



โครงการจำแนกประเภทข้อความ ที่มีความเสี่ยงเป็นโรคซึมเศร้า

จัดทำโดย

B6401221 นายนันทิพัฒน์ หัตถโกศล

B6423087 นางสาวภัทรพร พรหมเงิน

B6428273 นางสาวณัฐชนยา สุขลัต

เสนอ

รองศาสตราจารย์ ดร.จิติมนต์ อั้งสกุล

อาจารย์ ดร.อรรคพล วงศ์กอบลาก

บทคัดย่อ

ปัจจุบันอัตราของผู้ป่วยโรคซึมเศร้าในประเทศไทยมีแนวโน้มสูงขึ้นในทุกๆปี โดยทั่วไปผู้ป่วยโรคซึมเศร้า มักจะมีอาการซึมเศร้าก่อนและพัฒนาอย่างเป็นโรคซึมเศร้า ซึ่งส่วนใหญ่ไม่ได้รับการรักษาและแสดงออกผ่านทาง สังคมออนไลน์ ซึ่งทำให้มีผลกระทบต่อการใช้ชีวิตประจำวันในผู้ป่วย งานวิจัยนี้จึงมุ่งเน้นการวิเคราะห์ข้อมูลจาก ทวิตเตอร์ เพื่อจำแนกประเภทข้อความว่ามีความเสี่ยงที่จะเป็นโรคซึมเศร้าหรือไม่ โดยมีการดึงข้อมูลมาจากทวิต เตอร์ทั้งหมด 2400 ประ多い และมีการทำความสะอาดข้อมูลโดยมีการแบ่งข้อความออกเป็นข้อความที่เสี่ยงเป็นโรค ซึมเศร้า 99.7% ประ多い และ ข้อความที่ไม่เสี่ยงเป็นโรคซึมเศร้า 1403 ประ多い หลังจากนั้นได้ทำการเทรนโมเดล โดย อัลกอริทึมที่นำมาใช้มีดังนี้ Logistic Regression, Random Forest, Decision Tree และ k-Nearest Neighbors ผลการวิจัยพบว่า การประเมินประสิทธิภาพโมเดลที่มีความถูกต้องมากที่สุดคือโมเดล Logistic Regression มีค่า ความถูกต้อง 0.783% โมเดล Random Forest มีค่าความถูกต้อง 0.758% โมเดล Decision Tree มีค่าความ ถูกต้อง 0.742% และโมเดล k-Nearest Neighbors มีค่าความถูกต้อง 0.651%

วิธีการดำเนินงาน

1. ข้อมูลที่นำมาใช้

ไฟล์ข้อความรูปแบบตารางประเภทไฟล์ CSV ชื่อไฟล์ dataoutput1.CSV ภายในข้อมูลประกอบไปด้วย คอลัมน์ จำนวน 5 คอลัมน์ ได้แก่ column label username keywords และ content และมีชุดข้อมูลจำนวน 2400 ชุดข้อมูล แหล่งข้อมูลมาจากการแอพพลิเคชัน “ทวิตเตอร์” ใช้วิธีการ Scweet ในการดึงข้อมูล แบ่งข้อมูลเป็น train set กับ test set โดย train set 70% และ test set 30%

