Modeling Algorithms

Tree Ensemble Classification

นภัสกรณ์ รสหวาน 62070505207

พุฒิเมธ หิรัณย์อุฬาร 62070505212

เกรียงไกร แซ่ตั้น 62070505222

ปภาวิน ศักดาเพชรศิริ 62070505224

ฌานเมธ อัครกิตติโชค 62070505229

รายงานนี้เป็นส่วนหนึ่งของรายวิชา CPE378 MACHINE LEARNING

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

ปีการศึกษา 2564/2

Outline

1. **Introduction**
    a. Comparing of modeling
    b. Data source
    c. Tool used
2. **Exploratory Data Analysis**
    a. Preprocessing
3. **Models**
    a. Ensemble learning
        i. Bagging
        ii. Boosting
    b. Decision tree
    c. Random forest
    d. Adaboost
4. **Discussion**
    a. Advantages and disadvantages of Decision tree
    b. Advantages and disadvantages of Random Forest
    c. Advantages and disadvantages of Adaboost

Introduction

แบบจำลองการเรียนรู้แบบรวมกลุ่ม (Ensemble Learning) เป็นหนึ่งในเทคนิคของการเรียนรู้ ด้วยเครื่อง (Machine learning) ที่สามารถนำมาประยุกต์ใช้ได้ทั้งในงานด้านการจำแนก (Classification) และในงานด้านการถดถอย (Regression) ซึ่งในแบบจำลองการเรียนรู้แบบรวมกลุ่ม มีการใช้ตัวจำแนก (Classifier) มากกว่าหนึ่งตัวในการเรียนรู้ ซึ่งแต่ละตัวจำแนกจะมีกระบวนการทำงานของตัวเองและทุกตัวจำแนกจะกระทำกับข้อมูลเดียวกัน เมื่อได้ผลจากการจำแนกของแต่ละตัวจำแนกแล้ว จะนำผลลัพธ์เหล่านั้นมาผ่านวิธีการรวบรวม (Combination หรือ Vote) และนำไปตัดสินใจในขั้นตอนสุดท้าย โดยเทคนิคดังกล่าวมีจุดมุ่งหมายในการเพิ่มประสิทธิภาพความแม่นยำให้กับแบบจำลองการเรียนรู้ ซึ่งในการศึกษานี้ได้มีการหยิบเทคนิคการทำแบบจำลองการเรียนรู้แบบกลุ่มในงานด้านการจำแนก (Ensemble Classification) มาทั้งหมด 2 แบบ คือ Bagging โดยการใช้แบบจำลองการจำแนกประเภทการสุ่มป่าไม้ (Random Forest) และ Boosting โดยการใช้แบบจำลอง AdaBoost ซึ่งจะมีการนำมาเปรียบเทียบกับแบบจำลองการเรียนรู้ แบบต้นไม้ตัดสินใจ (Decision Tree) ซึ่งเป็นแบบจำลองการเรียนรู้ด้วยเครื่องแบบเชิงเดี่ยว โดยมีจุดประสงค์เพื่อต้องการศึกษาและเปรียบเทียบประสิทธิภาพการทำงานของแบบจำลองการเรียนรู้แบบรวมกลุ่มและแบบจำลองการเรียนรู้ด้วยเครื่องแบบเชิงเดี่ยว

Data source

1. **Real-word dataset**

การศึกษานี้ได้ใช้ข้อมูล Telecom-churn-prediction ที่ได้มีการนำข้อมูลมาจากแหล่งข้อมูล ในเว็บไซต์ Kaggle (https://www.kaggle.com/code/bandiatindra/telecom-churn-prediction/data) ซึ่งเป็นข้อมูลรายละเอียดของลูกค้าที่ใช้บริการบริษัทโทรคมนาคมที่สามารถนำข้อมูลมาวิเคราะห์ และทำแบบจำลองสำหรับการจำแนกโอกาสการต่อสัญญาของลูกค้าโดยมีการจำแนกทั้งหมด 2 รูปแบบคือ การต่อสัญญาหรือการไม่ต่อสัญญา

Exploratory Data Analysis

กระบวนการวิเคราะห์ข้อมูลโดยใช้เทคนิคการมองเห็น (Visualization) เพื่อเป็นการสำรวจแนวโน้ม หรือรูปแบบของข้อมูลเบื้องต้นก่อนทำการวิเคราะห์ข้อมูล

```
df.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSe
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	

5 rows × 21 columns

```
df.shape
```

```
(7043, 21)
```

ข้อมูล Telcom Customer Churn แต่ละแถวแสดงถึงลูกค้า แต่ละคอลัมน์ประกอบด้วยตัวแปรของลูกค้าแต่ละคน โดยข้อมูลดิบประกอบด้วย 7043 แถว คือลูกค้า 7043 คน โดยความหมายของคอลัมน์ ในแต่ละคอลัมน์หมายถึงดังนี้

1. customerID: รหัสลูกค้า
2. gender: ลูกค้าเป็นเพศชายหรือหญิง
3. SeniorCitizen: ลูกค้าจะเป็นผู้สูงอายุหรือไม่ (1, 0)
4. Partner: มีลูกหรือไม่ (Yes, No)
5. Dependents: มีผู้อยู่ในอุปการะหรือไม่ (Yes, No)
6. tenure: จำนวนเดือนที่ลูกค้าใช้บริการในบริษัท
7. PhoneService: ลูกค้ามีบริการโทรศัพท์หรือไม่ (Yes, No)
8. MultipleLines: ลูกค้ามีใช้โทรศัพท์หลายสายหรือไม่ (Yes, No, No phone service)
9. InternetService: ผู้ให้บริการอินเทอร์เน็ตของลูกค้า (DSL, Fiber optic, No)
10. OnlineSecurity: ลูกค้ามี online security หรือไม่ (Yes, No, No internet service)
11. OnlineBackup: ลูกค้ามีการสำรองข้อมูลออนไลน์หรือไม่ (Yes, No, No internet service)
12. DeviceProtection: ลูกค้ามีการป้องกันอุปกรณ์หรือไม่ (Yes, No, No internet service)
13. TechSupport: ลูกค้าจะมีการสนับสนุนด้านเทคนิคหรือไม่ (Yes, No, No internet service)
14. StreamingTV: ลูกค้ามีทีวีสตรีมมิ่งหรือไม่ TV or not (Yes, No, No internet service)

15. StreamingMovies: ลูกค้ามีสตรีมมิ่งภาพยนตร์หรือไม่ (Yes, No, No internet service)
16. Contract: อายุสัญญาของลูกค้า (Month-to-month, One year, Two year)
17. PaperlessBilling: ลูกค้ามีการเรียกเก็บเงินแบบไร้กระดาษหรือไม่t (Yes, No)
18. PaymentMethod: วิธีการชำระเงินของลูกค้า (Electronic check, mailed check, Bank transfer (automatic), Credit card (automatic))
19. MonthlyCharges: จำนวนเงินที่เรียกเก็บกับลูกค้ารายเดือน
20. TotalCharges: จำนวนเงินทั้งหมดที่เรียกเก็บจากลูกค้า
21. Churn: ลูกค้าต่อสัญญาหรือไม่ (Yes or No)

ซึ่งเป้าหมายที่เราจะใช้เป็นแนวทางในการสำรวจข้อมูลชุดนี้คือ Churn โดยแต่ละคอลัมน์สามารถอธิบายข้อมูลได้ดังนี้

ความหมาย	**ชื่อคอลัมน์**
ลูกค้าที่ออกในเดือนที่แล้ว	Churn
บริการที่ลูกค้าแต่ละรายสมัครใช้งาน	phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
ข้อมูลบัญชีลูกค้า	tenure, contract, payment method, paperless billing, monthly charges, and total charges
ข้อมูลทั่วไปของลูกค้า	gender, age range, partners and dependents

สำรวจคอลัมน์และประเภทของชุดข้อมูลด้วย ฟังก์ชัน info() จากรูปจะสังเกตได้ว่าไม่มีข้อมูลที่ขาดหายไป

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

โดยเมื่อสำรวจวิเคราะห์ข้อมูลในเชิงลึก จะสังเกตเห็นถึงการขาดหายไปในทางอ้อมของชุดข้อมูลที่อยู่ใน TotalCharges ซึ่งมีค่าหายไป 11 ค่า

```
df['TotalCharges'] = pd.to_numeric(df.TotalCharges, errors='coerce')
df.isnull().sum()
```

```
gender            0
SeniorCitizen     0
Partner           0
Dependents        0
tenure            0
PhoneService      0
MultipleLines     0
InternetService   0
OnlineSecurity    0
OnlineBackup      0
DeviceProtection  0
TechSupport       0
StreamingTV       0
StreamingMovies   0
Contract          0
PaperlessBilling  0
PaymentMethod     0
MonthlyCharges    0
TotalCharges      11
Churn             0
dtype: int64
```

เมื่อตรวจสอบใน TotalCharges พบว่ายังมี tenure หรือก็คือจำนวนเดือนที่ลูกค้าใช้บริการเป็น 0 ซึ่งขัดแย้งกับการมี MonthlyCharges หรือ ค่าบริการรายเดือน ซึ่งหมายถึงการที่ไม่ใช้บริการแต่มีค่าบริการรายเดือน

```
df[np.isnan(df['TotalCharges'])]
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	S
488	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	No	Yes	Yes	
753	Male	0	No	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	
936	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	Yes	Yes	No	
1082	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	
1340	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	Yes	Yes	Yes	
3331	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service	No internet service	
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service	No internet service	
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Yes	Yes	Yes	
6754	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Yes	No	Yes	

PaperlessBilling	PaymentMethod	MonthlyCharges	Tota
Yes	Bank transfer (automatic)	52.55	
No	Mailed check	20.25	
No	Mailed check	80.85	
No	Mailed check	25.75	
No	Credit card (automatic)	56.05	
No	Mailed check	19.85	
No	Mailed check	25.35	
No	Mailed check	20.00	
Yes	Mailed check	19.70	
No	Mailed check	73.35	
Yes	Bank transfer (automatic)	61.90	

เมื่อสำรวจเพิ่มเติมพบว่ามีเพียง 11 คอลัมน์ที่ข้อมูล tenure ขาดหายไป ดังนั้นการลบข้อมูลเหล่านี้จึงไม่ส่งผลกระทบต่อข้อมูล

```
[14] df[df['tenure'] == 0].index

     Int64Index([488, 753, 936, 1082, 1340, 3331, 3826, 4380, 5218, 6670, 6754], dtype='int64')

[15] df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)
     df[df['tenure'] == 0].index

     Int64Index([], dtype='int64')
```

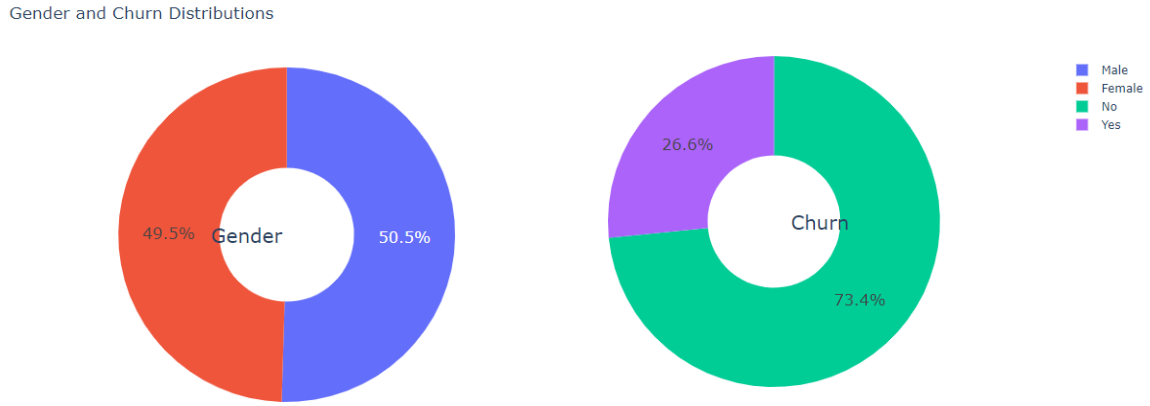
สำรวจข้อมูลหลังจากการลบข้อมูลที่ error ออกไปพบว่าไม่มีข้อมูลที่ขาดหายไป

```
df.isnull().sum()
```

