โครงงาน Hazard Detection การจำแนกประเภทข้อความที่พูดถึงเกี่ยวกับเหตุการณ์สาธารณภัย

จัดทำโดย

B6501242 นางสาวสุชารัตน์ กองฉลาด

B6501297 นางสาวธัญชนก กิ่งปรุ

B6501860 นายณัชพล ชูผล

เสนอ

รองศาสตราจารย์ ดร.จิติมนต์ อั่งสกุล

อาจารย์ ดร.อรรคพล วงศ์กอบลาภ

รายงานนี้เป็นส่วนหนึ่งของรายวิชา 1101273

โครงงานปัญญาประดิษฐ์สำหรับการวิเคราะห์ข้อมูล

ภาคการศึกษาที่ 2 ปีการศึกษา 2566

สำนักวิชาศาสตร์และศิลป์ดิจิทัล

มหาวิทยาลัยเทคโนโลยีสุรนารี

บทคัดย่อ

สาธารณภัยเป็นเรื่องใกล้ตัวมากขึ้นทุกวัน การแจ้งเตือนรายงานเหตุการณ์แก่ประชาชนอย่างถูกต้อง ทั่วถึง และทันต่อสถานการณ์เป็นสิ่งสำคัญ โดยเฉพาะการแจ้งเตือนเรียลไทม์จะส่งผลให้ประชาชนมีความปลอดภัยในชีวิตมากขึ้น งานวิจัยนี้จึงมุ่งเน้นการวิเคราะห์ข้อมูลจาก X (เอ็กซ์) เพื่อจำแนกประเภทข้อความว่าพูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ โดยการรวบรวมข้อความหรือโพสต์บน X (เอ็กซ์) ในรูปแบบประโยคจำนวน 4,000 ประโยค มีการทำความสะอาดข้อมูลโดยมีการแบ่งข้อความออกเป็นข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย จำนวน 1,139 ประโยค และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย จำนวน 2,861 ประโยค และแยกข้อความในแต่ละประเภทออกเป็นหมวดหมู่ ทั้งหมด 5 หมวดหมู่ ได้แก่ หมวดหมู่ทั่วไป, หมวดหมู่น้ำท่วม, หมวดหมู่ระเบิด, หมวดหมู่สงคราม และหมวดหมู่ไฟไหม้ หลังจากนั้นได้ทำการเทรนโมเดล โดยอัลกอริทึมที่นำมาใช้มีดังนี้ Logistic Regression, Random Forest, Decision Tree และ k-Nearest Neighbors, Neural Network Model, Convolutional Neural Network Model, Recurrent Neural Network Model และ Model with Dropout ผลการวิจัยพบว่า สำหรับการ Detect label โมเดล Logistic regression มีค่าความถูกต้อง 0.918 โมเดล Random Forest มีค่าความถูกต้อง 0.894 โมเดล Decision tree มีค่าความถูกต้อง 0.919 โมเดล K-Nearest Neighbors มีค่าความถูกต้อง 0.839 โมเดล Neural Network มีค่าความถูกต้อง 0.878 , โมเดล Convolutional Neural Network มีค่าความถูกต้อง 0.887 , โมเดล Recurrent Neural Network Model มีค่าความถูกต้อง 0.887 และ Model with Dropout มีค่าความถูกต้อง 0.887 สำหรับการ Detect label โมเดล Logistic regression มีค่าความถูกต้อง 0.915 โมเดล Random Forest มีค่าความถูกต้อง 0.885 โมเดล Decision tree มีค่าความถูกต้อง 0.896 โมเดล K-Nearest Neighbors มีค่าความถูกต้อง 0.839 โมเดล Neural Network มีค่าความถูกต้อง 0.718 , โมเดล Convolutional Neural Network มีค่าความถูกต้อง 0.863 , โมเดล Recurrent Neural Network Model มีค่าความถูกต้อง 0.860 และ Model with Dropout มีค่าความถูกต้อง 0.861

การแบ่งงาน

รหัสนักศึกษา	ชื่อ-นามสกุล	สัดส่วนการแบ่งงาน
B6501242	นางสาวสุชารัตน์ กองฉลาด	100%
B6501297	นางสาวธัญชนก กิ่งปรุ	100%
B6501860	นายณัชพล ชูผล	100%
สัดส่วนรวม (ต้องเท่ากับจำนวนสมาชิก*100)		300%

ที่มาและความสำคัญของโครงงาน

เนื่องจากสาธารณภัยเป็นเรื่องใกล้ตัวมากขึ้นทุกวัน การแจ้งเตือนรายงานเหตุการณ์แก่ประชาชนอย่างถูกต้อง ทั่วถึง และทันต่อสถานการณ์เป็นสิ่งสำคัญ โดยเฉพาะการแจ้งเตือนเรียลไทม์จะส่งผลให้ประชาชนมีความปลอดภัยในชีวิตมากขึ้น โดยการตรวจสอบ และรายงานเหตุการณ์สาธารณภัยเป็นระบบที่ใช้การวิเคราะห์ข้อมูลที่มีต้นทางมาจากสื่อสารสังคม (social media) หรือแหล่งข้อมูลออนไลน์อื่น ๆ โดยเน้นที่ข้อมูลที่ผู้คนแบ่งปันหรือโพสต์ในสื่อสารสังคมต่าง ๆ อย่างเช่น แอปพลิเคชัน ‘X (เอ็กซ์)’ หรือที่เคยเรียกว่า ‘Twitter (ทวิตเตอร์)’ ที่มีการสนทนาเกี่ยวกับเหตุการณ์สาธารณภัยเป็นจำนวนมาก แสดงให้เห็นผ่านการขึ้นเทรนด์ ทำให้ผู้คนให้ความสนใจกับเหตุการณ์นั้น ๆ มากขึ้น ซึ่งมีอิทธิพลต่อหน่วยงานที่เกี่ยวข้อง เพราะการที่หน่วยงานสามารถปรับตัว และตอบรับเหตุการณ์ในเวลาที่เหมาะสมทำให้มีผลการดำเนินงานที่มีประสิทธิภาพมากขึ้น และทำให้สามารถรักษาสภาพการณ์ได้ดีขึ้น การมีการแก้ไข และการปรับปรุงที่รวดเร็วนี้มีผลต่อความเชื่อมั่น และความพึงพอใจของประชาชนในการบริการหรือการจัดการสถานการณ์ต่าง ๆ ที่เกี่ยวข้อง ซึ่งเหตุการณ์สาธารณภัยต่าง ๆ ยกตัวอย่างเช่น น้ำท่วม, ไฟไหม้, การเสื่อมสภาพของถนน, ไฟป่า, การประท้วง, การระบาดของโรค เป็นต้น

ขอบเขตของโครงงาน

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

1 หมายถึง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

0 หมายถึง ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

และจะแยกข้อความในแต่ละประเภทออกเป็นหมวดหมู่ ทั้งหมด 5 หมวดหมู่ ได้แก่ หมวดหมู่ทั่วไป, หมวดหมู่น้ำท่วม, หมวดหมู่ระเบิด, หมวดหมู่สงคราม และหมวดหมู่ไฟไหม้

ประโยชน์ที่ได้รับ

1. ช่วยคัดแยกข้อความที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย
2. สามารถค้นหา ป้องกันอันตรายจากเหตุการณ์สาธารณภัยที่เกิดขึ้นได้รวดเร็วมากยิ่งขึ้น
3. ช่วยลดความเสี่ยง และจำนวนผู้ที่ได้รับอันตรายจากเหตุการณ์สาธารณภัย
4. สามารถนำข้อมูลที่ได้ไปวางแผนในการพัฒนา Emergency Alert ระบบเตือนภัย ยามเกิดเหตุฉุกเฉิน

ผลกระทบที่โครงงานมีต่อประชากรหรือชุมชน

1. มีประโยชน์ในการปรับปรุงคุณภาพชีวิตของประชาชน และสร้างความปลอดภัยของประชาชน โดยลดความเสี่ยงในการได้รับอันตรายจากเหตุการณ์นั้น ๆ
2. เพิ่มความรู้ และการตระหนักรู้ ทำให้ประชาชนมีการตระหนักรู้มากขึ้นเกี่ยวกับสถานการณ์รอบตัว และเพิ่มความรู้เกี่ยวกับวิธีการป้องกัน และจัดการกับเหตุการณ์นั้น
3. การแจ้งเตือน และการตอบสนองทันใจ สามารถช่วยลดความเสี่ยง และเพิ่มโอกาสในการรับมือกับสถานการณ์ในระยะเวลาที่สั้น ๆ ทำให้การตอบสนองเป็นไปได้อย่างทันที
4. ช่วยในการพัฒนานโยบาย, แผนการดำเนินงานเพื่อเตรียมความพร้อม และลดผลกระทบในอนาคต
5. สร้างความเชื่อมั่น และความพึงพอใจของประชาชนในการบริการหรือการจัดการสถานการณ์ต่าง ๆ จากหน่วยงานที่เกี่ยวข้อง

ความคิดสร้างสรรค์

1. การนำเอาเทคโนโลยี และการวิเคราะห์ข้อมูลทางด้านปัญญาประดิษฐ์มาใช้ในการเสริมสร้างความปลอดภัย และการจัดการเหตุการณ์สาธารณูปโภค
2. การนำเสนอแนวทางการป้องกัน และการจัดการเหตุฉุกเฉิน

วิธีการดำเนินงาน

1. การเก็บข้อมูล และข้อมูลที่นำมาใช้

ไฟล์ข้อความ รูปแบบตาราง ประเภท ไฟล์ CSV ชื่อไฟล์ aialldata.CSV ภายในข้อมูลประกอบไปด้วยคอลัมน์ จำนวน 4 คอลัมน์ ได้แก่ keyword, label, category และ text และ มีชุดข้อมูลจำนวน 4,000 ชุดข้อมูล แหล่งข้อมูลมาจาก แอปพลิเคชัน X (เอ็กซ์) ใช้วิธีการคัดลอกข้อความในแต่ละโพสต์ แบ่งข้อมูลเป็น train set กับ test set โดย train set 70% และ test set 30% จากการค้นคว้าข้อมูลเกี่ยวกับเหตุการณ์สาธารณภัย พบว่า “สาธารณภัย” หมายถึง เหตุอันมีผลกระทบต่อสาธารณชน ไม่ว่าเกิดจากธรรมชาติมีผู้ทำให้

เกิดขึ้น อุบัติเหตุ หรือเหตุอื่นใด ซึ่งก่อให้เกิดอันตรายแก่ชีวิต ร่างกายของประชาชนหรือความเสียหายแก่ทรัพย์สินของประชาชน หรือของรัฐ (กรมป้องกันและบรรเทาสาธารณภัย กระทรวงมหาดไทย, 2559)

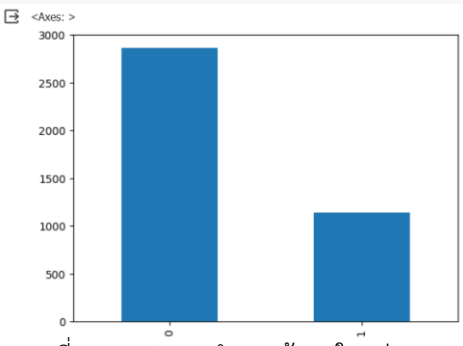
ผู้วิจัยจึงนำข้อมูลจากเหตุการณ์สาธารณภัยมากำหนดเป็นคีย์เวิร์ดในการรวบรวมข้อมูล จำนวน 5 คีย์เวิร์ด ได้แก่ น้ำท่วม, ไฟไหม้, ระเบิด, สงคราม และแผ่นดินไหว จากนั้นได้ทำการเลเบลข้อมูลในคอลัมน์ label ได้แก่ ได้แก่ 0 และ 1 “ 0 หมายถึง หมายถึง ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย” และ “1 หมายถึง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย” โดยแบ่งเลเบลเป็นข้อความที่เป็น 0 จำนวน 2,861 ข้อความ และ ข้อความที่เป็น 1 จำนวน 1,139 ข้อความ ดังรูปภาพต่อไปนี้

```
df[['keyword', 'label']].value_counts().reset_index(name='count')
```

	keyword	label	count
0	แผ่นดินไหว	1	360
1	สงคราม	0	305
2	ไฟไหม้	1	292
3	ระเบิด	0	266
4	น้ำท่วม	1	258
5	น้ำท่วม	0	142
6	ระเบิด	1	134
7	ไฟไหม้	0	108
8	สงคราม	1	95
9	แผ่นดินไหว	0	40

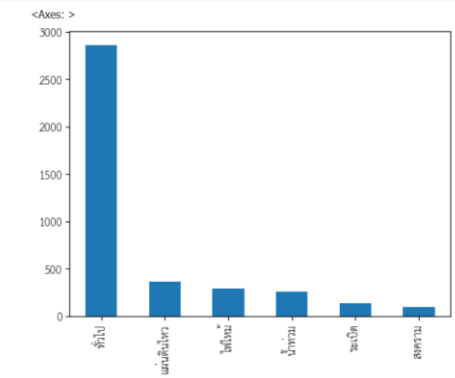
ภาพที่ 1 แสดงจำนวนข้อมูลของข้อความที่เป็น 0 และ 1 ของแต่ละคีย์เวิร์ด

