Suranaree University of Technology

Start

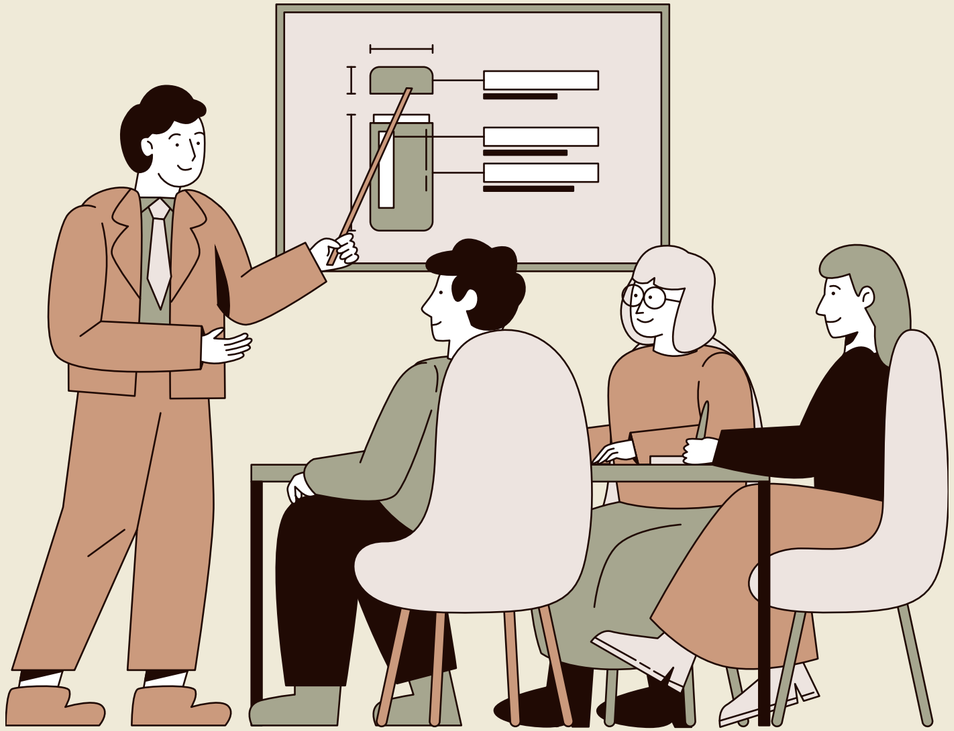
HAZARD DETECTION

การจำแนกประเภทข้อความที่พูดถึงเหตุการณ์เกี่ยวข้องกับสาธารณภัย

1101273 PROJECT IN ARTIFICIAL INTELLIGENCE FOR DATA ANALYTICS

สมาชิกในกลุ่ม

1. B6501242 นางสาวสุชารัตน์ กองฉลาด
2. B6501297 นางสาวธัญชนก กิ่งปรุ
3. B6501860 นายณัชพล ชูผล



ที่มา และความสำคัญของโครงงาน

เนื่องจากสาธารณภัยเป็นเรื่องใกล้ตัวมากขึ้นทุกวัน การแจ้งเตือนรายงานเหตุการณ์แก่ประชาชนอย่างถูกต้อง ทั่วถึง และทันต่อสถานการณ์เป็นสิ่งสำคัญ โดยเฉพาะการแจ้งเตือนเรียลไทม์จะส่งผลให้ประชาชนมีความปลอดภัยในชีวิตมากขึ้น โดยการตรวจสอบ และรายงานเหตุการณ์สาธารณภัยส่วนมากจะใช้การวิเคราะห์ข้อมูลที่มีต้นทางมาจากสื่อสังคม (social media) โดยเน้นที่ข้อมูลที่ผู้คนแบ่งปันหรือโพสต์ในสื่อสังคมต่าง ๆ อย่างเช่น แอปพลิเคชัน ‘X (เอ็กซ์)’ ที่มีการสนทนาเกี่ยวกับเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยเป็นจำนวนมาก แสดงให้เห็นผ่านการขึ้นเทรนด์ ทำให้ผู้คนให้ความสนใจกับเหตุการณ์นั้น ๆ มากขึ้น ซึ่งเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยต่าง ๆ ยกตัวอย่างเช่น น้ำท่วม, ไฟไหม้, การเสื่อมสภาพของถนน, ไฟป่า, การประท้วง, การระบาดของโรค เป็นต้น

ขอบเขตของโครงงาน

การตรวจสอบข้อความหรือโพสต์บน X (เอ็กซ์) เก็บตัวอย่างข้อมูลในรูปแบบประโยคจำนวน 4,000 ประโยค เพื่อนำมาวิเคราะห์ว่ามีข้อความถึงเหตุการณ์สาธารณภัยหรือไม่
โดยจะแยกข้อความออกเป็น 2 ประเภท ได้แก่

1 หมายถึง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

0 หมายถึง ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

และจะแยกข้อความในแต่ละประเภทออกเป็นหมวดหมู่ ทั้งหมด 6 หมวดหมู่ ได้แก่
หมวดหมู่ทั่วไป, หมวดหมู่น้ำท่วม, หมวดหมู่ระเบิด, หมวดหมู่สงคราม หมวดหมู่แผ่นดินไหวและหมวดหมู่ไฟไหม้

ประโยชน์ที่คาดว่าจะได้รับ

1. ช่วยคัดแยกข้อความที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

2. สามารถค้นหา ป้องกันอันตรายจากเหตุการณ์สาธารณภัยที่เกิดขึ้นได้รวดเร็วมากยิ่งขึ้น

3. ช่วยลดความเสี่ยง และจำนวนผู้ที่ได้รับอันตรายจากเหตุการณ์สาธารณภัย

4. สามารถนำข้อมูลที่ได้ ไปวางแผนในการพัฒนา Emergency Alert ระบบเตือนภัย ยามเกิดเหตุฉุกเฉิน

ความคิดสร้างสรรค์

1. การนำเอาเทคโนโลยี และการวิเคราะห์ข้อมูลทางด้านปัญญาประดิษฐ์มาใช้ในการเสริมสร้างความปลอดภัย และการจัดการเหตุการณ์สาธารณภัย

2. การนำเสนอแนวทางการป้องกัน และการจัดการเหตุฉุกเฉิน

ผลกระทบที่โครงงานมีต่อประชากรหรือชุมชน

01
มีประโยชน์ในการปรับปรุงคุณภาพชีวิตของประชาชน และสร้างความปลอดภัยของประชาชน โดยลดความเสี่ยงในการได้รับอันตรายจากเหตุการณ์นั้น ๆ

02
สร้างความเชื่อมั่น และความพึงพอใจของประชาชนในการบริการหรือการจัดการสถานการณ์ต่าง ๆ จากหน่วยงานที่เกี่ยวข้อง

03
การแจ้งเตือน และการตอบสนองทันใจ สามารถช่วยลดความเสี่ยง และเพิ่มโอกาสในการรับมือ ทำให้การตอบสนองเป็นไปได้อย่างทันที

04
เพิ่มความรู้ และการตระหนักรู้เกี่ยวกับวิธีการป้องกัน และจัดการกับเหตุการณ์นั้น

05
ช่วยในการพัฒนานโยบาย, แผนการดำเนินงานเพื่อเตรียมความพร้อม และลดผลกระทบในอนาคต

รายละเอียดการเก็บรวบรวมข้อมูล

วิธีการเก็บข้อมูล

เก็บข้อมูลด้วยมือ จาก ข้อความ หรือโพสต์บน X (เอ็กซ์) ลง excel ซึ่งเก็บตัวอย่างข้อมูลในรูปแบบประโยครวมทั้งหมด **จำนวน 4,000 ข้อมูล**

