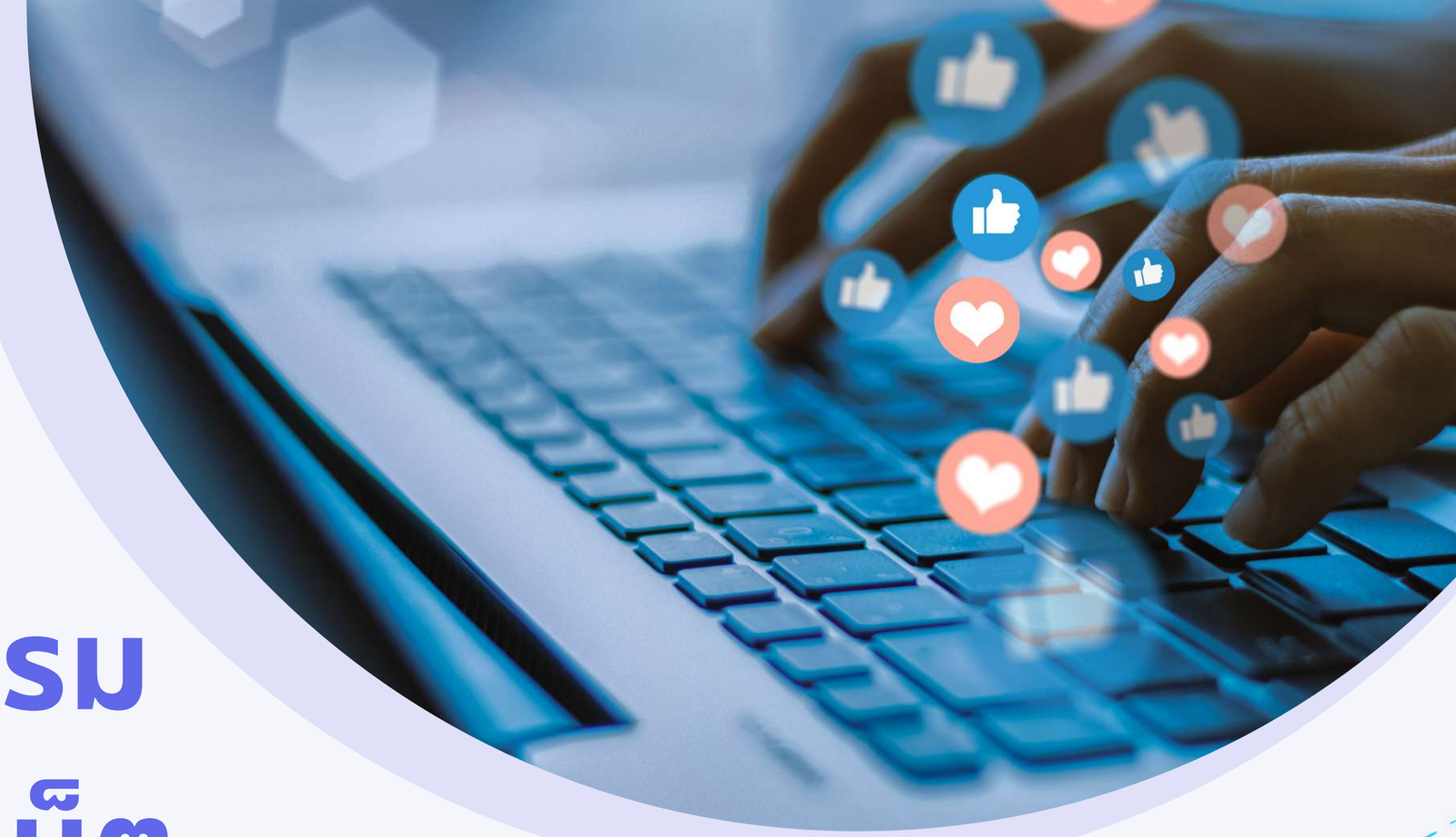


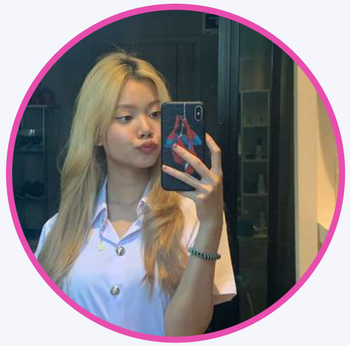
Decision tree ทำนาย พฤติกรรมการใช้อินเทอร์เน็ตในประเทศไทย

Presented by ได้หมดและจะดีหรอ

Our Team Meet

นายปิยพัทธ์ ปานะถึก

643020507-4



นางสาวพิมชนก วงศ์สายเชื้อ

643020510-5

นางสาววิภาดา ห่วงสูงเนิน

643020520-2

นางสาวหทัยชนก สรวงชัยภูมิ

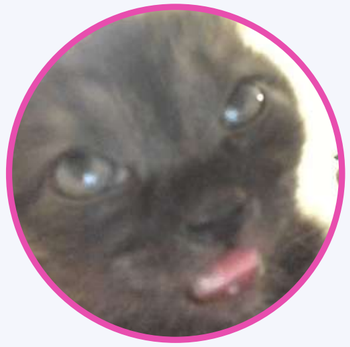
643020525-2

นายธนพร ก้านกิ่ง

643021264-9

นางสาวพัณณิตา ทองบ่อ

643020508-2



นายภูริศ เครือชารี

643020514-7

นางสาวสิรภัทร ไชยมาตย์

643020523-6

นายอาฤญช์ จรูญรักษ์

643020528-6

นางสาวจินดาพร โพธิ์ภูมี

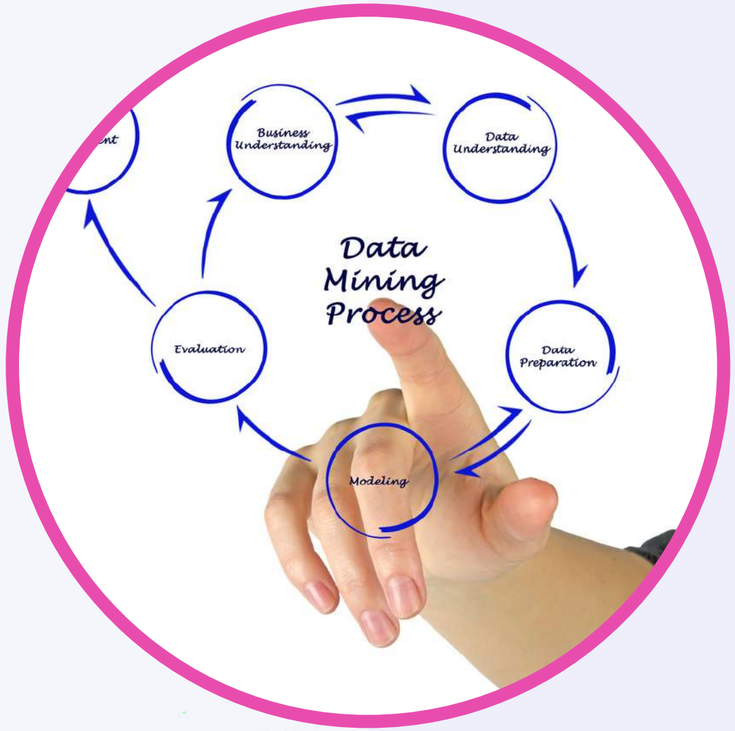
643021262-3

ที่มาและความสำคัญ

ปัจจุบันเข้าสู่สังคมออนไลน์ ผู้คนได้รับข่าวสารอย่างรวดเร็ว ทันเหตุการณ์
ในทางเดียวกันก็ยังประสบกับปัญหาข่าวปลอมที่ก่อให้เกิดความเข้าใจผิดกันในสังคม

สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (สพธอ.) ได้สำรวจพฤติกรรมการใช้อินเทอร์เน็ตเป็นประจำทุกปีตั้งแต่ปี 2556 โดยมีวัตถุประสงค์หลัก คือ เพื่อรวบรวมข้อมูลที่แสดงลักษณะและแนวโน้มพฤติกรรมการใช้อินเทอร์เน็ตของคนไทยที่เปลี่ยนแปลงอย่างต่อเนื่อง มีข้อมูลที่สำคัญและจำเป็นต่อการพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ไว้ให้บริการแก่ผู้ใช้ข้อมูลไม่ว่าจะเป็นภาครัฐ ภาคเอกชน และผู้สนใจ

การทำ Data Mining ช่วยในการเข้าใจถึงพฤติกรรมการใช้สื่อ ที่ส่งผลทำให้
เกิดการได้รับข่าวปลอม เพื่อใช้ในการหาแนวทางการแก้ไขปัญหาข่าวปลอมต่อไป



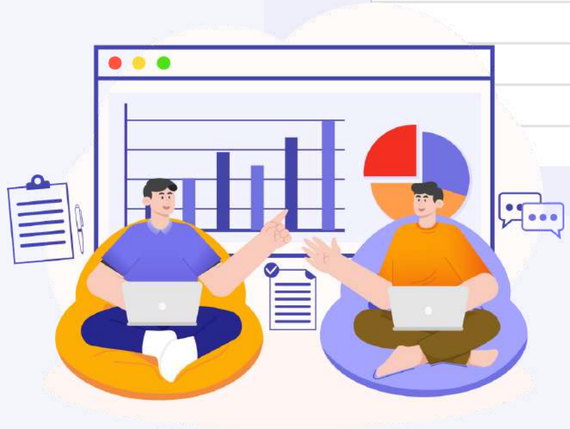
วัตถุประสงค์

เพื่อสร้างโมเดลทำนายการรับรู้ข่าวปลอม
จากข้อมูลสถิติการสำรวจพฤติกรรมผู้ใช้งาน
อินเทอร์เน็ตในประเทศไทย

