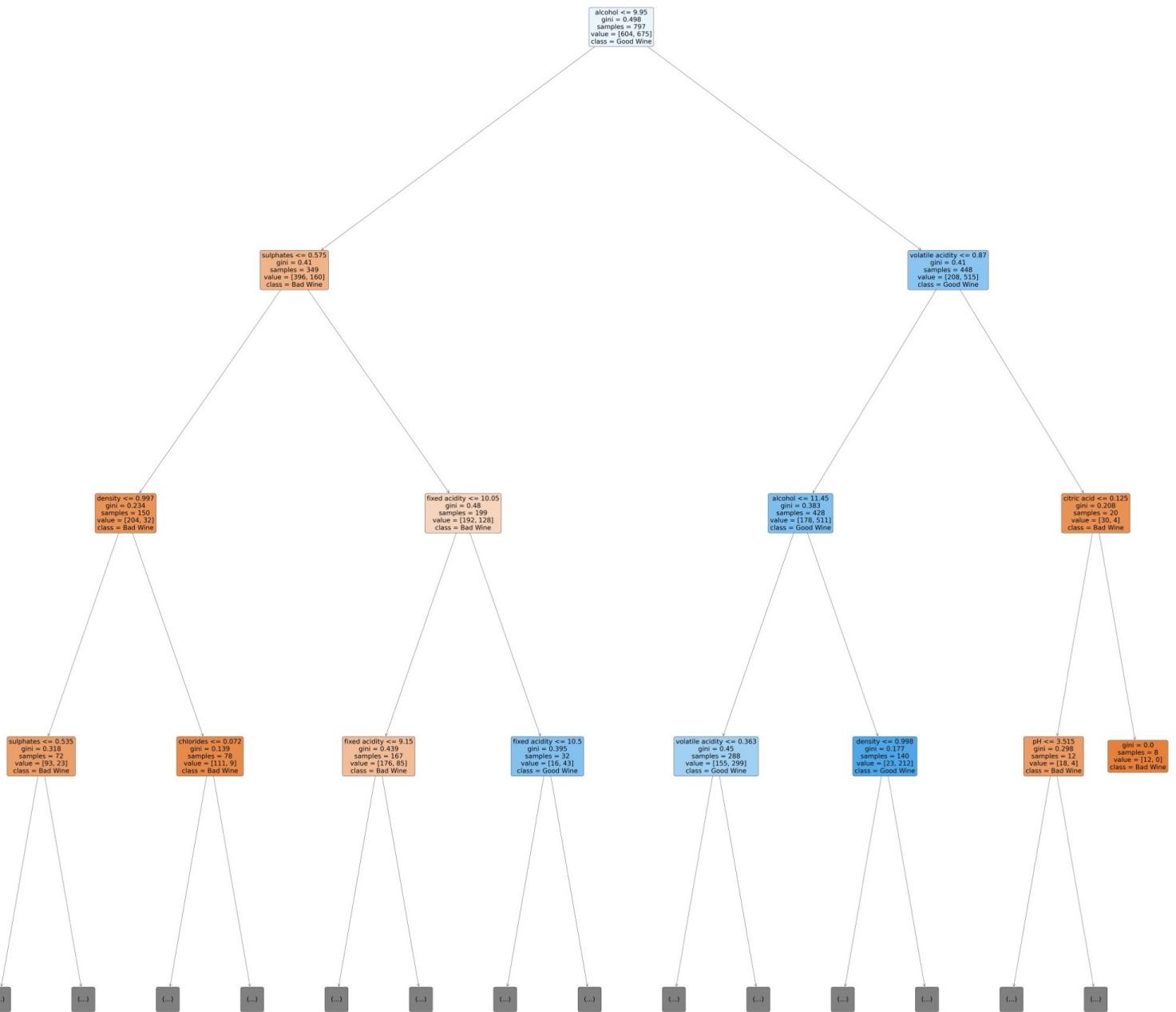
- แสดง tree แบบกราฟ

```

plt.figure(figsize=(100, 100))
plot_tree(tree, feature_names=X.columns,
          class_names=wine_classes, filled=True, rounded=True,
          max_depth=3)
plt.show()

```

= ต้น 3 ชั้น (without root)