1. ข้อมูลที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

ซึ่งกำหนดคีย์เวิร์ดในการค้นหาออกมาเป็น 5 คำ ดังนี้

1. น้ำท่วม 2. ไฟไหม้ 3. ระเบิด 4. สงคราม 5. แผ่นดินไหว

2. ข้อมูลที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

โดยจะแยกข้อความออกเป็น 2 ประเภท ได้แก่

1 หมายถึง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

0 หมายถึง ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

การแบ่งแยกกลุ่มเป็นหมวดหมู่ LABEL

LABEL 1

1. ข้อมูลที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย จำนวน 1,139 ข้อมูล

ตัวอย่างประโยค

"อิสราเอลประกาศอย่างเป็นทางการว่าขณะนี้เริ่มทำการส่งน้ำท่วมอุโมงค์"

LABEL 0

2. ข้อมูลที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย จำนวน 2,861 ข้อมูล

ตัวอย่างประโยค

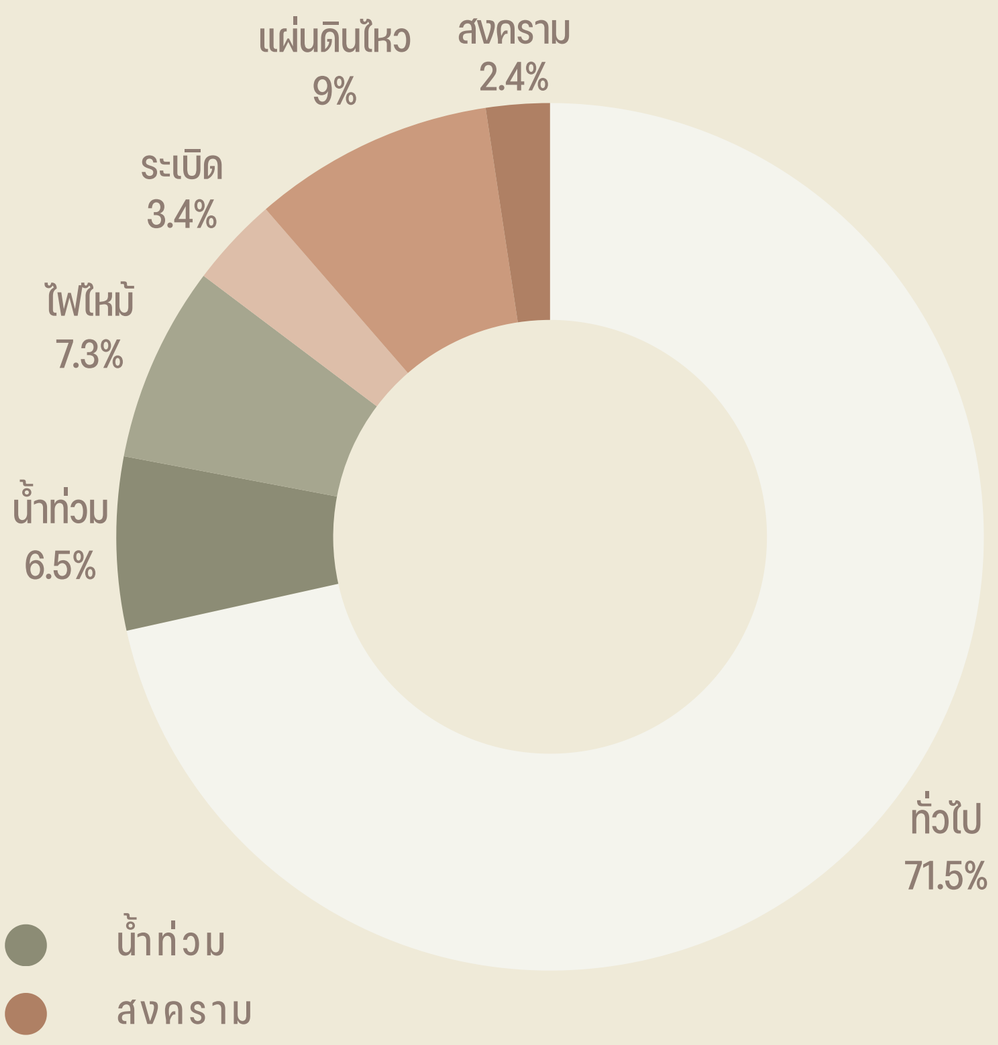
"ทุกคนอวดหมาเด็กแมวเด็กหน่อยค่ะ มันจำเป็นกับชีวิตเรามาก"

***สำหรับข้อมูลใดที่มีข้อความคีย์เวิร์ดในประโยค แต่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย จะถูกจัดอยู่ใน LABEL 0 และอยู่ในหมวดหมู่ทั่วไป เช่น "น้องแบมใส่ชุดหลุยส์มาด้วย หล่อระเบิดมาก"**

การแบ่งแยกกลุ่มเป็นหมวดหมู่ CATEGORY

สามารถจัดกลุ่มออกได้เป็น 6 หมวดหมู่ ได้แก่

1. หมวดหมู่ ทั่วไป จำนวน 2,861 ข้อมูล
2. หมวดหมู่ น้ำท่วม จำนวน 258 ข้อมูล
3. หมวดหมู่ ไฟไหม้ จำนวน 292 ข้อมูล
4. หมวดหมู่ ระเบิด จำนวน 134 ข้อมูล
5. หมวดหมู่ แผ่นดินไหว จำนวน 360 ข้อมูล
6. หมวดหมู่ สงคราม จำนวน 95 ข้อมูล



การจัดการข้อมูล

1. การนำเข้าข้อมูล

```
#import CSV file
import pandas as pd
df = pd.read_csv("/content/drive/MyDrive/273/aidata.csv")
df
```

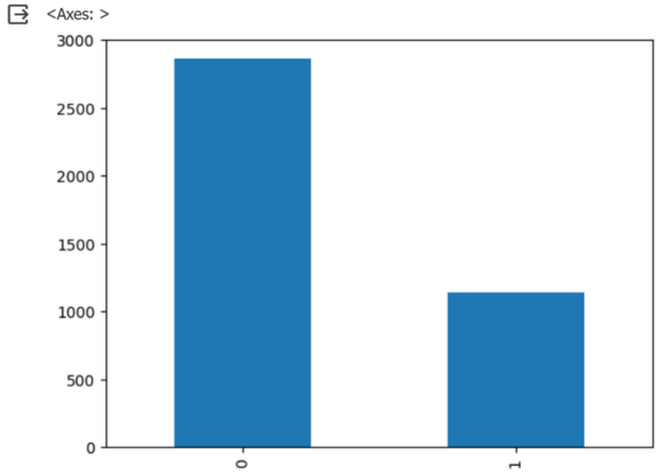
2. การสำรวจข้อมูล

```
#การสำรวจข้อมูล
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   keyword   2000 non-null   object
 1   label     4000 non-null   int64
 2   category  4000 non-null   object
 3   text      4000 non-null   object
dtypes: int64(1), object(3)
memory usage: 125.1+ KB
```

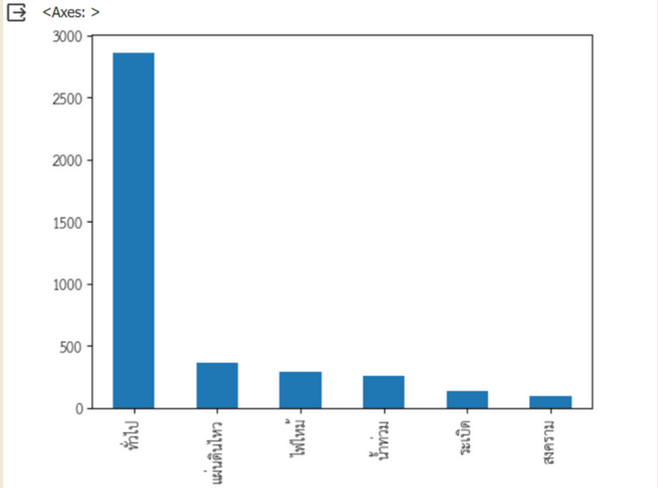
3. การแสดงข้อมูล

```
df.label.value_counts().plot(kind='bar')
```