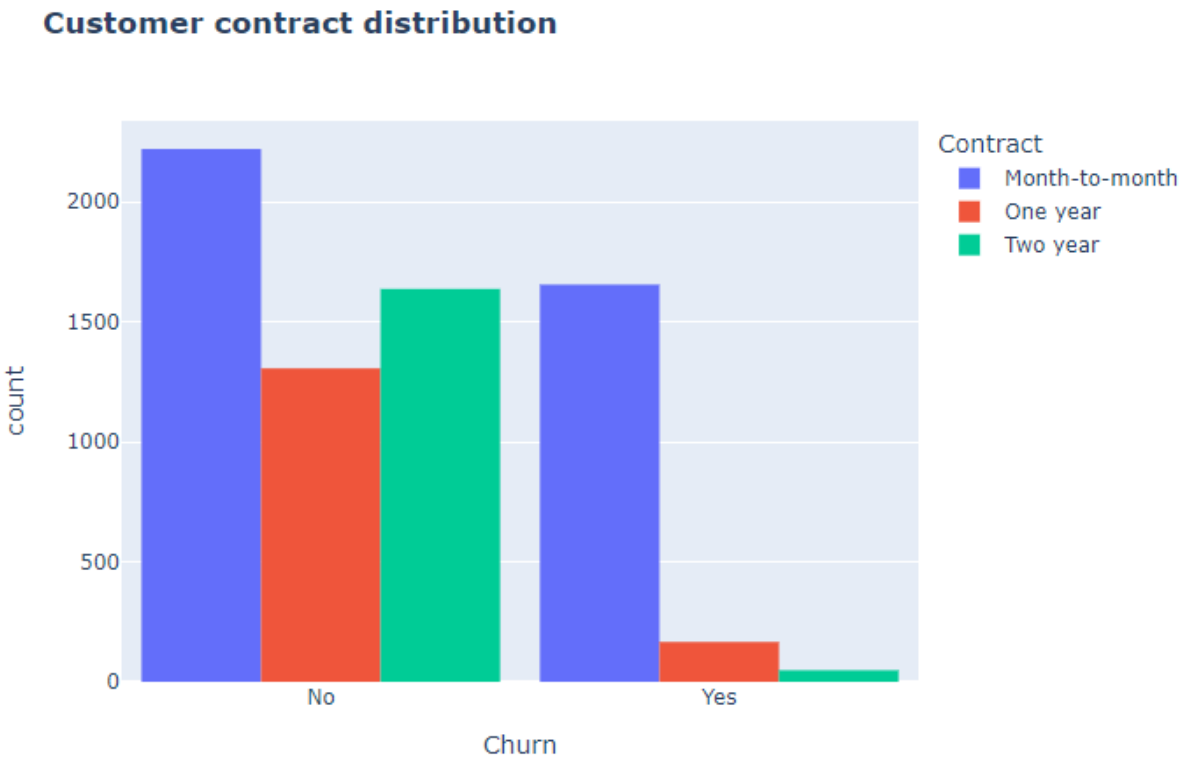
```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

Data Visualization

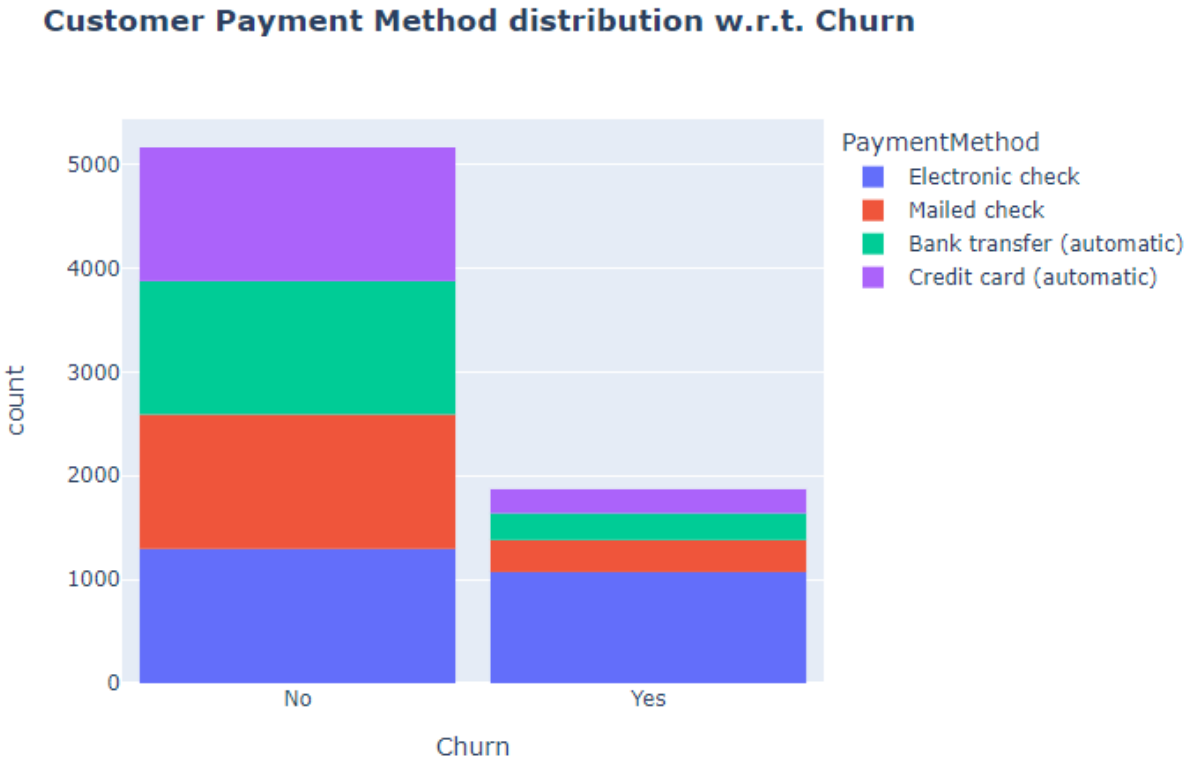
- ลูกค้าเป็นเพศชาย 50.5% และเพศหญิง 49.5%
- ลูกค้า 26.6% ย้ายไปใช้บริษัทอื่น



- ลูกค้าที่มีสัญญาแบบรายเดือนมีจำนวนการย้ายที่มากกว่าสัญญาแบบหนึ่งปีหรือสองปี

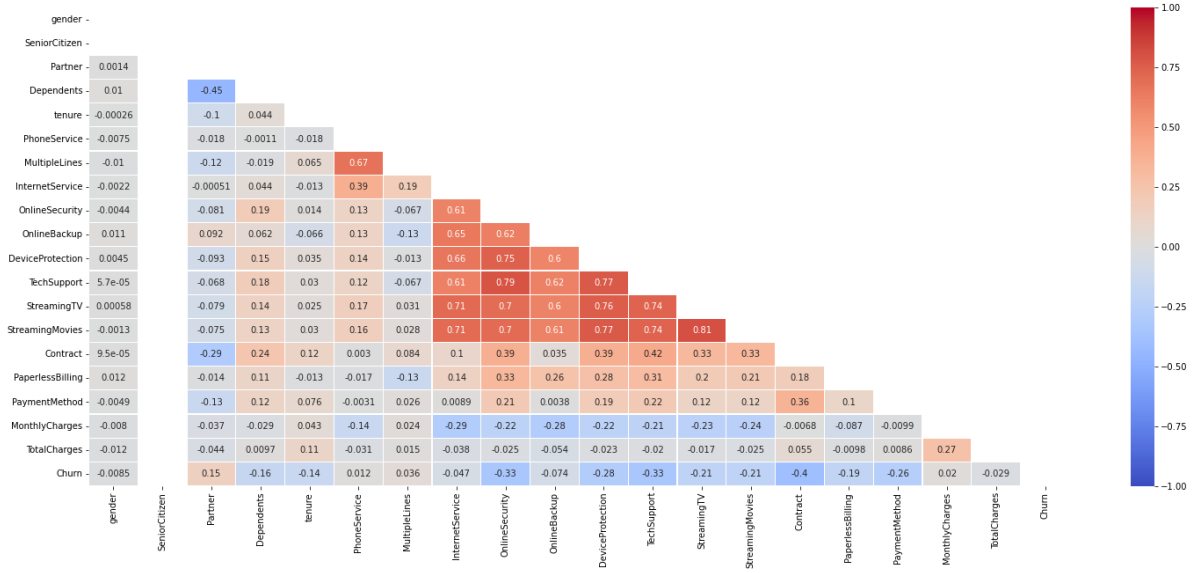


- ลูกค้าที่ย้ายออกส่วนมากใช้การชำระเงินแบบ Electronic Check



Correlation Heatmap

การวิเคราะห์สหสัมพันธ์เป็นการศึกษาความสัมพันธ์ระหว่างตัวแปร 2 ตัว ในทุกๆคู่ของตัวแปร เมื่อทำการวาด Correlation Heatmap โดยเพื่อหาความสัมพันธ์ระหว่างตัวแปรอื่นๆกับตัวแปรที่สนใจคือ Churn พบว่า MonthlyCharges มีความสัมพันธ์กับ Churn มากที่สุดคือ 0.1928




```
plt.figure(figsize=(14,7))
df.corr()['Churn'].sort_values(ascending = False)
```

```
Churn             1.000000
MonthlyCharges    0.192858
PaperlessBilling  0.191454
SeniorCitizen     0.150541
PaymentMethod     0.107852
MultipleLines     0.038043
PhoneService      0.011691
gender           -0.008545
StreamingTV      -0.036303
StreamingMovies  -0.038802
InternetService  -0.047097
Partner          -0.149982
Dependents       -0.163128
DeviceProtection -0.177883
OnlineBackup     -0.195290
TotalCharges     -0.199484
TechSupport      -0.282232
OnlineSecurity   -0.289050
tenure           -0.354049
Contract         -0.396150
Name: Churn, dtype: float64
```

Data Preprocessing

เป็นขั้นตอนการเตรียมข้อมูลให้เหมาะสมแก่การนำไปใช้งาน โดยการแปลงข้อความเป็นตัวเลข เพื่อให้เหมาะสมต่อการนำเข้าสู่การฝึกโมเดล

```
[22] def object_to_int(dataframe_series):
       if dataframe_series.dtype=='object':
         dataframe_series = LabelEncoder().fit_transform(dataframe_series)
       return dataframe_series
```

```
[23] df = df.apply(lambda x: object_to_int(x))
     df.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	Online
0	0	0	1	0	1	0	1	0	0	
1	1	0	0	0	34	1	0	0	2	
2	1	0	0	0	2	1	0	0	2	
3	1	0	0	0	45	0	1	0	2	
4	0	0	0	0	2	1	0	1	0	

แบ่งข้อมูลเป็น X เพื่อนำมาทำนายค่า y(Churn) เป็น train 70% และ test 30%

```
[26] X = df.drop(columns = ['Churn'])
     y = df['Churn'].values
```

```
[27] X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30, random_state = 40, stratify=y)
```

2. Synthetic datasets

การศึกษานี้ได้ใช้ข้อมูล 2 Class 2D with Noise จากการสร้างข้อมูลโดยใช้ sklearn.datasets import make_classification โดยมีการสร้าง noise เพิ่มเติม เพื่อทดสอบประสิทธิภาพของโมเดลในการจำแนกข้อมูล โดยมี noise ในกรณีที่ข้อมูลของเรามีข้อมูลรอบกวนในโลกความจริง

Generate Data

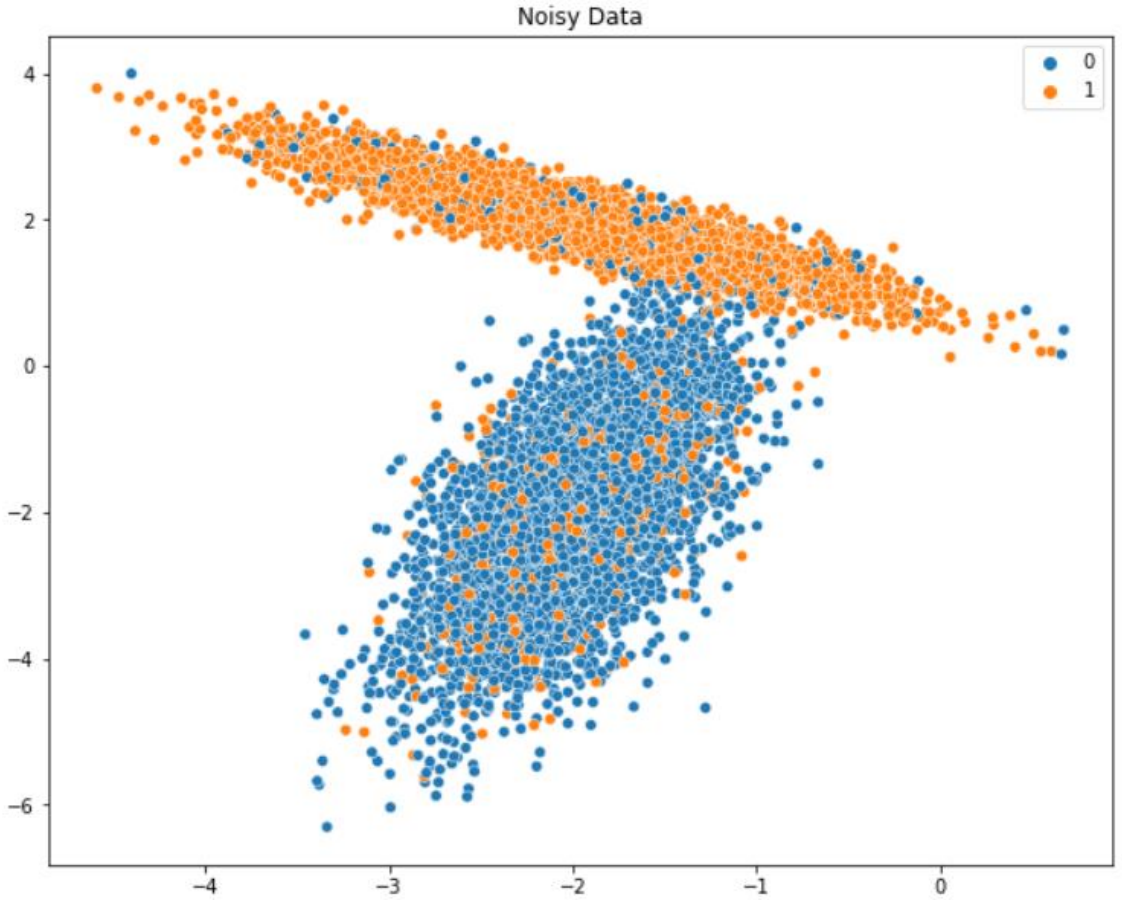
เราสร้างชุดข้อมูลจำนวน 10000 ข้อมูล แบ่งเป็น 2 class โดยมีจุดรอบกวน 0.2 โดยใช้คำสั่งดังนี้

```
X1, Y1 = make_classification(n_samples=10000, n_features=2, n_informative=2,
                             n_redundant=0, n_repeated=0, n_classes=2, n_clusters_per_class=1,
                             class_sep=2,flip_y=0.2,weights=[0.5,0.5], random_state=17)
```