ตัวอย่าง Data sets

ข้อมูลสถิติการสำรวจพฤติกรรมผู้ใช้งานอินเทอร์เน็ตในประเทศไทย

fake_news	online_acitivity_1	online_acitivity_2	online_acitivity_3	online_acitivity_4	online_acitivity_5	online_acitivity_6
2	1	0	1	1	1	1
2	1	0	0	1	0	0
2	1	1	1	1	1	1
2	1	0	1	1	1	0
2	0	1	0	1	0	1
2	1	0	0	0	1	0
2	1	0	1	1	1	1
2	1	0	0	0	0	0
1	1	1	0	1	1	0
2	1	1	0	1	1	1
2	1	0	0	1	1	1
2	1	0	0	1	1	0
2	1	0	0	0	1	0
2	1	0	0	1	1	0
2	1	0	1	1	1	1



variable
ตัวแปร

ตัวแปร Y : target variable

fake_news

ตัวแปร X : feature variables

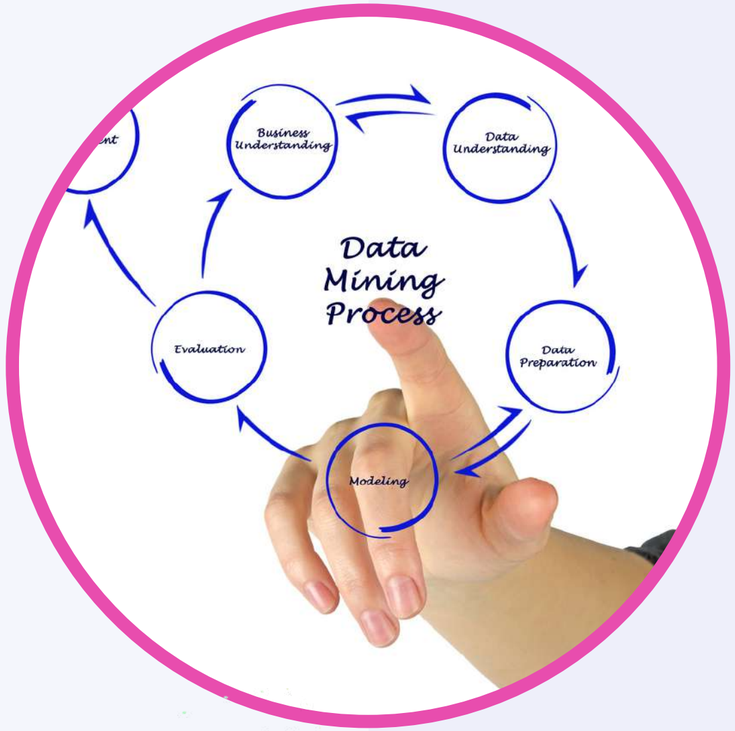
online_acitivity_1	online_acitivity_10
online_acitivity_2	online_acitivity_11
online_acitivity_3	online_acitivity_12
online_acitivity_4	online_acitivity_13
online_acitivity_5	online_acitivity_14
online_acitivity_6	online_acitivity_15
online_acitivity_7	online_acitivity_16
online_acitivity_8	online_acitivity_17
online_acitivity_9	online_acitivity_18

นิยามตัวแปร

fake_news	numeric override: numeric	1 = ไม่เคย 2 = เคย	ท่านเคยพบเห็นข่าวปลอม (Fake News) บนโลกออนไลน์หรือไม่
online_acitivity_1	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ใช้ Social Media เช่น Facebook, Twitter, Instagram
online_acitivity_2	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ใช้แอปพลิเคชันถ่ายทอดสด เช่น Facebook Live, Instagram Live ,YouTube live
online_acitivity_3	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	เล่นเกมออนไลน์
online_acitivity_4	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	รับ-ส่งอีเมล
online_acitivity_5	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ค้นหาข้อมูล (Search Engine) เช่น ค้นหาข้อมูลใน Google/Bing
online_acitivity_6	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	เรียนออนไลน์ (e-Learning)
online_acitivity_7	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	หางาน/สมัครงานทางออนไลน์
online_acitivity_8	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ซื้อขายสินทรัพย์เพื่อการลงทุน เช่น หุ้น กองทุนรวม
online_acitivity_9	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ติดต่อสื่อสารออนไลน์ ทั้งการโทรศัพท์ และพูดคุย (Chat)

นิยามตัวแปร

online_acitivity_10	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ขายสินค้าและบริการออนไลน์
online_acitivity_11	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ดาวน์โหลดซอฟต์แวร์/เพลง/ละคร/ภาพยนตร์/เกม/ไอเทมในเกม
online_acitivity_12	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ซื้อสินค้าและบริการออนไลน์
online_acitivity_13	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ดูโทรทัศน์/ดูคลิป/ดูหนัง/ฟังเพลงออนไลน์
online_acitivity_14	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	อ่านข่าว/บทความ/หนังสืออิเล็กทรอนิกส์ (e-Book)
online_acitivity_15	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ทำธุรกรรมทางการเงินออนไลน์
online_acitivity_16	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ใช้งานบริการภาครัฐผ่านระบบออนไลน์ เช่น ชำระภาษีออนไลน์
online_acitivity_17	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ทำงานผ่านระบบออนไลน์ ประชุมออนไลน์
online_acitivity_18	numeric override: numeric	0 = ไม่ได้ทำ 1= ทำ	ซื้อประกันออนไลน์ เช่น ประกันCOVID19 ประกันสุขภาพ ประกันอุบัติเหตุ ฯลฯ



Data Preprocessing

Step 1

จัดการ Missing Data

- หาค่า missing value จากข้อมูลทั้งหมด

```
[ ]  null_values = df.isnull().sum()

     print(null_values)
```

```
fake_news              0
online_acitivity_1     0
online_acitivity_2     0
online_acitivity_3     0
online_acitivity_4     0
online_acitivity_5     0
online_acitivity_6     0
online_acitivity_7     0
online_acitivity_8     0
online_acitivity_9     0
online_acitivity_10    0
online_acitivity_11    0
online_acitivity_12    0
online_acitivity_13    0
online_acitivity_14    0
online_acitivity_15    0
online_acitivity_16    0
online_acitivity_17    0
```

Step 2

Conditional Filtering

- ตรวจสอบค่าใน column fake_news ว่าเป็นค่า 1 และ 2 หรือไม่

```
# Check fake_news column
valid_fake_news = all(df['fake_news'].isin([1, 2]))
valid_fake_news
```

```
True
```

- ตรวจสอบค่าใน column online_acitivity ทั้งหมด ว่าเป็นค่า 0 และ 1 หรือไม่

```
[ ] # เลือกคอลัมน์ที่ 2 ถึงคอลัมน์สุดท้าย
    activity_columns = df.iloc[:, 1:]

    # ตรวจสอบว่าค่าทั้งหมดเป็น 0 หรือ 1
    valid_values = (activity_columns.isin([0, 1])).all().all()

    valid_values
```

```
False
```

Conditional Filtering

- ตรวจสอบค่าที่ไม่ใช่ตัวเลขในข้อมูล

```
17] non_numeric_values = df.apply(lambda x: pd.to_numeric(x, errors='coerce')).isnull().sum()

    print(non_numeric_values)
```

```
fake_news              0
online_acitivity_1     0
online_acitivity_2     0
online_acitivity_3     0
online_acitivity_4     0
online_acitivity_5     0
online_acitivity_6     0
online_acitivity_7     0
online_acitivity_8     0
online_acitivity_9     0
online_acitivity_10    0
online_acitivity_11    0
online_acitivity_12    0
online_acitivity_13    0
online_acitivity_14    0
online_acitivity_15    0
online_acitivity_16    0
online_acitivity_17    0
online_acitivity_18    1
dtype: int64
```

online_acitivity_18
S

- ลบ row ไม่ใช่ตัวเลข (เนื่องจากมีข้อมูลที่ผิดปกติไม่เกิน 5% ของข้อมูลทั้งหมด)

```
[41] non_numeric_rows = df.apply(lambda x: pd.to_numeric(x, errors='coerce')).isnull().any(axis=1)
     numeric_df = df[~non_numeric_rows]
     numeric_df
```

ตรวจสอบค่า Null หรือค่าผิดปกติ ใน DataFrame

- ตรวจสอบค่าใน column fake_news ได้ true

```
[5]  # Check fake_news column
     valid_fake_news = all(df['fake_news'].isin([1, 2]))
     valid_fake_news
```

```
True
```

- ตรวจสอบค่าใน column online_acitivity ทั้งหมด ได้ true

```
[6]  # เลือกคอลัมน์ที่ 2 ถึงคอลัมน์สุดท้าย
     activity_columns = df.iloc[:, 1:]

     # ตรวจสอบว่าค่าทั้งหมดเป็น 0 หรือ 1
     valid_values = (activity_columns.isin([0, 1])).all().all()

     valid_values
```

```
True
```

- ตรวจสอบค่าที่ไม่ใช่ตัวเลขในข้อมูล

```
non_numeric_values = df.apply(lambda x: pd.to_numeric(x, errors='coerce')).isnull().sum()

print(non_numeric_values)
```

```
fake_news              0
online_acitivity_1     0
online_acitivity_2     0
online_acitivity_3     0
online_acitivity_4     0
online_acitivity_5     0
online_acitivity_6     0
online_acitivity_7     0
online_acitivity_8     0
online_acitivity_9     0
online_acitivity_10    0
online_acitivity_11    0
online_acitivity_12    0
online_acitivity_13    0
online_acitivity_14    0
online_acitivity_15    0
online_acitivity_16    0
online_acitivity_17    0
online_acitivity_18    0
dtype: int64
```