กราฟแสดงจำนวนของข้อมูล
ในคอลัมน์ LABEL

```
df.category.value_counts().plot(kind='bar')
```



กราฟแสดงจำนวนของข้อมูล
ในคอลัมน์ CATEGORY

4. การทำความสะอาดข้อมูล

```
[ ]  # Data Preprocessing
     import numpy as np
     import re

     def cleantext(text):
         if type(text) == np.float64:
             return " "
         content = re.sub("@[\w]+", " ", text)  # ลบการกล่าวถึง
         content = re.sub(r'@\S+|https?://\S+', ' ', content)  # ลบการกล่าวถึงและ URL
         content = re.sub(r'#', ' ', content)  # ลบแฮชแท็ก
         content = re.sub(r'-', ' ', content)  # ลบ -
         content = re.sub(r'\*', ' ', content)  # ลบ *
         content = re.sub(r'([ก-ฮ])\{2,10}',r'\1', content) #ลบจำนวนตัวอักษรซ้ำ เช่น มากกกกกกก -> มาก
         content = "".join(c for c in content if c not in ('?', '-', '...', ';', ':', """, '!', 'ฯ', 'ๆ', '|', '/', ')', '(', ',' ,"", """))
         return content
```

5. การทำ WORD TOKENISE

```
[ ]  def my_tokenise(txt) :
         txt = word_tokenize(txt, keep_whitespace=False)
         return '  '.join(txt)

[ ]  df['token'] = df.cleantext.apply(my_tokenise)
```

6. การจัดการ THAI STOPWORD

```
[ ]  from wordcloud import WordCloud, STOPWORDS
     from pythainlp.corpus.common import thai_stopwords
     import matplotlib.pyplot as plt
     regexp = r"[\u0E00-\u0E7Fa-zA-Z']+" #[ก-๙a-zA-Z]
     thai_stopwords = list(thai_stopwords())
     thai_stopwords
```