- ใช้ model ที่ผ่านการเรียนรู้แล้วในการทำนาย

```
predictions = rf.predict(X_test)
predictions
```

```
array([0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
```

```

1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1,

```

វិធាន់នៅលើ False = រាយដែរ

វិធាន់នៅលើ positive, និង Negative

▼ Confusion matrix

វัดគម្រោងមែនយាំខែង model

- ក្រសួង binary classification (Yes, No)

$c_{\{0,0\}}$: true negatives $c_{\{0,1\}}$: false positives

$c_{\{1,0\}}$: false negatives $c_{\{1,1\}}$: true positives

- គឺជាក្រឹមគេរែងមែននៅក្នុងការវัดគម្រោងមែនយាំខែង model



```

from sklearn.metrics import confusion_matrix, accuracy_score, \
precision_score, recall_score, classification_report

```

- Confusion matrix

$y_{\text{ពិ.}} \quad y_{\text{ពិរាម}}$
`confusion_matrix(y_test, predictions)`
 $\begin{array}{cc} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{array} = \begin{array}{cc} 112 & 36 \\ 34 & 138 \end{array}$
 $\text{TP} = រាយដែរ នៅលើ នឹងបាន 174$
 $\text{TN} = រាយដែរ នៅលើ នឹងបាន 138$
 $\text{FP} = រាយដែរ នៅលើ នឹងបាន 34$
 $\text{FN} = រាយដែរ នៅលើ នឹងបាន 146$

`y_test: y_test.value_counts()`

$\begin{array}{cc} \text{1} & 172 \\ \text{0} & 148 \end{array}$
`Name: quality, dtype: int64`

- Accuracy = និងចូរកំណើន

```
accuracy_score(y_test, predictions)
```

0.78125 78%. → រាយដែរ 250 នៃ 326 នឹង

នាយករាជ .. $(112+138)/(172+148)$

0.78125

- Precision = និងចូរកំណើន វិធាន់នៅលើ positive និងចូរកំណើន វិធាន់នៅលើ negative

```
precision_score(y_test, predictions)
```

0.7931034482758621 79%

1

ນາທົກ.. 138/174

0.7931034482758621

- Recall = 94% of positive loans are correctly predicted

💡 ចົວຈຸບັນ precision, recall

recall_score(y_test, predictions)

0.8023255813953488 80%

ນາທົກ.. 138/172

0.8023255813953488

- สร้าง report

print(classification_report(y_test, predictions))

	precision	recall	f1-score	support
0	0.77	0.76	0.76	148
1	0.79	0.80	0.80	172
accuracy				320
macro avg	0.78	0.78	0.78	320
weighted avg	0.78	0.78	0.78	320

ຄົດກ່າວງ.
ໄລ້ໄວເຕັມມາດັ່ງ
accuracy
macro avg
weighted avg

ຄົດກ່າວງ.
ໄລ້ໄວເຕັມມາດັ່ງ : $\frac{(148 \times 77) + (172 \times 79)}{320}$

	Actual	
	Bad Loan (1)	Good Loan (0)
Predict	Bad Loan (1)	Good Loan (0)
Bad Loan = 1	✓ TP - 559	✗ FP - 0
Good Loan = 0	✗ FN - 33	✓ TN - 22

Bad loan predicted as a good loan

Precision: Out of the loan that is predicted as a bad loan, how many did we classify correctly?

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

$$= \frac{559}{559+0} = 100\%$$

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

$$= \frac{559}{559+33} = 94.5\%$$

High Recall, Low Precision: This means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.

Low Recall, High Precision: This shows that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP)