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



ภาพที่ 2 กราฟแสดงจำนวนข้อมูลในแต่ละ label

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



ภาพที่ 3 กราฟแสดงจำนวนข้อมูลในแต่ละ category

ตัวอย่างข้อมูลแต่ละประเภท

label	category	text
1	สงคราม น้ำท่วม ระเบิด ไฟไหม้ แผ่นดินไหว	- ‘บลิงเคน’ บินด่วนไปตะวันออกกลาง-หวั่น**สงคราม**อิสราเอล-ฮามาสขยายวง - นราธิวาส- **น้ำท่วม**บ้านเรือนที่ระแงะแล้ว ขณะที่ลุ่มน้ำบางนราจ่อล้น - เกิดการ**ระเบิด**ทำลายล้างสูง ใน Cherkasy ยูเครน - **ไฟไหม้**ชุมชนบ้านพักรถไฟบางซื่อเผาวอด 10 หลังคาเรือน - **แผ่นดินไหว**ญี่ปุ่น 7.6 แมกนิจูดเกิดคลื่นยักษ์สึนามิสูง 1.2ม. ซัดเข้าชายฝั่ง
0	ทั่วไป	- คนจะนอนแต่**สงคราม**ในเกมมันเรียกร้อง - พรุ่งนี้ศลปร้องเพลง แฟนคลับร้องไห้ **น้ำท่วม**สยามแน่ - ใส่แมสแล้วเท่**ระเบิด**เลย พอถอดแมสปุ๊บปากนิดจมูกหน่อยขนตายาวปากชมพูน่ารักไม่ไหวแระ - ทำงานที่นี่มาจะ 5 ปี ก็งาน**ไฟไหม้**ตลอด ไม่เคยมีสั่งล่วงหน้า - หล่อ**แผ่นดินไหว**มากแม่! - ผิ้งน้อยตัวอ้วนๆ

2. เทคนิคที่นำมาใช้

Logistic Regression การวิเคราะห์การถดถอยโลจิสติก เป็นเทคนิคการวิเคราะห์ตัวแปรเชิงพหุที่มีวัตถุประสงค์เพื่อประมาณค่าหรือทำนายเหตุการณ์ที่สนใจว่าจะเกิดหรือไม่เกิดเหตุการณ์นั้นภายใต้อิทธิของตัวปัจจัยแบบจำลองโลจิสติกประกอบด้วยตัวแปรตาม (หรือตัวแปรเกณฑ์) ที่ต้องเป็นตัวแปรแบบทวินาม (Dichotomous Variable) กล่าวคือมีได้สองค่า เช่น “เกิด” กับ “ไม่เกิด” หรือ “เสี่ยง” กับ “ไม่เสี่ยง” เป็นต้น และตัวแปรอิสระ (หรือตัวแปรทำนาย) ที่อาจมีตัวเดียวหรือหลายตัวที่เป็นได้ทั้งตัวแปรเชิงกลุ่ม(Categorical Variable) หรือตัวแปรแบบต่อเนื่อง (Continuous Variable) (ชูชีพ, 2018)

Decision Tree ต้นไม้ตัดสินใจ เป็นโมเดลทางคณิตศาสตร์ที่ใช้ทำนายประเภทของวัตถุโดยพิจารณาจากลักษณะของวัตถุ โดยมีลักษณะการตัดสินใจเป็นแบบแผนผังต้นไม้ประกอบด้วย ส่วนที่เป็นบัพภายใน (inner node) หรือที่เรียกว่าโหนดของต้นไม้ใช้แทนตัวแปรต่างๆ ที่ใช้ในการตัดสินใจ ส่วนที่กิ่ง (Branch) ใช้แทนค่าที่เป็นได้ของตัวแปร และส่วนที่เป็นบัพใบ (leaf node) หรือที่เรียกว่าโหนดใบ ใช้แทนค่าคำตอบของการตัดสินใจหรือการจำแนก (มหาวิทยาลัยนเรศวร)

Random Forest คือขั้นตอนวิธีหนึ่งของการเรียนรู้ของเครื่องที่นิยมใช้มากใช้ได้ดีในหลายๆ ปัญหาทั้งกับปัญหาแบบ Regression และ Classification โดย Random Forest เป็นขั้นตอนวิธีพัฒนาต่อยอดมาจากDecision Tree ต่างกันที่ Random Forest เป็นการเพิ่มจำนวนต้นไม้ (Tree) เป็นหลายๆ ต้นทำให้ประสิทธิภาพการทำงานและพยากรณ์สูงขึ้น Random Forest มีหลักการทำงาน คือ จะแบ่งข้อมูลออกเป็นต้นไม้ตัดสินใจ (Decision Tree) หลายๆ ต้น โดยแต่ละต้นจะได้รับคุณลักษณะ (Feature) และข้อมูล (Data)

ที่ไม่เหมือนกันทั้งหมด เพื่อทำให้ได้ต้นไม้ที่มีความหลากหลายและมีความอิสระต่อกันมากขึ้น(มหาวิทยาลัยนเรศวร)

K-nearest neighbors (KNN) นับเป็นเทคนิคที่มีวิธีการไม่ซับซ้อนและเข้าใจได้ง่ายที่สุดที่ใช้ในการจำแนกประเภทข้อมูล โดยหลักการทำงาน คือ จะใช้หลักการเปรียบเทียบความคล้ายคลึงกันของข้อมูลที่สนใจกับข้อมูลอื่นว่ามีความคล้ายคลึงหรืออยู่ใกล้กับข้อมูลใดมากที่สุด k ตัว จากนั้นจะทำการตัดสินใจว่าคำตอบของข้อมูลที่สนใจนั้นควรเป็นคำตอบเดียวกับข้อมูลที่อยู่ใกล้ที่สุด k ตัวนั้น ทั้งนี้ k คือความถี่ของข้อมูลที่อยู่ใกล้กับข้อมูลที่สนใจ (มหาวิทยาลัยนเรศวร)

เทคนิคการเรียนรู้เชิงลึก (Deep Learning) เป็นเทคนิคในกลุ่มโครงข่ายประสาทเทียม (Artificial Neural Network: ANN) ที่มีโครงสร้างขนาดใหญ่ประกอบด้วยนิวรอนและชั้นซ่อนจำนวนมาก เป็นอัลกอริทึมที่ถูกสร้างขึ้นเพื่อการเรียนรู้ของเครื่องจักร แต่ระดับ Hidden Layer ของการเรียนรู้เชิงลึกมีมากกว่า ANN ซึ่งแต่ละเลเยอร์จะเปรียบเสมือนประกอบด้วยเซลล์ประสาท (Neural) จำนวนมากที่มีหน้าที่ในการประมวลผล โดยเลเยอร์แรกสุดท้ายจะทำหน้าที่ในการรับข้อมูล (Input Layer) และส่งข้อมูลที่ประมวลผลเสร็จแล้วไปยังเลเยอร์สุดท้าย (Output Layer) การส่งข้อมูลแบบนี้มีข้อดีคือแต่ละเลเยอร์อาจทำให้มีค่าถ่วงน้ำหนัก (Weight) ค่าความเอนเอียงของข้อมูล (Bias) และวิธีการประมวลผลทางคณิตศาสตร์ (Activation Function) เป็นอิสระต่อกันถ้าป้อนข้อมูลเข้าไปให้กับโมเดลมากเท่าไหร่ แต่ละเลเยอร์ก็จะสามารถสกัดคุณลักษณะที่มีความซับซ้อนมากขึ้นทำให้ระบบสามารถตัดสินใจได้ใกล้เคียงกับมนุษย์มากยิ่งขึ้น (สมศักดิ์ ศรีสวการย์ และ สมัย ศรีสวย, 2563)

3. วิธีการ และการกำหนดค่าต่าง ๆ ของเทคนิคที่นำมาใช้

3.1 การนำเข้าข้อมูล

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

ภาพที่ 4 รายละเอียดข้อมูลนำเข้า

ไฟล์ข้อความรูปแบบตารางประเภทไฟล์ CSV ชื่อไฟล์ aidata.CSV ภายในไฟล์คือข้อมูลที่จัดเก็บข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย และข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

3.2 การติดตั้ง เครื่องมือ PyThaiNLP และการติดตั้ง font สำหรับภาษาไทย

3.2.1 การติดตั้งเครื่องมือ PyThaiNLP

```
#การติดตั้งเครื่องมือ PyThaiNLP
!pip install https://github.com/PyThaiNLP/pythainlp/archive/dev.zip
!pip install python-crfsuite
import pandas as pd
import re
from pythainlp.tokenize import word_tokenize
```

ภาพที่ 5 ติดตั้งเครื่องมือ PyThaiNLP

3.2.1 การติดตั้ง font สำหรับภาษาไทย

```
#การติดตั้ง font สำหรับภาษาไทย
!wget -q https://github.com/Phonbopit/sarabun-webfont/raw/master/fonts/thsarabunnew-webfont.ttf
font_path = '/content/thsarabunnew-webfont.tff'
```

ภาพที่ 6 ติดตั้ง font สำหรับภาษาไทย

3.3 Data Preprocessing

```
[52] # 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
```

ภาพที่ 7 การทำความสะอาดข้อมูลโดยใช้ฟังก์ชัน regex

- Import re เพื่อใช้งานฟังก์ชัน regex
- กำหนด คอลัมน์ text ให้อยู่ในตัวแปรที่ชื่อว่า content
- เรียกใช้ฟังก์ชัน def เพื่อลบข้อมูลและส่งข้อมูลตัวแปร content เข้ามา
- ทำการจัดการ แก้ไข และลบส่วนที่ไม่ต้องการออก ได้แก่ ลบลิงค์, ลบแฮชแท็ก, ลบอักขระพิเศษต่าง ๆ แทนด้วยช่องว่าง

```
[14] df['cleantext'] = df['text'].apply(cleantext)
     df
```

ภาพที่ 8 เรียกใช้ cleantext

เรียกใช้ df.text.apply(clean_text) ที่สร้างไว้เพื่อเก็บค่าเข้า คอลัมน์ df[‘cleantext’]

3.4 Word Tokenize การตัดคำต่าง ๆ ออกจากประโยค

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

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

ภาพที่ 9 Word Tokenize

- สร้างฟังก์ชันที่ชื่อว่า my_tokenise
- เรียกใช้ Word_tokenize เก็บค่าไว้ในตัวแปร txt

3.5 Word Cloud

3.5.1 นำเข้า library ต่าง ที่เกี่ยวกับการทำ word cloud แสดง thai stopwords

```
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
```

ภาพที่ 10 นำเข้า libraryต่าง ที่เกี่ยวกับการทำ word cloud แสดง thai stopwords

3.5.2 word cloud ของข้อมูลทั้งหมด

```
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()
```

ภาพที่ 11 word cloud ของข้อมูลทั้งหมด

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ token เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.3 word cloud ของ keyword น้ำท่วม

```
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[df.keyword == 'น้ำท่วม'].token.values)))
        #มีข้อความที่พูดถึงน้ำท่วม

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

ภาพที่ 12 word cloud ของ keyword น้ำท่วม

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword น้ำท่วม เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.4 word cloud ของ keyword น้ำท่วม ที่มีข้อความเป็น 1 (1 = ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'น้ำท่วม') & (df.label == 1)].token.values)))
        #มีข้อความที่พูดถึงน้ำท่วมที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 13 word cloud ของ keyword น้ำท่วม ที่มีข้อความเป็น 1

- เรียกใช้ font_path ที่ติดตั้งไว้

- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword น้ำท่วม ที่ label เป็น 1 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.5 word cloud ของ keyword น้ำท่วม ที่มีข้อความเป็น 0 (0 = ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'น้ำท่วม') & (df.label == 0)].token.values)))
        #มีข้อความที่พูดถึงน้ำท่วมที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 14 word cloud ของ keyword น้ำท่วม ที่มีข้อความเป็น 0

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword น้ำท่วม ที่ label เป็น 0 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.6 word cloud ของ keyword ระเบิด

```
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[df.keyword == 'ระเบิด'].token.values)))
        #มีข้อความที่พูดถึงระเบิด

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

ภาพที่ 15 word cloud ของ keyword ระเบิด

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword ระเบิด เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.7 word cloud ของ keyword ระเบิด ที่มีข้อความเป็น 1 (1 = ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'ระเบิด') & (df.label == 1)].token.values)))
        #มีข้อความที่พูดถึงระเบิดที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 16 word cloud ของ keyword ระเบิด ที่มีข้อความเป็น 1

- เรียกใช้ font_path ที่ติดตั้งไว้

- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword ระเบิด ที่ label เป็น 1 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ
-

3.5.8 word cloud ของ keyword ระเบิด ที่มีข้อความเป็น 0 (0 = ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'ระเบิด') & (df.label == 0)].token.values)))
        #มีข้อความที่พูดถึงระเบิดที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 17 word cloud ของ keyword ระเบิด ที่มีข้อความเป็น 0

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword ระเบิด ที่ label เป็น 0 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.9 word cloud ของ keyword สงคราม

```
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[df.keyword == 'สงคราม'].token.values)))
        #มีข้อความที่พูดถึงสงคราม

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

ภาพที่ 18 word cloud ของ keyword สงคราม

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword สงคราม เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.10 word cloud ของ keyword สงคราม ที่มีข้อความเป็น 1 (1 = ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'สงคราม') & (df.label == 1)].token.values)))
        #มีข้อความที่พูดถึงสงครามที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 19 word cloud ของ keyword สงคราม ที่มีข้อความเป็น 1

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword สงคราม ที่ label เป็น 1 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.11 word cloud ของ keyword สงคราม ที่มีข้อความเป็น 0 (0 = ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'สงคราม') & (df.label == 0)].token.values)))
        #มีข้อความที่พูดถึงสงครามที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 20 word cloud ของ keyword สงคราม ที่มีข้อความเป็น 0

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword สงคราม ที่ label เป็น 0 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.12 word cloud ของ keyword ไฟไหม้

```
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[df.keyword == 'ไฟไหม้'].token.values)))
        #มีข้อความที่พูดถึงไฟไหม้

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

ภาพที่ 21 word cloud ของ keyword ไฟไหม้

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword ไฟไหม้ เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.13 word cloud ของ keyword ไฟไหม้ ที่มีข้อความเป็น 1 (1 = ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'ไฟไหม้') & (df.label == 1)].token.values)))
        #มีข้อความที่พูดถึงไฟไหม้ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 22 word cloud ของ keyword ไฟไหม้ ที่มีข้อความเป็น 1

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword ไฟไหม้ ที่ label เป็น 1 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.14 word cloud ของ keyword ไฟไหม้ ที่มีข้อความเป็น 0 (0 = ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == "ไฟไหม้") & (df.label == 0)].token.values)))
        #มีข้อความที่พูดถึงไฟไหม้ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 23 word cloud ของ keyword ไฟไหม้ ที่มีข้อความเป็น 0

