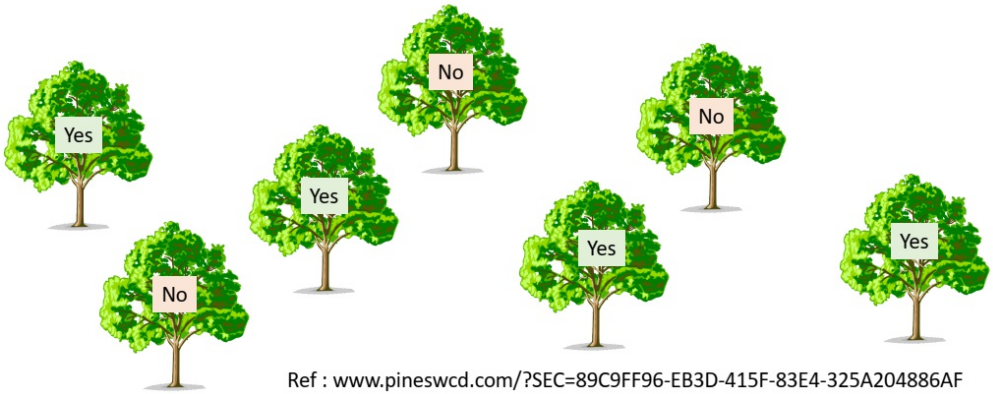
Ramdom Forest (สร้างป่า)

ตัวต่อของ Decision Tree(สร้างต้นไม้) : เพื่อลดความ Overfit

การทำ Decision Tree ให้ได้หลายๆ ต้นเพื่อหา Majority Vote (เหมือนคะแนนโหวตเลือกตั้ง)



Ref : www.pineswcd.com/?SEC=89C9FF96-EB3D-415F-83E4-325A204886AF

เช่น ในรูปมี Yes มากกว่า No ผลลัพธ์เลยเลือกตอบ Yes

ปกติการสร้างต้นไม้ Decision Tree เราจะหยิบ Feature มาเป็นหัวข้อคำถาม แล้วแตกย่อยแต่ละ Feature ให้ได้ 1 ต้นที่คิดว่าดีที่สุด

แล้วจะเลือกสร้างต้นไม้หลายต้นยังไง จากข้อมูล Feature เท่าเดิม => Ramdom Feature มาใช้แต่ละต้น

$Data =$

$X1$	$X2$	$\cdots$	XD	Y
x_1^1	x_1^2	$\cdots$	x_1^D	y_1
x_2^1	x_2^2	$\cdots$	x_2^D	y_2
x_3^1	x_3^2	$\cdots$	x_3^D	y_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_N^1	x_N^2	$\cdots$	x_N^D	y_N

X คือ ตัวแปรต้น (Feature)

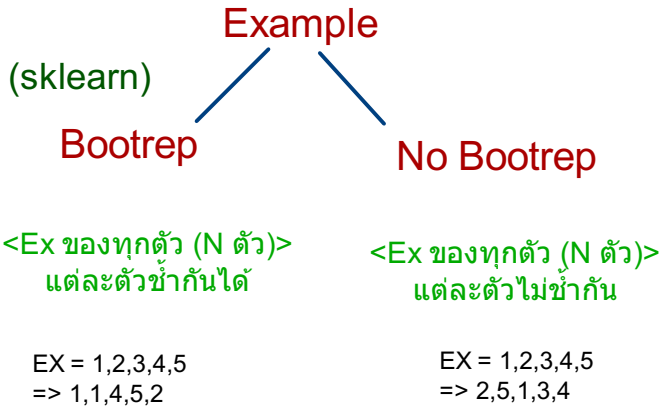
Y คือ ตัวแปรตาม (Target)