Data Visualization

แสดง Visualization ของข้อมูลที่สร้างขึ้นมา

```
f, (ax1) = plt.subplots(nrows=1, ncols=1,figsize=(10,8))
sns.scatterplot(X1[:,0],X1[:,1],hue=Y1,ax=ax1);
ax1.set_title("Noisy Data");
```



Data Preprocessing

แบ่งข้อมูล X เพื่อนำมาทำนายค่า y เป็น train 70% และ test 30%

```
X1 = pd.DataFrame(X1,columns=['x','y'])
y1 = pd.Series(Y1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X1,y1,test_size = 0.30, random_state = 40, stratify=y1)
```

Models

Ensemble learning

ข้อผิดพลาด (Error) ที่เกิดขึ้นในโมเดลใด ๆ สามารถแบ่งออกได้ดังสมการคณิตศาสตร์ต่อไปนี้

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\hat{f}(x) - E[\hat{f}(x)]\right]^2 + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

ซึ่งข้อผิดพลาด (Error) ในโมเดลเป็นสิ่งที่ต้องให้ความสำคัญ เพราะทั้งความโน้มเอียง (Bias)และความแปรปรวน (Variance) แลกเปลี่ยนกันอยู่ตลอดเวลา (Bias-Variance trade-off) การรักษาสมดุลของข้อผิดพลาดทั้งสองประเภทนี้ทำให้โมเดลมีประสิทธิภาพมากยิ่งขึ้น

โดยปกติทั่วไปการนำโมเดลเพียงโมเดลเดียวมาใช้ในการจำแนกข้อมูลนั้น ถึงแม้ว่าจะให้ผลลัพธ์ที่แม่นยำ แต่บางครั้งก็ทำให้เกิดปัญหาความโน้มเอียง (Bias) เกิดจากการที่โมเดลเราไม่สามารถฟิตกับข้อมูลได้ดีเท่าที่ควร หรือปัญหาความแปรปรวน (Variance) เพราะ Overfit ข้อมูล Train set หนทางหนึ่งที่ช่วยลดปัญหาที่เกิดขึ้นมาได้ นั่นคือ Ensemble Learning หรือ การเรียนรู้แบบร่วมกันตัดสินใจ

Ensemble learning หรือ การเรียนรู้แบบร่วมกันตัดสินใจ เป็นเทคนิคในการเพิ่มประสิทธิภาพการทำงานของโมเดล ในการเรียนรู้ด้วยเครื่อง (Machine learning) เป็นการทำนายโดยใช้หลาย ๆ โมเดลที่มีความแตกต่างกันและมีอิสระต่อกันในการทำงานร่วมกัน เพื่อที่จะทำให้ผลลัพธ์ออกมาดีที่สุด ทำให้โมเดลมีความยืดหยุ่นมากกว่าเดิม Bias และ Variance ก็ลดน้อยลงไปด้วย

เทคนิคที่นิยมใช้ในการทำ Ensemble learning มีอยู่ 2 เทคนิคด้วยกัน คือ Bagging และ Boosting

Bagging

คำว่า Bagging ย่อมาจาก Bootstrap Aggregation โดย Bagging เป็นเทคนิคที่ช่วยลดปัญหา Overfit และ Variance ของข้อมูล โดยคำว่า boostrap คือการสุ่มข้อมูลมาจากข้อมูลประชากร เพื่อใช้คำนวณค่าทางสถิติของประชากรกลุ่มเล็กๆที่เราสุ่มออกมา และ aggregation ก็คือการเอารวมกัน มีวิธีการทำงานแบบคู่ขนาน หรือ เรียกว่า Parallel Method

สามารถสรุปขั้นตอนการทำงานออกมาได้ดังนี้

1. สมมติเรามีข้อมูลตัวอย่าง bootstrap อยู่ L ชุด (การประมาณชุดข้อมูลอิสระ L) ขนาด B

$$\{z_1^1, z_2^1, ..., z_B^1\}, \{z_1^2, z_2^2, ..., z_B^2\}, ..., \{z_1^L, z_2^L, ..., z_B^L\} \qquad z_b^l \equiv b\text{-th observation of the } l\text{-th bootstrap sample}$$

2. แต่ละโมเดล (weak learner) L โมเดลฟิตกับข้อมูลตัวอย่าง bootstrap

$$w_1(.), w_2(.), ..., w_L(.)$$

3. จากนั้น aggregate เช่น ค่าที่พบมากที่สุด (voting) หรือใช้ค่าเฉลี่ย (Averaging) เพื่อให้ได้ผลลัพธ์โมเดลที่มีความแปรปรวนต่ำ

$$s_L(.) = \frac{1}{L}\sum_{l=1}^{L} w_l(.) \qquad \text{(simple average, for regression problem)}$$

$$s_L(.) = \arg\max_k [card(l|w_l(.) = k)] \qquad \text{(simple majority vote, for classification problem)}$$

Boosting

เป็นอีกวิธีที่นิยมใช้กันมากในปัจจุบัน ซึ่งหลักการคือการนำข้อผิดพลาด (Error) ของโมเดลก่อน ๆ มาฝึกให้กับโมเดลใหม่ โดยโมเดลทุกตัวจะใช้ตัวอย่างข้อมูลที่เหมือนกันหมดในการฝึก แต่ก็จะมี Error ของโมเดลตัวก่อนเข้ามาเทรนด้วยเช่นกัน มีวิธีการทำงานอย่างต่อเนื่องกัน หรือ เรียกว่า Sequential Method

สามารถสรุปขั้นตอนออกมาได้ดังนี้

1. กำหนดน้ำหนักที่เท่ากันให้กับตัวอย่างข้อมูลแต่ละรายการ ซึ่งจะป้อนข้อมูลไปยังโมเดลอัลกอริทึมพื้นฐาน จากนั้นอัลกอริทึมพื้นฐานจะดำเนินการทำนายสำหรับตัวอย่างข้อมูลแต่ละรายการ
2. ประเมินการทำนายของโมเดลและเพิ่มน้ำหนักของตัวอย่างด้วยข้อผิดพลาดที่มีนัยสำคัญมากขึ้น นอกจากนี้ยังกำหนดน้ำหนักตามประสิทธิภาพของโมเดลอีกด้วย โดยโมเดลที่ให้ผลการทำนายที่ยอดเยี่ยมจะมีอิทธิพลอย่างมากต่อการตัดสินใจขั้นสุดท้าย
3. อัลกอริทึมจะส่งข้อมูลถ่วงน้ำหนักไปยังโมเดลถัดไป
4. ทำซ้ำขั้นตอนที่ 2 และ 3 จนกว่างข้อผิดพลาดในการฝึกฝนจะต่ำกว่าเกณฑ์ที่กำหนด

ความแตกต่างของ Bagging เมื่อเทียบกับ Boosting

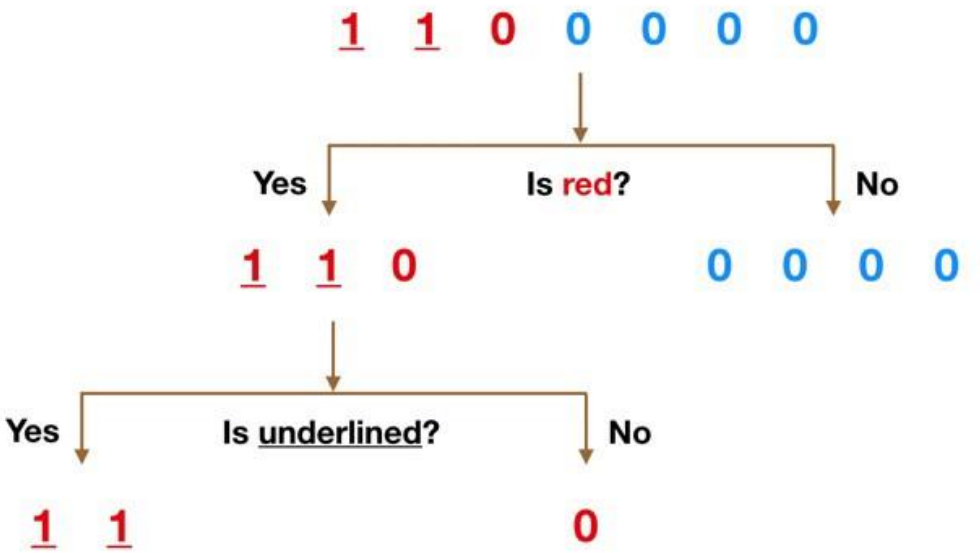
Bagging คือ การฝึกแต่ละโมเดลแบบคู่ขนาน (Parallel) แต่ละโมเดลจะถูกฝึกฝนโดยการสุ่มชุดข้อมูลย่อย (subset)

Boosting คือ การฝึกแต่ละโมเดลแบบเป็นลำดับขั้น (Sequential) แต่ละโมเดลจะเรียนรู้ข้อผิดพลาด (Error) ที่เกิดจากโมเดลก่อนหน้า

Decision tree

ต้นไม้การตัดสินใจ (Decision tree) เป็นโมเดลแบบสร้างสร้างกฎ if-else จากค่าของแต่ละ feature ของข้อมูล โดยไม่มีสมการที่คอยหาความสัมพันธ์ระหว่าง feature & target แต่ในการเลือก split ค่า feature แต่ละครั้ง มีเป้าหมายเพื่อ minimise ค่าของ cost functionให้น้อยที่สุด ต้นไม้การตัดสินใจประกอบด้วย

1. Nodes: ทดสอบค่าของตัวแปร
2. Edges/ Branch: เส้นเชื่อมระหว่าง Nodes กับ Nodes หรือ Leaf nodes โดยมีผลลั
3. Leaf nodes: Nodes ที่แสดงถึงประเภทหรือการกระจายประเภท



หลักการทำงานของ Decision Tree คือ การแบ่งข้อมูลออกทีละสองส่วนต่อไปเรื่อยๆ จาก node ล่างสุดของ tree เรียกว่า root node และนับขึ้นไปเรื่อยๆจนถึงบนสุด leaf node และทำการทำนายค่า class ของ target จะ split ข้อมูลจาก root node ไปจนถึง leaf node จะทำจนกว่าจะได้ตาม condition ที่กำหนด ยกตัวอย่างเช่น ความลึกของ tree ไม่เกิน 12 ชั้น (max dept) หรือจำนวนในแต่ละกลุ่มที่แบ่งออกมา (leaf node) มีจำนวนขั้นต่ำ 6 observations (min sample)

หลักการทำงานจะทำโดยแบ่งข้อมูลในแต่ละ node สำหรับข้อมูลที่มี k feature และ n observation มีดังนี้

1. เลือก 1 feature จาก k feature มาทำ sorting ข้อมูล ด้วยค่าของ feature ที่เลือกมา
2. หาจุดแบ่งข้อมูล ที่เป็นไปได้ทั้งหมด จากข้อมูล n observation สามารถหาจุดแบ่งข้อมูลที่เป็นไปได้ n-1 จุด
3. สำหรับการแบ่งข้อมูลแต่ละแบบที่เป็นไปได้ โดยอาศัย cost function 2 ตัวนี้

Gini impurity

ซึ่งเป็นการวัดความไม่บริสุทธิ์ของ class ในแต่ละกลุ่มข้อมูลที่แบ่งตามแต่ละ split point สำหรับปัญหา classification แบบ binary ที่มี target variable เป็น 0 หรือ 1 การ split ที่ดีควรที่จะได้ข้อมูลออกมา 2

กลุ่มที่สามารถแยกได้ชัดเจนในแต่ละกลุ่ม ยิ่งการแยก target ของ class ออกมาดีเท่าไร ค่า Gini impurity ก็ยิ่งต่ำเท่านั้น