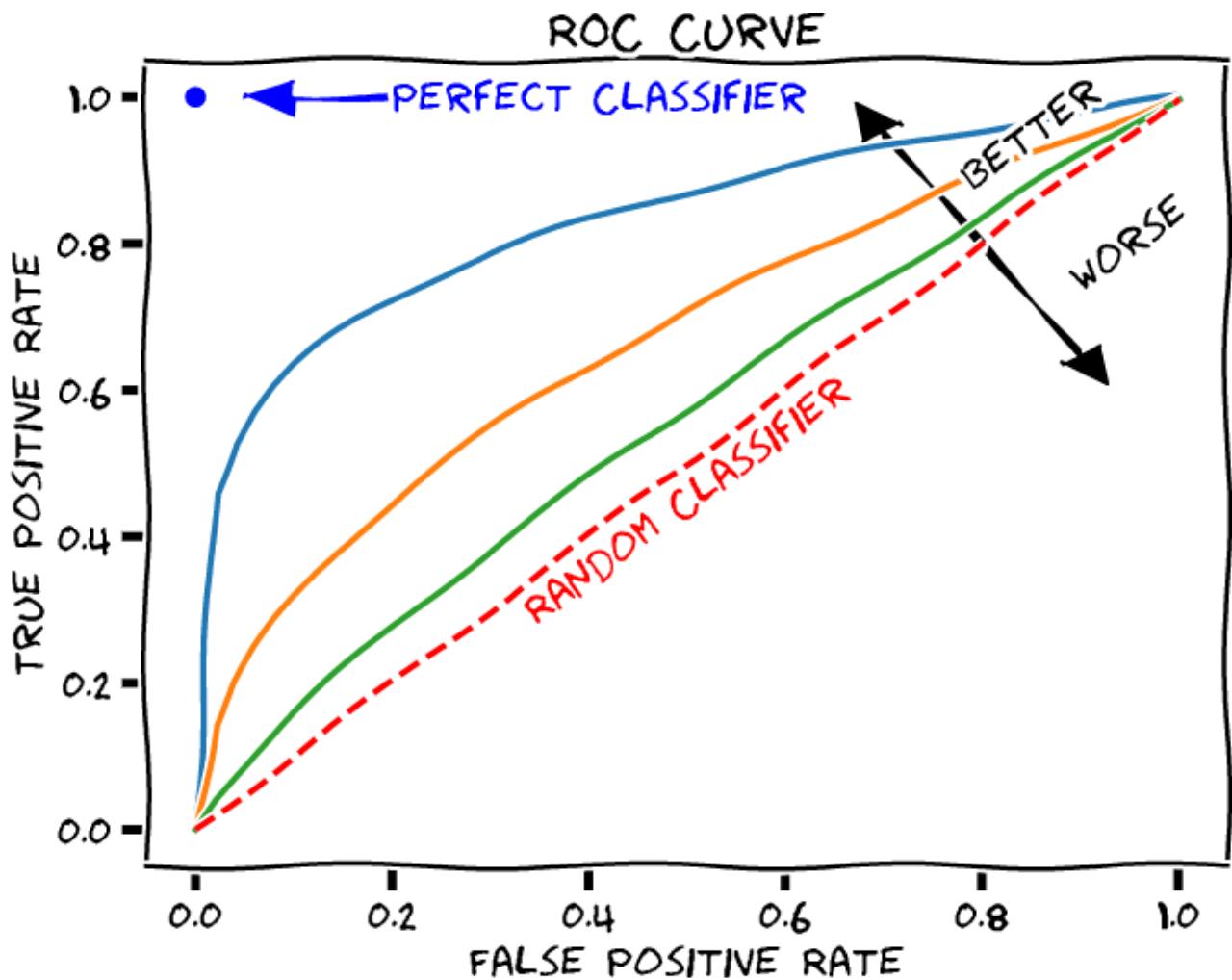
ເລີນວ່າ F1-score ມາຄົວໜາ.

print(classification_report(y_test, predictions))

support 04.