N คือ จำนวนตัวอย่างทั้งหมด

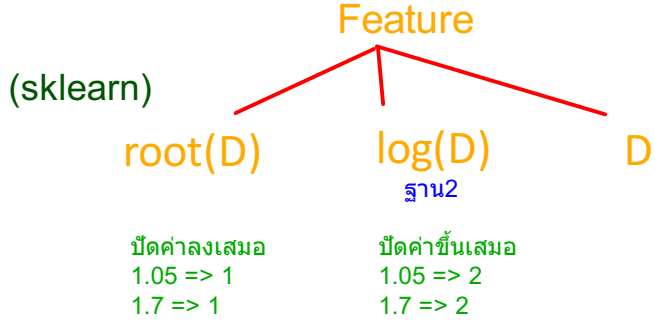
D คือ จำนวน Feature ทั้งหมด

เป็นการ Ramdom ที่ทุกตัวมีโอกาสเท่ากัน (Unifom)

การ Ramdom Example



การ Ramdom Feature (เพื่อเลือกใช้แต่ละครั้ง)



ใน sklearn เวลา run code กับข้อมูลชุดเดิม
จะได้ผลลัพธ์เหมือนเดิมเสมอ เพราะ Fix ค่า Ramdom

Code ที่ใช้เรียนในครั้งนี้ จะ Ramdom
ตลอดทุกครั้ง จึงจะได้ค่าไม่เท่าเดิม

เพศ	อายุ	ซื้อคอม ?
หญิง	40	ไม่ซื้อ
หญิง	50	ไม่ซื้อ
ชาย	20	ไม่ซื้อ
ชาย	40	ซื้อ
ชาย	50	ซื้อ
หญิง	20	ซื้อ