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword ระเบิด ที่ label เป็น 0 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.15 word cloud ของ keyword แผ่นดินไหว

```
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[df.keyword == 'แผ่นดินไหว'].token.values)))
        #มีข้อความที่พูดถึงแผ่นดินไหว

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

ภาพที่ 24 word cloud ของ keyword แผ่นดินไหว

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword แผ่นดินไหว เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.16 word cloud ของ keyword แผ่นดินไหว ที่มีข้อความเป็น 1 (1 = ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'แผ่นดินไหว') & (df.label == 1)].token.values)))
        #มีข้อความที่พูดถึงไฟไหม้ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 25 word cloud ของ keyword แผ่นดินไหว ที่มีข้อความเป็น 1

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword แผ่นดินไหว ที่ label เป็น 1 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.5.17 word cloud ของ keyword แผ่นดินไหว ที่มีข้อความเป็น 0 (0 = ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

```
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[(df.keyword == 'แผ่นดินไหว') & (df.label == 0)].token.values)))
        #มีข้อความที่พูดถึงไฟไหม้ที่เกี่ยวข้องกับเหตุการณ์สาธารณภัย

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

ภาพที่ 26 word cloud ของ keyword แผ่นดินไหว ที่มีข้อความเป็น 0

- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 1000
- กำหนดขนาด ความสูง = 500
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud = 200 คำ
- ลบ stopwords และ Generate เชื่อมกับคอลัมน์ keyword แผ่นดินไหว ที่ label เป็น 0 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงออกมาเป็นรูปภาพ

3.6 การทำแบ่ง Train set และ Test set (สำหรับการ Detect label)

```
[146] data = df.cleantext.values
      y = df.label.values
```

ภาพที่ 27 แสดงการเก็บค่าต่างๆไว้ในตัวแปร

- นำคอลัมน์ cleantext ไปเก็บไว้ในตัวแปร data
- นำคอลัมน์ label ไปเก็บไว้ในตัวแปร y

```
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)
```

ภาพที่ 28 แสดงการแบ่ง train set กับ test set

แบ่งข้อมูลเป็น train set กับ test set โดย train set 70% และ test set 30%

3.7 การทำ feature extraction (สำหรับการ Detect label)

3.7.1 การทำ feature extraction ด้วย CountVectorizer

```
[152] 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())
```

ภาพที่ 29 แสดงการทำfeature extraction ด้วย CountVectorizer

- นำเข้าLibrary ที่สำคัญในการสกัด CountVectorizer ดังนี้
- from sklearn.feature_extraction.text import CountVectorizer
- เรียกใช้ CountVectorizer เก็บค่าเข้าตัวแปร Vectorizer และนำมาvectorizer.fit_transform (data) ข้อมูลที่มีอยู่

3.7.2 การทำ feature extraction ด้วย TfidfVectorizer

```
[155] 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())
```

ภาพที่ 30 แสดงการทำfeature extraction ด้วย TfidfVectorizer

- นำเข้าLibrary ที่สำคัญในการสกัด TfidfVectorizer ดังนี้
- from sklearn.feature_extraction.text import TfidfVectorizer
- เรียกใช้ TfidfVectorizer กำหนด smooth_idf=False เก็บค่าเข้าตัวแปร tfidf
- และนำมา vectorizer.fit_transform(data) ข้อมูลที่มีอยู่
-

3.8 การ Train Model ด้วยการสกัด CountVectorizer (สำหรับการ Detect label)

3.8.1 Logistic Regression

```
[156] from sklearn.linear_model import LogisticRegression
      from sklearn import metrics

      logreg = LogisticRegression()
      logreg.fit(x_train_count,y_train)
      y_pred = logreg.predict(x_test_count)

      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 31 แสดงการเทรนโมเดล Logistic Regression ด้วยการสกัด CountVectorizer

3.8.2 Random Forest

```
[157] from sklearn.ensemble import RandomForestClassifier

      rf = RandomForestClassifier(n_estimators=50)
      rf.fit(x_train_count,y_train)
      y_pred = rf.predict(x_test_count)

      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 32 แสดงการเทรนโมเดล Random Forest ด้วยการสกัด CountVectorizer

3.8.3 Decision tree

```
[158] from sklearn import tree

      tree = tree.DecisionTreeClassifier(max_depth=4, class_weight='balanced')
      tree.fit(x_train_count,y_train)
      y_pred = tree.predict(x_test_count)
      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 33 แสดงการเทรนโมเดล Decision tree ด้วยการสกัด CountVectorizer

3.8.3 K-Nearest Neighbors

```
[161] from sklearn import neighbors

      knn = neighbors.KNeighborsClassifier()
      knn.fit(x_train_count,y_train)
      y_pred = knn.predict(x_test_count)

      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 34 แสดงการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด CountVectorizer

3.9 การ Train Model ด้วยการสกัด TfidfVectorizer (สำหรับการ Detect label)

3.9.1 Logistic Regression

```
[162] from sklearn.linear_model import LogisticRegression

      logreg = LogisticRegression()
      logreg.fit(x_train_tfidf,y_train)
      y_pred = logreg.predict(x_test_tfidf)

      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 35 แสดงการเทรนโมเดล Logistic Regression ด้วยการสกัด TfidfVectorizer

3.9.2 Random Forest

```
[163] from sklearn.ensemble import RandomForestClassifier

      rf = RandomForestClassifier(n_estimators=50)
      rf.fit(x_train_tfidf,y_train)
      y_pred = rf.predict(x_test_tfidf)

      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 36 แสดงการเทรนโมเดล Random Forest ด้วยการสกัด TfidfVectorizer

3.9.3 Decision tree

```
[164] from sklearn import tree

      tree = tree.DecisionTreeClassifier(max_depth=4, class_weight='balanced')
      tree.fit(x_train_tfidf,y_train)
      y_pred = tree.predict(x_test_tfidf)

      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 37 แสดงการเทรนโมเดล Decision tree ด้วยการสกัด TfidfVectorizer

3.9.4 K-Nearest Neighbors

```
[165] from sklearn import neighbors

      knn = neighbors.KNeighborsClassifier()
      knn.fit(x_train_tfidf,y_train)
      y_pred = knn.predict(x_test_tfidf)

      print(y_pred)
      print('Accuracy = %.3f' % (metrics.accuracy_score(y_test,y_pred)))
```

ภาพที่ 38 แสดงการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด TfidfVectorizer

3.10 การทำแบ่ง Train set และ Test set (สำหรับการ Detect category)

```
[61] data1 = df.cleantext.values
     y1 = df.category.values
```

ภาพที่ 39 แสดงการเก็บค่าต่าง ๆ ไว้ในตัวแปร

- นำคอลัมน์ cleantext ไปเก็บไว้ในตัวแปร data1
- นำคอลัมน์ category ไปเก็บไว้ในตัวแปร y

```
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)
```

ภาพที่ 40 แสดงการแบ่ง train set กับ test set

แบ่งข้อมูลเป็น train set กับ test set โดย train set 70% และ test set 30%

3.11 การทำ feature extraction (สำหรับการ Detect category)

3.11.1 การทำ feature extraction ด้วย CountVectorizer

```
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())
```

ภาพที่ 41 แสดงการทำfeature extraction ด้วย CountVectorizer

- นำเข้าLibrary ที่สำคัญในการสกัด CountVectorizer ดังนี้
- from sklearn.feature_extraction.text import CountVectorizer
- เรียกใช้ CountVectorizer เก็บค่าเข้าตัวแปร Vectorizer และนำมา vectorizer.fit_transform(data1) ข้อมูลที่มีอยู่

3.11.2 การทำ feature extraction ด้วย TfidfVectorizer

```
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())
```

ภาพที่ 42แสดงการทำfeature extraction ด้วย TfidfVectorizer

- นำเข้าLibrary ที่สำคัญในการสกัด TfidfVectorizer ดังนี้
- from sklearn.feature_extraction.text import TfidfVectorizer
- เรียกใช้ TfidfVectorizer กำหนด smooth_idf=False เก็บค่าเข้าตัวแปร tfidf
- และนำมา vectorizer.fit_transform(data) ข้อมูลที่มีอยู่

3.12 การ Train Model ด้วยการสกัด CountVectorizer (สำหรับการ Detect category)

3.12.1 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logreg1 = LogisticRegression()
logreg1.fit(x1_train_tfidf,y1_train)
y1_pred = logreg1.predict(x1_test_tfidf)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่ 43แสดงการเทรนโมเดล Logistic Regression ด้วยการสกัด CountVectorizer

3.12.2 Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf1 = RandomForestClassifier(n_estimators=50)
rf1.fit(x1_train_count,y1_train)
y1_pred = rf1.predict(x1_test_count)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่ 44 แสดงการเทรนโมเดล Random Forest ด้วยการสกัด CountVectorizer

3.12.3 Decision tree

```
from sklearn import tree

tree1 = tree.DecisionTreeClassifier(max_depth=4, class_weight='balanced')
tree1.fit(x1_train_count,y1_train)
y1_pred = tree1.predict(x1_test_count)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่ 45 แสดงการเทรนโมเดล Decision tree ด้วยการสกัด CountVectorizer

3.12.4 K-Nearest Neighbors

```
from sklearn import neighbors

knn1 = neighbors.KNeighborsClassifier()
knn1.fit(x1_train_count,y1_train)
y1_pred = knn1.predict(x1_test_count)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่ 46 แสดงการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด CountVectorizer

3.13 การ Train Model ด้วยการสกัด TfidfVectorizer (สำหรับการ Detect category)

3.13.1 Logistic Regression

```
from sklearn.linear_model import LogisticRegression

logreg1 = LogisticRegression()
logreg1.fit(x1_train_tfidf,y1_train)
y1_pred = logreg1.predict(x_test_tfidf)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่ 47 แสดงการเทรนโมเดล Logistic Regression ด้วยการสกัด TfidfVectorizer

3.13.2 Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf1 = RandomForestClassifier(n_estimators=50)
rf1.fit(x1_train_tfidf,y1_train)
y1_pred = rf1.predict(x1_test_tfidf)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่ 48 แสดงการเทรนโมเดล Random Forest ด้วยการสกัด TfidfVectorizer

3.13.3 Decision tree

```
from sklearn import tree

tree1 = tree.DecisionTreeClassifier(max_depth=4, class_weight='balanced')
tree1.fit(x1_train_tfidf,y1_train)
y1_pred = tree1.predict(x1_test_tfidf)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่ 49 แสดงการเทรนโมเดล Decision tree ด้วยการสกัด TfidfVectorizer

3.13.4 K-Nearest Neighbors

```
from sklearn import neighbors

knn1 = neighbors.KNeighborsClassifier()
knn1.fit(x1_train_tfidf,y1_train)
y1_pred = knn1.predict(x1_test_tfidf)

print(y1_pred)
print('Accuracy = %.3f' % (metrics.accuracy_score(y1_test,y1_pred)))
```

ภาพที่50 แสดงการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด TfidfVectorizer

3.14 การทำแบ่ง Train set และ Test set สำหรับการ Detect label

```
sentences = df['cleantext'].values
y = df['label'].values
```

ภาพที่ 51แสดงการเก็บค่าต่างๆไว้ในตัวแปร

- นำคอลัมน์ cleantext ไปเก็บไว้ในตัวแปร sentences
- นำคอลัมน์ label ไปเก็บไว้ในตัวแปร y

```
sentences_train, sentences_test, y_train, y_test = train_test_split(
  sentences, y, test_size=0.3, random_state=1000)
```

ภาพที่ 52 แสดงการแบ่ง train set กับ test set

- แบ่งข้อมูลเป็น train set กับ test set โดย train set 70% และ test set 30%

3.15 Deep Learning สำหรับการ Detect label

```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(sentences_train)

X_train = tokenizer.texts_to_sequences(sentences_train)
X_test = tokenizer.texts_to_sequences(sentences_test)

vocab_size = len(tokenizer.word_index) + 1  # Adding 1 because of reserved 0 index

print(sentences_train[4])
print(X_train[4])
```

ภาพที่ 53 แสดง Tokenization ของข้อความ (text) ทั้งในชุดฝึก (training set) และชุดทดสอบ (testing set)

- นำเข้าไลบารี่Tokenizer จาก Keras เพื่อทำ Tokenization ของข้อความ (text) ทั้งในชุดฝึก (training set) และชุดทดสอบ (testing set)
- กำหนด num_words=5000
- fit_on_texts ถูกใช้เพื่อปรับ Tokenizer กับข้อมูลฝึก (sentences_train)
- texts_to_sequences ถูกใช้เพื่อแปลงข้อความใน sentences_train และ sentences_test ให้เป็นลำดับของตัวเลข โดยให้แต่ละคำถูกแทนที่ด้วยดัชนีที่ได้จากการ Fit Tokenizer

```
def Average(lst):
  sum = 0
  for i in lst:
    sum += len(i)
  return sum / len(lst)

Average(X_train)
```

ภาพที่ 54 แสดงความยาวเฉลี่ยของลำดับ (sequence) ที่อยู่ในลิสต์ lst

- ฟังก์ชัน Average(X_train) คำนวณความยาวเฉลี่ยของลำดับทั้งหมดในชุดฝึก (X_train) ที่มีอยู่ในลิสต์ X_train และคืนค่านั้นออกมา

3.15.1 Neural Network (NN)

```
embedding_dim = 50

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           input_length=maxlen))
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