7. การทำ WORD CLOUD

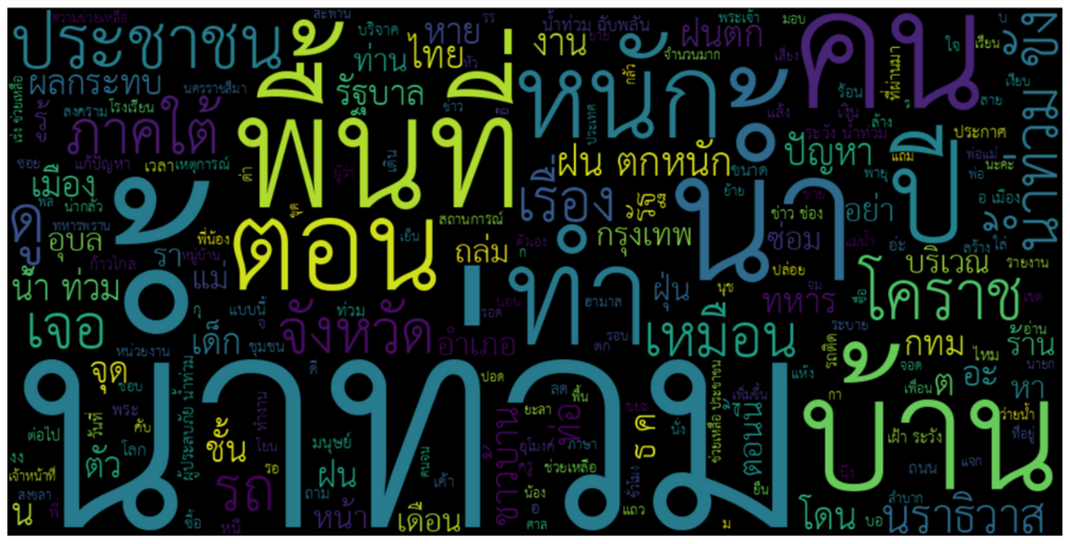
```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import copy
regexp = "[\u0E00-\u0E7F]+" #แสดงแค่ภาษาไทย

wordcloud = WordCloud(
        background_color = 'black',
        width = 1000,
        height = 500,
        regexp=regexp,
        max_words=200,
        font_path=font_path,
        stopwords =  thai_stopwords,).generate(str(' '.join(df.token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 16))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

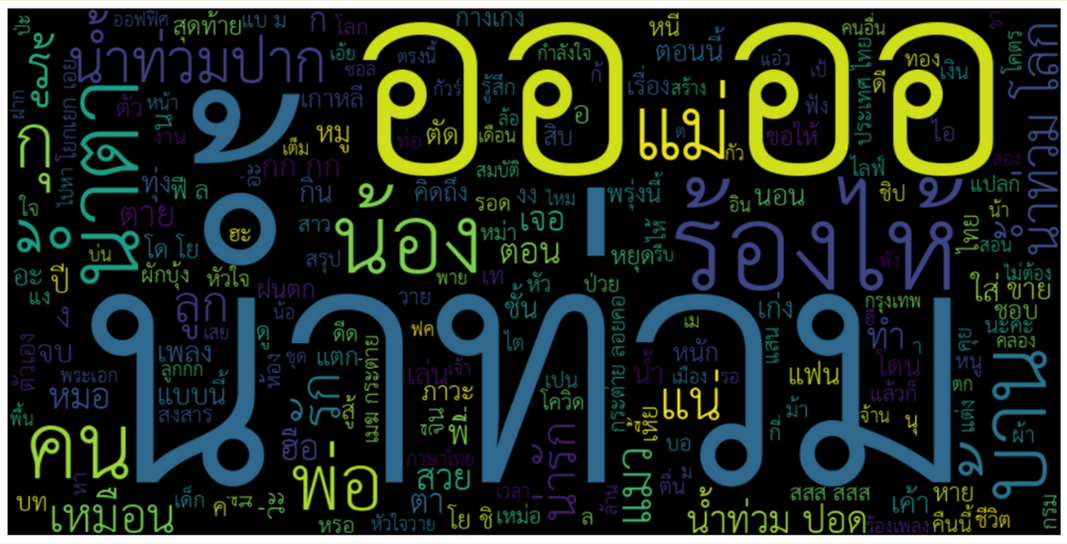
ตัวอย่างของ WORD CLOUD คีย์เวิร์ด น้ำท่วม

LABEL 1



WORD CLOUD ของคีย์เวิร์ด น้ำท่วม ที่ LABEL = 1

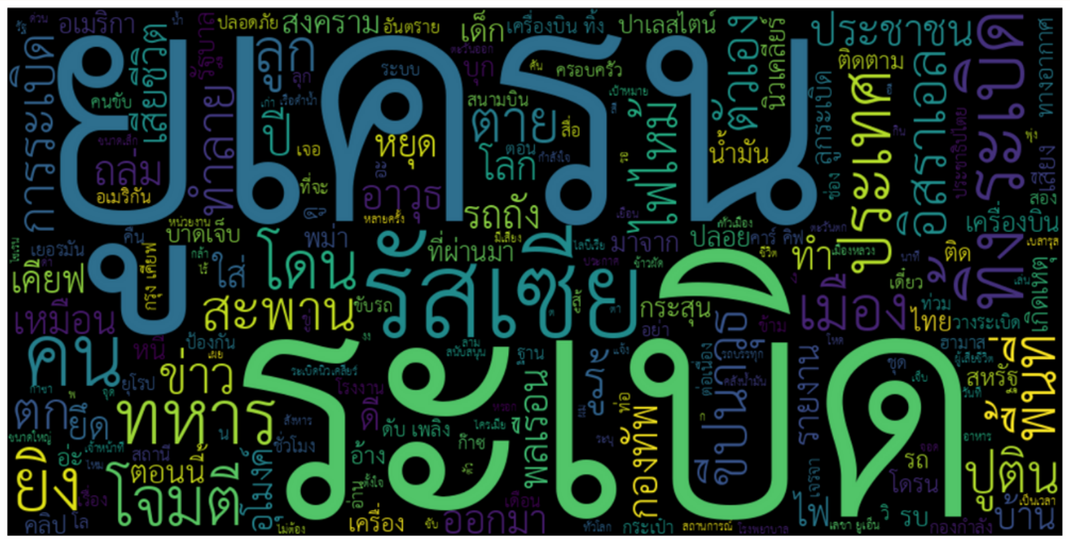
LABEL 0



WORD CLOUD ของคีย์เวิร์ด น้ำท่วม ที่ LABEL = 0

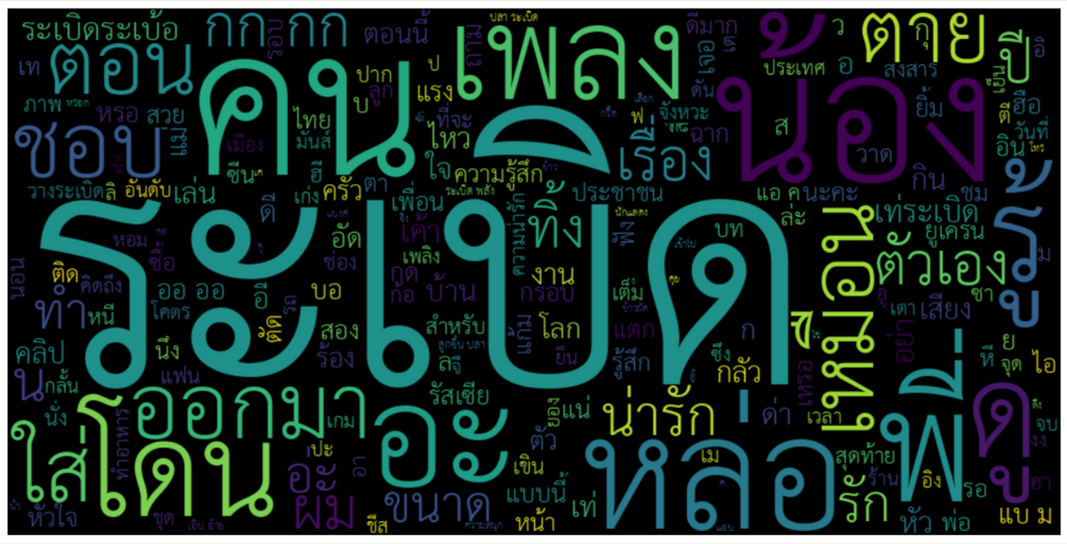
ตัวอย่างของ WORD CLOUD
คีย์เวิร์ด ระเบิด

LABEL 1



WORD CLOUD ของคีย์เวิร์ด ระเบิด ที่ LABEL = 1

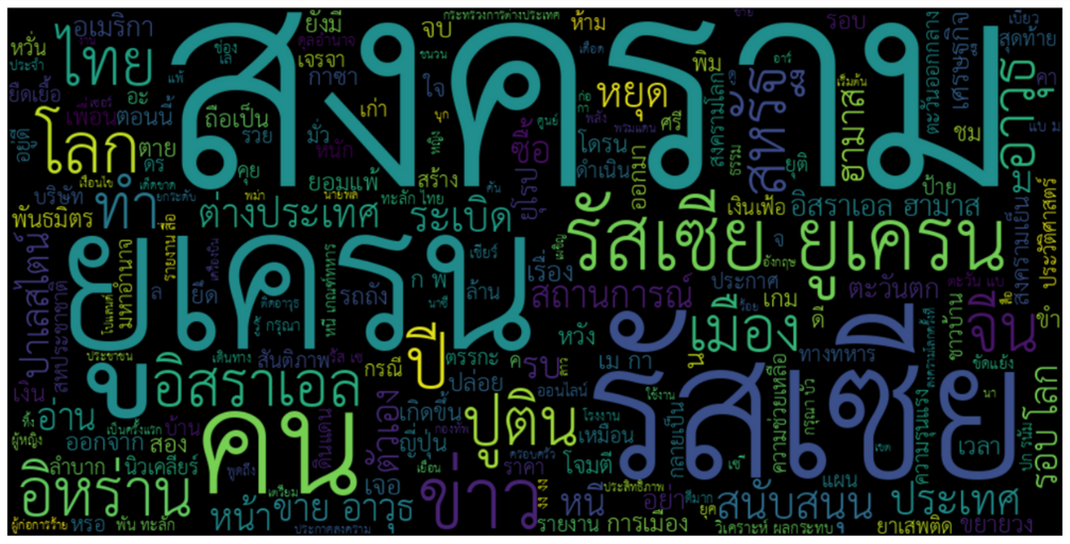
LABEL 0



WORD CLOUD ของคีย์เวิร์ด ระเบิด ที่ LABEL = 0

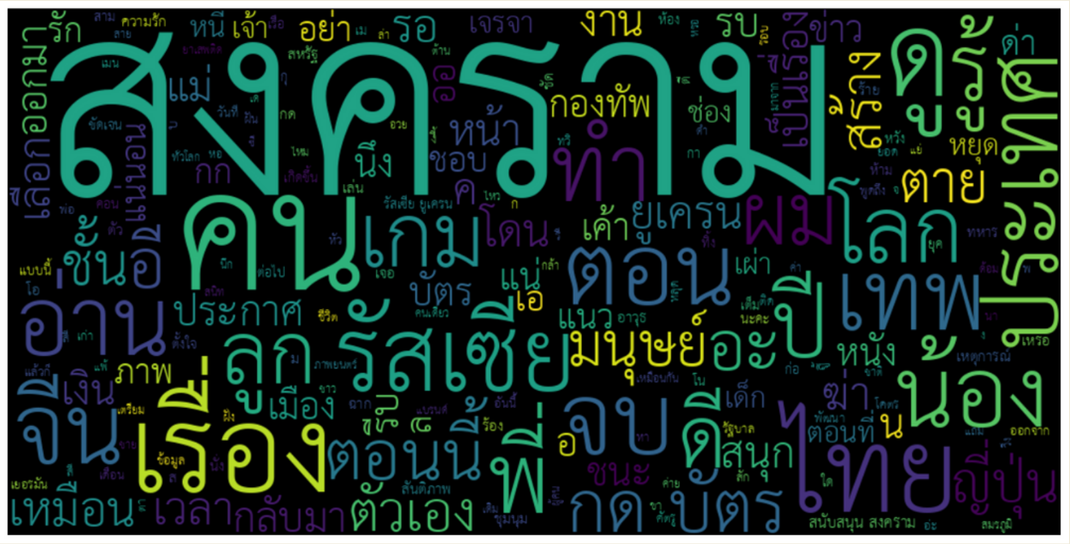
ตัวอย่างของ WORD CLOUD
คีย์เวิร์ด สงคราม

LABEL 1



WORD CLOUD ของคีย์เวิร์ด สงคราม ที่ LABEL = 1

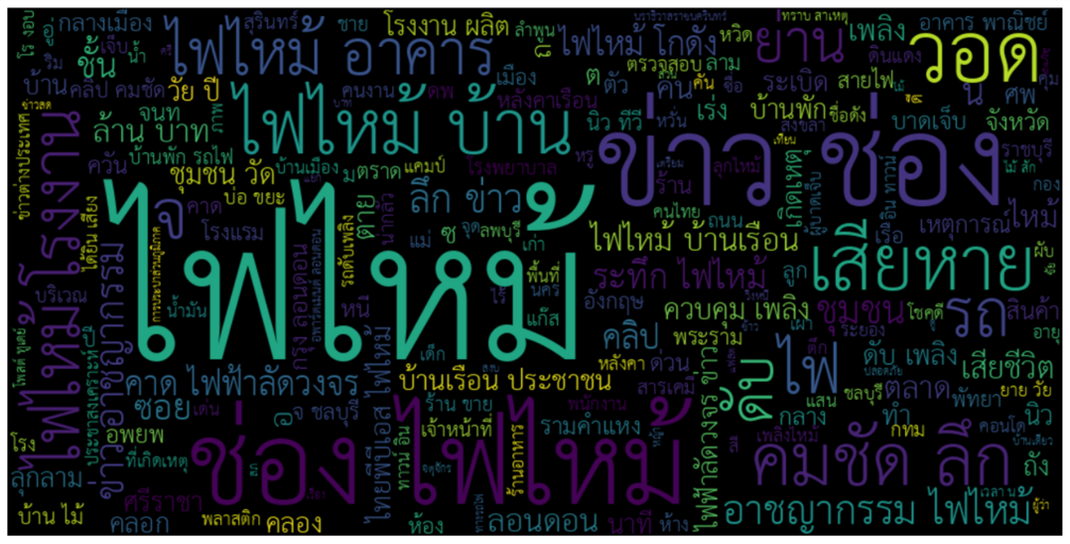
LABEL 0



WORD CLOUD ของคีย์เวิร์ดสงคราม ที่ LABEL = 0

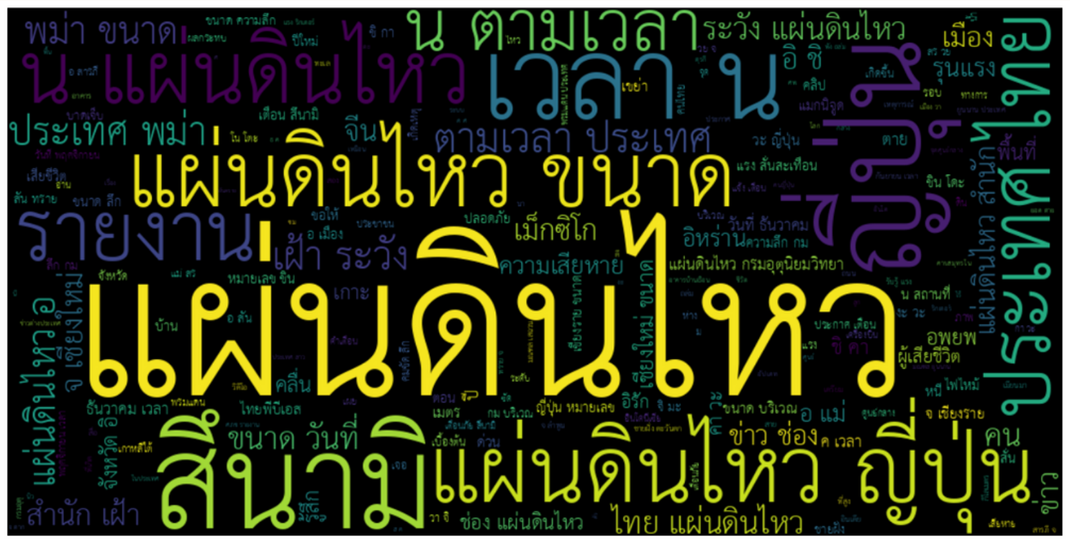
ตัวอย่างของ WORD CLOUD
คีย์เวิร์ด ไฟไหม้

LABEL 1


ไฟไหม้
ข่าว ช่อง
ไฟไหม้ บ้าน
ไฟไหม้ อาคาร
ไฟไหม้ โรงงาน
เสียหาย
วอด
ยาน
รถ
ดับ ไฟ
คมชัด ลึก
อาชญากรรม ไฟไหม้
ไฟไหม้ บ้านเรือน
ไฟไหม้ ช่อง
ไฟไหม้ ไฟฟ้าลัดวงจร
ไฟไหม้ โกดัง
ไฟไหม้ ชุมชน
ไฟไหม้ ไฟฟ้าลัดวงจร
ไฟไหม้ ไฟฟ้าลัดวงจร
ไฟไหม้ ไฟฟ้าลัดวงจร

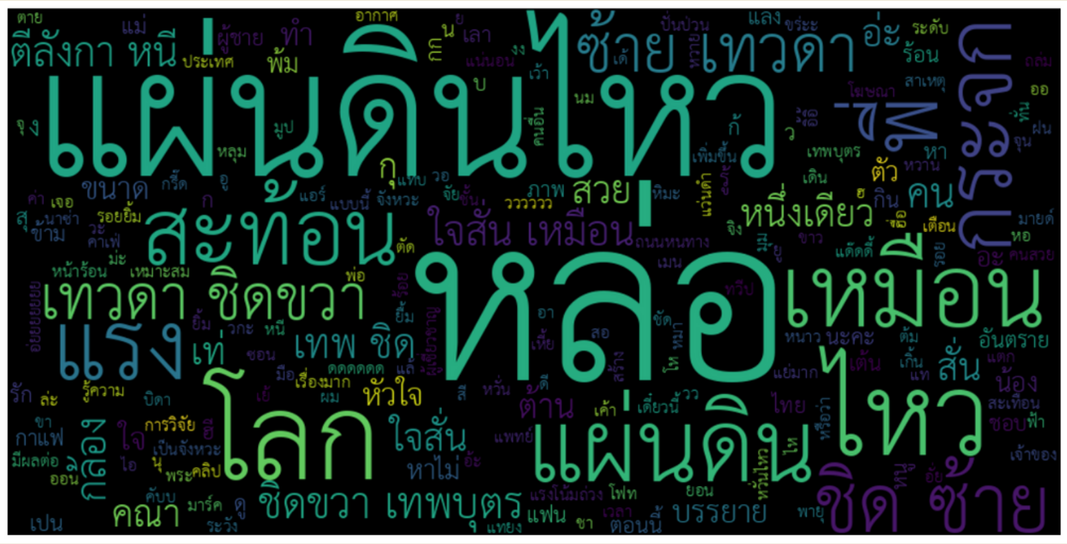
ตัวอย่างของ WORD CLOUD
คีย์เวิร์ด แผ่นดินไหว

LABEL 1



WORD CLOUD ของคีย์เวิร์ด แผ่นดินไหว ที่ LABEL = 1

LABEL 0



WORD CLOUD ของคีย์เวิร์ด แผ่นดินไหว ที่ LABEL = 0

การแบ่ง TRAIN SET และ TEST SET

โดยแบ่ง TRAIN SET 70% และ TEST SET 30%