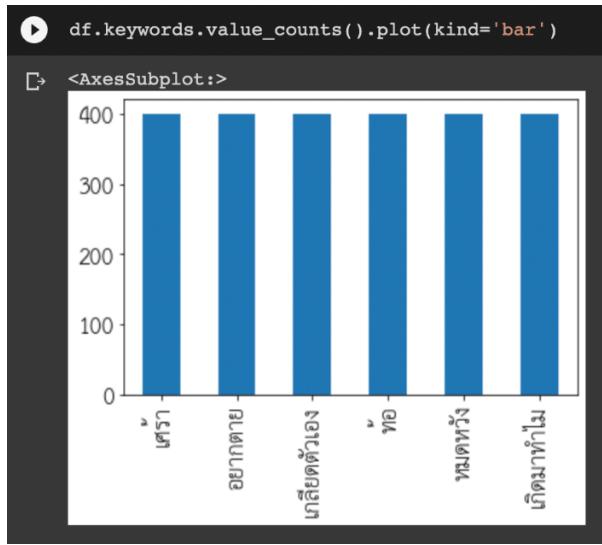
จากการค้นคว้าข้อมูลเกี่ยวกับผู้ที่เป็นโรคซึมเศร้าพบว่ามีอาการดังต่อไปนี้ 1. มีอารมณ์ซึมเศร้า(หงุดหงิด) ก้าวร้าว 2. ความสนใจในกิจกรรมต่าง ๆ แทบทั้งหมดลดลงอย่างมาก 3. น้ำหนักลดลงหรือเพิ่มขึ้นมาก เป็นอาหารหรือเจริญอาหารมาก 4. นอนไม่หลับหรือหลับมากเกินไป 5. กระวนกระวายอยู่ไม่สุขหรือเชื่องชาลง 6. อ่อนเพลีย ไร้เรี่ยวแรง 7. รู้สึกตนเองไร้ค่า 8. สามารถลดลง ใจโลய หรือลังเลใจไปหมด 9. คิดเรื่องการตายหรือคิดอยากร้าย (โรงพยาบาลบางปะกอก3, 2019)

ผู้วิจัยจึงนำข้อมูลจากการเหล่านี้มากำหนดเป็นคีย์เวิร์ดในการรวบรวมข้อมูล จำนวน 6 คีย์เวิร์ดได้แก่ เศร้า อยากร้าย เกลียดตัวเอง ห้อ หมวดหัวง และเกิดมาทำไม่ จากนั้น ได้ทำการเลเบลข้อมูลในคอลัมน์ label ได้แก่ 0 และ 1 “0 หมายถึง ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า” และ “1 หมายถึง ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า” โดยแบ่งเลเบลเป็นข้อความที่เป็น 0 จำนวน 1403 ข้อความ และ ข้อความที่เป็น 1 จำนวน 997 ข้อความ ดังรูปภาพต่อไปนี้

```
▶ df[['keywords', 'label']].value_counts()

   keywords      label
   หมวดหัวง      0        323
   เกิดมาทำไม่    0        280
   ห้อ            0        200
                   1        200
   อยากร้าย      0        200
                   1        200
   เกลียดตัวเอง  0        200
                   1        200
   เศร้า          0        200
                   1        200
   เกิดมาทำไม่    1        120
   หมวดหัวง      1         77
   dtype: int64
```

ภาพที่ 1 : แสดงจำนวนข้อมูลของข้อความที่เป็น 0 และ 1 ของแต่ละคีย์เวิร์ด



ภาพที่ 2 กราฟแสดงจำนวนข้อมูลของแต่ละคีย์เวิร์ด

2. เทคนิคที่นำมาใช้

Logistic Regression การวิเคราะห์การคาดถ้อยโลจิสติก เป็นเทคนิคการวิเคราะห์ตัวแปร เชิงพหุที่มีวัตถุประสงค์เพื่อประมาณค่าหรือทำนายเหตุการณ์ที่สนใจว่าจะเกิดหรือไม่เกิดเหตุการณ์นั้นๆ ภายใต้อิทธิของตัวปัจจัย แบบจำลองโลจิสติกประกอบด้วยตัวแปรตาม (หรือตัวแปรเกณฑ์) ที่ “องเป็นตัวแปรแบบทวินาม (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 เป็นโมเดลทางคณิตศาสตร์ที่ใช้ทำนาย ประเภทของวัตถุโดยพิจารณาจากลักษณะของวัตถุ โดยมีลักษณะการตัดสินใจเป็นแบบแผนผังต้นไม้ ประกอบด้วย ส่วนที่เป็นบัพภายใน (inner node) หรือที่เรียกว่าโหนดของต้นไม้ใช้แทนตัวแปรต่าง ๆ ที่ใช้ในการ ตัดสินใจ ส่วนที่กิ่ง (Branch) ใช้แทนค่าที่เป็นได้ของตัวแปร และส่วนที่เป็นบัพใบ (leaf node) หรือที่เรียกว่า โหนดใบ ใช้แทนค่าคำตอบของการตัดสินใจ หรือการจำแนก (มหาวิทยาลัยนเรศวร)