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

โดยที่ K คือ จำนวนคลาส

p คือ ความน่าจะเป็นที่เกี่ยวข้องกับคลาส k

Entropy

ซึ่งเป็นการวัดความไม่แน่นอนของข้อมูล เช่น การโยนเหรียญจะเกิด entropy เท่ากับ (½*log2(½) + ½*log2(½)) = 1 ซึ่งถือว่าเป็นค่า entropy ที่สูงสุด เพราะไม่สามารถคาดเดาเหตุการณ์ที่ไม่มี bias แบบนี้ได้ โดยเทคนิคในการทำให้โมเดล มีประสิทธิภาพ คือ ต้องลดความไม่แน่นอนให้น้อยที่สุด เพราะ เป็นการแยก class ของ target ให้ได้สัดส่วนของ class ใด class หนึ่งมากที่สุด เพื่อเพิ่มความแม่นยำในการทำนาย

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

โดยที่ K คือ จำนวนคลาส

p คือ ความน่าจะเป็นที่เกี่ยวข้องกับคลาส k

โดยต้นไม้การตัดสินใจ จะใช้ Entropy ในการคำนวณการรับข้อมูล ดังนั้น ด้วยการคำนวณการลดลงของการวัดเอนโทรปีของแต่ละคุณลักษณะ เราจึงสามารถคำนวณการรับข้อมูลได้ คุณลักษณะที่มีการรับข้อมูลสูงสุดจะถูกเลือกเป็นคุณลักษณะในการการแยกโหนด

การ overfitting ใน Decision Tree

การ overfitting ถือเป็นปัญหาระหว่างการสร้าง Decision Tree model โดยปัญหาของการ overfitting นั้นเกิดเมื่ออัลกอริทึมได้คำนวณแล้วทำ leaf node ลึกไปเรื่อยๆ เพื่อที่ต้องการจะลด training-set error แต่ผลลัพธ์จากการใช้กับ test-set เกิด error ที่เพิ่มขึ้น ดังนั้น ความแม่นยำของการทำนายของโมเดลก็จะต่ำลง เหตุการณ์แบบนี้จะเกิดขึ้นทั่วไป เมื่อเราต้องการที่จะสร้างโมเดลที่มีหลาย branches จาก outliers และ irregularities ในชุดข้อมูล

วิธีการแก้ปัญหา overfitting แบ่งออกเป็น 2 วิธี ดังนี้

1. Pre-Prunning

สำหรับ pre-pruning นั้นจะหยุด tree construction ก่อนเวลาเล็กน้อย เพราะว่าไม่ต้องการที่จะแยก node หากการวัดประสิทธิภาพของ node นั้นต่ำกว่าเกณฑ์ แต่วิธีการนี้จะยากในการที่จะเลือกจุดที่เหมาะสมที่สุดของโมเดล

1. Post-Pruning

สำหรับ post-pruning นั้นเราจะสร้างต้นไม้ที่สมบูรณ์ก่อน หากต้นไม้มีปัญหาการใส่มากเกินไป การตัดแต่งกิ่งจะทำเป็นขั้นตอนหลัง ซึ่งจะอาศัยการใช้ cross-validation ในการเช็คผลลัพธ์จากการทำ post-pruning โดยการใช้ validation data นั้นจะทดสอบว่าการขยาย node นั้นช่วยทำให้ผลลัพธ์ดีขึ้นหรือแย่ลง ถ้าดีขึ้นก็จะให้สามารถขยายไป node นั้นได้ แต่หากเห็นว่าไม่ช่วยทำให้ดีขึ้นก็จะไม่ขยาย node ไป

Random forest

Random Forest อยู่ในกลุ่มของ Ensemble learning คือ อัลกอริทึมที่ใช้เทคนิค Bagging โดยมี Decision tree เป็นอัลกอริทึมพื้นฐาน กล่าวคือเป็นเทคนิคในการนำ Decision tree หลาย ๆ โมเดลมาทำงานร่วมกัน โดยมีหลักการทำงาน คือ การสร้างโมเดลจากการนำ Decision tree หลาย ๆ โมเดลย่อย ๆ ตั้งแต่ 10 ถึงมากกว่า 1000 โมเดล ซึ่งแต่ละโมเดลจะได้รับชุดข้อมูลและคุณลักษณะของข้อมูล (feature)ไม่เหมือนกัน ซึ่งเป็นชุดข้อมูลย่อยของชุดข้อมูลทั้งหมด โดย Decision tree จะคำนวณผลลัพธ์จากการ prediction ด้วยการ vote output ที่ถูกเลือกโดย Decision Tree มากที่สุด) โดย Decision Tree ใน Random Forest ถือว่าเป็น weak learner เพราะว่าเป็นโมเดลที่ทำนายค่อนข้างไม่แม่นยำ แต่เมื่อได้นำแต่ละ Decision Tree มาทำการทำนายร่วมกัน ก็จะกลายเป็นโมเดลที่มีความแม่นยำที่ค่อนข้างสูง หากเทียบกับ Decision Tree ที่ prediction แค่ตัวเดียว

หลักการในการทำ Random Forest

1. ทำการสร้างชุดข้อมูลย่อย จากชุดข้อมูลทั้งหมด ให้ได้ชุดข้อมูลย่อย n ชุด ซึ่ง n คือจำนวนของ Decision Tree ใน Random Forest เช่น ชุดข้อมูลตั้งต้นมีอยู่ 20 Feature ซึ่งแต่ละ Decision Tree จะได้ Feature ไม่เหมือนกัน ซึ่งรวมไปถึงข้อมูลที่ได้ก็ไม่ครบทุก row จากชุดข้อมูลรวมทั้งหมดเช่นกัน
2. สร้าง Model Decision Tree สำหรับแต่ละชุดข้อมูลย่อย
3. ทำการ Aggregation ผลลัพธ์ จากการ bagging model เช่น การนำ voting ของแต่ละ Decision Tree มาคิดรวมเพื่อหาว่าสรุปแล้ว voting ไปทางไหน

Random forest algorithm

Classification and Regression Tree (CART) เป็น heuristic method สำหรับการกระตุ้น DT เพื่อแบ่งพาร์ติชันจากบนลงล่าง, greedy, recursive และ binary ของ training data set โดยจะเริ่มต้นจาก node เดียว ซึ่งให้เป็นรูทในการที่ครอบคลุมอินสแตนซ์ของข้อมูล training data ทั้งหมด การที่ใช้ตัวแปรที่ไม่ใช่อัลกอริทึม CART จะดำเนินการโดยการสร้าง node ในการตัดสินใจ เพื่อแบ่งอินสแตนซ์ให้มีความ pure มากขึ้นตาม

คลาสของชนิดนั้นๆ โดยในการวนซ้ำในแต่ละครั้ง จะcandidate split เพื่อ ประเมินผลสำหรับทุก node ที่อยู่ end of decision path แต่ถ้ายังไม่เจอการ stop criteria ส่วนของ candidate spilt ก็จะ generate two child nodes พร้อมกับ lowest weighted total Gini Impurity โดย child nodes จะถูกนำไปเพิ่มใน growing tree

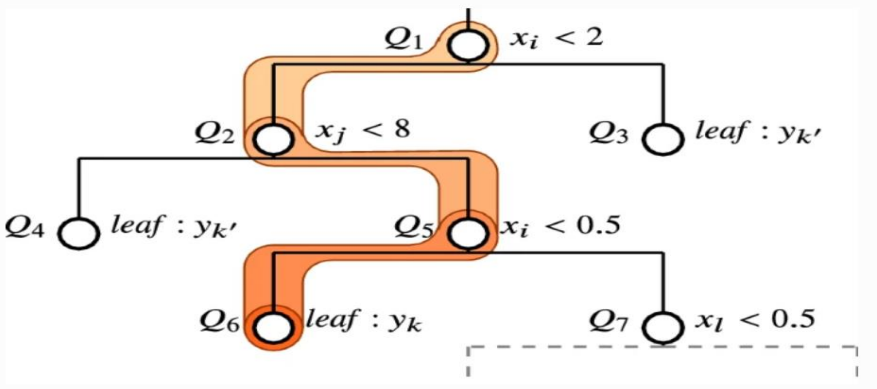
$$I_G(Q) = \sum_{k=1}^{K} p_k \cdot (1 - p_k)$$

ซึ่งค่า p_k เป็น proportion ของอินสแตนซ์ ที่มีการ label y_k ใน node Q และ K ซึ่งเป็นเลขของ class

$$\text{Weighted Total } I_G = \frac{I_G(Q_{\text{first}}) \cdot |Q_{\text{first}}|}{N} + \frac{I_G(Q_{\text{second}}) \cdot |Q_{\text{second}}|}{N}$$

ซึ่ง $\{Q_{\text{first}}, Q_{\text{second}}\}$ เป็น set ของ child node ซึ่งถูกสร้างโดย candidate split และ lQl คือ จำนวนของ training อินสแตนซ์ที่ครอบคลุมโดย node เป็นการกระทำที่เป็นอิสระจาก การวนซ้ำเก่าๆ หรือครั้งต่อๆไป ซึ่งแปลได้ว่าไม่มีทางที่จะเคลื่อนที่กลับหรือหาการแยกที่ดีกว่าโดยอาศัยการพิจารณาจากการโต้ตอบของฟีเจอร์ เป็นการแสดงถึง greedy heuristic มีประสิทธิภาพในการคำนวณมากกว่าการค้นหาอย่างละเอียดถี่ถ้วน แต่ไม่มีการรับประกันว่าจะพบ Decision Tree ที่เหมาะสมที่สุด วิธีการนี้นั้นสามารถกำหนดได้สำหรับต้นไม้ต้นเดียวและ static sample

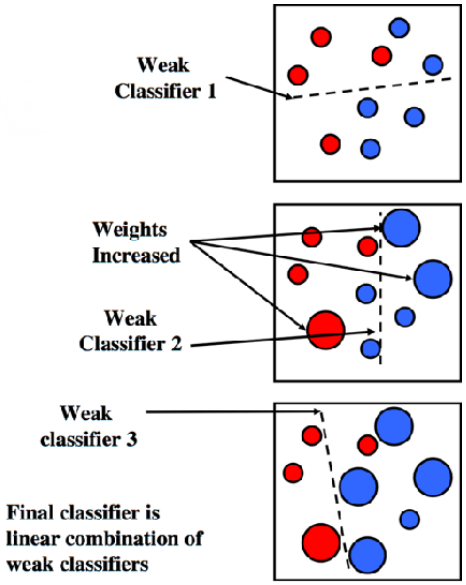
โดยการจำแนกอินสแตนซ์ที่มองไม่เห็นก่อนหน้า สามารถทำได้โดยดูจากเส้นทางของอินสแตนซ์ลงไปที่ทรี โดยเริ่มจาก root แล้วปฏิบัติตามเงื่อนไขการแยกว่าเป็นจริงหรือเท็จสำหรับอินสแตนซ์ ดังนั้นอินสแตนซ์สามารถติดตามได้เพียงเส้นทางเดียวเท่านั้น และมาถึง node ปลายสุดได้เพียงที่เดียวเท่านั้น โดย node ปลายสุดแต่ละ node จะครอบคลุมชุดย่อยของอินสแตนซ์ของ training data set และ รีเทิร์น คลาสที่ได้ label ออกมาจากการวิเคราะห์ผล



โดย Random Forest เป็น ensemble ของ Decision Tree classifiers ตั้งแต่ 10 ถึง 1000 ต้น ซึ่ง ensemble ทำงานแบบคู่ขนาน และจำแนกตามการ voting คุณสมบัตินี้จะช่วยปรับปรุงประสิทธิภาพการจำแนกประเภทในกลุ่มเมื่อเทียบกับตัวแยกประเภทเดียว สมมติว่าตัวแยกประเภทแต่ละตัวใน ensemble ทำงานได้ดีกว่าการเดาแบบสุ่ม และมีความแตกต่างกัน (ถ้าไม่สัมพันธ์กันทั้งหมด) ในข้อผิดพลาด ซึ่ง ensemble การจัดประเภทที่ผิดพลาดนั้นจะมีจำนวนน้อยกว่า ทำให้การจัดประเภทนั้นมีความถูกต้องและแม่นยำมาก เพื่อส่งเสริมความหลากหลายทางโครงสร้างที่จำเป็นระหว่างตัวแยกประเภทพื้นฐาน กระบวนการสุ่มถูกนำมาใช้ในสองขั้นตอนใน Decision Tree โดยประการแรกอาศัยการ bootstrap ใน training set data ประการที่สองเป็นการ voting เพื่อหา I_g score ที่ดีที่สุดของ candidate split ซึ่งจะเพิ่มลงใน Tree ในแต่ละการวนซ้ำในแต่ละครั้ง ซึ่ง candidate splits คือ การลิมิตการสุ่มตัวอย่างย่อยของ candidate โดย Tree จะเติบโตเต็มที่เพื่อให้ค่า pure ของใบไม้แต่ละใบครอบคลุมอินสแตนซ์การฝึกของคลาสเดียวเท่านั้น ผลลัพธ์ของต้นไม้อาจมีความลึกและเป็นพุ่มมาก โดย โมเดล Random Forest มี่สร้างขึ้นด้วยวิธีนี้มีความสามารถในการหาความถูกต้อง ทนทานต่อการ over fitting การไม่สมดุลของคลาส ใน data set ซึ่งจะมีพารามิเตอร์ในการดัดแปลงคุณสมบัติ ไม่ว่าจะเป็น จำนวนของต้นไม้ หรือจำนวนของ feature ที่ใช้ในการสร้าง candidate splits นี่เป็นสิ่งที่ทำให้ Random Forest นั้นง่ายต่อการไปใช้งานและนำไปฝึกฝน โดยความแตกต่างของโครงสร้าง Decision Tree และ Random Forest คือ Random Forest เป็น black box model กล่าวคือ เป็นโมเดลที่ค่อนข้างที่จะอธิบายได้ยากหากเทียบกับ Decision Tree