```
data = df.cleantext.values
y = df.label.values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, y, test_size=0.3,random_state=1)


print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

สำหรับ DETECT LABEL = 0/1

```
data1 = df.cleantext.values
y1 = df.category.values

from sklearn.model_selection import train_test_split
x1_train, x1_test, y1_train, y1_test = train_test_split(data1, y1, test_size=0.3,random_state=1)


print(x1_train)
print(x1_test)
print(y1_train)
print(y1_test)
```

สำหรับ DETECT CATEGORY

การทำ FEATURE EXTRACTION

สำหรับ DETECT LABEL = 0/1

COUNT VECTORIZER

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
vectorizer.fit_transform(data)
x_train_count = vectorizer.transform(x_train)
x_test_count = vectorizer.transform(x_test)

print(x_train_count.shape)
print(x_test_count.shape)
print(x_train_count.toarray())
print(x_test_count.toarray())
print(vectorizer.get_feature_names_out())
```

TFIDF VECTORIZER

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(smooth_idf=False)
tfidf.fit_transform(data)
x_train_tfidf = vectorizer.transform(x_train)
x_test_tfidf = vectorizer.transform(x_test)

print(x_train_tfidf.shape)
print(x_test_tfidf.shape)
print(x_train_tfidf.toarray())
print(x_test_tfidf.toarray())
print(vectorizer.get_feature_names_out())
```