ภาพที่ 55 แสดงการสร้างโมเดล Deep Learning ด้วย Neural Network

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด output_dim เท่ากับ 50
- Flatten layer ถูกใช้เพื่อแปลงเวกเตอร์ที่ได้จาก Embedding layer เป็นลักษณะเชิงแถว (1D)
- Dense layer ที่มี 10 โหนด และ activation function เป็น 'relu' ใช้ในการเรียนรู้ลักษณะของข้อมูลที่ซับซ้อน
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy'

```
history = model.fit(X_train, y_train,
                    epochs=20,
                    verbose=False,
                    validation_data=(X_test, y_test),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่56 แสดงการเทรนโมเดล Deep Learning ด้วย Neural Network

- history = model.fit(...): เทรนด้วยข้อมูลฝึก X_train และ y_train โดยใช้ฟังก์ชัน fit
- Epochs = 20 ระบุจำนวนรอบการเทรนทั้งหมดที่โมเดลจะถูกฝึก เท่ากับ 20
- verbose=False ระบุให้ไม่แสดงผลลัพธ์ทุกรอบการเทรน

- validation_data=(X_test, y_test) ระบุชุดข้อมูลทดสอบ เพื่อทำการตรวจสอบประสิทธิภาพของโมเดลบนชุดทดสอบขณะเทรน
- batch_size=10 ระบุขนาดของแบทช์ (batch) ที่จะถูกใช้ในการเทรน เท่ากับ 10
- ทดสอบด้วยข้อมูลฝึก X_train และ y_train ด้วยฟังก์ชัน evaluate

3.15.2 Convolutional Neural Network (CNN)

```
embedding_dim = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

ภาพที่57 แสดงการสร้างโมเดล Deep Learning ด้วย Convolutional Neural Network

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด embedding_dim เท่ากับ 100
- Conv1D layer ถูกใช้เพื่อทำการ Convolution บนลำดับของตัวเลขที่ได้จาก Embedding layer ซึ่งมี 128 ตัวกรอง (filters) ที่มีขนาด 5 (kernel size)
- Activation function ที่ใช้คือ 'relu' (Rectified Linear Unit)
- GlobalMaxPooling1D layer ถูกใช้เพื่อทำการ max pooling บนลำดับของตัวเลขที่ผ่านการ Convolution
- Dense layer ที่มี 10 โหนด และ activation function เป็น 'relu' ใช้ในการเรียนรู้ลักษณะของข้อมูลที่ซับซ้อน
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy'

```
history = model.fit(X_train, y_train,
                    epochs=10,
                    verbose=False,
                    validation_data=(X_test, y_test),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่58 แสดงการเทรนโมเดล Deep Learning ด้วย Convolutional Neural Network

- history = model.fit(...): เทรนด้วยข้อมูลฝึก X_train และ y_train โดยใช้ฟังก์ชัน fit
- Epochs = 10 ระบุจำนวนรอบการเทรนทั้งหมดที่โมเดลจะถูกฝึก เท่ากับ 10
- verbose=False ระบุให้ไม่แสดงผลลัพธ์ทุกรอบการเทรน
- validation_data=(X_test, y_test) ระบุชุดข้อมูลทดสอบ เพื่อทำการตรวจสอบประสิทธิภาพของโมเดลบนชุดทดสอบขณะเทรน

- batch_size=10 ระบุขนาดของแบทช์ (batch) ที่จะถูกใช้ในการเทรน เท่ากับ 10
- ทดสอบด้วยข้อมูลฝึก X_train และ y_train ด้วยฟังก์ชัน evaluate

3.15.3 Recurrent Neural Network (RNN)

```
embedding_dim = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Bidirectional(layers.LSTM(128)))  # ใช้ LSTM ทั่วไป
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

print(model.summary())
```

ภาพที่59 แสดงการสร้างโมเดล Deep Learning ด้วย Recurrent Neural Network

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด embedding_dim เท่ากับ 100
- LSTM(128) สร้าง LSTM ที่มี 128 โหนด (units)
- Dense layer ที่มี 32 โหนดและ activation function เป็น 'relu'
- Dense layer ที่มี 16 โหนดและ activation function เป็น 'relu'
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy

```
history = model.fit(X_train, y_train,
                    epochs=10,
                    verbose=False,
                    validation_data=(X_test, y_test),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่ 60 แสดงการเทรนโมเดล Deep Learning ด้วย Recurrent Neural Network

- history = model.fit(...): เทรนด้วยข้อมูลฝึก X_train และ y_train โดยใช้ฟังก์ชัน fit
- Epochs = 10 ระบุจำนวนรอบการเทรนทั้งหมดที่โมเดลจะถูกฝึก เท่ากับ 10
- verbose=False ระบุให้ไม่แสดงผลลัพธ์ทุกรอบการเทรน
- validation_data=(X_test, y_test) ระบุชุดข้อมูลทดสอบ เพื่อทำการตรวจสอบประสิทธิภาพของโมเดลบนชุดทดสอบขณะเทรน
- batch_size=10 ระบุขนาดของแบทช์ (batch) ที่จะถูกใช้ในการเทรน เท่ากับ 10
- ทดสอบด้วยข้อมูลฝึก X_train และ y_train ด้วยฟังก์ชัน evaluate

3.15.4 Model with Dropout

```
embedding_dim = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Bidirectional(layers.LSTM(128)))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(32,activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(3,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

print(model.summary())
```

ภาพที่ 61แสดงการสร้างโมเดล Deep Learning ด้วย Model with Dropout

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด embedding_dim เท่ากับ 100
- LSTM(128) สร้าง LSTM ที่มี 128 โหนด (units)
- Dense layer ที่มี 32 โหนดและ activation function เป็น 'relu'
- Dense layer ที่มี 16 โหนดและ activation function เป็น 'relu'
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy

```
history = model.fit(X_train, y_train,
                    epochs=10,
                    verbose=False,
                    validation_data=(X_test, y_test),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่ 62 แสดงการเทรนโมเดล Deep Learning ด้วย Model with Dropout

- history = model.fit(...): เทรนด้วยข้อมูลฝึก X_train และ y_train โดยใช้ฟังก์ชัน fit
- Epochs = 10 ระบุจำนวนรอบการเทรนทั้งหมดที่โมเดลจะถูกฝึก เท่ากับ 10
- verbose=False ระบุให้ไม่แสดงผลลัพธ์ทุกรอบการเทรน
- validation_data=(X_test, y_test) ระบุชุดข้อมูลทดสอบ เพื่อทำการตรวจสอบประสิทธิภาพของโมเดลบนชุดทดสอบขณะเทรน
- batch_size=10 ระบุขนาดของแบทช์ (batch) ที่จะถูกใช้ในการเทรน เท่ากับ 10
- ทดสอบด้วยข้อมูลฝึก X_train และ y_train ด้วยฟังก์ชัน evaluate

3.16 Deep Learning สำหรับการ Detect category

```
sentences = df['cleantext'].values
y = df['category'].values
```

ภาพที่ 63 แสดงการเก็บค่าต่างๆไว้ในตัวแปร

- นำคอลัมน์ cleantext ไปเก็บไว้ในตัวแปร sentences
- นำคอลัมน์ category ไปเก็บไว้ในตัวแปร y

```
sentences_train, sentences_test, y_train, y_test = train_test_split(
  sentences, y, test_size=0.3, random_state=1000)
```

ภาพที่ 64 แสดงการแบ่ง train set กับ test set

- แบ่งข้อมูลเป็น train set กับ test set โดย train set 70% และ test set 30%

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

ภาพที่ 65 แสดงการแปลงค่าในชุดข้อมูล

- การแปลงค่าในชุดข้อมูลฝึก (y_train): label_encoder.fit_transform(y_train) จะทำการ Fit label encoder ด้วยค่าคลาสทั้งหมดในชุดข้อมูลฝึก (y_train) และแปลงค่าคลาสเหล่านั้นเป็นตัวเลข
- การแปลงค่าในชุดข้อมูลทดสอบ (y_test): label_encoder.transform(y_test) จะใช้ label encoder ที่ถูก Fit จากชุดข้อมูลฝึกเพื่อแปลงค่าคลาสในชุดข้อมูลทดสอบ (y_test) เป็นตัวเลขตาม mapping ที่ได้จากการ Fit ก่อนหน้า

3.16.1 Neural Network (NN)

```
embedding_dim = 50

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           input_length=maxlen))
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

ภาพที่ 66 แสดงการสร้างโมเดล Deep Learning ด้วย Neural Network

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด embedding_dim เท่ากับ 50
- Flatten layer ถูกใช้เพื่อแปลงข้อมูลจากลำดับที่มีมิติไปเป็นเวกเตอร์แบบแนวนอน
- Dense layer ที่มี 10 โหนดและ activation function เป็น 'relu'
- Dense layer ที่มี 10 โหนดและ activation function เป็น 'softmax'
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy'

```
#sparse_categorical_crossentropy
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

loss, accuracy  = model.evaluate(X_train, y_train_encoded, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))

loss, accuracy = model.evaluate(X_test, y_test_encoded, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่ 67 แสดงการเทรนโมเดล Deep Learning ด้วย Neural Network

- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy' และ metric เป็น 'accuracy'
- model.evaluate ถูกใช้เพื่อประเมินประสิทธิภาพของโมเดลทั้งในชุดฝึกและทดสอบ

3.16.2 Convolutional Neural Network (CNN)

```
embedding_dim = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(6, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

ภาพที่ 68 แสดงการสร้างโมเดล Deep Learning ด้วย Convolutional Neural Network

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด embedding_dim เท่ากับ 100
- สร้างชั้น Convolutional ที่มี 128 filters, kernel size 5, และ activation function เป็น 'relu'
- สร้างชั้น Global Max Pooling ที่ใช้ในการลดขนาดของข้อมูลหลังจาก Convolutional Layer
- Dense layer ที่มี 10 โหนดและ activation function เป็น 'relu'
- Dense layer ที่มี 6 โหนดและ activation function เป็น 'softmax'
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy'

```
history = model.fit(X_train,y_train_encoded,
                    epochs=10,
                    verbose=False,
                    validation_data=(X_test, y_test_encoded),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train_encoded, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test_encoded, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่ 69 แสดงการเทรนโมเดล Deep Learning ด้วย Convolutional Neural Network

- history = model.fit(...): เทรนด้วยข้อมูลฝึก X_train และ y_train_encoded โดยใช้ฟังก์ชัน fit
- Epochs = 10 ระบุจำนวนรอบการเทรนทั้งหมดที่โมเดลจะถูกฝึก เท่ากับ 10
- verbose=False ระบุให้ไม่แสดงผลลัพธ์ทุกรอบการเทรน
- validation_data=(X_test, y_test_encoded) ระบุชุดข้อมูลทดสอบ เพื่อทำการตรวจสอบประสิทธิภาพของโมเดลบนชุดทดสอบขณะเทรน
- batch_size=10 ระบุขนาดของแบทช์ (batch) ที่จะถูกใช้ในการเทรน เท่ากับ 10
- ทดสอบด้วยข้อมูลฝึก X_train และ y_train_encoded ด้วยฟังก์ชัน evaluate

3.16.3 Recurrent Neural Network (RNN)

```
embedding_dim = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Bidirectional(layers.LSTM(128)))
model.add(layers.Dense(32,activation='relu'))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dense(10,activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

print(model.summary())
```

ภาพที่ 70 แสดงการสร้างโมเดล Deep Learning ด้วย Recurrent Neural Network

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด embedding_dim เท่ากับ 100
- LSTM(128) สร้าง LSTM ที่มี 128 โหนด (units)
- Dense layer ที่มี 32 โหนดและ activation function เป็น 'relu'
- Dense layer ที่มี 16 โหนดและ activation function เป็น 'relu'
- Dense layer ที่มี 10 โหนดและ activation function เป็น 'softmax'
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy'

```
history = model.fit(X_train, y_train_encoded,
                    epochs=10,
                    verbose=False,
                    validation_data=(X_test, y_test_encoded),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train_encoded, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test_encoded, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่ 71 แสดงการเทรนโมเดล Deep Learning ด้วย Recurrent Neural Network

- history = model.fit(...): เทรนด้วยข้อมูลฝึก X_train และ y_train_encoded โดยใช้ฟังก์ชัน fit
- Epochs = 10 ระบุจำนวนรอบการเทรนทั้งหมดที่โมเดลจะถูกฝึก เท่ากับ 10
- verbose=False ระบุให้ไม่แสดงผลลัพธ์ทุกรอบการเทรน
- validation_data=(X_test, y_test_encoded) ระบุชุดข้อมูลทดสอบ เพื่อทำการตรวจสอบประสิทธิภาพของโมเดลบนชุดทดสอบขณะเทรน
- batch_size=10 ระบุขนาดของแบทช์ (batch) ที่จะถูกใช้ในการเทรน เท่ากับ 10
- ทดสอบด้วยข้อมูลฝึก X_train และ y_train_encoded ด้วยฟังก์ชัน evaluate

3.16.4 Model with Dropout

```
embedding_dim = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Bidirectional(layers.LSTM(128)))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(32,activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(10,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

print(model.summary())
```

ภาพที่ 72 แสดงการสร้างโมเดล Deep Learning ด้วย Model with Dropout

- Embedding layer ถูกใช้เพื่อแปลงตัวเลข (indexes) ที่ได้จาก Tokenization เป็น dense vectors ขนาด embedding_dim เท่ากับ 100
- LSTM(128) สร้าง LSTM ที่มี 128 โหนด (units)
- Dropout(0.2) ถูกใช้เพื่อลด Overfitting โดยการสุ่มปิดบางโหนดในชั้นที่กำลังปรับปรุง
- Dense layer ที่มี 32 โหนดและ activation function เป็น 'relu'
- Dropout(0.2) ถูกใช้ในการลด Overfitting หลังจากชั้น Dense แรก
- Dense layer ที่มี 16 โหนดและ activation function เป็น 'relu'
- Dropout(0.2) ถูกใช้ในการลด Overfitting หลังจากชั้น Dense ที่สอง
- Dense layer ที่มี 10 โหนดและ activation function เป็น 'softmax'
- compile ด้วย optimizer เป็น 'adam', loss function เป็น 'sparse_categorical_crossentropy'

```
history = model.fit(X_train, y_train_encoded,
                    epochs=10,
                    verbose=False,
                    validation_data=(X_test, y_test_encoded),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train_encoded, verbose=False)
print("Training Accuracy: {:.3f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test_encoded, verbose=False)
print("Testing Accuracy:  {:.3f}".format(accuracy))
```