∴ ຂົດກ່າວງໃຫຍ່ກ່າວງກັນ

ROC & AUC



imgae from: <https://commons.wikimedia.org/wiki/File:Roc-draft-xkcd-style.svg>

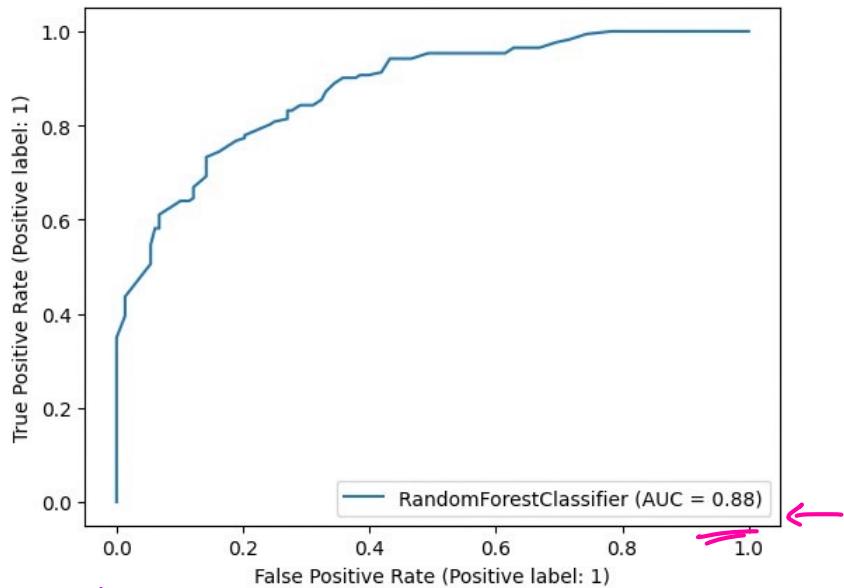
- อีกวิธีในการวัดประสิทธิภาพของ model
- ROC - Receiver Operating Characteristic
- AUC - Area Under the Curve *wn. 9 ตาราง*
- เรียกใช้ RocCurveDisplay

AUC < 0.5 (50%)
= แปลงเป็นภาษาไทย
เตาหูมีผู้คนมากกว่า

```
from sklearn.metrics import RocCurveDisplay, roc_auc_score
```

- แสดงกราฟ

```
RocCurveDisplay.from_estimator(rf, X_test, y_test)
plt.show()
```



→ စုက်.အောင်နှာ ပါယာ။ Wine

rf.predict_proba(X_test)[:,1]

တော်း col index 1

= ပါယာ Wine ပါယာ
အကဲ လုပ်က.အောင်နှာ
30% (ပါယာ ၇၀%
အနောက် ပါယာ Wine
သမား ပါယာ)