การทำ FEATURE EXTRACTION

สำหรับ DETECT CATEGORY

COUNT VECTORIZER

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
vectorizer.fit_transform(data1)
x1_train_count = vectorizer.transform(x1_train)
x1_test_count = vectorizer.transform(x1_test)

print(x1_train_count.shape)
print(x1_test_count.shape)
print(x1_train_count.toarray())
print(x1_test_count.toarray())
print(vectorizer.get_feature_names_out())
```

TFIDF VECTORIZER

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(smooth_idf=False)
tfidf.fit_transform(data1)
x1_train_tfidf = vectorizer.transform(x1_train)
x1_test_tfidf = vectorizer.transform(x1_test)

print(x1_train_tfidf.shape)
print(x1_test_tfidf.shape)
print(x1_train_tfidf.toarray())
print(x1_test_tfidf.toarray())
print(vectorizer.get_feature_names_out())
```

โมเดลที่ใช้

สำหรับ DETECT LABEL และ DETECT CATEGORY

- LOGISTIC REGRESSION
- DECISION TREE
- RANDOM FOREST
- K-NEAREST NEIGHBORS (KNN)
- NEURAL NETWORK
- CONVOLUTIONAL NEURAL NETWORK
- RECURRENT NEURAL NETWORK
- MODEL WITH DROPOUT

COUNT VECTORIZER

สำหรับ DETECT LABEL = 0/1

1. LOGISTIC REGRESSION

```
from sklearn.metrics import classification_report
y_true = y_test
y_pred =logreg.predict(x_test_count)
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       870
           1       0.92      0.77      0.84       330

    accuracy                           0.92      1200
   macro avg       0.92      0.87      0.89      1200
weighted avg       0.92      0.92      0.92      1200
```

2. DECISION TREE

```
from sklearn.metrics import classification_report
y_true = y_test
y_pred =tree.predict(x_test_count)
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       870
           1       0.85      0.86      0.85       330

    accuracy                           0.92      1200
   macro avg       0.90      0.90      0.90      1200
weighted avg       0.92      0.92      0.92      1200
```

3. RANDOM FOREST

```
from sklearn.metrics import classification_report
y_true = y_test
y_pred =rf.predict(x_test_count)
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.93       870
           1       0.99      0.64      0.77       330

    accuracy                           0.90      1200
   macro avg       0.93      0.82      0.85      1200
weighted avg       0.91      0.90      0.89      1200
```

4. K-NEAREST NEIGHBORS (KNN)

```
from sklearn.metrics import classification_report
y_true = y_test
y_pred =knn.predict(x_test_count)
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      1.00      0.90       870
           1       0.99      0.42      0.59       330

    accuracy                           0.84      1200
   macro avg       0.91      0.71      0.74      1200
weighted avg       0.87      0.84      0.81      1200
```

TFIDF VECTORIZER

สำหรับ DETECT LABEL = 0/1

1. LOGISTIC REGRESSION

```
from sklearn.metrics import classification_report
y_true = y_test
y_pred =logreg.predict(x_test_tfidf)
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       870
           1       0.92      0.77      0.84       330

    accuracy                           0.92      1200
   macro avg       0.92      0.87      0.89      1200
weighted avg       0.92      0.92      0.92      1200
```

2. DECISION TREE

```
from sklearn.metrics import classification_report
y_true = y_test
y_pred =tree.predict(x_test_tfidf)
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       870
           1       0.85      0.86      0.85       330

    accuracy                           0.92      1200
   macro avg       0.90      0.90      0.90      1200
weighted avg       0.92      0.92      0.92      1200
```

3. RANDOM FOREST

```
] from sklearn.metrics import classification_report
  y_true = y_test
  y_pred =rf.predict(x_test_tfidf)
  target_names = ['class 0', 'class 1', 'class 2']
  print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       870
           1       1.00      0.65      0.78       330

    accuracy                           0.90      1200
   macro avg       0.94      0.82      0.86      1200
weighted avg       0.91      0.90      0.89      1200
```

4. K-NEAREST NEIGHBORS (KNN)

```
| from sklearn.metrics import classification_report
  y_true = y_test
  y_pred =knn.predict(x_test_tfidf)
  target_names = ['class 0', 'class 1', 'class 2']
  print(classification_report(y_true,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      1.00      0.90       870
           1       0.99      0.42      0.59       330

    accuracy                           0.84      1200
   macro avg       0.91      0.71      0.74      1200
weighted avg       0.87      0.84      0.81      1200
```

COUNT VECTORIZER

สำหรับ DETECT CATEGORY

1. LOGISTIC REGRESSION

```
from sklearn.metrics import classification_report
y1_true = y1_test
y1_pred =logreg1.predict(x1_test_count)
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.91	0.99	0.95	870
น้ำท่วม	0.85	0.68	0.76	78
ระเบิด	0.82	0.21	0.34	42
สงคราม	0.67	0.12	0.20	34
แผ่นดินไหว	1.00	1.00	1.00	97
ไฟไหม้	0.97	0.97	0.97	79
accuracy			0.92	1200
macro avg	0.87	0.66	0.70	1200
weighted avg	0.91	0.92	0.90	1200

2. DECISION TREE

```
5] from sklearn.metrics import classification_report
   y1_true = y1_test
   y1_pred =tree1.predict(x1_test_count)
   target1_names = ['class 0', 'class 1', 'class 2']
   print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.95	0.93	0.94	870
น้ำท่วม	0.64	0.99	0.78	78
ระเบิด	0.00	0.00	0.00	42
สงคราม	0.35	0.35	0.35	34
แผ่นดินไหว	0.97	1.00	0.98	97
ไฟไหม้	0.88	1.00	0.93	79
accuracy			0.90	1200
macro avg	0.63	0.71	0.66	1200
weighted avg	0.87	0.90	0.88	1200

3. RANDOM FOREST

```
from sklearn.metrics import classification_report
y1_true = y1_test
y1_pred =rf1.predict(x1_test_count)
target1_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.86	1.00	0.93	870
น้ำท่วม	1.00	0.22	0.36	78
ระเบิด	1.00	0.05	0.09	42
สงคราม	1.00	0.06	0.11	34
แผ่นดินไหว	0.99	0.98	0.98	97
ไฟไหม้	0.99	0.95	0.97	79
accuracy			0.88	1200
macro avg	0.97	0.54	0.57	1200
weighted avg	0.90	0.88	0.84	1200

4. K-NEAREST NEIGHBORS (KNN)

```
7] from sklearn.metrics import classification_report
   y1_true = y1_test
   y1_pred =knn1.predict(x1_test_count)
   target1_names = ['class 0', 'class 1', 'class 2']
   print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.82	1.00	0.90	870
น้ำท่วม	0.90	0.12	0.20	78
ระเบิด	0.00	0.00	0.00	42
สงคราม	1.00	0.03	0.06	34
แผ่นดินไหว	1.00	0.70	0.82	97
ไฟไหม้	1.00	0.76	0.86	79
accuracy			0.84	1200
macro avg	0.79	0.43	0.47	1200
weighted avg	0.83	0.84	0.79	1200

TFIDF VECTORIZER

สำหรับ DETECT CATEGORY

1. LOGISTIC REGRESSION

```
from sklearn.metrics import classification_report
y1_true = y1_test
y1_pred =logreg1.predict(x1_test_tfidf)
target1_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.91	0.99	0.95	870
น้ำท่วม	0.85	0.68	0.76	78
ระเบิด	0.82	0.21	0.34	42
สงคราม	0.67	0.12	0.20	34
แผ่นดินไหว	1.00	1.00	1.00	97
ไฟไหม้	0.97	0.97	0.97	79
accuracy			0.92	1200
macro avg	0.87	0.66	0.70	1200
weighted avg	0.91	0.92	0.90	1200

2. DECISION TREE