Random Forest คือขั้นตอนวิธีหนึ่งของการเรียนรู้ของเครื่องที่นิยมใช้มากใช้ได้ในหลาย ๆ ปัญหา ทั้งกับปัญหาแบบ Regression และ Classification โดย Random Forest เป็นขั้นตอนวิธีพัฒนาต่อยอดมาจาก Decision Tree ต่างกันที่ Random Forest เป็นการเพิ่มจำนวนต้นไม้ (Tree) เป็นหลาย ๆ ต้นทำให้ ประสิทธิภาพการทำงานและพยากรณ์สูงขึ้น Random Forest มีหลักการทำงาน คือ จะแบ่งข้อมูลออกเป็น ต้นไม้ตัดสินใจ (Decision Tree) หลาย ๆ ต้น โดยแต่ละต้นจะได้รับคุณลักษณะ (Feature) และข้อมูล (Data) ที่ไม่เหมือนกัน ทั้งหมด เพื่อทำให้ได้ต้นไม้ที่มีความหลากหลายและมีความอิสระต่อกันมากขึ้น (มหาวิทยาลัยนเรศวร)

K-nearest neighbors (KNN) นับเป็นเทคนิคที่มีวิธีการไม่ซับซ้อนและเข้าใจได้ง่ายที่สุดที่ใช้ในการจำแนกประเภทข้อมูล โดยหลักการทำงาน คือ จะใช้หลักการเปรียบเทียบความคล้ายคลึงกันของข้อมูลที่ สนใจกับข้อมูลอื่นว่ามีความคล้ายคลึงหรืออยู่ใกล้กับข้อมูลมากที่สุด k ตัว 既然นี้จะทำการตัดสินใจว่า คำตอบของข้อมูลที่สนใจนั้นควรเป็นคำตอบเดียวกับข้อมูลที่อยู่ใกล้ที่สุด k ตัวนั้น ทั้งนี้ k คือความถี่ของข้อมูลที่อยู่ใกล้กับข้อมูลที่สนใจ (มหาวิทยาลัยนเรศวร)

3. วิธีการ และการกำหนดค่าต่าง ๆ ของเทคนิคที่นำมาใช้

3.1 การนำเข้าข้อมูล

```
[ ] #import CSV file
    import pandas as pd

    df = pd.read_csv('/content/drive/MyDrive/Project 73/dataoutput1.csv')
```

ภาพที่ 3 รายละเอียดข้อมูลนำเข้า

ไฟล์ข้อความรูปแบบตารางประเภทไฟล์ CSV ชื่อไฟล์ dataoutput1.CSV ภายในไฟล์คือข้อมูลที่จัดเก็บข้อความที่มีความเสี่ยงเป็นโรคซึมเศร้า และข้อความที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า

3.2 การติดตั้งเครื่องมือ PyThaiNLP และ การติดตั้ง font สำหรับภาษาไทย

3.2.1 การติดตั้งเครื่องมือ PyThaiNLP

```
▶ # https://github.com/PyThaiNLP/pythainlp
# https://github.com/PyThaiNLP/wisesight-sentiment

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

ภาพที่ 4 การติดตั้งเครื่องมือ PyThaiNLP

ติดตั้งเครื่องมือ PyThaiNLP เพื่อช่วยในการตัดในประโยคที่เป็นภาษาไทย และ import libraly ที่จำเป็นได้แก่ import pandas as pd, import re from pythainlp.tokenize, และ import word_tokenize

3.2.1 การติดตั้ง font สำหรับภาษาไทย