AdaBoost

AdaBoost (Adaptive Boosting) เป็น Sequential ensemble method ที่มีการรวม weak learner หลายๆ ตัวเข้าด้วยกัน แล้วสร้างเป็น strong learner ทำให้ประสิทธิภาพของการทำนายสุดท้ายเพิ่มขึ้น โดยการใช้โมเดลการจำแนกประเภทที่มีความซับซ้อนน้อย เช่น Decision tree มาเรียงต่อกันเป็นลำดับ แต่ละรอบของการฝึกจะมีการกำหนดค่าน้ำหนักของแต่ละจุดข้อมูล และมีการเรียนรู้จากค่าน้ำหนักเหล่านั้น โดยให้ความสำคัญกับจุดข้อมูลที่ทำนายผิด เพื่อให้ในรอบต่อไปมีโอกาสที่จำแนกได้ถูกต้อง โดยะมีการปรับเพิ่มค่าน้ำหนักของจุดข้อมูลที่ทำนายผิด และปรับลดค่าน้ำหนักของจุดข้อมูลที่ทำนายถูก แล้วนำจุดข้อมูลเหล่านี้ไปฝึก และสร้าง weak learner ตัวใหม่ ขั้นตอนเหล่านี้จะทำแบบ sequential และจะมีการปรับค่า weight ในทุกรอบ



ที่มา: https://sirawichjaichuen.medium.com/adaboost-algorithm-cfe6b58e60fa

การทำนายของ AdaBoost คือการถ่วงน้ำหนักเข้ากับจุดข้อมูล แล้วเลือกคำตอบของการทำนายจากประเภทที่มีค่าถ่วงน้ำหนักมากที่สุด

AdaBoost Algorithm:

1. Initialize weight โดยเริ่มต้นให้ทุกจุดข้อมูลมีค่า weight ตั้งต้นของข้อมูลแต่ละรายการเป็น 1/m โดย m คือจำนวนรายการข้อมูลทั้งหมด

$$w^{(i)} = \frac{1}{m}$$

2. ทำนายประเภทจุดข้อมูลรอบแรก และคำนวนอัตราความคลาดเคลื่อน r1 โดยเปรียบเทียบระหว่างผลรวมของน้ำหนักของทุกๆ รายการที่ทำนายผิด กับผลรวมของน้ำหนักของทุกๆ รายการ (Weighted error rate ของ Classifier j)

$$r_j = \frac{\sum_{i=1}^{m} w^{(i)}_{[\hat{y}_j^{(i)} \neq y^{(i)}]}}{\sum_{i=1}^{m} w^{(i)}}$$

3. นำ Weighted error rate นี้มาคำนวนหาน้ำหนัก α_j ของ Classifier j ดังนี้:

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$

- η คือ Learning rate hyperparameter ที่ควบคุมอัตราเร็วในการเรียนรู้

4. นำน้ำหนัก α_j ของ Classifier j ไปอับเดตน้ำหนัก $w_{(i)}$ของข้อมูลแต่ละรายการ ตามเงื่อนไขดังนี้

for i=1, 2,…,m

$$w^{(i)} = \begin{cases} w^{(i)} & \text{if } \hat{y}^{(i)} = y^{(i)} \\ w^{(i)} e^{\alpha_j} & \text{if } \hat{y}^{(i)} \neq y^{(i)} \end{cases}$$

แล้ว Normalise น้ำหนัก w(i) ของข้อมูลแต่ละรายการ ด้วยการหารด้วยผลรวมของ w(i) ทั้งหมด

$$w^{(i)} := \frac{w^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$$

ทำกระบวนการที่ 2) ถึง 4) วนซ้ำไปเรื่อย ๆ โดยจะหยุดก็ต่อเมื่อ:

- วนซ้ำครบจำนวน Classifier instance ที่กำหนด หรือ
- ไม่พบรายการที่ทำนายผิดพลาดเลยแม้แต่รายการเดียว

5. เมื่อต้องการทำนาย จะคำนวนค่าการทำนาย $\hat{y}(i)$ของทุกๆการจำแนกประเภท โดยถ่วงน้ำหนักค่าการทำนายของแต่ละประเภทด้วยค่าน้ำหนัก α_j ของการจำแนกแต่ละประเภท ดังนั้น ค่าการทำนายของประเภทที่มีความแม่นยำมากจะมีค่ามาก และได้รับการพิจารณามาก โดยจะมีค่าถ่วงน้ำหนักมากที่สุด

ดังนั้น AdaBoost มีเป้าหมายที่จะทำให้ค่าความถูกต้องของจุดข้อมูลที่ผ่านการทำนายไปแล้วมีค่าสูง เมื่อต้องการทำนาย

Evaluation criteria

การประเมินผล (Evaluation) เป็นขั้นตอนที่สำคัญเพื่อประเมินความถูกต้องของโมเดล เพื่อตรวจสอบว่าโมเดลที่ได้ทำไปนั้นมีความเหมาะสมหรือดีพอที่จะนำไปใช้งานได้หรือไม่ โดยสามารถพิจารณาได้จากการประเมินผลที่ได้รับความนิยมคือ เมทริกซ์ความสับสน (Confusion matrix) เพื่อที่จะตรวจสอบหาค่าต่างๆ เช่น ค่าความถูกต้อง (Accuracy), ค่าความแม่นยำ (Precision), ความไว (Sensitivity หรือ Recall), F1-score ที่สามารถแสดงผลลัพธ์และความเหมาะสมของโมเดล

	ลูกค้าต่อสัญญา (Actually positive)	ลูกค้าไม่ต่อสัญญา (Actually negative)
ผลบวกจากโมเดล (Predicted positive)	ผลบวกจริง (True positive: TP)	ผลลบลวง (False negative: FN)
ผลลบจากโมเดล (Predicted negative)	ผลบวกลวง (False positive: FP)	ผลลบจริง (True negative: TN)

ค่าความถูกต้อง (Accuracy) คือ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล โดยจะเป็นสัดส่วนของผลรวมผลบวกจริงและผลลบจริง ต่อจำนวนของผลตรวจทั้งหมด

$$Accuracy = \frac{\mathrm{TP} + TN}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

ค่าความแม่นยำ (Precision) คือ ค่าที่บ่งบอกความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง โดยจะเป็นสัดส่วนของผลบวกจริงต่อผลบวกจริงและผลบวกลวง

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

ความไว (Sensitivity หรือ Recall) คือ ค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง โดยจะเป็นสัดส่วนของผลบวกจริงต่อผลบวกทั้งหมดจากโมเดล

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

ความจำเพาะ (Specificity) คือ ค่าที่แสดงถึงโอกาสที่จะแยกความถูกต้องระหว่างคลาส โดยจะเป็นสัดส่วนของผลลบจริงต่อผลลบทั้งหมดจากโมเดล

$$Specificity = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

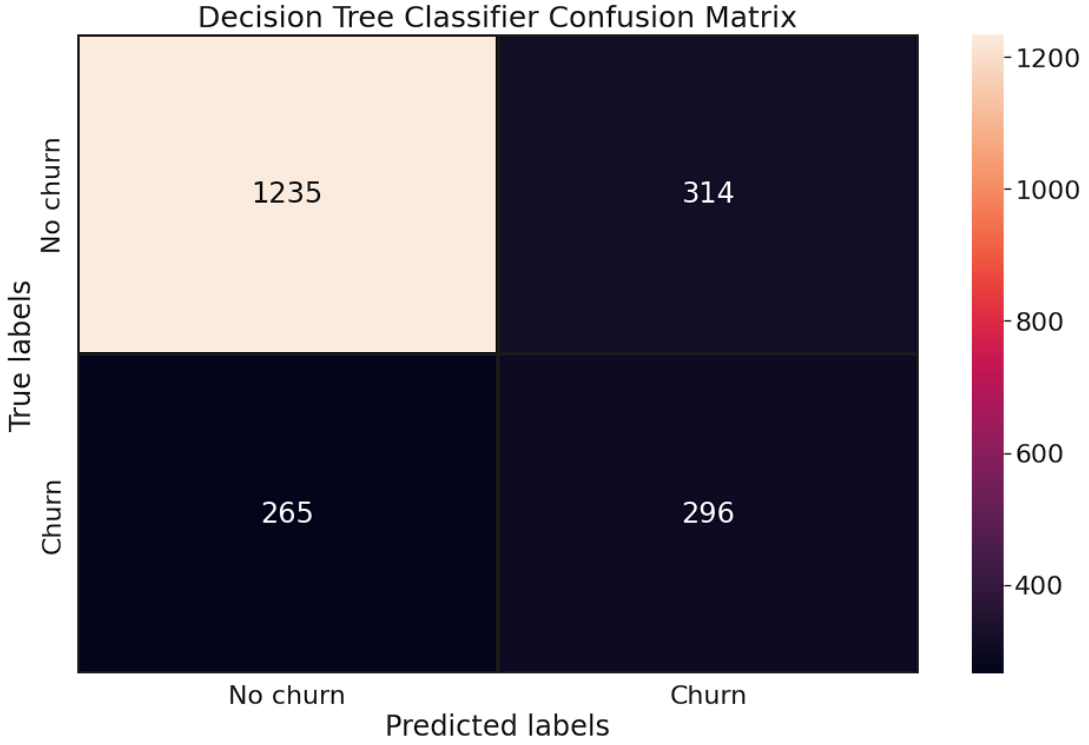
F1-score คือค่าเฉลี่ยของค่าความแม่นยำและความไว ใช้เพื่อจะลดผลที่ทำให้เกิดผลลบลวงสูง

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right)$$

Result

1. Real-world dataset

	precision	recall	f1-score	support
No Churn	0.82	0.80	0.81	1549
Churn	0.49	0.53	0.51	561
accuracy			0.73	2110
macro avg	0.65	0.66	0.66	2110
weighted avg	0.73	0.73	0.73	2110



รูปภาพ Confusion Matrix Decision Tree with Real-World Dataset

จากผลลัพธ์การทำ Confusion Matrix กับ Decision Tree จะเห็นว่า F1-score ของการต่อสัญญามีค่า 0.81 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.81 โดยเฉลี่ย และ F1-score ของการไม่ต่อสัญญามีค่า 0.51 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.51 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำต่ำมากหากเทียบกับการทำนายการต่อสัญญา โดยโมเดล

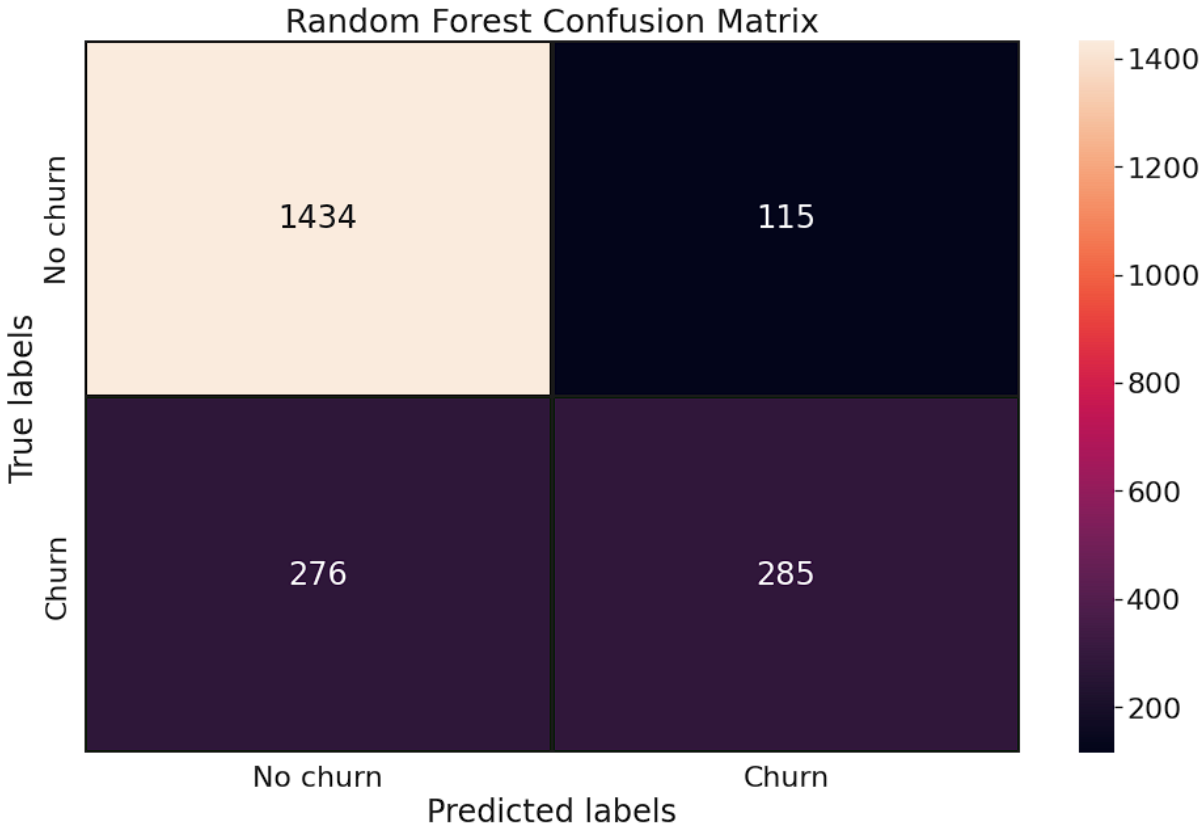
Decision Tree นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.73 ถือว่าค่อนข้างดีในระดับหนึ่ง

```
Time Consuming :  28.985429048538208
Best Accuracy Parameters
{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 10}
Decison Tree Accuracy Testing is :  0.7729857819905214
              precision    recall  f1-score   support

    No churn       0.83      0.87      0.85      1549
       Churn       0.58      0.52      0.55       561

    accuracy                           0.77      2110
   macro avg       0.71      0.69      0.70      2110
weighted avg       0.77      0.77      0.77      2110
```