```
] from sklearn.metrics import classification_report
y1_true = y1_test
y1_pred =tree1.predict(x1_test_tfidf)
target1_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.95	0.93	0.94	870
น้ำท่วม	0.64	0.99	0.78	78
ระเบิด	0.00	0.00	0.00	42
สงคราม	0.35	0.35	0.35	34
แผ่นดินไหว	0.97	1.00	0.98	97
ไฟไหม้	0.88	1.00	0.93	79
accuracy			0.90	1200
macro avg	0.63	0.71	0.66	1200
weighted avg	0.87	0.90	0.88	1200

3. RANDOM FOREST

```
] from sklearn.metrics import classification_report
y1_true = y1_test
y1_pred =rf1.predict(x1_test_tfidf)
target1_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.87	1.00	0.93	870
น้ำท่วม	1.00	0.33	0.50	78
ระเบิด	1.00	0.05	0.09	42
สงคราม	1.00	0.06	0.11	34
แผ่นดินไหว	0.99	0.99	0.99	97
ไฟไหม้	1.00	0.96	0.98	79
accuracy			0.89	1200
macro avg	0.98	0.57	0.60	1200
weighted avg	0.91	0.89	0.86	1200

4. K-NEAREST NEIGHBORS (KNN)

```
] from sklearn.metrics import classification_report
y1_true = y1_test
y1_pred =knn1.predict(x1_test_tfidf)
target1_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y1_true,y1_pred))
```

	precision	recall	f1-score	support
ทั่วไป	0.82	1.00	0.90	870
น้ำท่วม	0.90	0.12	0.20	78
ระเบิด	0.00	0.00	0.00	42
สงคราม	1.00	0.03	0.06	34
แผ่นดินไหว	1.00	0.70	0.82	97
ไฟไหม้	1.00	0.76	0.86	79
accuracy			0.84	1200
macro avg	0.79	0.43	0.47	1200
weighted avg	0.83	0.84	0.79	1200

DEEP LEARNING

สำหรับ DETECT LABEL = 0/1

1. NEURAL NETWORK

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test, y_pred))
```

```
38/38 [==============================]
              precision    recall  f1-score   support

           0       0.87      0.97      0.92       862
           1       0.90      0.64      0.75       338

    accuracy                           0.88      1200
   macro avg       0.89      0.80      0.83      1200
weighted avg       0.88      0.88      0.87      1200
```

2. CONVOLUTIONAL NEURAL NETWORK

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test, y_pred))
```

```
38/38 [==============================]
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       862
           1       0.96      0.62      0.76       338

    accuracy                           0.89      1200
   macro avg       0.91      0.81      0.84      1200
weighted avg       0.90      0.89      0.88      1200
```

3. RECURRENT NEURAL NETWORK

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test, y_pred))
```

```
38/38 [==============================]
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       862
           1       0.92      0.65      0.77       338

    accuracy                           0.89      1200
   macro avg       0.90      0.82      0.85      1200
weighted avg       0.89      0.89      0.88      1200
```

4. MODEL WITH DROPOUT

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test, y_pred))
```

```
38/38 [==============================]
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       862
           1       0.94      0.64      0.76       338

    accuracy                           0.89      1200
   macro avg       0.91      0.81      0.84      1200
weighted avg       0.89      0.89      0.88      1200
```

DEEP LEARNING

สำหรับ DETECT CATEGORY

1. NEURAL NETWORK

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test_encoded, y_pred))
```

```
38/38 [==============================]
              precision    recall  f1-score   support

           0       0.72      1.00      0.84       862
           1       0.00      0.00      0.00        81
           2       0.00      0.00      0.00        38
           3       0.00      0.00      0.00        26
           4       0.00      0.00      0.00       106
           5       0.00      0.00      0.00        87
           6       0.00      0.00      0.00         0

    accuracy                           0.72      1200
   macro avg       0.10      0.14      0.12      1200
weighted avg       0.52      0.72      0.60      1200
```

2. CONVOLUTIONAL NEURAL NETWORK

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test_encoded, y_pred))
```

```
38/38 [==============================]
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       862
           1       0.39      0.19      0.25        81
           2       0.19      0.11      0.14        38
           3       0.00      0.00      0.00        26
           4       1.00      0.86      0.92       106
           5       0.96      0.92      0.94        87

    accuracy                           0.86      1200
   macro avg       0.57      0.51      0.53      1200
weighted avg       0.82      0.86      0.84      1200
```

3. RECURRENT NEURAL NETWORK

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test_encoded, y_pred))
```

```
38/38 [==============================]
              precision    recall  f1-score   support

           0       0.87      0.98      0.92       862
           1       0.50      0.16      0.24        81
           2       0.25      0.13      0.17        38
           3       0.07      0.04      0.05        26
           4       1.00      0.85      0.92       106
           5       0.98      0.92      0.95        87

    accuracy                           0.86      1200
   macro avg       0.61      0.51      0.54      1200
weighted avg       0.83      0.86      0.83      1200
```

4. MODEL WITH DROPOUT

```
y_pred_prob = model.predict(X_test)
y_pred = tf.argmax(y_pred_prob, axis=1).numpy()
print(classification_report(y_test_encoded, y_pred))
```

```
38/38 [==============================
              precision    recall  f1-score   support

           0       0.87      0.97      0.92       862
           1       0.36      0.16      0.22        81
           2       0.31      0.13      0.19        38
           3       0.00      0.00      0.00        26
           4       1.00      0.90      0.95       106
           5       0.95      0.93      0.94        87

    accuracy                           0.86      1200
   macro avg       0.58      0.52      0.54      1200
weighted avg       0.82      0.86      0.83      1200
```

ผลการทดลองค่าประสิทธิภาพ Model สำหรับการ Detect label

การสกัด	โมเดล	ค่าประสิทธิภาพโมเดล			
		Accuracy	Precision	Recall	F-measure
CountVectorizer	Logistic Regression	0.918	0.918	0.918	0.916
	Random Forest	0.894	0.905	0.894	0.886
	Decision Tree	0.919	0.919	0.919	0.919
	K-nearest Neighbors	0.839	0.867	0.839	0.814
TfidfVectorizer	Logistic Regression	0.918	0.918	0.918	0.916
	Random Forest	0.891	0.902	0.891	0.882
	Decision Tree	0.919	0.919	0.919	0.919
	K-nearest Neighbors	0.839	0.867	0.839	0.814
	Neural Network Model	0.878	0.881	0.878	0.871
	Convolutional Neural Network Model	0.887	0.895	0.887	0.878
	Recurrent Neural Network Model	0.887	0.891	0.887	0.881
	Model with Dropout	0.887	0.892	0.887	0.879

ผลการทดลองค่าประสิทธิภาพ Model สำหรับการ Detect category

การสกัด	โมเดล	ค่าประสิทธิภาพโมเดล			
		Accuracy	Precision	Recall	F-measure
CountVectorizer	Logistic Regression	0.915	0.906	0.915	0.897
	Random Forest	0.885	0.884	0.885	0.846
	Decision Tree	0.896	0.874	0.896	0.882
	K-nearest Neighbors	0.839	0.827	0.839	0.791
TfidfVectorizer	Logistic Regression	0.915	0.906	0.915	0.897
	Random Forest	0.885	0.884	0.885	0.845
	Decision Tree	0.896	0.874	0.896	0.882
	K-nearest Neighbors	0.839	0.827	0.839	0.791
	Neural Network Model	0.718	0.517	0.718	0.601
	Convolutional Neural Network Model	0.863	0.821	0.863	0.836
	Recurrent Neural Network Model	0.860	0.828	0.860	0.835
	Model with Dropout	0.861	0.819	0.861	0.834