กำหนด Features และ Target varible (X,Y)

```
[12] data = numeric_df.copy()
```

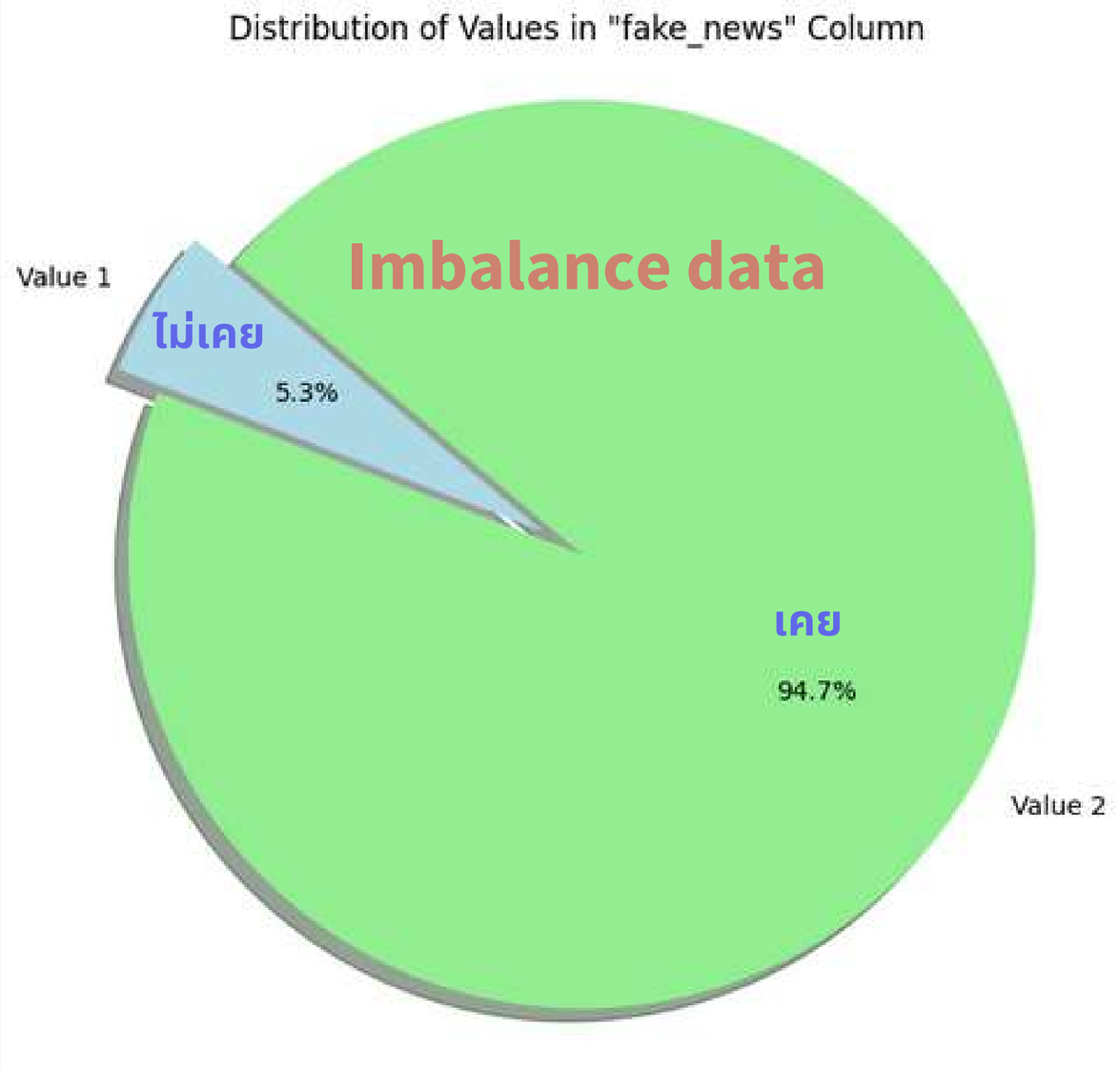
0 วินาที

```
[13] # Features (X)
     X = data.drop(columns=['fake_news'])  # คอลัมน์กิจกรรมออนไลน์ทั้งหมด

     # Target variable (y)
     y = data['fake_news']  # เลือกคอลัมน์ 'fake_news' เป็นตัวแปรเป้าหมาย
```

0 วินาที

สัดส่วน Fake_new



จัดการ Imbalance data

Method 1: Oversampling & Undersampling

- ปรับ train test เป็น 80% 20%

```
[16] from imblearn.over_sampling import RandomOverSampler
     from imblearn.under_sampling import RandomUnderSampler

     # แบ่งข้อมูลออกเป็นชุด
     X_train_OU, X_test_OU, y_train_OU, y_test_OU = train_test_split(X, y, test_size=0.2, random_state=0)
```

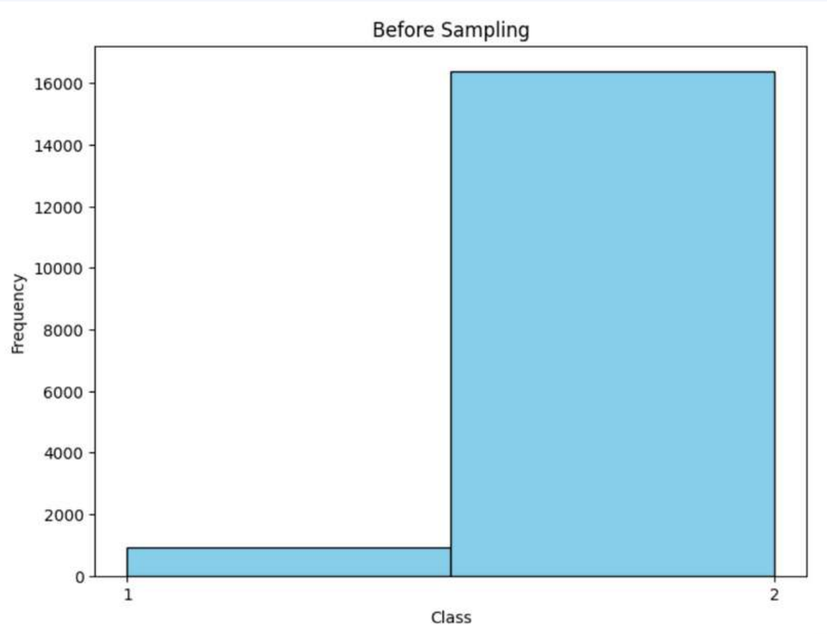
```
Count of values in y_train_OU:
Counter({2: 16400, 1: 918})

Count of values in y_test_OU:
Counter({2: 4097, 1: 233})
```

แสดง Histogram ของ y_train_OU ก่อนการทำ Over and Under Sampling

```
[17] import matplotlib.pyplot as plt

     # แสดง Histogram ของ y_train_OU ก่อนการทำ Over and Under Sampling
     plt.figure(figsize=(8, 6))
     plt.hist(y_train_OU, bins=2, color='skyblue', edgecolor='black')
     plt.title('Before Sampling')
     plt.xlabel('Class')
     plt.ylabel('Frequency')
     plt.xticks([1, 2])  # กำหนดแกน x เป็นคลาส 1 และ 2
     plt.show()
```

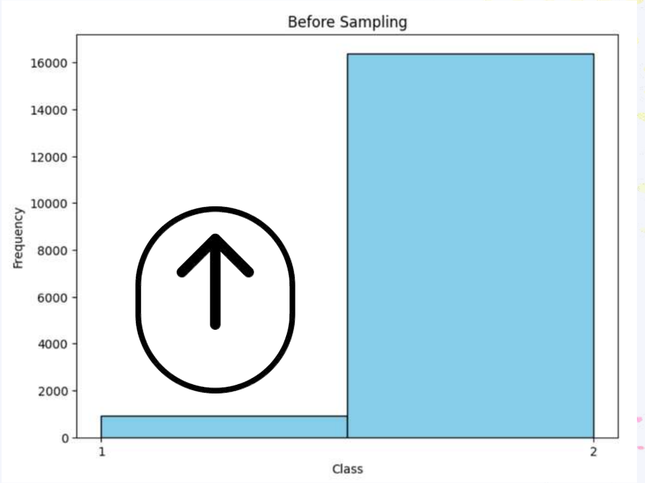



```
[ ] count_fake_news_encounters = df[df['fake_news'] == 1].shape[0]
    count_fake_news_encounters

    1151
```

Oversampling

การทำ Oversampling ช่วยเพิ่มประสิทธิภาพในการฝึกโมเดล โดยให้โมเดลมีข้อมูลมากขึ้นในคลาสที่มีจำนวนน้อย และช่วยลดโอกาสในการเกิด overfitting โดยไม่ต้องลดจำนวนข้อมูลในคลาสที่มีจำนวนมาก ทำให้สามารถสร้างโมเดลที่มีประสิทธิภาพและความสามารถในการทำนายได้ดีขึ้น

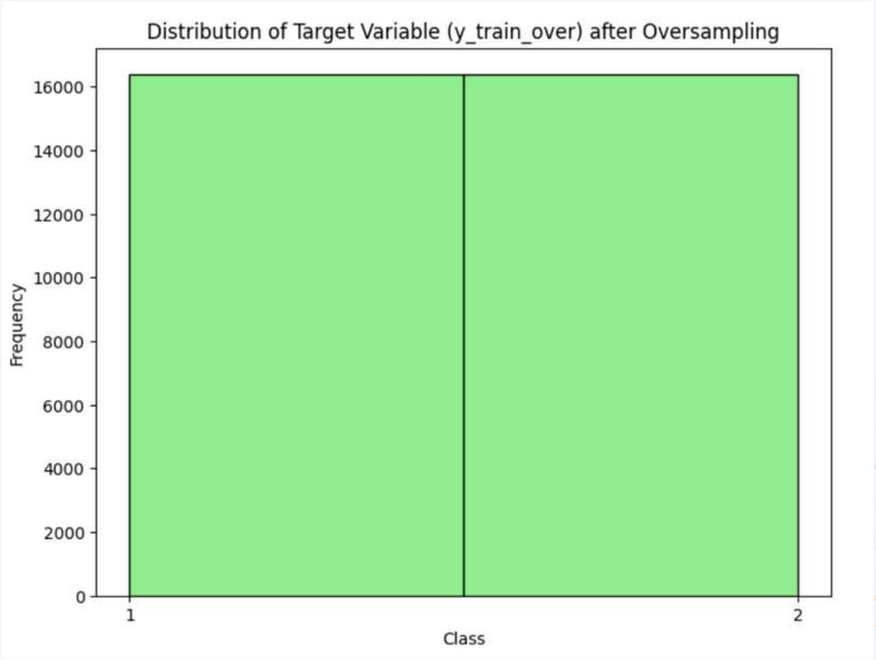


fit_resample ช่วยสร้างข้อมูลใหม่ในคลาส ที่มีข้อมูลน้อยกว่า และทำให้จำนวนข้อมูลในแต่ละคลาสเท่ากันหรือใกล้เคียงกัน

```
[18] # Oversampling
     oversampler = RandomOverSampler(random_state=0)
     X_train_over, y_train_over = oversampler.fit_resample(X_train_OU, y_train_OU) #fit_
```

```
import matplotlib.pyplot as plt

# แสดง Histogram ของ y_train_over หลัง Oversampling
plt.figure(figsize=(8, 6))
plt.hist(y_train_over, bins=2, color='lightgreen', edgecolor='black')
plt.title('Distribution of Target Variable (y_train_over) after Oversampling')
plt.xlabel('Class')
plt.ylabel('Frequency')
plt.xticks([1, 2])  # กำหนดแกน x เป็นคลาส 1 และ 2
plt.show()
```



1.1.Oversampling

```
# Oversampling
oversampler = RandomOverSampler(random_state=0)
X_train_over, y_train_over = oversampler.fit_resample(X_train_OU, y_train_OU) #fit_resample ช่วยสร้างข้อมูลใหม่ในคลาส ที่มีข้อมูลน้อยกว่า และทำให้จำนวนข้อมูลในแต่ละคลาสเท่ากันหรือใกล้เคียงกัน
```