ภาพที่ 73 แสดงการเทรนโมเดล Deep Learning ด้วย Model with Dropout

- history = model.fit(...): เทรนด้วยข้อมูลฝึก X_train และ y_train_encoded โดยใช้ฟังก์ชัน fit
- Epochs = 10 ระบุจำนวนรอบการเทรนทั้งหมดที่โมเดลจะถูกฝึก เท่ากับ 10
- verbose=False ระบุให้ไม่แสดงผลลัพธ์ทุกรอบการเทรน
- validation_data=(X_test, y_test_encoded) ระบุชุดข้อมูลทดสอบ เพื่อทำการตรวจสอบประสิทธิภาพของโมเดลบนชุดทดสอบขณะเทรน
- batch_size=10 ระบุขนาดของแบทช์ (batch) ที่จะถูกใช้ในการเทรน เท่ากับ 10
- ทดสอบด้วยข้อมูลฝึก X_train และ y_train_encoded ด้วยฟังก์ชัน evaluate

ผลการทดลอง และการอภิปรายผล

สภาพแวดล้อมในการทดลอง

ข้อมูลที่ใช้คือข้อมูลที่เป็นข้อความจากการทวิต รวบรวมมาจาก แอปพลิเคชัน X (เอ็กซ์) จำนวน 4,000 ชุดข้อมูล โดยมี 5 คีย์เวิร์ด ได้แก่ น้ำท่วม, ไฟไหม้, ระเบิด, สงคราม และแผ่นดินไหว และแยกประเภทของข้อความเป็น 2 ประเภท ได้แก่ 0 และ 1 “ 0 หมายถึง ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย” และ “1 หมายถึง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย” และจะแยกข้อความในแต่ละประเภทออกเป็นหมวดหมู่ ทั้งหมด 5 หมวดหมู่ ได้แก่ หมวดหมู่ทั่วไป, หมวดหมู่น้ำท่วม, หมวดหมู่ระเบิด, หมวดหมู่สงคราม และหมวดหมู่ไฟไหม้ มาประเมินโมเดลจากค่าความถูกต้อง

ผลการทดลองที่ได้รับ

4.1 **การแสดง Word Cloud**

4.1.1 ภาพ word cloud ของชุดข้อมูลทั้งหมด

ภาพที่ 74 ผลลัพธ์ word cloud ของ keyword น้ำท่วม

4.1.2 ภาพ word cloud ของ keyword น้ำท่วม

ภาพที่ 75 ผลลัพธ์ word cloud ของ keyword น้ำท่วม

4.1.3 ภาพ word cloud ของ keyword น้ำท่วม ที่ข้อความเป็น 1 (1=ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

ภาพที่ 76 ผลลัพธ์ word cloud ของ keyword น้ำท่วม ที่ข้อความเป็น 1

4.1.4 ภาพ word cloud ของ keyword น้ำท่วม ที่ข้อความเป็น 0 (0=ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

น้ำท่วม
ออ
ออ
แม่
น้ำท่วมปาก
น้อง
ร้องไห้
คน
พ่อ
เหมือน
แน่
น้ำท่วม ปอด
ลูก
ตอน
เจอ
ใส่ ขาย
ชอบ
แฟน
ภาวะ
สวย
ตา
ตาย
ผักบุ้ง
หัวใจ
ฝนตก
เพลง
แบบนี้
หมอ
ดู
ติด
แตก
ฮือ
ฮะ
ปี
อะ
จบ
ไม่ต้อง
นะคะ
คล้อง
ทำ
ไดน
ม้า
ล้าน
นุ
ตก
หนู
ตาย
ไม่ต้อง
ไม่ต้อง
ไม่ต้อง

4.1.5 ภาพ word cloud ของ keyword ไฟไหม้

ไฟไหม้
ข่าว
ช่อง
ปัน
งาน
เสียหาย
ไฟไหม้ บ้าน
ไฟไหม้ โรงงาน
งาน ไฟไหม้
วอด
ดด
ไลน์
ย่าน
รถ
ข่าว อาชญากรรม
อาชญากรรม ไฟไหม้
ลิก
คมชัด
ดับ
ไฟไหม้ อาคาร
ไฟไหม้ โกดัง
ไฟไหม้ บ้านเรือน
ไฟไหม้ ชุมชน
ไฟไหม้ ตลาด
ไฟไหม้ ลอนดอน
ไฟไหม้ ไฟฟ้าลัดวงจร
ไฟไหม้ ประชาชน
ไฟไหม้ ไฟฟ้าลัดวงจร
ไฟไหม้ ชุมชน
ไฟไหม้ บ้านเรือน
ไฟไหม้ ระทึก
ไฟไหม้ ลิก ข่าว
ไฟไหม้ ลิก
ไฟไหม้ ดับ
ไฟไหม้ ไฟ
ไฟไหม้ ช่อง
ไฟไหม้ ย่าน
ไฟไหม้ ไลน์
ไฟไหม้ ดด
ไฟไหม้ เดด
ไฟไหม้ ปัน
ไฟไหม้ งาน
ไฟไหม้ เสียหาย
ไฟไหม้ วอด
ไฟไหม้ ทำ
ไฟไหม้ ดวน
ไฟไหม้ ไฟ
ไฟไหม้ ลิก

จากผลการทดลอง การแสดง word cloud ของ keyword ไฟไหม้พบว่าคำที่เกิดขึ้นบ่อย ได้แก่ ไฟไหม้, ข่าว, ช่อง, วอด, โรงงาน, งาน, คมชัด และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย แสดงใน Word Cloud ได้แก่ ไฟไหม้, ข่าว, ช่อง, วอด, บ้าน, เสียหาย และอื่น ๆ และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า คำที่แสดงใน Word Cloud ได้แก่ ปั่น, งาน, เดด, ไลน์ และอื่น ๆ เป็นต้น เห็นได้ชัดว่าข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นปัญหาไฟไหม้อาคาร บ้านเรือน หรือโรงงาน และการรายงานข่าวต่าง ๆ เช่น “ไฟไหม้แผงร้านค้าตลาดนัดสวนจตุจักร” แต่ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำเหล่านี้ไปในทางประโยคบอกเล่า เช่น “ทำงานที่นี่มาจะ 5 ปี ก็งานไฟไหม้ตลอด ไม่เคยมีสั่งล่วงหน้า”(ตัวอย่างประโยค ให้เอามาจากไฟล์ csv)

4.1.8 ภาพ word cloud ของ keyword ระเบิด

ภาพที่ 81 ผลลัพธ์ word cloud ของ keyword ระเบิด

4.1.9 ภาพ word cloud ของ keyword ระเบิด ที่ข้อความเป็น 1 (1=ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

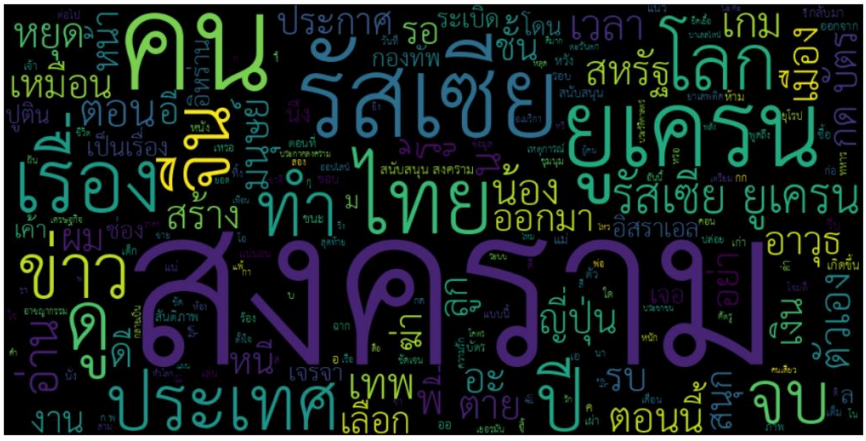
ภาพที่ 82 ผลลัพธ์ word cloud ของ keyword ระเบิด ที่ข้อความเป็น 1

4.1.10 ภาพ word cloud ของ keyword ระเบิด ที่ข้อความเป็น 0 (0=ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

ภาพที่ 83 ผลลัพธ์ word cloud ของ keyword ระเบิดที่ข้อความเป็น 0

จากผลการทดลอง การแสดง word cloud ของ keywords ระเบิดพบว่าคำที่เกิดขึ้นบ่อย ได้แก่ ระเบิด, ยูเครน, รัสเซีย, ตาย, คน และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย แสดงใน Word Cloud ได้แก่ ยูเครน, ระเบิด, รัสเซีย, ทิ้งระเบิด, โจมตี และอื่น ๆ และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า คำที่แสดงใน Word Cloud ได้แก่ ระเบิด, คน, เพลง, น้อง, หล่อ และอื่น ๆ เป็นต้น เห็นได้ชัดว่าข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นเหตุระเบิด ณ พื้นที่ ประเทศต่าง ๆ และผลกระทบของเหตุระเบิด เช่น “เกิดเหตุระเบิดหลายครั้งในภูมิภาค Kyiv ชั่วโมงที่ผ่านมา” แต่ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำเหล่านี้ไปในทางประโยคบอกเล่า เช่น “มุงกุอยู่ไม่ไหวหรอกนะ ซึงกวานใน ver นี้ กุรับไม่ไหวนะ ใจจะระเบิด”

4.1.11 ภาพ word cloud ของ keyword สงคราม

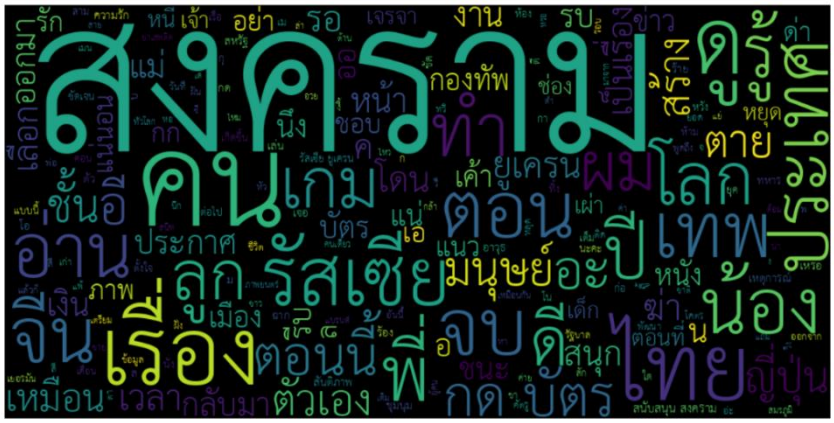


ภาพที่ 84 ผลลัพธ์ word cloud ของ keyword สงคราม

4.1.12 ภาพ word cloud ของ keyword สงคราม ที่ข้อความเป็น 1 (1=ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)

ภาพที่ 85 ผลลัพธ์ word cloud ของ keyword สงคราม ที่ข้อความเป็น 1

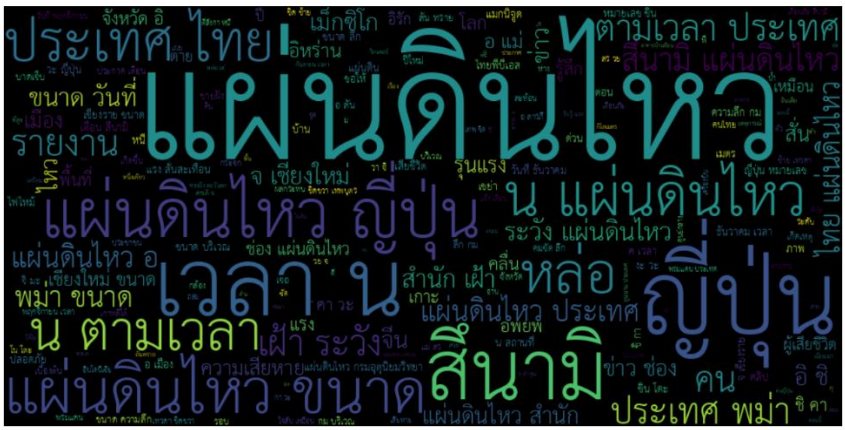
4.1.13 ภาพ word cloud ของ keyword สงครามที่ข้อความเป็น 0 (0=ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)



ภาพที่ 86 ผลลัพธ์ word cloud ของ keyword สงคราม ที่ข้อความเป็น 0

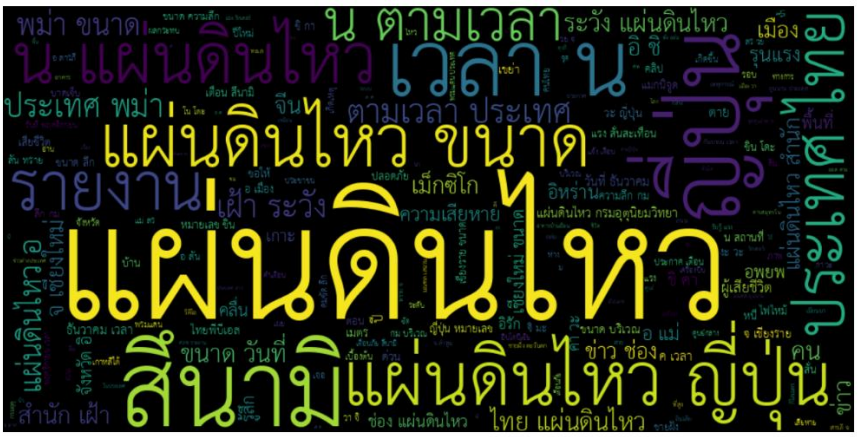
จากผลการทดลอง การแสดง word cloud ของ keywords สงครามพบว่าคำที่เกิดขึ้นบ่อย ได้แก่ สงคราม, ยูเครน, รัสเซีย, คน, จีน, ประเทศ และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย แสดงใน Word Cloud ได้แก่ สงคราม, ยูเครน, รัสเซีย, อิสราเอล, อิหร่าน, ปูติน และอื่น ๆ และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า คำที่แสดงใน Word Cloud ได้แก่ สงคราม, เรื่อง, คน, เกม, เทพ และอื่น ๆ เป็นต้น เห็นได้ชัดว่าข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นเหตุการณ์ที่เกี่ยวข้องกับสงครามในประเทศต่าง ๆ และผลกระทบของสงคราม เช่น “สหประชาชาติหวั่นสงครามในฉนวนกาซาลุกลามบานปลายกลายเป็นสงครามภูมิภาคหลังอิสราเอลโจมตีเบรุตการเกิดระเบิดครั้งใหญ่ในอิหร่าน” แต่ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำเหล่านี้ไปในทางประโยคบอกเล่า เช่น “บัตร 35000 ราคาอันนี้สงสัยไปรบในสงครามมา ได้ขายทีหวังรวย”