สรุปผลค่าประสิทธิภาพ Model

DETECT LABEL

โมเดล Logistic regression และ Decision tree มีค่าความถูกต้องสูงที่สุด 0.918, โมเดล Random Forest มีค่าความถูกต้องเป็นอันดับรองลงมา 0.894 และ K Nearest Neighbors มีค่าความถูกต้องต่ำที่สุด 0.839

DETECT CATEGORY

โมเดล Logistic regression มีค่าความถูกต้องสูงที่สุด 0.915, โมเดล Decision tree มีค่าความถูกต้องเป็นอันดับรองลงมา 0.896 และ Neural Network Model มีค่าความถูกต้องต่ำที่สุด 0.718

การทดสอบโมเดล

DETECT LABEL

```python
inp = input()
cleaned = cleantext(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned=vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)
result = logreg.predict(cleaned)

print(end="\n")
print('ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)')
print('ผลลัพธ์ = ',result)

#สร้างเงื่อนไข
if result[0] == 0:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย')
else:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย')
```

```
ฮื่อออ หล่อมากจิงคับบ แบบแผ่นดินไหวแต่เราม่ะหวายย
ฮื่อออ หล่อมากจิงคับบ แบบแผ่นดินไหวแต่เราม่ะหวายย
ฮื่อ ออ  หล่อ มาก จิง คับบ  แบบ แผ่นดินไหว แต่ เรา ม่ะ หวาย ย
[[0 0 0 ... 0 0 0]]

ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)
ผลลัพธ์ =  [0]
คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย
```

ผลลัพธ์ = [0]
คำอธิบาย : ข้อความนี้ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

```python
inp = input()
cleaned = cleantext(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned=vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)
result = logreg.predict(cleaned)

print(end="\n")
print('ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)')
print('ผลลัพธ์ = ',result)

#สร้างเงื่อนไข
if result[0] == 0:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย')
else:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย')
```

```
คนญี่ปุ่นไลฟ์สตรีมอยู่แล้วมีแจ้งเตือนแผ่นดินไหวเข้าอย่างเร็ว #แผ่นดินไหว #ญี่ปุ่น #สึนามิ
คนญี่ปุ่นไลฟ์สตรีมอยู่แล้วมีแจ้งเตือนแผ่นดินไหวเข้าอย่างเร็ว  แผ่นดินไหว  ญี่ปุ่น  สึนามิ
คนญี่ปุ่น  ไลฟ์  สตรีม  อยู่แล้ว  มี  แจ้ง  เตือน  แผ่นดินไหว  เข้า  อย่าง  เร็ว  แผ่นดินไหว  ญี่ปุ่น  สึนามิ
[[0 0 0 ... 0 0 0]]

ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)
ผลลัพธ์ =  [1]
คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย
```

ผลลัพธ์ = [1]
คำอธิบาย : ข้อความนี้เกี่ยวข้องกับเหตุการณ์สาธารณภัย

การทดสอบโมเดล

DETECT CATEGORY

```
inp = input()
cleaned = cleantext(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned=vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)

result1 = logreg1.predict(cleaned)

print(end="\n")
print('ข้อความนี้ถูกจัดให้อยู่ในหมวดหมู่อะไร ?')
print('ผลลัพธ์ = ',result1)
print('คำอธิบาย : ข้อความนี้จัดอยู่ในหมวดหมู่',result1[0])
```

```
ปั่นงานออกแบบไฟไหม้ไปเลยวันนี้ ดองงานไว้ นี่สิ่นะผลกรรมม
ปั่นงานออกแบบไฟไหม้ไปเลยวันนี้ ดองงานไว้ นี่สิ่นะผลกรรมม
ปั่น งาน ออกแบบ ไฟไหม้ ไป เลย วันนี้ ดอง งาน ไว้ นี่ สิ่ นะ ผลกรรม ม
[[0 0 0 ... 0 0 0]]

ข้อความนี้ถูกจัดให้อยู่ในหมวดหมู่อะไร ?
ผลลัพธ์ =  ['ทั่วไป']
คำอธิบาย : ข้อความนี้จัดอยู่ในหมวดหมู่ ทั่วไป
```

ผลลัพธ์ = ทั่วไป

คำอธิบาย : ข้อความนี้จัดอยู่ในหมวดหมู่ ทั่วไป

```
inp = input()
cleaned = cleantext(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned=vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)

result1 = logreg1.predict(cleaned)

print(end="\n")
print('ข้อความนี้ถูกจัดให้อยู่ในหมวดหมู่อะไร ?')
print('ผลลัพธ์ = ',result1)
print('คำอธิบาย : ข้อความนี้จัดอยู่ในหมวดหมู่',result1[0])
```

```
ระทึก!รถบัสทัวร์จีนไฟไหม้กลางเมืองภูเก็ต36ชีวิตหนีตายอลมาน
ระทึกรถบัสทัวร์จีนไฟไหม้กลางเมืองภูเก็ต36ชีวิตหนีตายอลมาน
ระทึก รถบัส ทัวร์ จีน ไฟไหม้ กลางเมือง ภูเก็ต 36 ชีวิต หนี ตา ยอ ล มาน
[[0 0 0 ... 0 0 0]]

ข้อความนี้ถูกจัดให้อยู่ในหมวดหมู่อะไร ?
ผลลัพธ์ =  ["ไฟไหม้"]
คำอธิบาย : ข้อความนี้จัดอยู่ในหมวดหมู่ ไฟไหม้
```

ผลลัพธ์ = ไฟไหม้

คำอธิบาย : ข้อความนี้จัดอยู่ในหมวดหมู่ ไฟไหม้

การทดสอบโมเดล

DETECT LABEL และ DETECT CATEGORY

```
inp = input()
cleaned = cleantext(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned=vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)
result = logreg.predict(cleaned)
result1 = logreg1.predict(cleaned)

print(end="\n")
print('ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)')
print('ผลลัพธ์ = ',result,result1)

#สร้างเงื่อนไข
if result[0] == 0:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่',result1[0])
else:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่',result1[0])
```

```
ร้องไห้น้ำท่วมจอแล้วกุ
ร้องไห้น้ำท่วมจอแล้วกุ
ร้องไห้ น้ำท่วม จอ แล้ว กุ
[[0 0 0 ... 0 0 0]]

ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)
ผลลัพธ์ =  [0] ['ทั่วไป']
คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่ ทั่วไป
```

ผลลัพธ์ = [0] ทั่วไป

คำอธิบาย : ข้อความนี้ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่ ทั่วไป

```
inp = input()
cleaned = cleantext(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned=vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)
result = logreg.predict(cleaned)
result1 = logreg1.predict(cleaned)

print(end="\n")
print('ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)')
print('ผลลัพธ์ = ',result,result1)

#สร้างเงื่อนไข
if result[0] == 0:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่',result1[0])
else:
    print('คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่',result1[0])
```

```
ฝนตกตั้งแต่เมื่อคืนจนถึงตอนนี้ฝนแรงขึ้นเรื่อยๆ ล่าสุดน้ำท่วมถนนหน้าบ้านแล้ว
ฝนตกตั้งแต่เมื่อคืนจนถึงตอนนี้ฝนแรงขึ้นเรื่อย ล่าสุดน้ำท่วมถนนหน้าบ้านแล้ว
ฝนตก ตั้งแต่ เมื่อคืน จนถึง ตอนนี้ ฝน แรง ขึ้น เรื่อย ล่าสุด น้ำท่วม ถนน หน้าบ้าน แล้ว
[[0 0 0 ... 0 0 0]]

ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)
ผลลัพธ์ =  [1] ['น้ำท่วม']
คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่ น้ำท่วม
```

ผลลัพธ์ = [1] น้ำท่วม

คำอธิบาย : ข้อความนี้เกี่ยวข้องกับเหตุการณ์สาธารณภัย และจัดอยู่ในหมวดหมู่ น้ำท่วม

THANK YOU