จำนวนข้อมูล y_train_over หลังการทำ Oversampling

```
count_y_train_over = Counter(y_train_over)

print(count_y_train_over)
print()
```

```
Counter({2: 16393, 1: 16393})
```

ฝึกโมเดล Decision Tree

```
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train_over, y_train_over)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

Cross - Validation
Oversampling

กำหนดจำนวน K ใน K-Fold Cross Validation
และ ประเมินโมเดลโดยใช้ Cross Validation

```
# กำหนดจำนวน K ใน K-Fold Cross Validation
k_fold = KFold(n_splits=5, shuffle=True, random_state=0)

# ประเมินโมเดลโดยใช้ Cross Validation
scores = cross_val_score(tree_over, X_test_OU, y_test_OU, cv=k_fold)
```

Cross Validation scores:
[0.8949 0.8972 0.8972 0.8845 0.8868]
Average Cross Validation score: 0.8921

Confusion Matrix
Oversampling

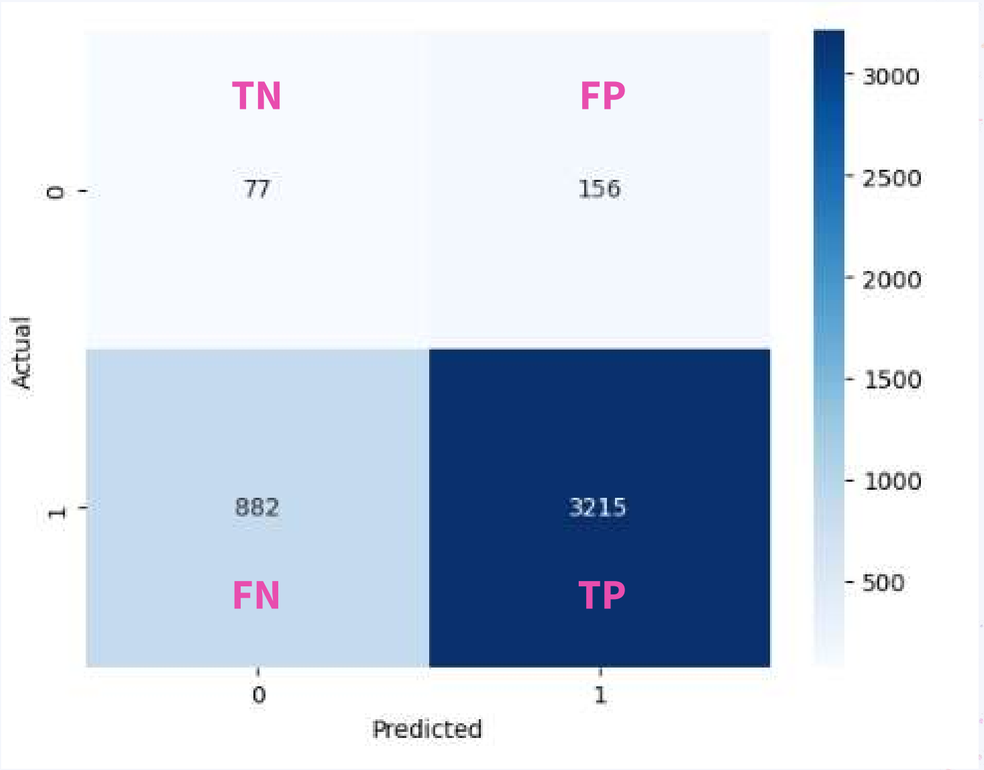
ทำนายบนชุดข้อมูลทดสอบ

```
[29] from sklearn.metrics import confusion_matrix

[30] # ทำนายบนชุดข้อมูลทดสอบ
     y_pred = tree_over.predict(X_test_OU)

[31] cm = confusion_matrix(y_test_OU, y_pred)

[32] sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.show()
```



วัดประสิทธิภาพของโมเดล

```python
# วัดประสิทธิภาพของโมเดล
accuracy = accuracy_score(y_test_OU, y_pred)
precision = precision_score(y_test_OU, y_pred, average='binary')  # binary average สำหรับประสิทธิภาพของคลาสเดียว
recall = recall_score(y_test_OU, y_pred, average='binary')
f1 = f1_score(y_test_OU, y_pred, average='binary')

# แสดงค่าประสิทธิภาพของโมเดล
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

ประสิทธิภาพของโมเดล

Accuracy: 0.7602

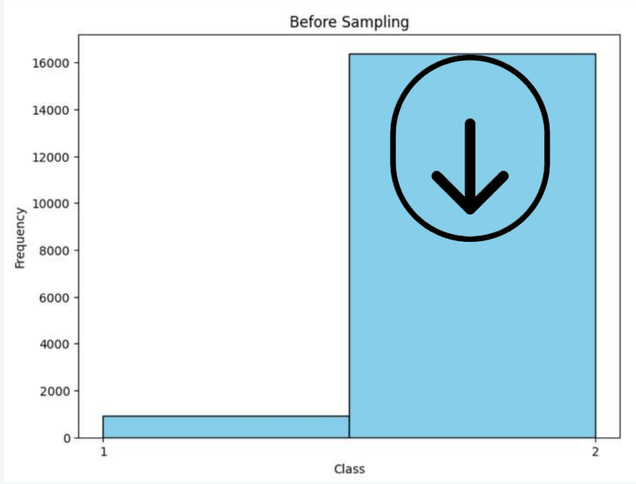
Precision: 0.0802

Recall: 0.3304

F1 Score: 0.1291

1.2.Undersampling

ในกรณีที่จำนวนของตัวอย่างในคลาสหนึ่งมีจำนวนมากกว่าคลาสอื่นๆ อย่างมาก เทคนิคนี้ทำงานโดยการลดจำนวนตัวอย่างในคลาสที่มีจำนวนมาก เพื่อให้สัดส่วนของคลาสในชุดข้อมูลมีความสมดุลมากขึ้น อย่างไรก็ตาม อาจทำให้สูญเสียข้อมูลที่มีความสำคัญ และยังมีความเสี่ยงที่จะทำให้โมเดลไม่สามารถจำแนกคลาสที่มีความน่าสนใจได้อย่างเพียงพอ

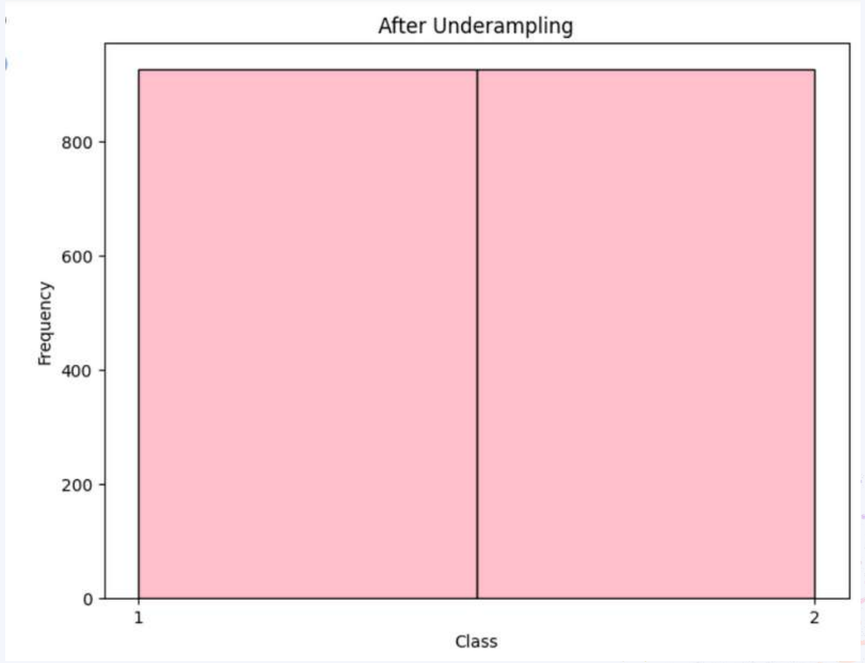


ทำการสุ่มตัวอย่างในคลาสที่มีจำนวนมาก ให้มีจำนวนเท่ากับคลาสที่มีจำนวนน้อย

```
[36] # Undersampling
     undersampler = RandomUnderSampler(random_state=0)
     X_train_under, y_train_under = undersampler.fit_resample(X_train_OU, y_train_OU)
```

```
✓ 0 วินาที
[37] # แสดงกราฟ
     import matplotlib.pyplot as plt

     # แสดง Histogram ของ y_train_over หลัง Oversampling
     plt.figure(figsize=(8, 6))
     plt.hist(y_train_under, bins=2, color='pink', edgecolor='black')
     plt.title('After Underampling')
     plt.xlabel('Class')
     plt.ylabel('Frequency')
     plt.xticks([1, 2])  # กำหนดแกน x เป็นคลาส 1 และ 2
     plt.show()
```



จำนวนข้อมูล y_train_over หลังการทำ Undersampling

```
count_y_train_under = Counter(y_train_under)

print(count_y_train_under)
print()
```

```
Counter({1: 918, 2: 918})
```

ฝึกโมเดล Decision Tree

```
[38] # ฝึกโมเดล Decision Tree
     tree = DecisionTreeClassifier(random_state=0)
     tree.fit(X_train_under, y_train_under)
```

```
DecisionTreeClassifier(random_state=0)
```

Cross - Validation

Undersampling

สร้าง KFold cross-validator
คำนวณคะแนน cross-validation

```
# กำหนดจำนวน K ใน K-Fold Cross Validation
k_fold = KFold(n_splits=5, shuffle=True, random_state=0)

# ประเมินโมเดลโดยใช้ Cross Validation
scores = cross_val_score(tree_under, X_test_OU, y_test_OU, cv=k_fold)
```

Cross Validation scores: [0.8949 0.8972 0.8972 0.8845 0.8868]
Average Cross Validation score: 0.8921

Confusion Matrix
Undersampling

ทำนายบนชุดข้อมูลทดสอบ

```
[ ]  from sklearn.metrics import confusion_matrix

[ ]  # ทำนายบนชุดข้อมูลทดสอบ
     y_pred = tree_under.predict(X_test_OU)

[ ]  cm = confusion_matrix(y_test_OU, y_pred)

     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.show()
```