Bootrep <Ex ของทุกตัว (N ตัว)>
แต่ละตัวซ้ำกันได้

root(D) => D = 2 => root(2) = 1.4 ปัดลงได้ 1

N

D=1

เพศ	ซื้อคอม ?
หญิง	ไม่ซื้อ
หญิง	ไม่ซื้อ
ชาย	ไม่ซื้อ
ชาย	ซื้อ
ชาย	ซื้อ
หญิง	ซื้อ

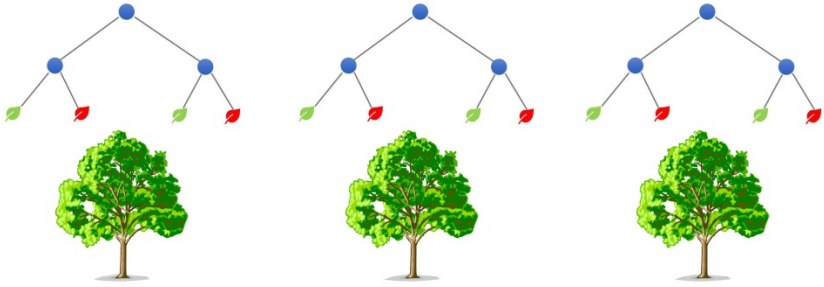
D=1

อายุ	ซื้อคอม ?
40	ไม่ซื้อ
50	ไม่ซื้อ
20	ไม่ซื้อ
40	ซื้อ
50	ซื้อ
20	ซื้อ

D=1

เพศ	ซื้อคอม ?
หญิง	ไม่ซื้อ
หญิง	ไม่ซื้อ
ชาย	ไม่ซื้อ
ชาย	ซื้อ
ชาย	ซื้อ
หญิง	ซื้อ

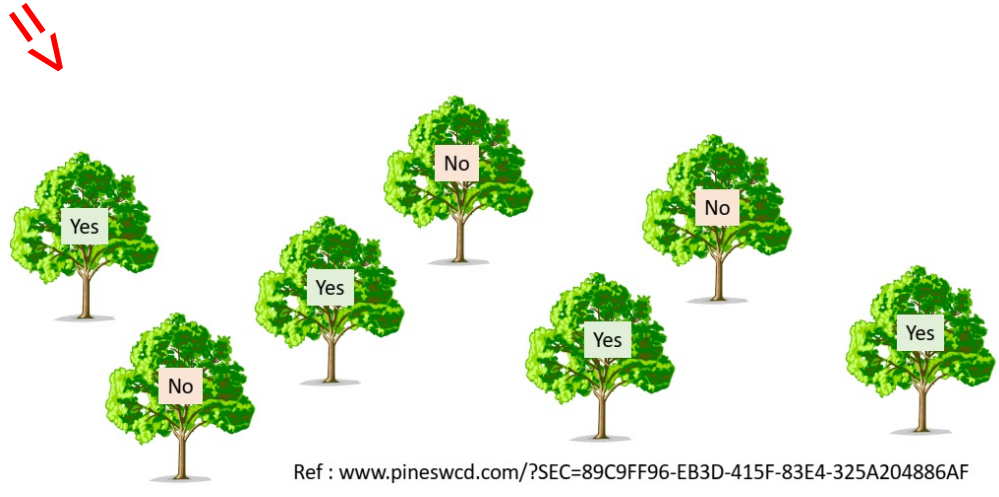
Ref : www.pineswcd.com/?SEC=89C9FF96-EB3D-415F-83E4-325A204886AF



ข้อมูล Test

เพศ	อายุ	ซื้อคอม ?
หญิง	45	?

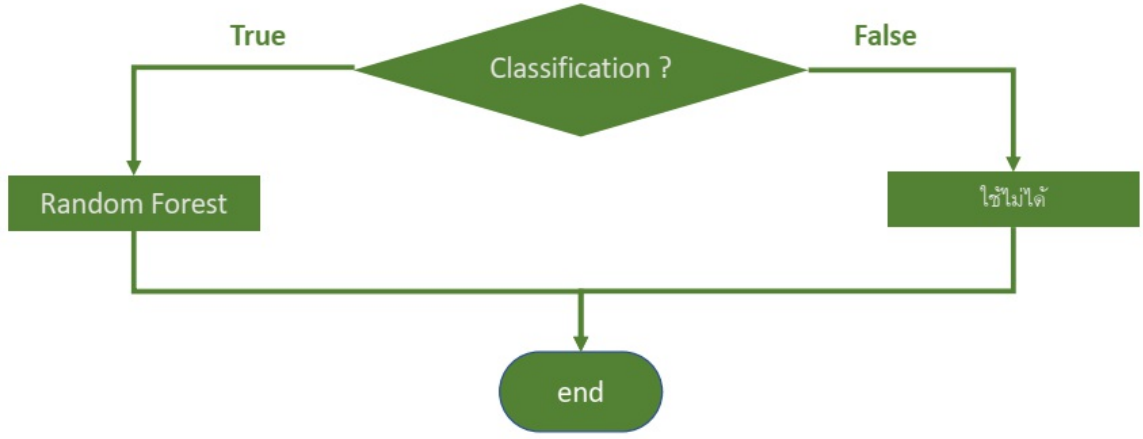
Ref : www.pineswcd.com/?SEC=89C9FF96-EB3D-415F-83E4-325A204886AF