รูปภาพ Confusion Matrix Decision Tree with Real-World Dataset GridsearchCV

จากผลลัพธ์การทำ Confusion Matrix กับ Decision Tree จะเห็นว่า F1-score ของการต่อสัญญามีค่า 0.85 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.85 โดยเฉลี่ย และ F1-score ของการไม่ต่อสัญญามีค่า 0.55 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.55 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำต่ำมากหากเทียบกับการทำนายการต่อสัญญา โดยโมเดล Decision Tree นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.77 ถือว่าค่อนข้างดีกว่า Decision Tree ที่ยังไม่ได้มีการใช้ GridsearchCV ในการหาค่าที่ดีที่สุด

	precision	recall	f1-score	support
No Churn	0.84	0.93	0.88	1549
Churn	0.71	0.51	0.59	561
accuracy			0.81	2110
macro avg	0.78	0.72	0.74	2110
weighted avg	0.81	0.81	0.80	2110

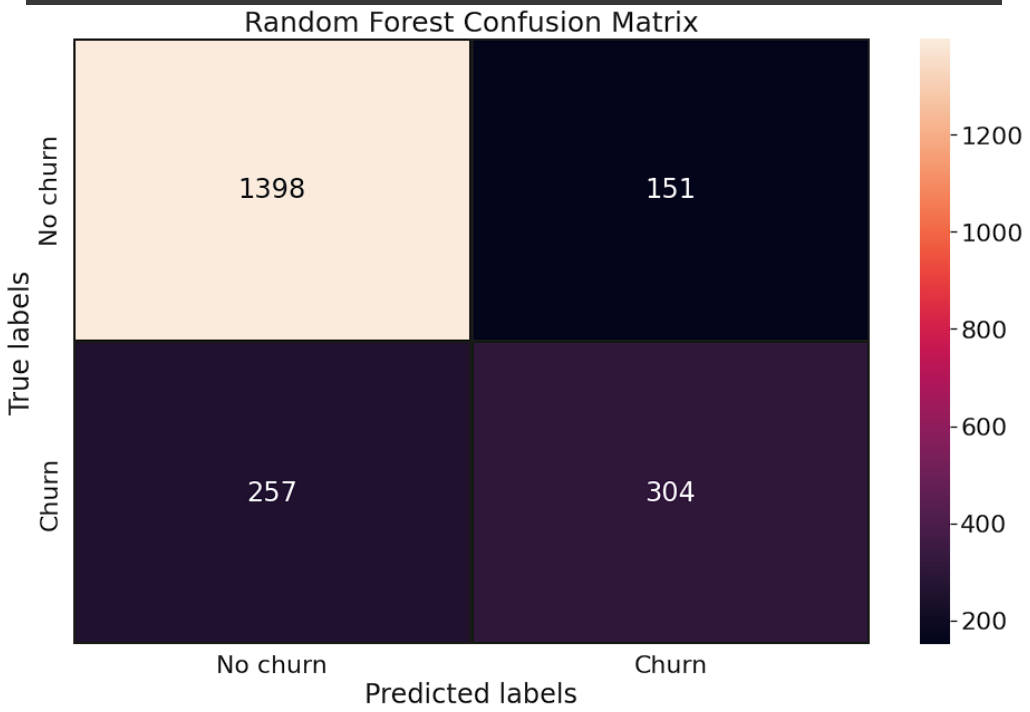


รูปภาพ Confusion Matrix Random Forest with Real-World Dataset

จากผลลัพธ์การทำ Confusion Matrix กับ Random Forest จะเห็นว่า F1-score ของการต่อสัญญามีค่า 0.88 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.88 โดยเฉลี่ย และ F1-score ของการไม่ต่อสัญญามีค่า 0.59 หมายความว่าความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.59 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำต่ำมากหากเทียบกับการทำนายการต่อสัญญา โดยโมเดล

Random Forest นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.81 ถือว่าแม่นยำกว่า Decision Tree model

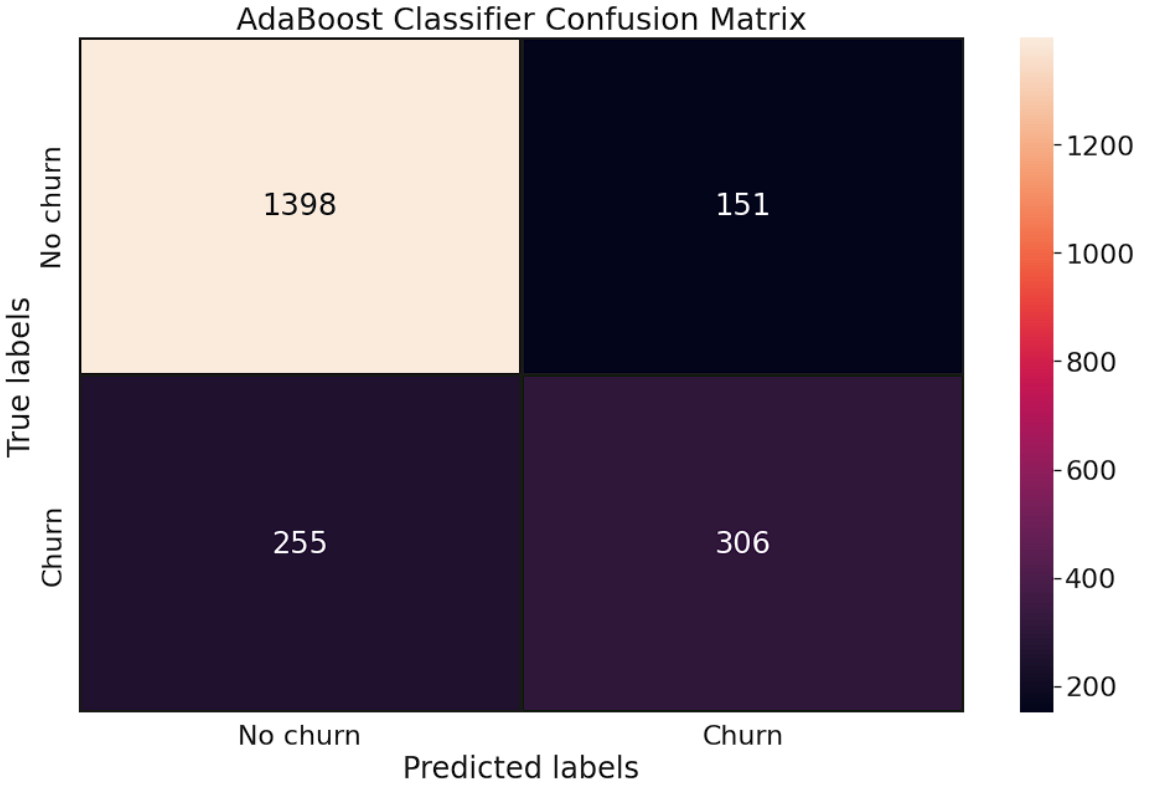
```
Fitting 4 folds for each of 80 candidates, totalling 320 fits
Time consuming : 680.8813948631287
Accuracy Testing score : 0.8066350710900474
Best Accuracy Parameters
{'n_estimators': 550, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_features': 0.3, 'max_depth': 90, 'criterion': 'gini'}
              precision    recall  f1-score   support

    No Churn       0.84      0.90      0.87      1549
       Churn       0.67      0.54      0.60       561

    accuracy                           0.81      2110
   macro avg       0.76      0.72      0.74      2110
weighted avg       0.80      0.81      0.80      2110
```



รูปภาพ Confusion Matrix Random Forest with Real-World Dataset RandomizeSearchCV

จากผลลัพธ์การทำ Confusion Matrix กับ Random Forest จะเห็นว่า F1-score ของการต่อสัญญามีค่า 0.88 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.87 โดยเฉลี่ย และ F1-score ของการไม่ต่อสัญญามีค่า 0.6 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.6 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำต่ำมากหากเทียบกับการทำนายการต่อสัญญา โดยโมเดล Random Forest นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.81 ถือว่าแม่นยำกว่า Decision Tree model แต่มีประสิทธิภาพเท่าแบบไม่ได้จูน

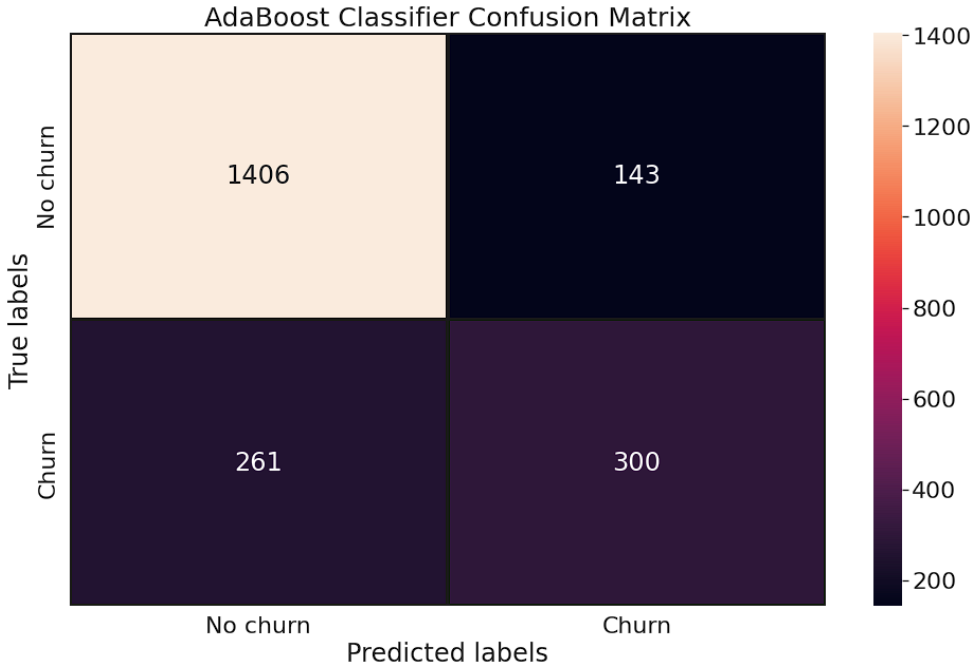
	precision	recall	f1-score	support
No Churn	0.85	0.90	0.87	1549
Churn	0.67	0.55	0.60	561
accuracy			0.81	2110
macro avg	0.76	0.72	0.74	2110
weighted avg	0.80	0.81	0.80	2110



รูปภาพ Confusion Matrix AdaBoost with Real-World Dataset

จากผลลัพธ์การทำ Confusion Matrix กับ AdaBoost จะเห็นว่า F1-score ของการต่อสัญญามีค่า 0.87 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.87 โดยเฉลี่ย และ F1-score ของการไม่ต่อสัญญามีค่า 0.6 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.6 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำต่ำมากหากเทียบกับการทำนายการต่อสัญญา โดยโมเดล AdaBoost นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.81 ถือว่าแม่นยำกว่า Decision Tree model แต่มีประสิทธิภาพใกล้เคียงกับ Random Forest model

```
Fitting 4 folds for each of 20 candidates, totalling 80 fits
Time Consuming :  53.50905394554138
Best Accuracy Parameters
{'n_estimators': 500, 'learning_rate': 0.1}
Accuracy Testing Score :  0.8085308056872038
              precision    recall  f1-score   support

    No Churn       0.84      0.91      0.87      1549
       Churn       0.68      0.53      0.60       561

    accuracy                           0.81      2110
   macro avg       0.76      0.72      0.74      2110
weighted avg       0.80      0.81      0.80      2110
```