วัดประสิทธิภาพของโมเดล

```
[ ]  # วัดประสิทธิภาพของโมเดล
     accuracy = accuracy_score(y_test_OU, y_pred)
     precision = precision_score(y_test_OU, y_pred, average='binary')  # binary average สำหรับประสิทธิภาพของคลาสเดียว
     recall = recall_score(y_test_OU, y_pred, average='binary')
     f1 = f1_score(y_test_OU, y_pred, average='binary')

     # แสดงค่าประสิทธิภาพของโมเดล
     print("Accuracy:", accuracy)
     print("Precision:", precision)
     print("Recall:", recall)
     print("F1 Score:", f1)
```

Accuracy: 0.5340
Precision: 0.0620
Recall: 0.5407
F1 Score: 0.1110

Method 2: Cost-sensitive Learning

2.1 class_weights = {0: 10, 1: 1}

```
[49] # แบ่งข้อมูลออกเป็นชุด
     X_train_cost, X_test_cost, y_train_cost, y_test_cost = train_test_split(X, y, test_size=0.2, random_state=0)
```

แบ่งข้อมูลออกเป็นชุด

replace

- แทนที่ '2' ด้วย '1' และ '1' ด้วย '0' ในคลาส เพราะ Cost-sensitive Learning ต้องใช้ตัวเลขที่เป็นไบนารีเท่านั้น หรือ 0 , 1

```
[50] y_train_binary = y_train_cost.replace({2: 1, 1: 0})
     y_test_binary = y_test_cost.replace({2: 1, 1: 0})
```

```
[51] class_weights = {0: 10, 1: 1}
```

weight

- ปรับน้ำหนัก โมเดลจะให้ความสำคัญกับคลาสที่มีค่าเป็น 1 มากกว่าคลาสที่มีค่าเป็น 0
- คลาสที่มีค่าเป็น 0 จะมีค่าน้ำหนักเท่ากับ 1 คลาสที่มีค่าเป็น 1 จะมีค่าน้ำหนักเท่ากับ 10

Train Model

```
tree_cost = DecisionTreeClassifier(random_state=0, class_weight=class_weights)
tree_cost.fit(X_train_cost, y_train_binary)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(class_weight={0: 10, 1: 1}, random_state=0)
```

Cross - Validation

2.1 class_weights = {0: 10, 1: 1}

กำหนดจำนวน K ใน K-Fold Cross Validation
และ ประเมินโมเดลโดยใช้ Cross Validation

```
# กำหนดจำนวน K ใน K-Fold Cross Validation
k_fold = KFold(n_splits=5, shuffle=True, random_state=0)

# ประเมินโมเดลโดยใช้ Cross Validation
scores = cross_val_score(tree_cost, X_test_cost, y_test_binary, cv=k_fold)
```

Cross Validation scores: 0.8325 0.8394 0.8487 0.8510 0.8233

Average Cross Validation score: 0.8390

Confusion Matrix

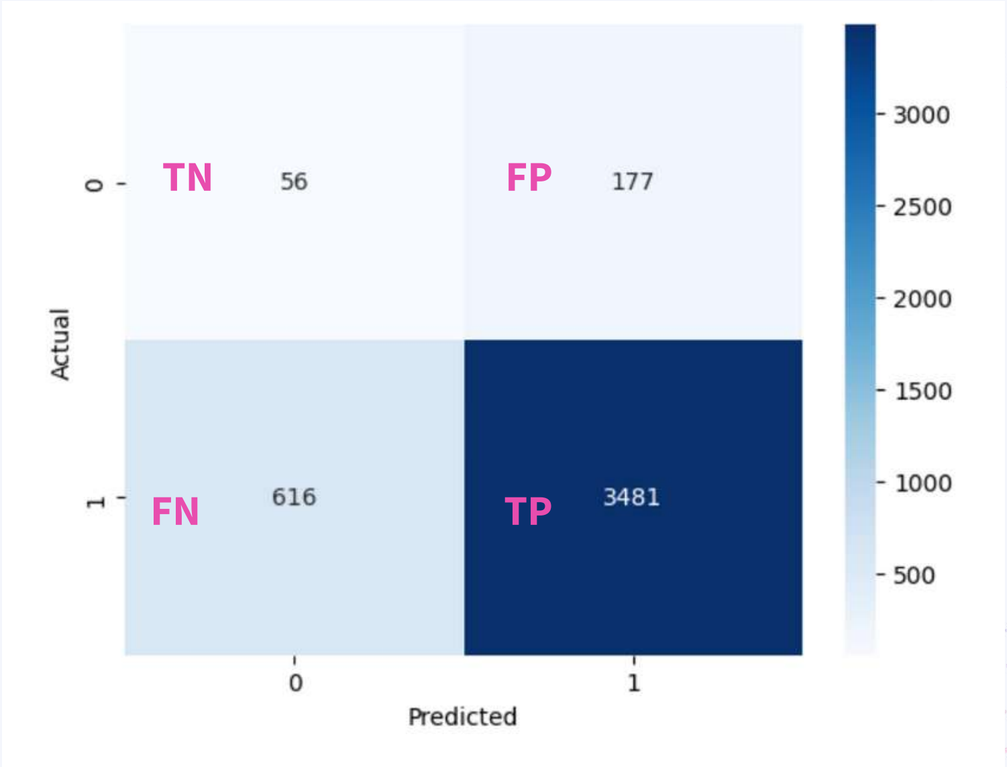
2.1 class_weights = {0: 10, 1: 1}

ทำนายบนชุดข้อมูลทดสอบ

```
    y_pred = tree_cost.predict(X_test_cost)

[56] cm = confusion_matrix(y_test_binary, y_pred)

[57] sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.show()
```



วัดประสิทธิภาพของโมเดล

```python
# ประเมิน model
accuracy = accuracy_score(y_test_binary, y_pred)
precision = precision_score(y_test_binary, y_pred)
recall = recall_score(y_test_binary, y_pred)
f1 = f1_score(y_test_binary, y_pred)

# แสดงค่าประสิทธิภาพของโมเดล
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision:, {precision:.4f}")
print(f"Recall:, {recall:.4f}")
print(f"F1 Score:, {f1:.4f}")
```

Accuracy: 0.8169

Precision: 0.9516

Recall:, 0.8496

F1 Score:, 0.8977

2.2 class_weight='balanced'

```
from collections import Counter

# นับค่าที่ปรากฏซ้ำกันใน y_train_cw
count_y_train_cw = Counter(y_train_cw)

# นับค่าที่ปรากฏซ้ำกันใน y_test_cw
count_y_test_cw = Counter(y_test_cw)


print("Count of values in y_train_cw:")
print(count_y_train_cw)
print()

print("Count of values in y_test_cw:")
print(count_y_test_cw)
print()
```

Count of values in y_train_cw:
Counter({2: 16400, 1: 918})
Count of values in y_test_cw:
Counter({2: 4097, 1: 233})

Cross - Validatoin

2.2 class_weight='balanced'

สร้าง KFold cross-validator
คำนวณคะแนน cross-validation

Cross-validation scores:
[0.8129 0.8302 0.8290
0.8348 0.8267]
Mean score: 0.8267

```
# สร้าง KFold cross-validator
kf = KFold(n_splits=5, shuffle=True, random_state=0)

# คำนวณคะแนน cross-validation
cv_scores = cross_val_score(tree_cw, X_test_cw,y_test_cw, cv=kf)
```

Confusion Matrix

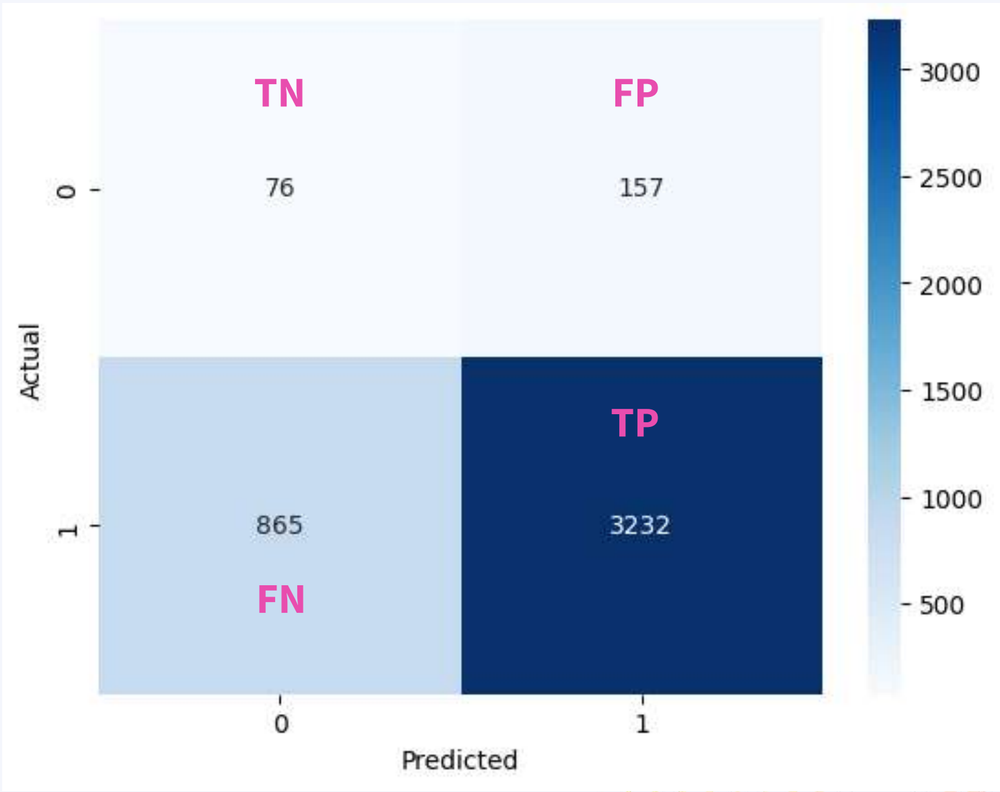
2.2 class_weight='balanced'

ทำนายบนชุดข้อมูลทดสอบ

```
[ ]  # ทำนายบนชุดข้อมูลทดสอบ
     y_pred = tree_cw.predict(X_test_cw)

[ ]  cm = confusion_matrix(y_test_cw, y_pred)

     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.show()
```



วัดประสิทธิภาพของโมเดล

Accuracy: 0.7640

Precision:, 0.0808

Recall:, 0.3262

F1 Score:, 0.1295

```
# คำนวณและพิมพ์ความแม่นยำของโมเดล
accuracy = accuracy_score(y_test_cw, y_pred)
precision = precision_score(y_test_cw, y_pred)
recall = recall_score(y_test_cw, y_pred)
f1 = f1_score(y_test_cw, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision:, {precision:.4f}")
print(f"Recall:, {recall:.4f}")
print(f"F1 Score:, {f1:.4f}")
```