4.1.14 ภาพ word cloud ของ keyword แผ่นดินไหว



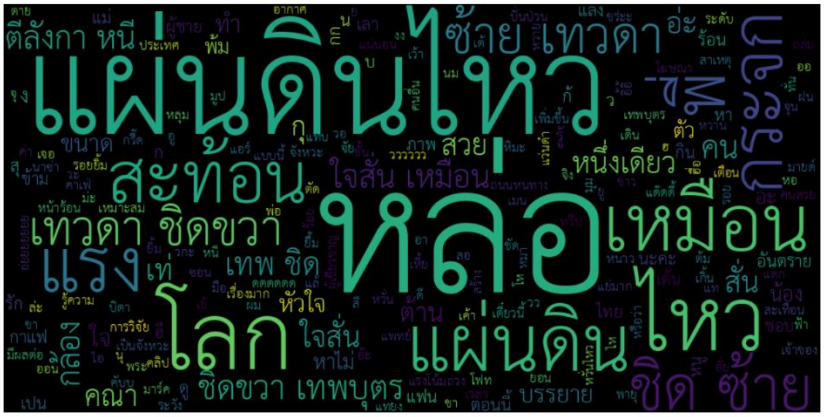
ภาพที่ 87 ผลลัพธ์ word cloud ของ keyword แผ่นดินไหว

4.1.15 ภาพ word cloud ของ keyword แผ่นดินไหว ที่ข้อความเป็น 1 (1=ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)



ภาพที่ 88 ผลลัพธ์ word cloud ของ keyword แผ่นดินไหว ที่ข้อความเป็น 1

4.1.16 ภาพ word cloud ของ keyword แผ่นดินไหว ที่ข้อความเป็น 0 (0=ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย)



ภาพที่ 89 ผลลัพธ์ word cloud ของ keyword แผ่นดินไหว ที่ข้อความเป็น 0

จากผลการทดลอง การแสดง word cloud ของ keyword แผ่นดินไหวพบว่าคำที่เกิดขึ้นบ่อย ได้แก่ แผ่นดินไหว, ญี่ปุ่น, สึนามิ, ขนาด, ตามเวลา, เวลา และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย แสดงใน Word Cloud ได้แก่ แผ่นดินไหว, ญี่ปุ่น, สึนามิ, ขนาด, ตามเวลา, รายงาน และอื่น ๆ และข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยพบว่า คำที่แสดงใน Word Cloud ได้แก่ แผ่นดินไหว, หล่อ, โลก, สะท้อน, เทวดา และอื่น ๆ เป็นต้น เห็นได้ชัดว่า ข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะการรายงานเหตุการณ์แผ่นดินไหว ระดับความรุนแรง และสถานที่เกิด เช่น “อาคารหลายหลังพังทลายลง ณ ใจกลางเมืองวาจิมะ จังหวัดอิชิคาวะ หลังเกิดแผ่นดินไหวรุนแรงขึ้นในพื้นที่ในช่วงเช้าของวันนี้” แต่ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยจะใช้คำเหล่านี้ไปในทางประโยคบอกเล่า เช่น “พี่ซอนอูกับแว่นดำคือแบบหล่อโลกสะเทือน แผ่นดินไหวแต่กุไม่ไหวค่า”(ตัวอย่างประโยค ให้เอามาจากไฟล์ csv)

4.2 ผลการทดลองการTrain Model ด้วยการสกัด CountVectorizer (สำหรับการ Detect label)

4.2.1 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
```

ภาพที่ 90 แสดงผลการเทรนโมเดล Logistic Regression ด้วยการสกัด CountVectorizer

4.2.2 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
```

ภาพที่ 91 แสดงผลการเทรนโมเดล Random Forest ด้วยการสกัด CountVectorizer

4.2.3 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
```

ภาพที่ 92 แสดงผลการเทรนโมเดล Decision Tree ด้วยการสกัด CountVectorizer

4.2.4 K-Nearest Neighbors

```
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
```

ภาพที่ 93 แสดงผลการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด CountVectorizer

4.3 ผลการทดลองการTrain Model ด้วยการสกัด TfidfVectorizer (สำหรับการ Detect label)

4.3.1 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
```

ภาพที่ 94 แสดงผลการเทรนโมเดล Logistic Regression ด้วยการสกัด TfidfVectorizer

4.3.2 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
```

ภาพที่ 95 แสดงผลการเทรนโมเดล Random Forest ด้วยการสกัด TfidfVectorizer

4.3.3 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
```

ภาพที่ 96แสดงผลการเทรนโมเดล Decision Tree ด้วยการสกัด TfidfVectorizer

4.3.4 K-Nearest Neighbors

```
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
```

ภาพที่ 97 แสดงผลการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด TfidfVectorizer

4.4 ผลการทดลองการ Train Model ด้วยการสกัด CountVectorizer (สำหรับการ Detect category)

4.4.1 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
```

ภาพที่ 98 แสดงผลการเทรนโมเดล Logistic Regression ด้วยการสกัด CountVectorizer

4.4.2 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
```

ภาพที่ 99 แสดงผลการเทรนโมเดล Random Forest ด้วยการสกัด CountVectorizer

4.4.3 Decision Tree

```
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
```

ภาพที่100 แสดงผลการเทรนโมเดล Decision Tree ด้วยการสกัด CountVectorizer

4.4.4 K-Nearest Neighbors

```
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
```

ภาพที่ 101 แสดงผลการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด CountVectorizer

4.5 ผลการทดลองการ Train Model ด้วยการสกัด TfidfVectorizer (สำหรับการ Detect category))

4.5.1 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
```

ภาพที่ 102 แสดงผลการเทรนโมเดล Logistic Regression ด้วยการสกัด TfidfVectorizer

4.5.2 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
```

ภาพที่ 103 แสดงผลการเทรนโมเดล Random Forest ด้วยการสกัด TfidfVectorizer

4.5.3 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
```

ภาพที่ 104 แสดงผลการเทรนโมเดล Decision Tree ด้วยการสกัด TfidfVectorizer

4.5.4 K-Nearest Neighbors

```
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
```

ภาพที่ 105 แสดงผลการเทรนโมเดล K-Nearest Neighbors ด้วยการสกัด TfidfVectorizer

4.6 ผลการทดลอง Deep Learning (สำหรับการ Detect label)

4.6.1 Neural Network Model (NN model)

```
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
```

ภาพที่ 106 แสดงผลการเทรนโมเดล NN model - สำหรับการ Detect label

4.6.2 Convolutional Neural Network Model (CNN model)

```
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
```

ภาพที่ 107 แสดงผลการเทรนโมเดล CNN model - สำหรับการ Detect label

4.6.3 Recurrent Neural Network Model (RNN model)

```
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
```

ภาพที่ 108 แสดงผลการเทรนโมเดล RNN model - สำหรับการ Detect label

4.6.4 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
```

ภาพที่ 109 แสดงผลการเทรนโมเดล Model with dropout - สำหรับการ Detect label

4.7 ผลการทดลอง Deep Learning (สำหรับการ Detect category)

4.7.1 Neural Network Model (NN model)

```
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
```

ภาพที่ 110 แสดงผลการเทรนโมเดล NN model - สำหรับการ Detect category

4.7.2 Convolutional Neural Network Model (CNN model)

```
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
```

ภาพที่ 111 แสดงผลการเทรนโมเดล CNN model - สำหรับการ Detect category

4.7.3 Recurrent Neural Network Model (RNN model)

```
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
```

ภาพที่ 112 แสดงผลการเทรนโมเดล RNN model - สำหรับการ Detect category

4.7.4 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
```

ภาพที่ 113 แสดงผลการเทรนโมเดล Model with dropout - สำหรับการ Detect category

4.8 ผลการทดลองค่าประสิทธิภาพ Model ในรูปแบบตาราง สำหรับการ Detect label

ตารางที่ 1 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

4.8 ผลการทดลองค่าประสิทธิภาพ Model ในรูปแบบตาราง สำหรับการ Detect category

ตารางที่ 2 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

4.9 ผลการทดสอบ Model สำหรับการ Detect label

4.9.1 ผลการทดสอบ Model ที่มีคีย์เวิร์ด คือ 'น้ำท่วม'

```
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')
```

```
ความรักก์เหมือนผักบุ้งพอน้ำท่วมทุ่ง มันก็โหรงเหรงว่ะ
ความรักก์เหมือนผักบุ้งพอน้ำท่วมทุ่ง มันก็โหรงเหรงว่ะ
ความรัก ก์ เหมือน ผักบุ้ง พอ น้ำท่วม ทุ่ง มัน ก็ โหรงเหรง ว่ะ
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 114 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'น้ำท่วม'

```
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')
```

```
#อีสเราเอล วางท่อน้ำจากทะเลหันน้ำทะเลเข้าอุโมงค์ของ #ฮามาสใน #กาซ่า เพื่อให้อุโมงค์น้ำท่วมขับพวกฮามาสออกมาเหมือนล้างท่อ ล้างส้วม ล้างสิ่งปฏิกูล คิดได้เนาะ ไม่ใช้อาวุธ ใช้ธรรมชาติแม่งเลย
 อีสเราเอล วางท่อน้ำจากทะเลหันน้ำทะเลเข้าอุโมงค์ของ ฮามาสใน กาซ่า เพื่อให้อุโมงค์น้ำท่วมขับพวกฮามาสออกมาเหมือนล้างท่อ ล้างส้วม ล้างสิ่งปฏิกูล คิดได้เนาะ ไม่ใช้อาวุธ ใช้ธรรมชาติแม่งเลย
อีส เรา เอ ล วาง ท่อน้ำ จาก ทะเล หัน น้ำทะเล เข้า อุโมงค์ ของ ฮามาส ใน กา ซ่า เพื่อให้ อุโมงค์ น้ำท่วม ขับ พวก ฮามาส ออกมา เหมือน ล้าง ท่อ ล้าง ส้วม ล้าง สิ่งปฏิกูล คิดได้ เนาะ ไม่ ใช้ อาวุธ ใช้ ธรรมชาติ แม่ ง เลย
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 115 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'น้ำท่วม'

4.9.2 ผลการทดสอบ Model ที่มีคีย์เวิร์ด คือ 'ระเบิด'

```
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')
```

```
ทำไมรู้สึกอินกับฉากนี้มาก มะนาวคงเก็บความรู้สึกไว้เพื่อที่จะเข้าใจและจะไม่ทำให้พี่เกี่ยวไม่สบายใจจนถึงจุดระเบิดเลยทำให้ทะเลาะกันนักแสดงแสดงออกมาได้ดีมาก อันดาเล่นให้เรารู้สึกเจ็บปวดไปด้วย ส่วนลูกแก้วก็ทำมันได้ดีจิงๆ อินกับทั้งสองคน #LoveseniorEP8
ทำไมรู้สึกอินกับฉากนี้มาก มะนาวคงเก็บความรู้สึกไว้เพื่อที่จะเข้าใจและจะไม่ทำให้พี่เกี่ยวไม่สบายใจจนถึงจุดระเบิดเลยทำให้ทะเลาะกันนักแสดงแสดงออกมาได้ดีมาก อันดาเล่นให้เรารู้สึกเจ็บปวดไปด้วย ส่วนลูกแก้วก็ทำมันได้ดีจิง อินกับทั้งสองคน LoveseniorEP8
ทำไม รู้สึก อิน กับ ฉาก นี้ มาก มะนาว คง เก็บ ความรู้สึก ไว้ เพื่อที่จะ เข้าใจ และ จะ ไม่ ทำให้ พี่ เกี่ยว ไม่ สบายใจ จนถึง จุด ระเบิด เลย ทำให้ ทะเลาะ กัน นักแสดง แสดง ออกมา ได้ ดีมาก อัน ดา เล่น ให้ เรา รู้สึก เจ็บปวด ไป ด้วย ส่วน
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 116 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'ระเบิด'

```
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')
```

```
วีดีโอตัวเต็มของฮามาสที่โดนระเบิดของ IDF ได้สวดภาวนาก่อนที่เขาจะเสียชีวิตในพื้นที่ KhanYunis
วีดีโอตัวเต็มของฮามาสที่โดนระเบิดของ IDF ได้สวดภาวนาก่อนที่เขาจะเสียชีวิตในพื้นที่ KhanYunis
วีดีโอ ตัว เต็ม ของ ฮามาส ที่ โดน ระเบิด ของ IDF ได้ สวด ภาวนา ก่อนที่ เขา จะ เสียชีวิต ใน พื้นที่ KhanYunis
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 117 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'ระเบิด'

4.9.3 ผลการทดสอบ Model ที่มีคีย์เวิร์ด คือ 'สงคราม'

```
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('ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)')
print('ผลลัพธ์ = ',result)

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

```
25 นาที ยอดไลค์สามพันห้า สงครามกดบัตรอันใหญ่หลวง #เนเน่
25 นาที ยอดไลค์สามพันห้า สงครามกดบัตรอันใหญ่หลวง  เนเน่
25  นาที  ยอด  ไลค์  สาม  พัน  ห้า  สงคราม  กด  บัตร  อัน  ใหญ่หลวง  เนเน่
[[0 0 0 ... 0 0 0]]
ข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ ? (โดยที่ 0 = ไม่เกี่ยวข้อง, 1 = เกี่ยวข้อง)
ผลลัพธ์ =  [0]
คำอธิบาย : ข้อความนี้พูดถึงเหตุการณ์ที่ไม่เกี่ยวข้องกับเหตุการณ์สาธารณภัย
```