Ref : www.pineswcd.com/?SEC=89C9FF96-EB3D-415F-83E4-325A204886AF

Random Forest : Code

I. เรียนรู้
II. พยากรณ์



I. เรียนรู้ (สร้าง Tree)

```
def RF_fit(X_Train, Y_Train, Feature_Name, All_Class, n_tree=10, boostrap=True, max_depth=np.inf, depth=1,
           max_majority=np.inf, min_leaf=-np.inf):
    N, D = X_Train.shape
    if boostrap == True:                          Random (Bootrep ซ้ำได้)
        example_index = np.random.choice(N, N)
        X_Train = X_Train[example_index]
        Y_Train = Y_Train[example_index]
    forest = []
    for i in range(n_tree):                       Random Feature =>root(D)
        n_filter_feature = int(np.sqrt(D))
        feature_index = random.sample(range(D), n_filter_feature)
        Feature_to_forest = Feature_Name[feature_index]
        X_to_forest = X_Train[:, feature_index]
Decision Tree =>  tree = DT_fit(X_to_forest, Y_Train, Feature_to_forest, feature_index, All_Class, max_depth=max_depth,
                      max_majority=max_majority, min_leaf=min_leaf)
        forest.append(tree)                       index ของ Feature ที่ Random มา (คอลัมน์ไหนบ้าง)
    return forest
```

```
np.random.choice(5, 5) # np.arange(5) , size 3:
```

```
array([3, 3, 4, 0, 2])
```

```
range(5)
```

```
range(0, 5)
```

```
print(random.sample(range(5),3))
```

```
[3, 1, 0]
```

```python
def DT_find_best_question(X, Y, Feature_Name, Feature_Index, All_Class):
    max_Gain = -np.inf
    isComplete = False
    Gini_Parent = DT_compute_Gini(Y, All_Class)
    Question_Dict = DT_create_Question(X, Feature_Name)
    for d, fn in enumerate(Feature_Name):
        N = X.shape[0]
        if fn in Question_Dict:
            unique_value = Question_Dict[fn]['unique_value']
            check_type = Question_Dict[fn]['type_of_feature']
            for i, uv in enumerate(unique_value):
                filter_true, filter_false = DT_find_filter(X, check_type, d, uv)
                X_True = X[filter_true]; Y_True = Y[filter_true];
                X_False = X[filter_false]; Y_False = Y[filter_false];
                weight_true, weight_false = DT_compute_weight_true_false(filter_true, filter_false, N)
                Gini_True, Gini_False = DT_compute_Gini_True_False(Y_True, Y_False, All_Class)
                Gini_Children = DT_compute_Gini_Children(weight_true, Gini_True, weight_false, Gini_False)
                Gain = DT_compute_Gain(Gini_Parent, Gini_Children)
                if Gain >= max_Gain:
                    max_Gain = Gain
                    best = {}
                    best['fn'] = fn
                    best['findex'] = Feature_Index[d]  ## เพิ่มตรงนี้มาโดยเฉพาะ (ค่าถามที่ดีสุดอยู่ Index ไหน)
                    best['uv'] = uv
                    best['X_True'] = X_True
                    best['Y_True'] = Y_True
                    best['X_False'] = X_False
                    best['Y_False'] = Y_False
                if max_Gain == Gini_Parent:
                    isComplete = True
                    return best, isComplete
    return best, isComplete
```

II. พยากรณ์ (For loop ต้นไม้แต่ละต้น เพื่อได้ผลลัพธ์แต่ละต้น)

Decision Tree =>

เป็นเปอร์เซ็นต์เรียบร้อย

```
def RF_predict(X_Test, forest):
    N = X_Test.shape[0]
    n_tree = len(forest)
    Yhat_forest = np.empty([N, 0])
    for i in range(n_tree):
        tree = forest[i]
        Yhat_each_tree = DT_predict(X_Test, tree)[:, 0:1]
        Yhat_forest = np.hstack([Yhat_forest, Yhat_each_tree])
    Yhat_Test = []
    for Y in Yhat_forest:
        unique_prediction, count_unique_prediction = np.unique(Y, return_counts=True)
        prediction = unique_prediction[count_unique_prediction.argmax()]
        percent = count_unique_prediction.max()/n_tree
        yhat_test = np.array([prediction, percent])
        Yhat_Test.append(yhat_test)
    return np.array(Yhat_Test)
```

```
>>> a = np.array((1,2,3))
>>> b = np.array((2,3,4))
>>> np.hstack((a,b))
array([1, 2, 3, 2, 3, 4])
>>> a = np.array([[1],[2],[3]])
>>> b = np.array([[2],[3],[4]])
>>> np.hstack((a,b))
array([[1, 2],
       [2, 3],
       [3, 4]])
```

```
>>> np.unique([1, 1, 2, 2, 3, 3])
array([1, 2, 3])
>>> a = np.array([[1, 1], [2, 3]])
>>> np.unique(a)
array([1, 2, 3])
```