Method 3: Ensemble learning

Ensemble learning

แบ่งชุดข้อมูล

```
X_train_ssg, X_test_ssg, y_train_ssg, y_test_ssg = train_test_split(X, y, test_size=0.2, random_state=0)
```

นับค่าที่ปรากฏซ้ำกันใน y_train_ssg
นับค่าที่ปรากฏซ้ำกันใน y_test_ssg

```
# นับค่าที่ปรากฏซ้ำกันใน y_train_ssg
count_y_train_ssg = Counter(y_train_ssg)

# นับค่าที่ปรากฏซ้ำกันใน y_test_ssg
count_y_test_ssg = Counter(y_test_ssg)


print("Count of values in y_train_ssg:")
print(count_y_train_ssg)
print()

print("Count of values in y_test_ssg:")
print(count_y_test_ssg)
print()
```

Count of values in y_train_ssg:
Counter({2: 16393, 1: 926})
Count of values in y_test_ssg:
Counter({2: 4105, 1: 225})

Ensemble learning

```
smote = SMOTE(random_state=0)
X_train_smote, y_train_smote = smote.fit_resample(X_train_ssg, y_train_ssg)
```

ใช้ SMOTE เพื่อจัดการกับข้อมูลไม่สมดุล

```
[72] from collections import Counter

     y_train_smote_counts = Counter(y_train_smote)

     print(y_train_smote_counts)
```

Counter({2: 16393, 1: 16393})

```
tree_smote = DecisionTreeClassifier(random_state=0)
tree_smote.fit(X_train_smote, y_train_smote)
```

```
▼ DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

สร้างและฝึกโมเดล

Cross - Validatoin

Ensemble learning

กำหนดจำนวน K ใน K-Fold Cross Validation
และ ประเมินโมเดลโดยใช้ Cross Validation

```
# กำหนดจำนวน K ใน K-Fold Cross Validation
k_fold = KFold(n_splits=5, shuffle=True, random_state=0)

# ประเมินโมเดลโดยใช้ Cross Validation
scores = cross_val_score(tree_smote, X_test_ssg, y_test_ssg, cv=k_fold)
```

Cross Validation scores: [0.8949 0.8972 0.8972 0.88450.8868]
Average Cross Validation score: 0.8921

Confusion Matrix

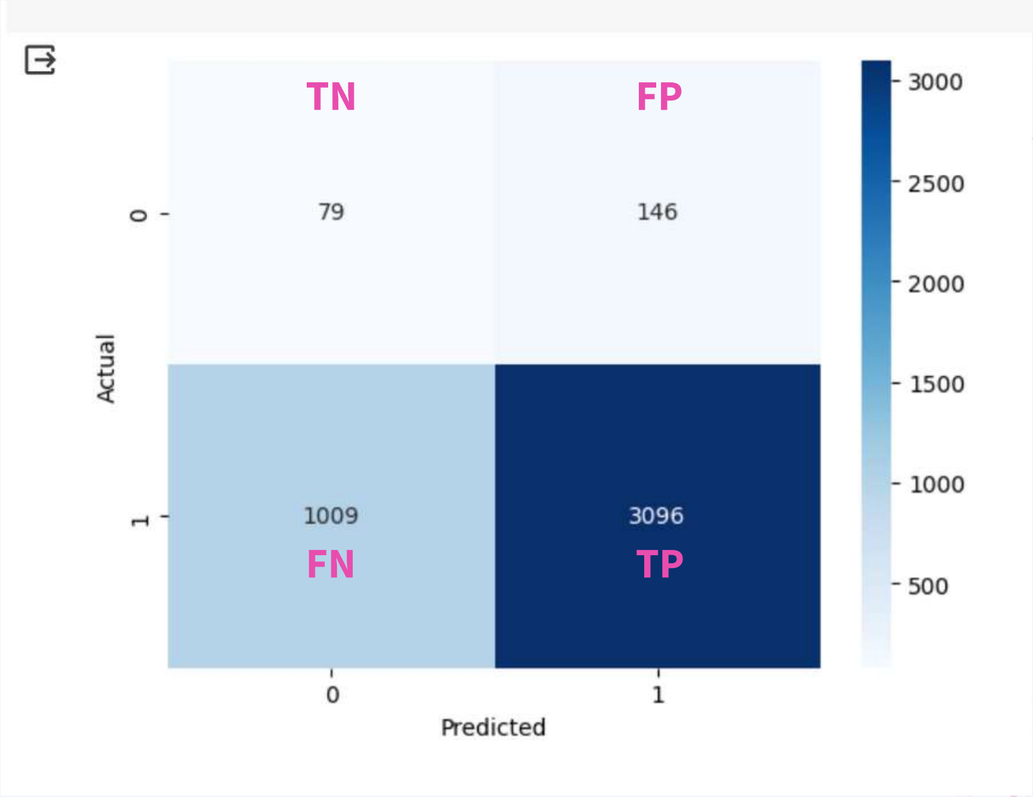
Ensemble learning

ทำนายบนชุดข้อมูลทดสอบ

```
[75] # ทำนายบนชุดข้อมูลทดสอบ
     y_pred = tree_smote.predict(X_test_ssg)

[76] cm = confusion_matrix(y_test_ssg, y_pred)

     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.show()
```



วัดประสิทธิภาพของโมเดล

```python
# วัดประสิทธิภาพของโมเดล
accuracy = accuracy_score(y_test_ssg, y_pred)
precision = precision_score(y_test_ssg, y_pred, average='binary')
recall = recall_score(y_test_ssg, y_pred, average='binary')
f1 = f1_score(y_test_ssg, y_pred, average='binary')

# แสดงค่าประสิทธิภาพของโมเดล
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

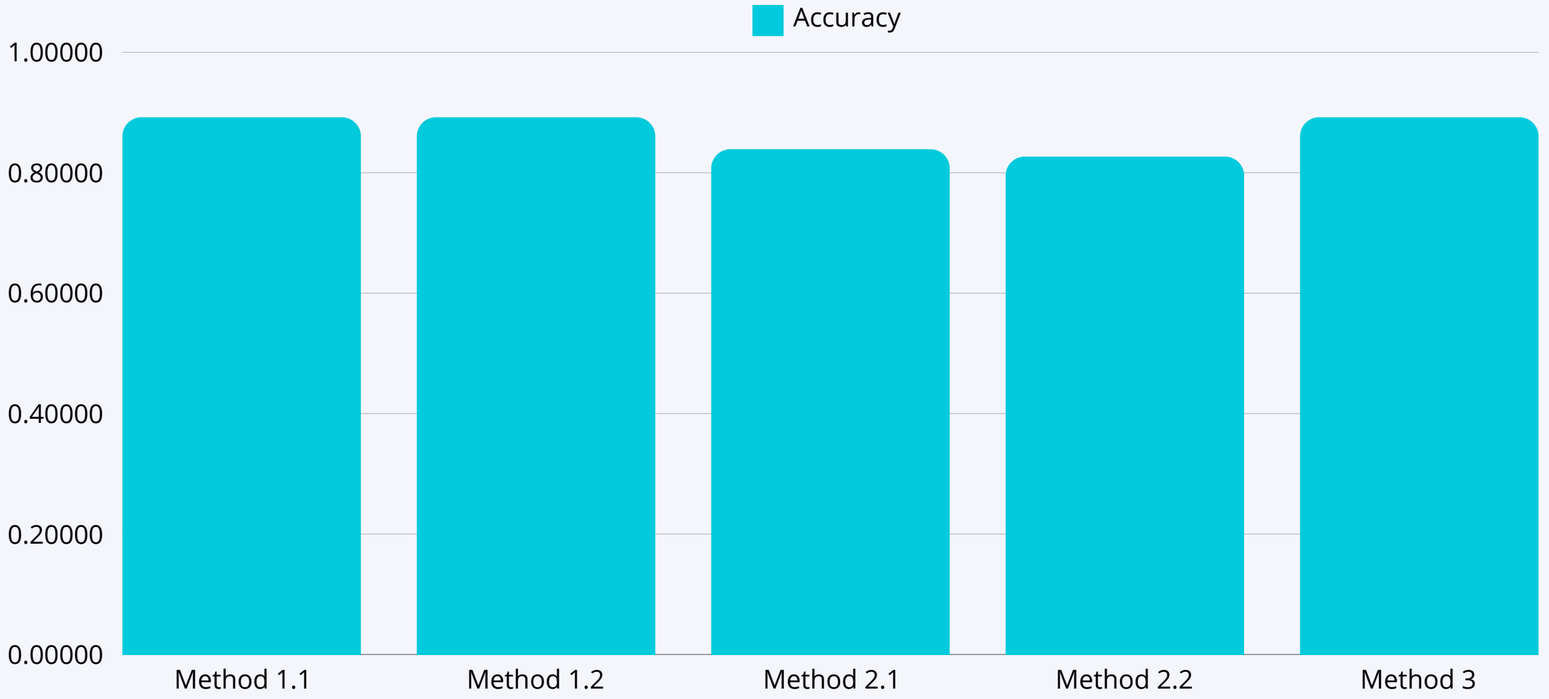
Accuracy: 0.7385

Precision: 0.0731

Recall: 0.3304

F1 Score: 0.1193

เปรียบเทียบ Mean score จาก cross valid



Method 1.1
Oversampling

Method 1.2
Undersampling

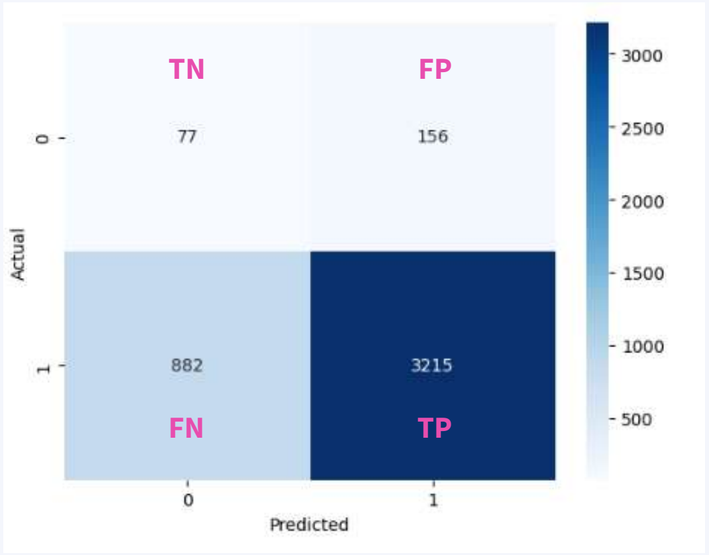
Method 2.1
Cost-sensitive Learning
class_weights