ภาพที่ 118 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'สงคราม'

```
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')
```

```
อิสราเอล โจมตีโดรน! ใส่อิหร่านบุกปาเลสไตน์ เดือดลุกเป็นไฟ สื่อรายงานว่าอิสราเอลโดยการสนับสนุนจากลุกที่ขาใหญ่เปิดสงครามโจมตีอิหร่านด้วยอากาศยานโดรนติดอาวุธ หวังทำลายอุตสาหกรรมทางทหารของอิหร่านที่ศูนย์ผลิตกระสุนกลาโหม โรงงานโดรน คลังสิ
อิสราเอล โจมตีโดรน ใส่อิหร่านบุกปาเลสไตน์ เดือดลุกเป็นไฟ สื่อรายงานว่าอิสราเอลโดยการสนับสนุนจากลุกที่ขาใหญ่เปิดสงครามโจมตีอิหร่านด้วยอากาศยานโดรนติดอาวุธ หวังทำลายอุตสาหกรรมทางทหารของอิหร่านที่ศูนย์ผลิตกระสุนกลาโหม โรงงานโดรน คลังสิ
อิสราเอล โจมตี โดรน ใส่ อิหร่าน บุก ปาเลสไตน์ เดือด ลุกเป็นไฟ สื่อ รายงาน ว่า อิสราเอล โดย การ สนับสนุน จาก ลุกที่ ขาใหญ่ เปิด สงคราม โจมตี อิหร่าน ด้วย อากาศยาน โดรน ติดอาวุธ หวัง ทำลาย อุตสาหกรรม ทางทหาร ของ อิหร่าน
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 119 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'สงคราม'

4.9.4 ผลการทดสอบ Model ที่มีคีย์เวิร์ด คือ 'แผ่นดินไหว'

```
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')
```

```
ไอเหี้ย มือกุสั่นแรงยิ่งกว่าแผ่นดินไหว พี่แทแม่งหล่อมาก เอากุไปดัมเดี๋ยวนี้ ตอนนี้ ไม่ไหวแล้ว
ไอเหี้ย มือกุสั่นแรงยิ่งกว่าแผ่นดินไหว พี่แทแม่งหล่อมาก เอากุไปดัมเดี๋ยวนี้ ตอนนี้ ไม่ไหวแล้ว
ไอ เหี้ย มือ กุ สั่น แรง ยิ่งกว่า แผ่นดินไหว พี่ แท แม่ ง หล่อ มาก เอา กุ ไป ดัม เดี๋ยวนี้ ตอนนี้ ไม่ ไหว แล้ว
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 120 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'แผ่นดินไหว'

```
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
```

ภาพที่ 121 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'แผ่นดินไหว'

4.9.5 ผลการทดสอบ Model ที่มีคีย์เวิร์ด คือ 'ไฟไหม้'

```
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')
```

```
งานไฟไหม้ มาพร้อมกับคนไฟลุก (คนฮอต) อย่ามองแบบนั้นค่ะลูก แค่นี้ก็หลงจะตายแล้วว #เนเน่xทททท Nene Amazing Thailand 2024
งานไฟไหม้ มาพร้อมกับคนไฟลุก คนฮอต อย่ามองแบบนั้นค่ะลูก แค่นี้ก็หลงจะตายแล้วว เนเน่xทททท Nene Amazing Thailand 2024
งาน ไฟไหม้ มา พร้อมกับ คน ไฟ ลุก คน ฮอต อย่า มอง แบบ นั้น ค่ะ ลูก แค่นี้ ก็ หลง จะ ตาย แล้ว ว เนเน่ x ทท ท Nene Amazing Thailand 2024
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 122 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'ไฟไหม้'

```
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')
```

```
อาคารที่ไฟไหม้เป็นอาคารคอนกรีตเสริมเหล็ก เมื่อไฟไหมคอนกรีตจะร่อนเหล็กสัมผัสกับไฟโดยตรงทำให้แรงรับน้ำหนักน้อยลง/นายกวิศวฯ #PPTVHd36 #ไฟไหม้
อาคารที่ไฟไหม้เป็นอาคารคอนกรีตเสริมเหล็ก เมื่อไฟไหมคอนกรีตจะร่อนเหล็กสัมผัสกับไฟโดยตรงทำให้แรงรับน้ำหนักน้อยลงนายกวิศว PPTVHd36 ไฟไหม้
อาคาร ที่ ไฟไหม้ เป็น อาคาร คอนกรีตเสริมเหล็ก เมื่อ ไฟ ไหม คอนกรีต จะ ร่อน เหล็ก สัมผัส กับ ไฟ โดยตรง ทำ ให้แรง รับ น้ำหนัก น้อยลง นายก วิศว PPTVHd 36 ไฟไหม้
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 123 ผลการทดสอบโมเดล Detect label ที่มีคีย์เวิร์ด คือ 'ไฟไหม้'

4.10 ผลการทดสอบ Model สำหรับการ Detect category

4.10.1 ผลการทดสอบ Model หมวดหมู่ทั่วไป

```
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])
```

```
ทำไมพัสดุยังไม่มาส่งอีกวะ สั่งไปตั้งสองนาทีแล้ว
ทำไมพัสดุยังไม่มาส่งอีกวะ สั่งไปตั้งสองนาทีแล้ว
ทำไม พัสดุ ยัง ไม่ มา ส่ง อีก วะ สั่ง ไป ตั้ง สอง นาที แล้ว
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 124 ผลการทดสอบโมเดล

4.10.2 ผลการทดสอบ Model หมวดหมู่ทั่วไป ที่ประโยคมีคำคีย์เวิร์ด คือ 'น้ำท่วม'

```
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])
```

```
โยกเยกเอย น้ำท่วมเมฆกระด่ายลอยคอ...
โยกเยกเอย น้ำท่วมเมฆกระด่ายลอยคอ...
โยกเยก เอย น้ำท่วม เมฆ กระด่าย ลอยคอ ...
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 125 ผลการทดสอบโมเดล

4.10.3 ผลการทดสอบ Model หมวดหมู่ทั่วไป ที่ประโยคมีคำคีย์เวิร์ด คือ 'ระเบิด'

```
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])
```

```
มาแค่คลิปสั้นๆก็รู้แล้วว่าต้องออกมาดีมากแน่ๆ พูห์พาเวลเคมีระเบิดระเบ้อจริง #poohpavel #พูห์พาเวล
มาแค่คลิปสั้นก็รู้แล้วว่าต้องออกมาดีมากแน่ พูห์พาเวลเคมีระเบิดระเบ้อจริง  poohpavel  พูห์พาเวล
มา แค่ คลิป สั้น ก็ รู้ แล้ว ว่า ต้อง ออกมา ดีมาก แน่ พูห์ พา เวล เคมี ระเบิดระเบ้อ จริง poohpavel พูห์ พา เวล
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 126 ผลการทดสอบโมเดล

4.10.4 ผลการทดสอบ Model หมวดหมู่ทั่วไป ที่ประโยคมีคำคีย์เวิร์ด คือ 'สงคราม'

```
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])
```

```
พลังความรักเป็นอะไรที่แสนยิ่งใหญ่ มันเกินคำว่ารักไปมากกล้าพูดเลยว่าถ้ามีมีตคู่เกิดขึ้น สงครามต้อมแน่นอนนนนนน #หวานใจมิวกลัฟ #WaanjaiMewGulf
พลังความรักเป็นอะไรที่แสนยิ่งใหญ่ มันเกินคำว่ารักไปมากกล้าพูดเลยว่าถ้ามีมีตคู่เกิดขึ้น สงครามต้อมแน่นอนนนนนน  หวานใจมิวกลัฟ  WaanjaiMewGulf
พลัง ความรัก เป็น อะไร ที่ แสน ยิ่งใหญ่ มัน เกิน คำ ว่า รัก ไป มาก กล้า พูด เลย ว่า ถ้า มี มีต คู่ เกิดขึ้น สงคราม ต้อม แน่นอน นนนนน หวานใจ มิ วก ลัฟ WaanjaiMewGulf
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 127 ผลการทดสอบโมเดล

4.10.5 ผลการทดสอบ Model หมวดหมู่ทั่วไป ที่ประโยคมีคำคีย์เวิร์ด คือ 'แผ่นดินไหว'

```
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])
```

```
แผ่นดินไหวแต่เราอ่ะไม่ไหว ดูความพระเจ้าของผู้ชายคนนี้เถอะ หล่อเกินเรื่องมาก ตายๆๆๆๆ ตัวพ่อ ตัวบิดา ตัวแด๊ดดี้สุดๆๆ
แผ่นดินไหวแต่เราอ่ะไม่ไหว ดูความพระเจ้าของผู้ชายคนนี้เถอะ หล่อเกินเรื่องมาก ตาย ตัวพ่อ ตัวบิดา ตัวแด๊ดดี้สุด
แผ่นดินไหว แต่ เรา อ่ะ ไม่ ไหว ดู ความ พระ เจ้าของ ผู้ชาย คน นี้ เถอะ หล่อ เกิน เรื่องมาก ตาย ตัว พ่อ ตัว บิดา ตัว แด๊ดดี้ สุด
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 128 ผลการทดสอบโมเดล

4.10.6 ผลการทดสอบ Model หมวดหมู่ทั่วไป ที่ประโยคมีคำคีย์เวิร์ด คือ 'ไฟไหม้'

```
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])
```

```
กลิ่นไฟไหม้เพราะเดตไลน์ ไม่ใช่ โอ้ กลิ่นไหม้นี่ยอดฮิตตามกลิ่นไม้กับธูปมาติดๆเลยฮะ น่าสนใจ
กลิ่นไฟไหม้เพราะเดตไลน์ ไม่ใช่ โอ้ กลิ่นไหม้นี่ยอดฮิตตามกลิ่นไม้กับธูปมาติดเลยฮะ น่าสนใจ
กลิ่น ไฟไหม้ เพราะ เด ต ไลน์ ไม่ ใช่ โอ้ กลิ่น ไหม้ นี่ ยอดฮิต ตาม กลิ่น ไม้ กับ ธูป มา ติด เลย ฮะ น่าสนใจ
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 129 ผลการทดสอบโมเดล

4.10.7 ผลการทดสอบ Model หมวดหมู่น้ำท่วม

```
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])
```

```
#อิสเราเอลวางท่อน้ำจากทะเลหันน้ำทะเลเข้าอุโมงค์ของ #ฮามาสใน #กาซ่า เพื่อให้อุโมงค์น้ำท่วมขับพวกฮามาสออกมาเหมือนล้างท่อ ล้างส้วม ล้างสิ่งปฏิกูล คิดได้เนาะไม่ใช้อาวุธใช้ธรรมชาติแม่งเลย
 อิสเราเอลวางท่อน้ำจากทะเลหันน้ำทะเลเข้าอุโมงค์ของ  ฮามาสใน  กาซ่า เพื่อให้อุโมงค์น้ำท่วมขับพวกฮามาสออกมาเหมือนล้างท่อ ล้างส้วม ล้างสิ่งปฏิกูล คิดได้เนาะไม่ใช้อาวุธใช้ธรรมชาติแม่งเลย
อิส เรา เอ ล วาง ท่อน้ำ จาก ทะเล หัน น้ำทะเล เข้า อุโมงค์ ของ ฮามาส ใน กา ซ่า เพื่อให้ อุโมงค์ น้ำท่วม ขับ พวก ฮามาส ออกมา เหมือน ล้าง ท่อ ล้าง ส้วม ล้าง สิ่งปฏิกูล คิดได้ เนาะ ไม่ ใช้ อาวุธ ใช้ ธรรมชาติ แม่ ง เลย
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 130 ผลการทดสอบโมเดล

4.10.8 ผลการทดสอบ Model หมวดหมู่ระเบิด

```
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])
```

```
ระเบิดทำลายระบบอุโมงค์และห้องใต้ดินที่ซ่อนของ Sinwar #อิสราเอล #Israel #Gaza #ปาเลสไตน์ #กลุ่มฮามาส #Hamas #ฉนวนกาซา #ฮามาส #สงครามอิสราเอล #อิสราเอลปาเลสไตน์ #อุโมงค์
ระเบิดทำลายระบบอุโมงค์และห้องใต้ดินที่ซ่อนของ Sinwar  อิสราเอล  Israel  Gaza  ปาเลสไตน์  กลุ่มฮามาส  Hamas  ฉนวนกาซา  ฮามาส  สงครามอิสราเอล  อิสราเอลปาเลสไตน์  อุโมงค์
ระเบิดทำลาย  ระบบ  อุโมงค์  และ  ห้องใต้ดิน  ที่ซ่อน  ของ  Sinwar  อิสราเอล  Israel  Gaza  ปาเลสไตน์  กลุ่ม  ฮามาส  Hamas  ฉนวน  กาซา  ฮามาส  สงคราม  อิสราเอล  อิสราเอล  ปาเลสไตน์  อุโมงค์
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 131 ผลการทดสอบโมเดล

4.10.9 ผลการทดสอบ Model หมวดหมู่สงคราม

```
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])
```

```
ยุโรปเสี่ยง Recession ผลกระทบจากสงครามทั้งในตะวันออกกลาง สงครามรัสเซียยูเครนไปจนถึงอิสราเลฮามาสที่กำลังดำเนิน และยังไม่มีท่าทีว่าจะจบลงง่ายๆ
ยุโรปเสี่ยง Recession ผลกระทบจากสงครามทั้งในตะวันออกกลาง สงครามรัสเซียยูเครนไปจนถึงอิสราเลฮามาสที่กำลังดำเนิน และยังไม่มีท่าทีว่าจะจบลงง่าย
ยุโรป เสี่ยง Recession ผลกระทบ จาก สงคราม ทั้ง ใน ตะวันออกกลาง สงคราม รัสเซีย ยูเครน ไป จนถึง อิส รา เล ฮามาส ที่ กำลัง ดำเนิน และ ยัง ไม่ มี ท่าที ว่า จะ จบ ลง ง่าย
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 132 ผลการทดสอบโมเดล

4.10.10 ผลการทดสอบ Model หมวดหมู่แผ่นดินไหว