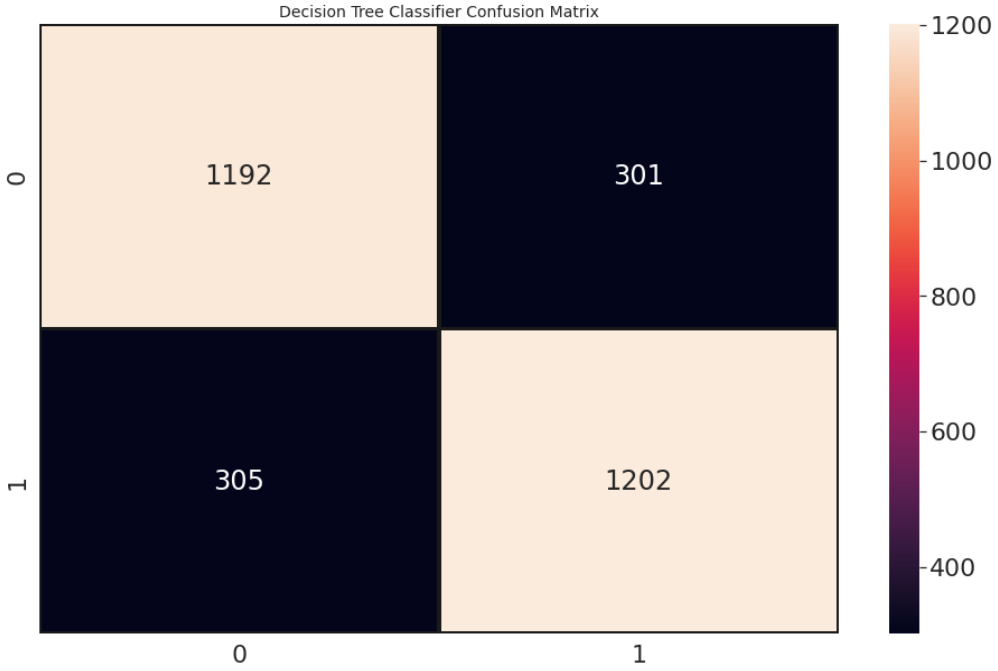


รูปภาพ Confusion Matrix AdaBoost with Real-World Dataset RandomizeSearchCV

จากผลลัพธ์การทำ Confusion Matrix กับ AdaBoost จะเห็นว่า F1-score ของการต่อสัญญามีค่า 0.87 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.87 โดยเฉลี่ย และ F1-score ของการไม่ต่อสัญญามีค่า 0.6 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.6 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำต่ำมากหากเทียบกับการทำนายการต่อสัญญา โดยโมเดล AdaBoost นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.81 ถือว่าแม่นยำกว่า Decision Tree model แต่มีประสิทธิภาพใกล้เคียงกับ Random Forest model และ AdaBoost แบบไม่ได้จูน

2. Synthetic dataset

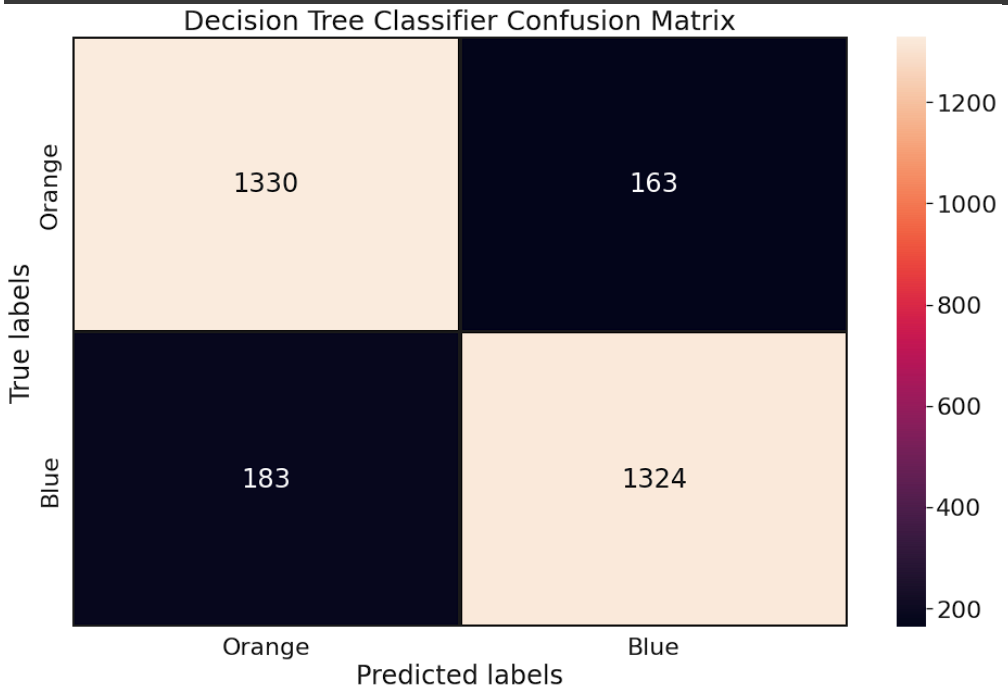
	precision	recall	f1-score	support
0	0.80	0.80	0.80	1493
1	0.80	0.80	0.80	1507
accuracy			0.80	3000
macro avg	0.80	0.80	0.80	3000
weighted avg	0.80	0.80	0.80	3000



รูปภาพ Confusion Matrix Decision Tree with Synthetic Dataset

จากผลลัพธ์การทำ **Confusion Matrix** กับ **Decision Tree** จะเห็นว่า **F1-score** ของการหาจุดสีส้มมีค่า **0.8** หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง **0.8** โดยเฉลี่ย และ **F1-score** ของการหาจุดสีฟ้ามีค่า **0.8** หมายความว่าความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง **0.8** โดยเฉลี่ย ซึ่งถือว่าความแม่นยำเทียบเท่ากับการทำนายการหาจุดสีส้ม โดยโมเดล **Decision Tree** นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ **accuracy** มีค่า **0.8** ถือว่าค่อนข้างดีในระดับหนึ่ง

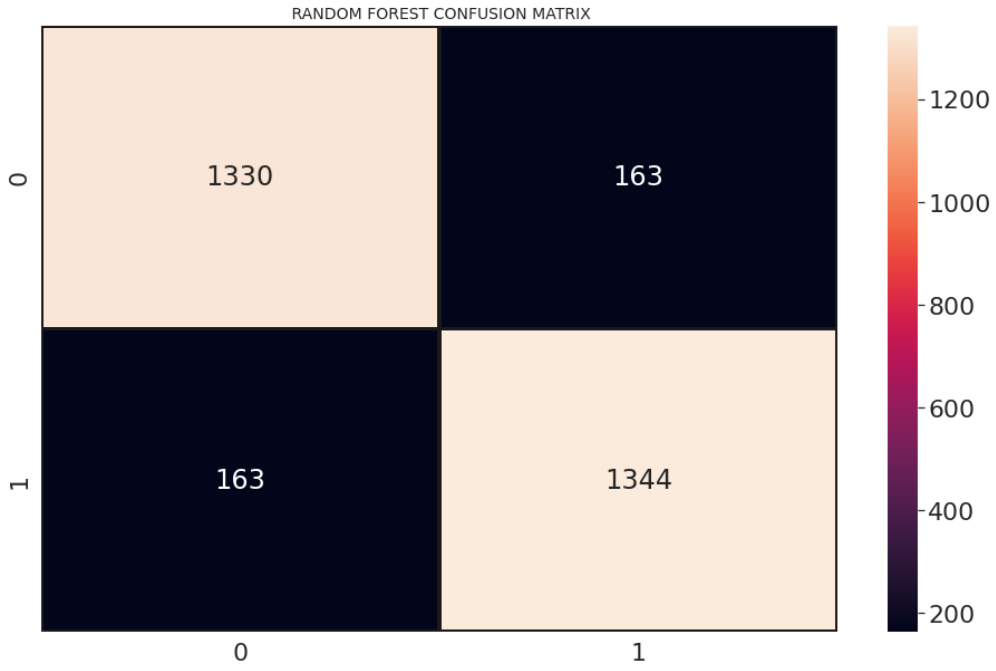
```
Time Consuming :  26.061567068099976
Best Accuracy Parameters
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 10}
Decison Tree Accuracy Testing is :  0.8846666666666667
              precision    recall  f1-score   support

           0       0.88      0.89      0.88      1493
           1       0.89      0.88      0.88      1507

    accuracy                           0.88      3000
   macro avg       0.88      0.88      0.88      3000
weighted avg       0.88      0.88      0.88      3000
```



รูปภาพ Confusion Matrix Decision Tree with Synthetic Dataset GridSearchCV

จากผลลัพธ์การทำ Confusion Matrix กับ Decision Tree จะเห็นว่า F1-score ของการหาจุดสีส้มมีค่า 0.88 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.88 โดยเฉลี่ย และ F1-score ของการหาจุดสีฟ้ามีค่า 0.88 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.88 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำเทียบเท่ากับการทำนายการหาจุดสีส้ม โดยโมเดล Decision Tree นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.88 ถือว่าดีขึ้นจากการใช้ Decision Tree แต่ไม่ได้ใช้ GridSearchCV

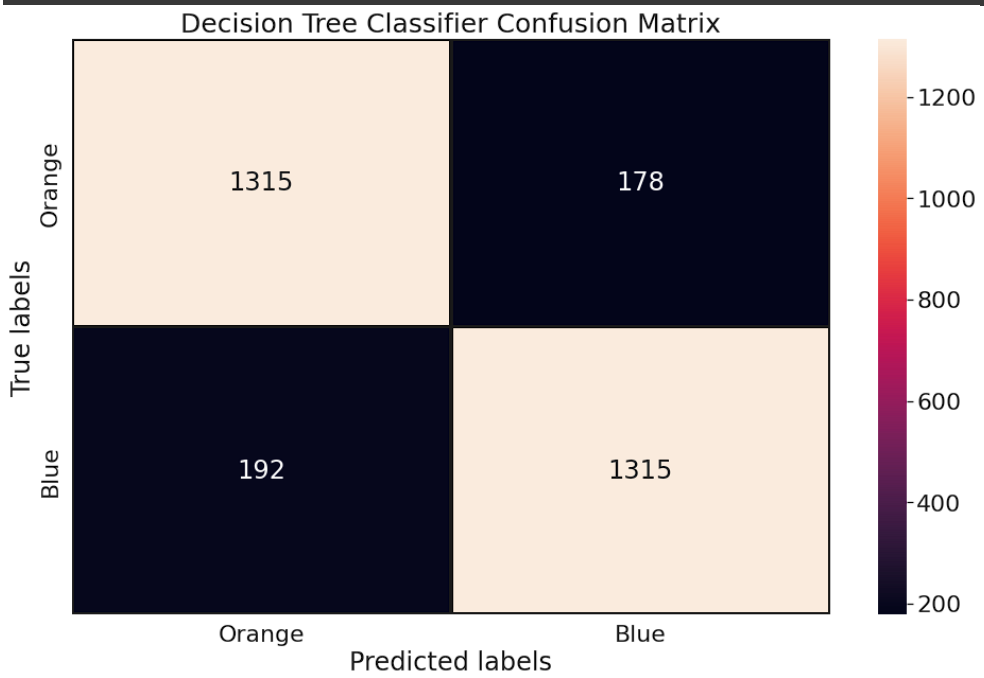
	precision	recall	f1-score	support
0	0.89	0.89	0.89	1493
1	0.89	0.89	0.89	1507
accuracy			0.89	3000
macro avg	0.89	0.89	0.89	3000
weighted avg	0.89	0.89	0.89	3000



รูปภาพ Confusion Matrix Random Forest with Synthetic Dataset

จากผลลัพธ์การทำ **Confusion Matrix** กับ **Random Forest** จะเห็นว่า **F1-score** ของการหาจุดสีส้มมีค่า **0.89** หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง **0.89** โดยเฉลี่ย และ **F1-score** ของการหาจุดสีฟ้ามีค่า **0.89** หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง **0.89** โดยเฉลี่ย ซึ่งถือว่าความแม่นยำเทียบเท่ากับการทำนายการหาจุดสีส้ม โดยโมเดล **Decision Tree** นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ **accuracy** มีค่า **0.89** ถือว่าค่อนข้างดีกว่าการใช้ **Decision Tree** ทั้งแบบใช้ **GridSearchCV** และไม่ใช้ **GridSearchCV**