Method 2.1
Cost-sensitive Learning
class_weight='balanced'

Method 3
Ensemble learning

จากค่า Average Cross Validation score พบว่ามี 3 วิธีที่มีค่ามากที่สุดคือ
วิธีที่1.1Oversampling
วิธีที่1.2 Undersampling
วิธีที่ 3 Ensemble learning
ซึ่งทั้ง3วิธีมีค่า Accuracy สูงที่สุด คือ 0.89215

เปรียบเทียบ Confusion Matrix

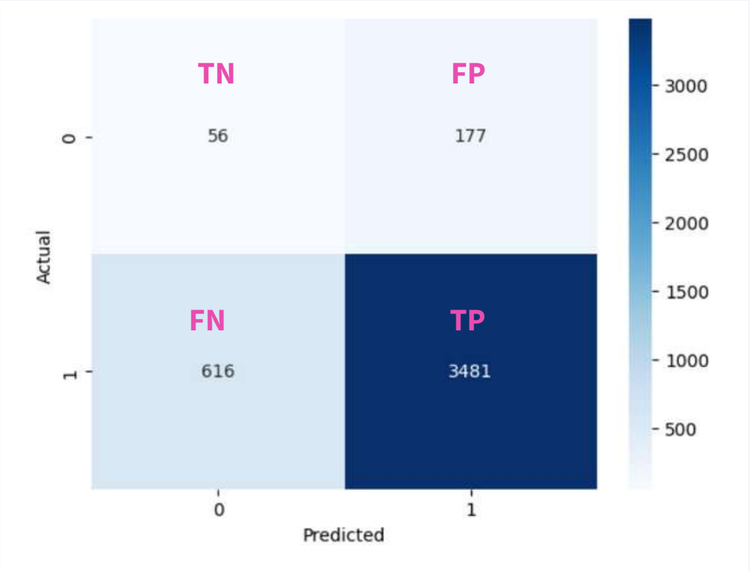


Method 1.1
Oversampling

Accuracy: 0.7602
Precision: 0.0802
Recall: 0.3304
F1 Score: 0.1291

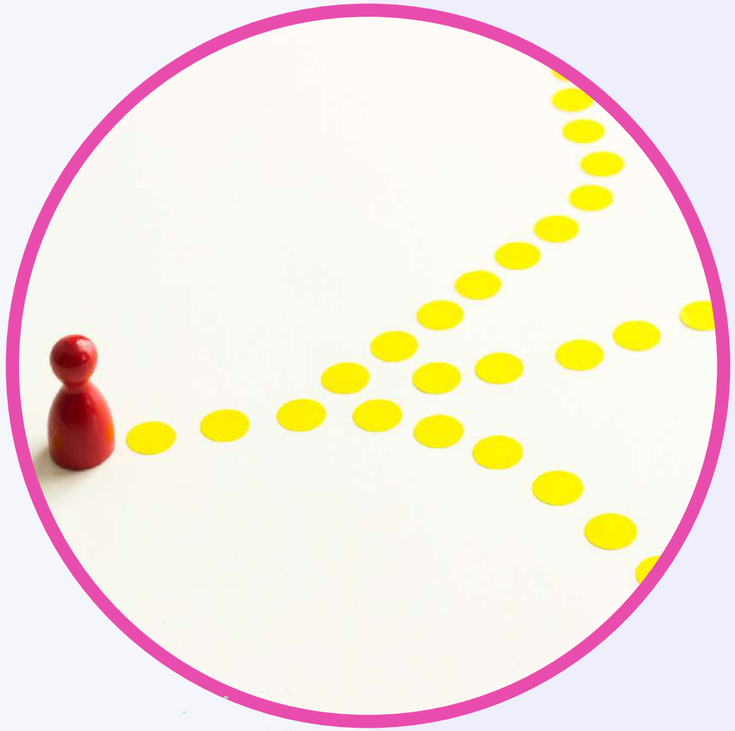
Method 1.2
Undersampling

Accuracy: 0.5339
Precision: 0.0618
Recall: 0.5407
F1 Score: 0.1110



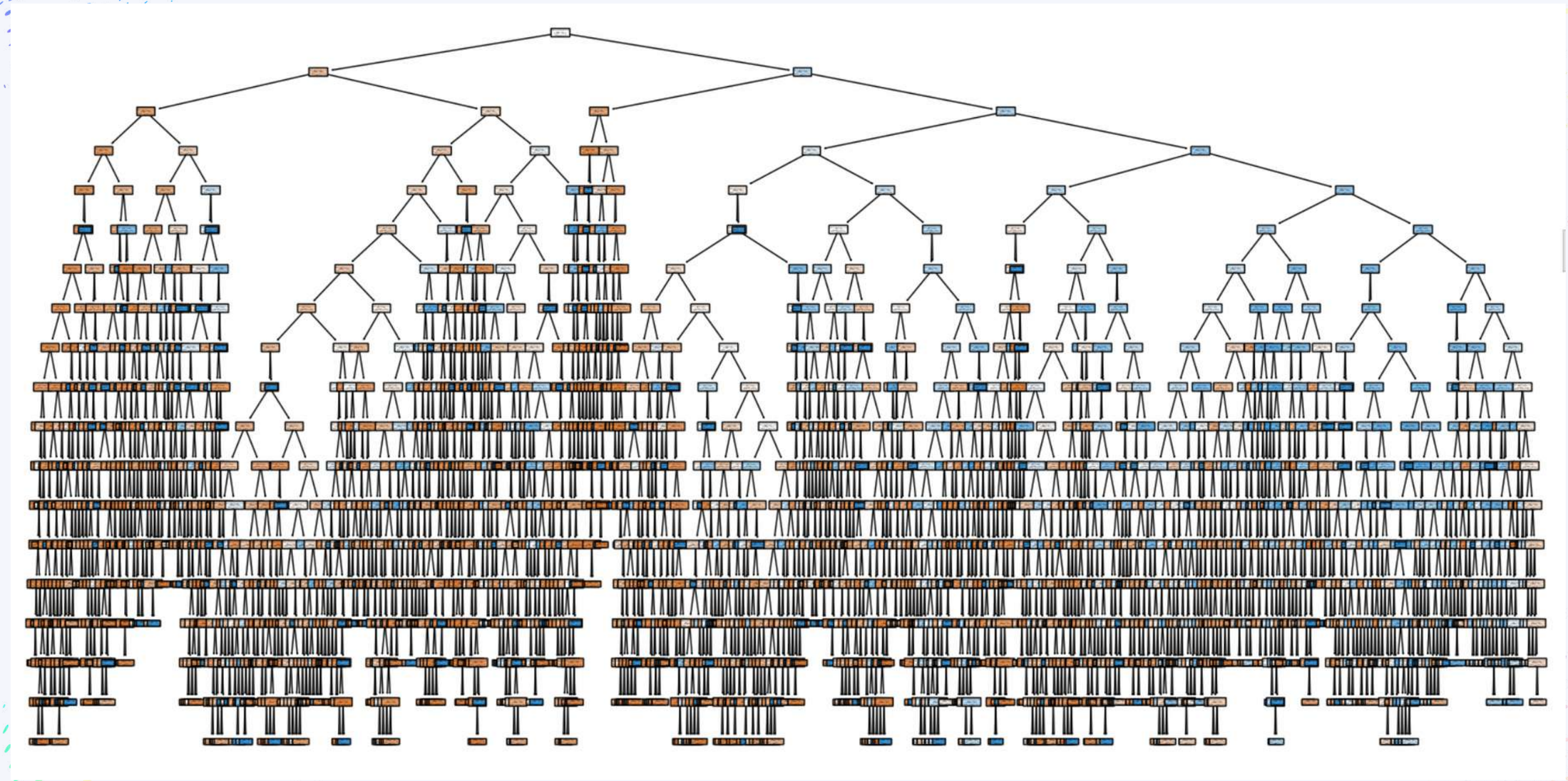
Method 3
Ensemble learning

Accuracy: 0.7385
Precision: 0.0731
Recall: 0.3304
F1 Score: 0.1193



Decision Tree

Decision tree ก่อนกำหนดพารามิเตอร์



หาค่าพารามิเตอร์ที่เหมาะสมสำหรับ Decition tree โดยใช้ Grid SearchCV

ตั้งค่าพารามิเตอร์ที่ต้องการทดสอบ
เพื่อให้การสร้างโมเดล Decision Tree มีความแม่นยำมากที่สุด

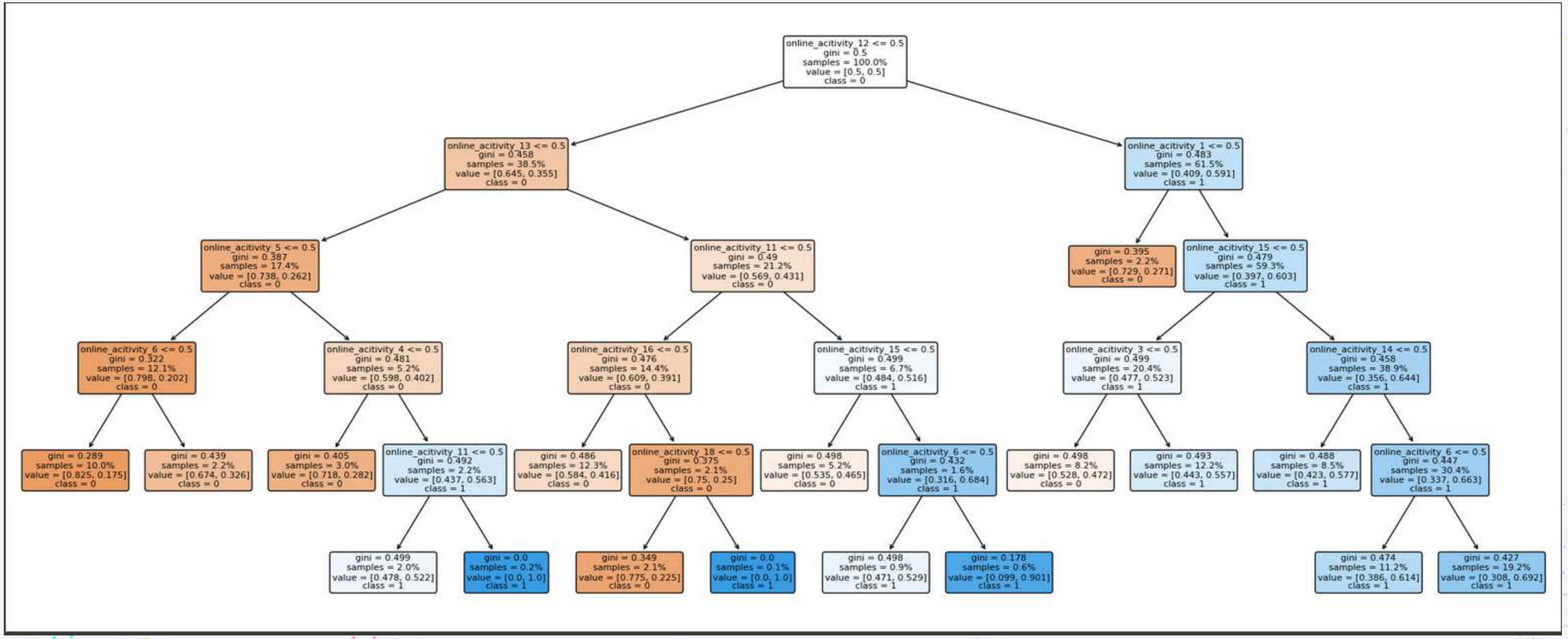
```
# Define parameter grid
param_grid = {
    'max_depth': [2, 3, 4, 5, 6, 7 ],
    'max_leaf_nodes' : [7, 9, 11, 13, 15, 17],
    'min_samples_split': [ 3, 5, 7, 10],
    'min_samples_leaf': [ 1, 3, 5, 7],
    'max_features': [None, 'sqrt'],
    'criterion': ['gini', 'entropy']
    # ,    # 'random_state' : [1]
}
```