တို့ကို တွက်နောက် ပါယာ မျှတော်း ပါယာ မျှတော်း၊ ဒါ ဘုံး ပေးပို့
သလုပ်လောက်

```
array([0.30487805, 0.97560976, 0.75609756, 0.02439024, 0.13414634,
       0.5 , 0.86585366, 0.37804878, 0.54878049, 0.19512195,
       0.29268293, 0.85365854, 0.64634146, 0.54878049, 0.24390244,
       0.15853659, 0.51219512, 0.04878049, 0.86585366, 0.54878049,
       0.8902439 , 0.51219512, 0.19512195, 0.1097561 , 0.5 ,
       1. , 0.13414634, 0.87804878, 0.12195122, 0.96341463,
       0.48780488, 0.7195122 , 0.18292683, 0.87804878, 0.1097561 ,
       0.7804878 , 0.37804878, 0.63414634, 0.3902439 , 0.30487805,
       0.91463415, 0.97560976, 0.45121951, 0.31707317, 0.64634146,
       0.1097561 , 0.30487805, 0.92682927, 0.86585366, 0.41463415,
       0.14634146, 0.6097561 , 0.69512195, 0.97560976, 0.84146341,
       0.90243902, 0.87804878, 0.7195122 , 0.92682927, 0.20731707,
       0.1097561 , 0.40243902, 0.96341463, 0.65853659, 0.07317073,
       0.51219512, 0.53658537, 0.64634146, 1. , 0.53658537,
       0.13414634, 0.68292683, 0.84146341, 0.19512195, 0.1097561 ,
       0.67073171, 0.46341463, 0.3902439 , 0.84146341, 0.26829268,
       0.70731707, 0.7804878 , 0.81707317, 0.1097561 , 0.1097561 ,
       0.84146341, 0.92682927, 0.20731707, 0.75609756, 1. ,
       0.80487805, 0.81707317, 0.34146341, 0.26829268, 0.43902439,
       0.73170732, 0.95121951, 0.12195122, 0.85365854, 0.04878049,
       0.64634146, 0.3902439 , 0.08536585, 0.58536585, 0.18292683,
       0.86585366, 0.91463415, 0.95121951, 0.06097561, 0.01219512,
       0.95121951, 0.7804878 , 0.31707317, 0.37804878, 0.58536585,
       0.30487805, 0.79268293, 0.51219512, 0.97560976, 0.30487805,
       0.01219512, 0.81707317, 1. , 0.96341463, 0.17073171,
       0.81707317, 0.93902439, 0.15853659, 0.2195122 , 0.2804878 ,
       0.20731707, 0.20731707, 1. , 0.26829268, 0.53658537,
       0.75609756, 0.56097561, 0.93902439, 0.62195122, 0.12195122,
       0.30487805, 0.13414634, 0.43902439, 0.08536585, 0.93902439,
       0.24390244, 0.86585366, 0.25609756, 0.12195122, 1. ,
       0.7804878 , 0.85365854, 0.2195122 , 0.06097561, 0.81707317,
       0.68292683, 0.93902439, 0.69512195, 0.54878049, 0.52439024,
       0.43902439, 0.51219512, 0.3902439 , 0.98780488, 0.96341463,
       0.92682927, 0.75609756, 0.18292683, 0.47560976, 0.97560976,
       0.95121951, 0.42682927, 0.56097561, 0.64634146, 0.51219512,
       0.2195122 , 0.86585366, 0.97560976, 1. , 0.80487805,
       0.29268293, 0.15853659, 0.63414634, 0.51219512, 0.07317073,
       0.80487805, 0.51219512, 0.14634146, 0.08536585, 0.79268293,
       0.76829268, 0.20731707, 0.64634146, 0.12195122, 0.85365854,
       0.06097561, 0.20731707, 0.93902439, 0.84146341, 0.95121951,
```

```
0.95121951, 0.87804878, 0.17073171, 0.23170732, 0.48780488,  
0.48780488, 0.93902439, 0.42682927, 0.63414634, 0.80487805,  
0.62195122, 0.87804878, 0.63414634, 0.09756098, 0.31707317,  
0.14634146, 0.25609756, 0.91463415, 0.95121951, 0.93902439,  
0.73170732, 0.96341463, 0.8902439, 0.93902439, 0.32926829,  
0.95121951, 0.1097561, 0.35365854, 0.19512195, 0.8902439,  
0.56097561, 0.18292683, 0.13414634, 0.17073171, 0.03658537,  
0.73170732, 0.95121951, 0.51219512, 0.48780488, 0.3902439,  
0.40243902, 0.98780488, 0.95121951, 0.96341463, 0.34146341,  
0.8902439, 0.20731707, 0.73170732, 0.18292683, 0.35365854,  
1, 0.18292683, 0.24390244, 0.02439024, 0.47560976,  
0.51219512, 0.75609756, 0.02439024, 0.29268293, 0.7195122,  
0.32926829, 0.91463415, 0.42682927, 0.85365854, 0.81707317,  
0.85365854, 0.92682927, 0.84146341, 0.07317073, 0.70731707,  
0.95121951, 0.91463415, 0.64634146, 0.36585366, 0.40243902,  
0.43902439, 0.14634146, 0.12195122, 0.41463415, 0.01219512,  
0.31707317, 0.79268293, 0.41463415, 0.86585366, 0.96341463,  
0.7195122, 0.13414634, 0.2804878, 0.41463415, 0.7804878,
```

9 auc score:

```
roc_auc_score(y_test, rf.predict_proba(X_test)[:,1])
```

Code n: 9

0.8773570081709617

Model នានា នីមួយៗ

```
from sklearn.model_selection import "cross_val_score"  
X, y នូវនៅរាយការណ៍បង្កើតបង្កើត (នានាដំឡើង)  
cross_val_score(rf, X, y, cv=5).mean() = ឲ្យបង្កើតបង្កើតមែនវាដំឡើង
```

e.g. ឲ្យ 5 នៅ

សម 1 : 2-5 វាដំឡើង 1 នៅ test
" 2 : 1,3-5 — 2 —
" 3 : 1-2,4-5 — 3 —

1/5 នៅ mean នៃ 5 សែនុកា 0.7267065047021944

5 នៅ, 4 នៅ train
5 នៅ + 1 នៅ test

73%

ឲ្យនានាដំឡើងនឹងរាយការណីរបស់វា. វាដំឡើង

```
scores = []  
for n in range(50, 101): នីម 50 នូវ 100  
    rf = RandomForestClassifier(n_estimators=n, random_state=555)  
    scores.append(cross_val_score(rf, X, y, cv=5).mean())  
max_score = max(scores)  
max_index = scores.index(max_score)+50  
print(max_index, max_score)
```

98 0.7323354231974921

↓
n_estimators នៅ accuracy 93%
នឹងការបង្កើត
សម្រាប់

Colab paid products - Cancel contracts here