```
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])
```

```
แผ่นดินไหวญี่ปุ่น 7.6 แมกนิจูด เกิดคลื่นยักษ์สึนามิสูง 1.2ม. ซัดเข้าชายฝั่ง #แผ่นดินไหว #ญี่ปุ่น #แผ่นดินไหวญี่ปุ่น #TNN #TNNThailand #TNNช่อง16 #TNNONLINE #TNNWorld
แผ่นดินไหวญี่ปุ่น 7.6 แมกนิจูด เกิดคลื่นยักษ์สึนามิสูง 1.2ม. ซัดเข้าชายฝั่ง แผ่นดินไหว ญี่ปุ่น แผ่นดินไหวญี่ปุ่น TNN TNNThailand TNNช่อง16 TNNONLINE TNNWorld
แผ่นดินไหว ญี่ปุ่น 7.6 แมกนิจูด เกิด คลื่น ยักษ์ สึนามิ สูง 1.2 ม. ซัด เข้า ชายฝั่ง แผ่นดินไหว ญี่ปุ่น แผ่นดินไหว ญี่ปุ่น TNN TNNThailand TNN ช่อง 16 TNNONLINE TNNWorld
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 133 ผลการทดสอบโมเดล

4.10.11 ผลการทดสอบ Model หมวดหมู่ไฟไหม้

```
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])
```

```
10.25 น. ไฟไหม้โรงงานสีทาบ้านย่านบางพลีเพลิงสงบแล้ว เหลือกลุ่มควันหลังคาอาคารทรุดตัวลงมา #ThaiPBS #ไฟไหม้
10.25 น. ไฟไหม้โรงงานสีทาบ้านย่านบางพลีเพลิงสงบแล้ว เหลือกลุ่มควันหลังคาอาคารทรุดตัวลงมา ThaiPBS ไฟไหม้
10.25 น. ไฟไหม้ โรงงาน สี ทา บ้าน ย่าน บาง พลี เพลิง สงบ แล้ว เหลือ กลุ่ม ควัน หลังคา อาคาร ทรุดตัว ลงมา ThaiPBS ไฟไหม้
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 134 ผลการทดสอบโมเดล

4.11 ผลการทดสอบทั้ง 2 Model พร้อมกัน (ทั้ง label และ category)

4.11.1 ผลการทดสอบทั้ง 2 Model พร้อมกัน โดยไม่มีคีย์เวิร์ด

```
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])
```

```
ทำไมพัสดุยังไม่มาส่งอีกวะ สั่งไปตั้งสองนาทีแล้ว
ทำไมพัสดุยังไม่มาส่งอีกวะ สั่งไปตั้งสองนาทีแล้ว
ทำไม พัสดุ ยัง ไม่ มา ส่ง อีก วะ สั่ง ไป ตั้ง สอง นาที แล้ว
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 135 ผลการทดสอบโมเดล

4.11.2 ผลการทดสอบทั้ง 2 Model พร้อมกัน โดยมีคีย์เวิร์ด คือ น้ำท่วม

```
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
```

ภาพที่ 136 ผลการทดสอบโมเดล

```
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
```

ภาพที่ 137 ผลการทดสอบโมเดล

4.11.3 ผลการทดสอบทั้ง 2 Model พร้อมกัน โดยมีคีย์เวิร์ด คือ ระเบิด

```
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])
```

```
น้องแบมใส่ชุดหลุยส์มาด้วยหล่อระเบิดมาก
น้องแบมใส่ชุดหลุยส์มาด้วยหล่อระเบิดมาก
น้อง แบ ม ใส่ ชุด หลุยส์ มา ด้วย หล่อ ระเบิด มาก
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 138 ผลการทดสอบโมเดล

```
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])
```

```
กษิดภิรมย์เปิดปมแค้น 30ปี นาโต้VSรัสเซียก่อนระเบิดในสมรภูมิยูเครน
กษิดภิรมย์เปิดปมแค้น 30ปี นาโต้VSรัสเซียก่อนระเบิดในสมรภูมิยูเครน
กษิด ภิรมย์ เปิด ปม แค้น 30 ปี นาโต้ VS รัสเซีย ก่อน ระเบิด ใน สมรภูมิ ยูเครน
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 139 ผลการทดสอบโมเดล

4.11.4 ผลการทดสอบทั้ง 2 Model พร้อมกัน โดยมีคีย์เวิร์ด คือ สงคราม

```
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])
```

```
25 นาที ยอดไลค์สามพันห้า สงครามกดบัตรอันใหญ่หลว
25 นาที ยอดไลค์สามพันห้า สงครามกดบัตรอันใหญ่หลว
25 นาที ยอด ไลค์ สาม พัน ห้า สงคราม กด บัตร อัน ใหญ่ หลว
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 140 ผลการทดสอบโมเดล

```
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])
```

```
หยุดสงครามทำยังไงหรอ รัสเซียหยุดยิง พิมมาเนี่ยยูเครนยิงรัสเซียหรอ?หรือจะให้ยูเครนยอมง่ายๆก่อนจะว่าคนสนับสนุนสงครามแยกให้ออกระหว่างสนับสนุนกับช่วยได้ก่อนปะ อยากจบสงครามรัสเซียถอนทหารออกง่ายกว่าปะ ไม่ต้องมาพิมร่ายสายยาย อย่าเพิ่ง
หยุดสงครามทำยังไงหรอ รัสเซียหยุดยิง พิมมาเนี่ยยูเครนยิงรัสเซียหรอหรือจะให้ยูเครนยอมง่ายก่อนจะว่าคนสนับสนุนสงครามแยกให้ออกระหว่างสนับสนุนกับช่วยได้ก่อนปะ อยากจบสงครามรัสเซียถอนทหารออกง่ายกว่าปะ ไม่ต้องมาพิมร่ายสายยาย อย่าเพิ่งม
หยุด สงคราม ทำ ยังไง หรอ รัสเซีย หยุด ยิง พิม มา เนี่ย ยูเครน ยิง รัสเซีย หรอ หรือ จะ ให้ ยูเครน ยอม ง่าย ก่อน จะ ว่า คน สนับสนุน สงคราม แยก ให้ ออก ระหว่าง สนับสนุน กับ ช่วย ได้ ก่อน ปะ อยาก จบ สงคราม รัสเซีย
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 141 ผลการทดสอบโมเดล

4.11.5 ผลการทดสอบทั้ง 2 Model พร้อมกัน โดยมีคีย์เวิร์ด คือ แผ่นดินไหว

```
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])
```

```
มึงพี่แทยงเค้าหล่อแบบเทพบุตรอ่ะ เอาดีๆ แผ่นดินไหวแต่กูไม่ไหวค่ะ
มึงพี่แทยงเค้าหล่อแบบเทพบุตรอ่ะ เอาดี แผ่นดินไหวแต่กูไม่ไหวค่ะ
มึง พี่ แทยง เค้า หล่อ แบบ เทพบุตร อ่ะ เอา ดี แผ่นดินไหว แต่ กู ไม่ ไหว ค่ะ
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 142 ผลการทดสอบโมเดล

```
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])
```

```
แผ่นดินไหวญี่ปุ่น 7.6 แมกนิจูด เกิดคลื่นยักษ์สึนามิสูง 1.2ม. ซัดเข้าชายฝั่ง #แผ่นดินไหว #ญี่ปุ่น #แผ่นดินไหวญี่ปุ่น #TNN #TNNThailand #TNNช่อง16 #TNNONLINE #TNNWorld
แผ่นดินไหวญี่ปุ่น 7.6 แมกนิจูด เกิดคลื่นยักษ์สึนามิสูง 1.2ม. ซัดเข้าชายฝั่ง แผ่นดินไหว ญี่ปุ่น แผ่นดินไหวญี่ปุ่น TNN TNNThailand TNNช่อง16 TNNONLINE TNNWorld
แผ่นดินไหว ญี่ปุ่น 7.6 แมกนิจูด เกิด คลื่น ยักษ์ สึนามิ สูง 1.2 ม. ซัด เข้า ชายฝั่ง แผ่นดินไหว ญี่ปุ่น แผ่นดินไหว ญี่ปุ่น TNN TNNThailand TNN ช่อง 16 TNNONLINE TNNWorld
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 143 ผลการทดสอบโมเดล

4.11.6 ผลการทดสอบทั้ง 2 Model พร้อมกัน โดยมีคีย์เวิร์ด คือ ไฟไหม้

```
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])
```

```
ชอบมากเดดไลน์ งานไฟเผา ไฟลุก ไฟไหม้ ไฟเพลิง ขับเคลื่อนชีวิตด้วยเดดไลน์
ชอบมากเดดไลน์ งานไฟเผา ไฟลุก ไฟไหม้ ไฟเพลิง ขับเคลื่อนชีวิตด้วยเดดไลน์
ชอบ มาก เด ด ไลน์ งาน ไฟ เผา ไฟ ลุก ไฟไหม้ ไฟ เพลิง ขับเคลื่อน ชีวิต ด้วย เด ด ไลน์
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 144 ผลการทดสอบโมเดล

```
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])
```

```
ไฟไหม้โรงงานโออิชิ พนักงานวิ่งหนีตาย
ไฟไหม้โรงงานโออิชิ พนักงานวิ่งหนีตาย
ไฟไหม้ โรงงาน โออิชิ พนักงาน วิ่งหนี ตาย
[[0 0 0 ... 0 0 0]]

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

ภาพที่ 145 ผลการทดสอบโมเดล

จากผลการทดลองการทดสอบโมเดลที่มีค่าความถูกต้องสูงที่สุด สำหรับการ Detect label คือโมเดล Logistic regression และ Decision tree มีค่าความถูกต้อง 92% พบว่าเมื่อป้อนข้อความ “ยอดไลค์สามพันห้า สงครามกดบัตรอันใหญ่หลวง” ผลลัพธ์ = 0 หมายถึง ข้อความที่ไม่ได้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย และเมื่อป้อนข้อความ “อิสราเอล โจมตีโดรน! ใส่อิหร่านบุกปาเลสไตน์ เดือดลุกเป็นไฟ สื่อรายงานว่าอิสราเอลโดยการสนับสนุนจากลูกพี่ขาใหญ่เปิดสงครามโจมตีอิหร่านด้วยอากาศยานโดรนติดอาวุธ” ผลลัพธ์ = 1 คือข้อความที่พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัย

จากผลการทดลองการทดสอบโมเดลที่มีค่าความถูกต้องสูงที่สุด สำหรับการ Detect category คือโมเดล Logistic regression มีค่าความถูกต้อง 92% พบว่าเมื่อป้อนข้อความ “ยอดไลค์สามพันห้า สงครามกดบัตรอันใหญ่หลวง” ผลลัพธ์ = ทั่วไป หมายถึง ข้อความนี้จัดอยู่ในหมวดหมู่ทั่วไป และเมื่อป้อนข้อความ

“อิสราเอล โจมตีโดรน! ใส่อิหร่านบุกปาเลสไตน์ เดือดลุกเป็นไฟ สื่อรายงานว่าอิสราเอลโดยการสนับสนุนจากลูกพี่ขาใหญ่เปิดสงครามโจมตีอิหร่านด้วยอากาศยานโดรนติดอาวุธ” ผลลัพธ์ = สงคราม คือข้อความนี้จัดอยู่ในหมวดหมู่สงคราม

การอภิปรายผล

จากผลการทดลองด้วยข้อมูลที่ใช้คือข้อมูลที่เป็นข้อความจากการทวิต รวบรวมมาจาก แอปพลิเคชัน X (เอ็กซ์) จำนวน 4,000 ชุดข้อมูล โดยใช้โมเดล Logistic Regression, Random Forest, Decision Tree, K-Nearest Neighbors, Neural Network Model, Convolutional Neural Network Model, Recurrent Neural Network Model และ Model with Dropout ทำการเปรียบเทียบและประเมินประสิทธิภาพของโมเดลทำนายพบว่า ผลจากการทำนายจากทั้ง 8 โมเดลนั้น สำหรับการ Detect label โมเดล Logistic regression และ Decision tree มีค่าความถูกต้อง สูงที่สุด โมเดล Random Forest มีค่าความถูกต้องเป็นอันดับรองลงมา และ K-Nearest Neighbors มีค่าความถูกต้องต่ำที่สุด สำหรับการ Detect category โมเดล Logistic regression มีค่าความถูกต้องสูงที่สุด โมเดล Decision tree มีค่าความถูกต้องเป็นอันดับรองลงมา และ Neural Network Model มีค่าความถูกต้องต่ำที่สุด

ในส่วนของการทำนายผู้วิจัยจึงได้นำโมเดล Logistic Regression ที่มีความถูกต้องสูงที่สุดมาใช้ในการทดสอบการหาผลลัพธ์จากข้อความใน X (เอ็กซ์) ว่าข้อความนี้พูดถึงเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยหรือไม่ โดยโมเดลอาจจะทำนายโดยยึดจากคีย์เวิร์ดเป็นหลัก ซึ่งในความเป็นจริงนั้นภาษาไทยมีรูปแบบประโยค และการใช้คำที่หลากหลายมากกว่าคีย์เวิร์ดที่ผู้วิจัยได้กำหนด และนำมาใช้ในงานวิจัยนี้ ซึ่งอาจเป็นคีย์เวิร์ดเพียงส่วนหนึ่งของเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยที่นำมาใช้ในการโพสต์ลงแอปพลิเคชัน X (เอ็กซ์) ในอนาคตควรจะรวบรวมข้อมูลเหตุการณ์ที่เกี่ยวข้องกับสาธารณภัยให้มีหลากหลายมากขึ้น และเพิ่มปริมาณชุดข้อมูลมากขึ้นเพื่อเพิ่มประสิทธิภาพของโมเดลการทำนาย

อ้างอิง

กรมป้องกันและบรรเทาสาธารณภัย กระทรวงมหาดไทย. (2559). **การลดความเสี่ยง จากสาธารณภัย** (พิมพ์ครั้งที่ 3). กรุงเทพฯ: บริษัท เวิร์ค พริ้นติ้ง จำกัด.

มหาวิทยาลัยนเรศวร. (ม.ป.ป.). เข้าถึงได้จากhttps://csit.nu.ac.th/kraisak/ds/ds/chapter05/Chapter05.pdf

โรงพยาบาลบางปะกอก3. (2019). เข้าถึงได้จาก https://www.bangpakok3.com/care_blog/view/