Best parameters: {'criterion': 'gini', 'max_depth': 5, 'max_features': None, 'max_leaf_nodes': 17, 'min_samples_leaf': 1, 'min_samples_split': 3} Best score: 0.633689024390243 9

Decision tree หลังจากกำหนดพารามิเตอร์

มีทั้งหมด 17 Leaf Node

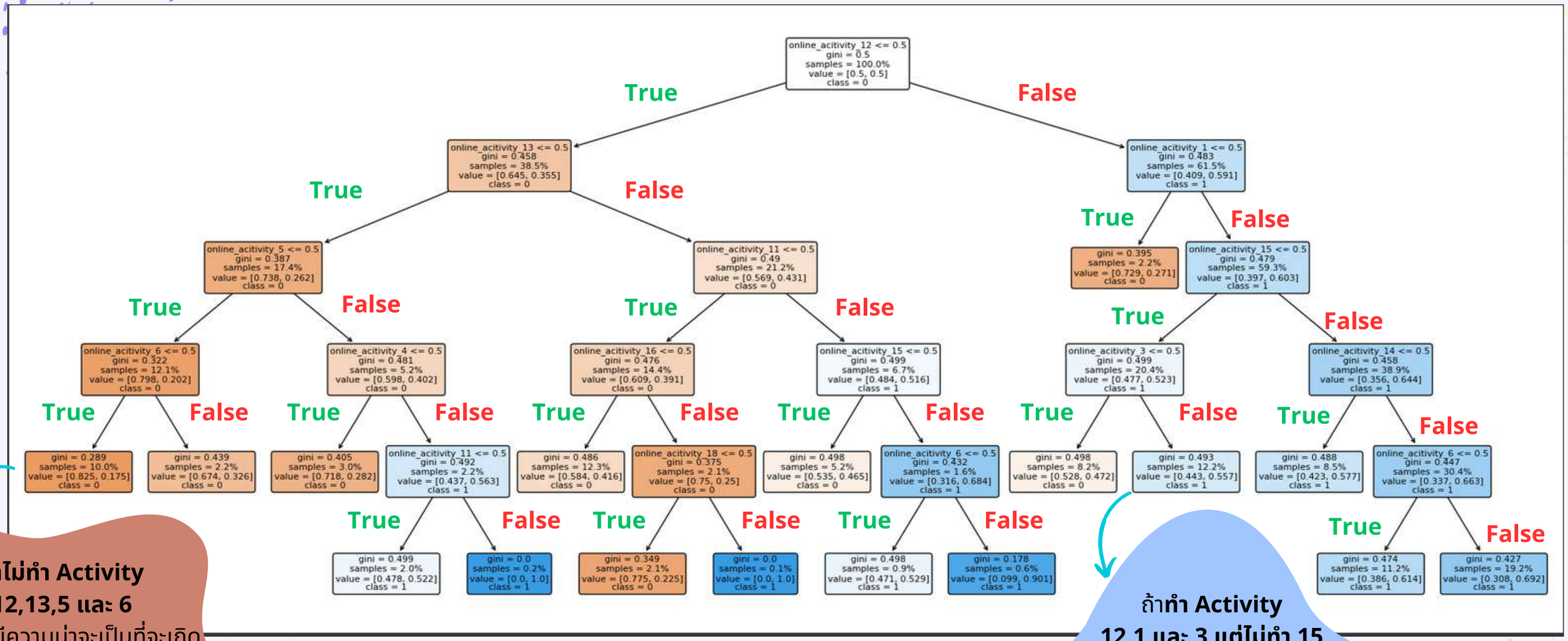
- **Leaf node ที่ทำนายได้ Class 0 หรือไม่เคยเจอข่าวปลอม มีจำนวน 8 Leaf node**
- **Leaf node ที่ทำนายได้ Class 1 หรือเคยเจอข่าวปลอม มีจำนวน 9 Leaf node**



Accuracy: 0.6789 **Recall: 0.5622**

Precision: 0.0923 **F1 Score: 0.1586**

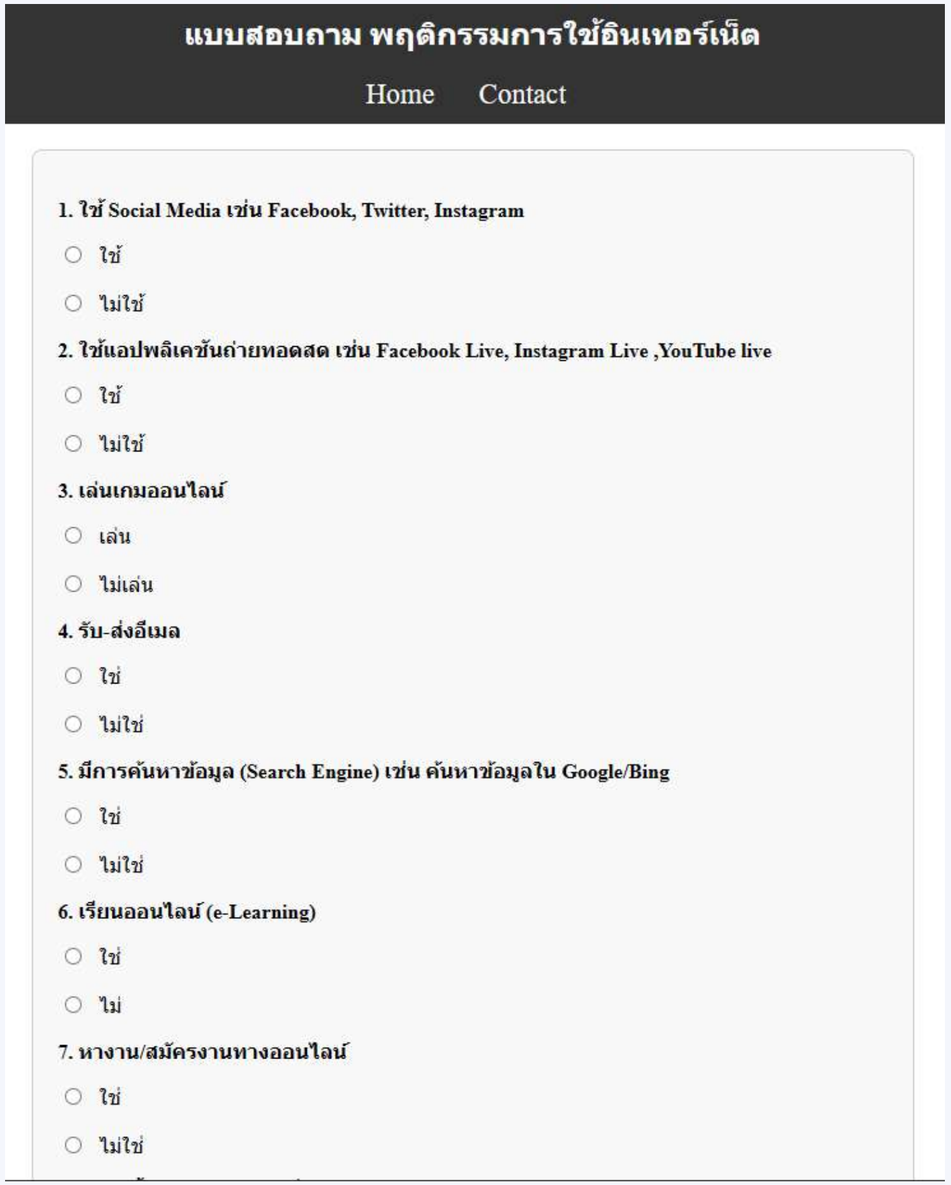
การแปรผล Decision tree



ถ้า**ไม่ทำ Activity 12,13,5 และ 6** จะทำให้ มีความน่าจะเป็นที่จะเกิดเหตุการณ์ 0 หรือ**ไม่เคยเจอข่าวปลอม**เท่ากับ 0.825 หรือ คิดเป็น 82.5%

ถ้า**ทำ Activity 12,1 และ 3 แต่ไม่ทำ 15** จะทำให้ มีความน่าจะเป็นที่จะเกิดเหตุการณ์ 1 หรือ**เคยเจอข่าวปลอม** เท่ากับ 0.557 หรือ คิดเป็น 55.7%

Prototype web Prediction fake news



ประโยชน์ที่ได้จากการทำนาย

ใช้เป็นเครื่องมือในการทำนายพฤติกรรมการใช้อินเทอร์เน็ตของคนไทยในอนาคตได้ ซึ่งสามารถช่วยในการวางแผนและตัดสินใจในการดำเนินกลยุทธ์ต่างๆ นำไปสู่การพัฒนานโยบายหรือการดำเนินการที่เหมาะสมเพื่อส่งเสริมการใช้งานอินเทอร์เน็ตที่ปลอดภัยและมีประสิทธิภาพ

Thank You