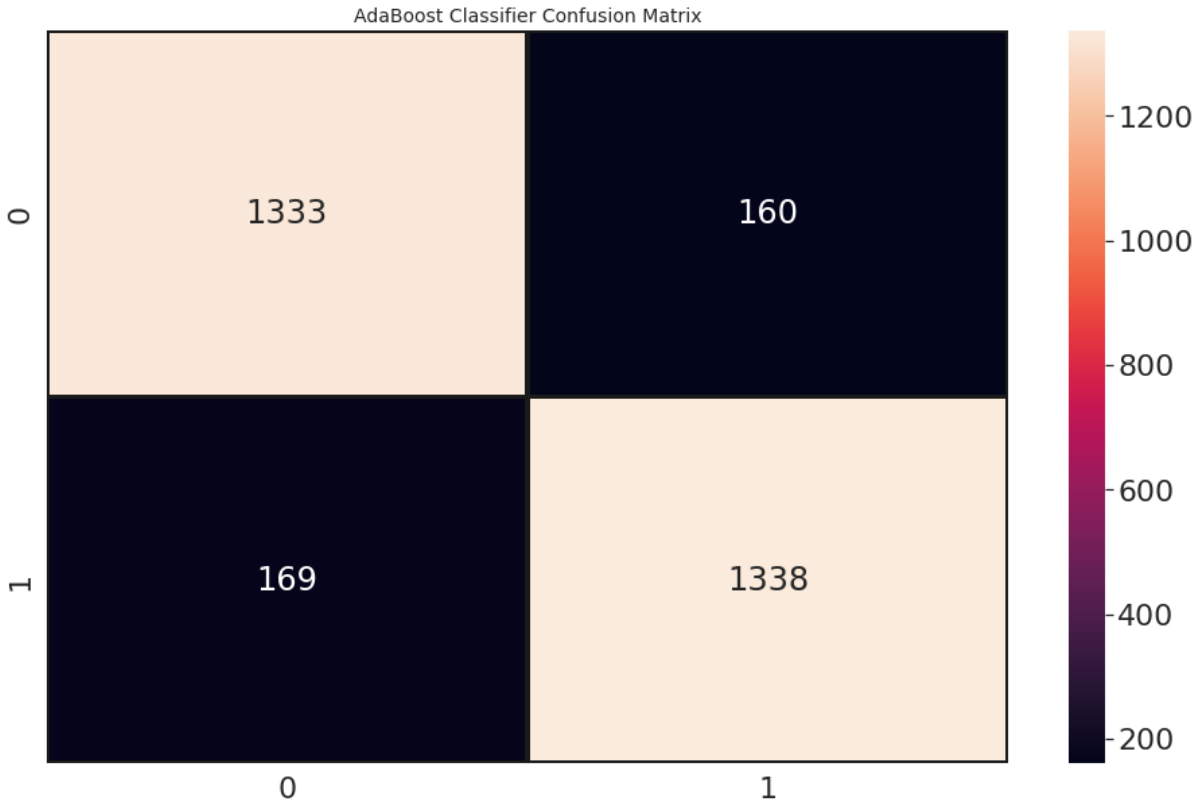
```
Time Consuming :  35.179890394210815
Best Accuracy Parameters
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 10}
Decison Tree Accuracy Testing is :  0.8846666666666667
              precision    recall  f1-score   support

      Orange       0.88      0.89      0.88      1493
        Blue       0.89      0.88      0.88      1507

    accuracy                           0.88      3000
   macro avg       0.88      0.88      0.88      3000
weighted avg       0.88      0.88      0.88      3000
```



รูปภาพ Confusion Matrix Random Forest with Synthetic Dataset RandomizeSearchCV

จากผลลัพธ์การทำ Confusion Matrix กับ Random Forest จะเห็นว่า F1-score ของการหาจุดสีส้มมีค่า 0.88 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.88 โดยเฉลี่ย และ F1-score ของการหาจุดสีฟ้ามีค่า 0.88 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.88 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำเทียบเท่ากับการทำนายการหาจุดสีส้ม โดยโมเดล Decision Tree นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.88 ถือว่าค่อนข้างดีกว่าการใช้ Decision Tree ทั้งแบบใช้ GridSearchCV และไม่ใช้ GridSearchCV แต่ประสิทธิภาพเทียบเท่ากับการใช้ Random Forest แบบยังไม่จูน

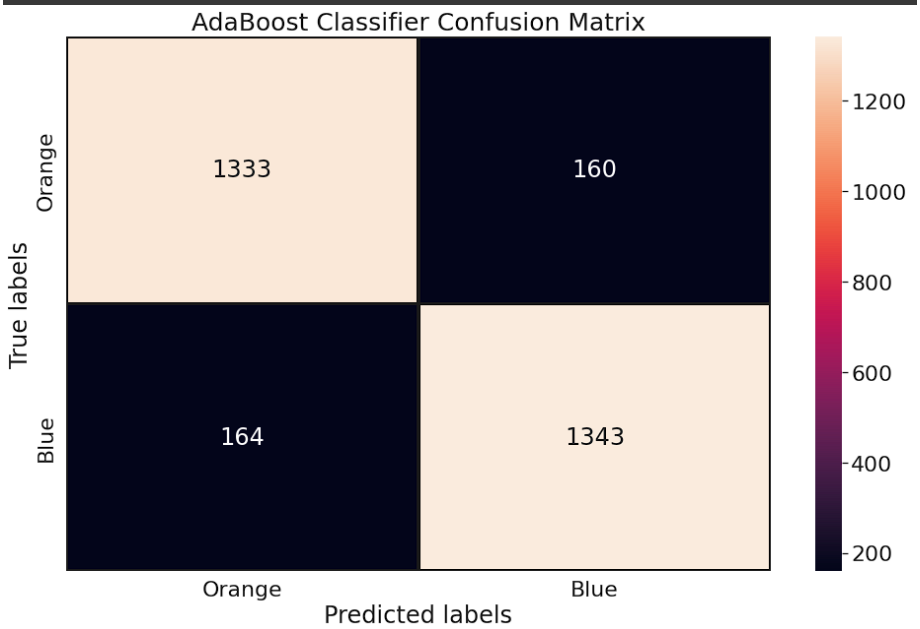
	precision	recall	f1-score	support
0	0.89	0.89	0.89	1493
1	0.89	0.89	0.89	1507
accuracy			0.89	3000
macro avg	0.89	0.89	0.89	3000
weighted avg	0.89	0.89	0.89	3000



รูปภาพ Confusion Matrix AdaBoost with Synthetic Dataset

จากผลลัพธ์การทำ **Confusion Matrix** กับ **AdaBoost** จะเห็นว่า **F1-score** ของการหาจุดสีส้มมีค่า 0.89 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.89 โดยเฉลี่ย และ **F1-score** ของการหาจุดสีฟ้ามีค่า 0.89 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.89 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำเทียบเท่ากับการทำนายการหาจุดสีส้ม โดยโมเดล **Decision Tree** นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ **accuracy** มีค่า 0.89 ถือว่าค่อนข้างดีกว่าการใช้ **Decision Tree** ทั้งแบบใช้ **GridSearchCV** และไม่ใช้ **GridSearchCV** แต่ประสิทธิภาพเทียบเท่ากับการใช้ **Random Forest** แบบยังไม่จูนและแบบจูน

```
Fitting 4 folds for each of 20 candidates, totalling 80 fits
Time Consuming :  49.78249478340149
Best Accuracy Parameters
{'n_estimators': 100, 'learning_rate': 0.1}
Accuracy Testing Score :  0.892
              precision    recall  f1-score   support

      Orange       0.89      0.89      0.89      1493
        Blue       0.89      0.89      0.89      1507

    accuracy                           0.89      3000
   macro avg       0.89      0.89      0.89      3000
weighted avg       0.89      0.89      0.89      3000
```



รูปภาพ Confusion Matrix AdaBoost with Synthetic Dataset RandomizedSearchCV

จากผลลัพธ์การทำ Confusion Matrix กับ AdaBoost จะเห็นว่า F1-score ของการหาจุดสีส้มมีค่า 0.89 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.89 โดยเฉลี่ย และ F1-score ของการหาจุดสีฟ้ามีค่า 0.89 หมายความว่า ความถูกต้องของค่าที่บอกว่าโมเดลทำนายว่าจริง และค่าที่แสดงถึงโอกาสที่จะทำนายได้อย่างถูกต้อง มีความแม่นยำถึง 0.89 โดยเฉลี่ย ซึ่งถือว่าความแม่นยำเทียบเท่ากับการทำนายการหาจุดสีส้ม โดยโมเดล Decision Tree นี้ ค่าที่บ่งบอกถึงความสามารถในการทำนายค่าความถูกต้องของโมเดล หรือ accuracy มีค่า 0.89 ถือว่าค่อนข้างดีกว่าการใช้ Decision Tree ทั้งแบบใช้ GridSearchCV และไม่ใช้ GridSearchCV แต่ประสิทธิภาพเทียบเท่ากับการใช้ Random Forest แบบยังไม่จูนและแบบจูน และผลลัพธ์เท่ากับตอนที่ยังไม่จูน Adaboost RandomizedSearchCV เช่นกัน

Discussion

Advantages and Disadvantages of Decision Tree

ข้อดี

- Decision tree ได้ถูกออกแบบมาเพื่อรองรับกับข้อมูลแบบ non-linear เนื่องจากข้อมูลจริงในธรรมชาติที่เราได้พบเห็นอยู่บ่อย ๆ นั้นไม่ใช่ข้อมูลแบบเชิงเส้น (linear) เสมอไป
- Decision tree ให้ผลการฝึกอบรมโมเดลที่สามารถตีความได้ง่าย (Interpretability)
- Decision tree สามารถนำมาใช้ได้ทั้ง Regerssion และ Classification

ข้อเสีย

- ความแม่นยำ (Accuracy) ไม่สูงมากนักเมื่อเปรียบเทียบกับโมเดลอื่น
- หาก Decision tree มีความลึกมากไป อาจทำให้เกิดปัญหา Overfit ได้

Advantages and Disadvantages of Random Forest

ข้อดี

- Random forest สามารถนำมาใช้ได้ทั้ง Regerssion และ Classification
- Random forest สามารถจัดการกับชุดข้อมูลขนาดใหญ่ที่มีจำนวนมากได้มีประสิทธิภาพ ซึ่งเหมาะสมกับงานที่มีความซับซ้อน
- Random forest จะสนใจตัวแปรหรือค่าที่หายไปจากข้อมูล โดยอ้างอิงจากตัวแปรที่ปรากฏให้เห็นมากที่สุด

ข้อเสีย

- Random forest เป็นเครื่องมือที่ใช้เพียงแค่จำแนกประเภท ไม่ได้ใช้ในการอธิบายข้อมูล
- หากตัวโมเดลมีจำนวนต้นไม้ที่เยอะจะทำให้การทำงานของอัลกอริธึมช้าลงและไม่เหมาะสมที่จะนำมาใช้ในการทำนายแบบ real time

Advantages and Disadvantages of Adaboost

ข้อดี

- Adaboost สามารถนำมาใช้ได้ทั้ง Regerssion และ Classification
- เป็นวิธีที่ค่อนข้างยืดหยุ่น และ สามารถใช้ได้กับ Learning Method ที่ค่อนข้างหลากหลาย
- ปรับลด Bias ได้ดี และช่วยแก้ปัญหาความแม่นยำต่ำ

ข้อเสีย

- Adaboost ต้องการข้อมูลที่มีคุณภาพสูง เนื่องจากถ้าหากมี Noise หรือ Outliner จะทำให้ประสิทธิภาพแย่ลงอย่างมาก

- เนื่องจาก Adaboost ทำงานเป็นแบบ Sequential จึงอาจจะประมวลผลได้ช้ากว่าเมื่อเทียบกับ Random forest ที่ทำงานเป็นแบบ Parallel

อ้างอิง

https://link.springer.com/article/10.1007/s10462-020-09833-6

Hatwell, J., Gaber, M. M., & Azad, R. M. A. (2020). CHIRPS: Explaining random forest classification. Artificial Intelligence Review, 53(8), 5747–5788.

Ensemble Learning | Ensemble Learning Techniques (analyticsvidhya.com)

Srivastava, T. (2015, August 2). Ensemble learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/

Ensemble methods: bagging, boosting and stacking | by Joseph Rocca | Towards Data Science (medium.com)

Rocca, J. (2019, April 23). Ensemble methods: bagging, boosting and stacking. Towards Data Science. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Ensemble Learning คืออะไร - Arnondora

Puitrakul, A. (2020, October 28). Ensemble Learning คืออะไร. Arnondora. https://arnondora.in.th/what-is-ensemble-learning/

Decision Tree Classification. A Decision Tree is a simple… | by Afroz Chakure | The Startup | Medium

hakure, A. (2019, July 6). Decision Tree classification - the Startup - medium. The Startup. https://medium.com/swlh/decision-tree-classification-de64fc4d5aac

Decision-Tree Classifier Tutorial | Kaggle

rashant. (2020, March 13). Decision-tree classifier tutorial. Kaggle.Com; Kaggle. https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial

Hyperparameter Tuning สำหรับ Machine Learning Model - Big Data Thailand

Marlaithong, T., & Thamjaroenporn, P. (2021, December 7). เพิ่มประสิทธิภาพของ machine learning model ด้วย Hyperparameter Optimization. Big Data Thailand. https://bigdata.go.th/big-data-101/machine-learning-model-hyperparameter-optimization/

Hyperparameter-Tuning-/hyperparameter_optimization.ipynb at main · Linktnk/Hyperparameter-Tuning-(github.com)

https://towardsdatascience.com/https-medium-com-faizanahemad-generating-synthetic-classification-data-using-scikit-1590c1632922

Ahemad, F. (2019, March 13). Generating Synthetic Classification Data using Scikit. Towards Data Science. https://towardsdatascience.com/https-medium-com-faizanahemad-generating-synthetic-classification-data-using-scikit-1590c1632922