```
▶ !wget -q https://github.com/Phonbopit/sarabun-webfont/raw/master/fonts/thsarabunnew-webfont.ttf

▶ font_path = '/content/thsarabunnew-webfont.ttf'
```

ภาพที่ 5 การติดตั้ง front ภาษาไทย

ติดตั้ง font ภาษาไทย TH Saranun New เพื่อให้ Word Cloud สามารถอ่านภาษาไทยได้

3.3 Data Preprocessing

```
import numpy as np
import re

temp = df.content.values
def clean_text(temp):
    if type(temp) == np.float:
        return ""
    temp = re.sub('http\S+', 'url', temp )
    temp = re.sub('[?#@]', '', temp )
    temp = re.sub('[A-Za-z0-9_]+', '', temp )
    temp = re.sub('#[A-Za-z0-9_]+', '', temp )
    temp = "".join(c for c in temp if c not in ("?", "^", ".", "/", "+", "!", "'", "!", "!", "|", "/", "\\", "\\", "\\", "\\", "("))

    return temp
```

ภาพที่ 6 การทำความสะอาดข้อมูลโดยใช้ฟังก์ชัน regex

- Import re เพื่อใช้งานฟังก์ชัน regex
 - กำหนด คอลัมน์ content ให้อยู่ในตัวแปรที่ชื่อว่า temp
 - เรียกใช้ฟังก์ชัน def เพื่อlobข้อมูล และส่งข้อมูลตัวแปร temp เข้ามา
 - ทำการเช็คข้อมูลว่าเป็นตัวเลขหรือไม่ ถ้าเป็นให้เริ่นกลับไปเป็นซองว่าง หากไม่ใช่ตัวเลขแต่เป็นข้อมูล String จะทำการจัดการ แก้ไข และลบส่วนที่ไม่ต้องการออก ได้แก่ ลบลิปิค์ , ลบตัวอักษร A-Z a-z และ ลบเลข 0-9 , ลบแซชแท็ก, ลบอักษรพิเศษต่าง ๆ แทนด้วยซองว่าง

```
df['clean'] = df.content.apply(clean_tweet)
```

ภาพที่ 7 เรียกใช้ clean tweet

เรียกใช้ `df.content.apply(clean_tweet)` ที่สร้างไว้เพื่อเก็บค่าเข้า คอลัมน์ `df['clean']`

3.4 Word Tokenize การตัดคำต่าง ๆ ออกจากประโยค

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

ภาพที่ 8 Word Tokenize

- สร้างฟังก์ชันที่ชื่อว่า my_tokenise
 - เรียกใช้ Word tokenize เก็บค่าไว้ในตัวแปร txt

3.5 Word Cloud

3.5.1 นำเข้า library ต่าง ๆ ที่เกี่ยวกับการทำ word cloud และ thai stopwords และนำเข้า import copy เพื่อคัดลอก thai stopwords ไปใช้ต่อไป

```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

regexp = r"[\u0E00-\u0E7Fa-zA-Z]+"
thai_stopwords = set(STOPWORDS)
thai_stopwords.update(["ໃຊ້", "ທາກ", "ທັງນັນ", "ໄມ່", "ພລາຍ", "ດ້າ", "ໄປ", "ຫລັງຈາກ", "ດູກ", "ໄດ້", "ທັງ", "ລົງ",
"ໃໝ່", "ທ່ອງ", "ໃນ", "ເໜື່ງ", "ຕ່າງໆ", "ໂດຍ", "ສ່ວນ", "ຕ່າງ", "ແກ່", "ສົ່ງ", "ຕອ", "ວັນຈານ", "ນີ້", "ແມ່", "ຈ",
"ແລ້ວ", "ສຸດ", "ຄົນ", "ແຂບ", "ສໍາເລັບ", "ຕົ້ນຕໍ່", "ແກຣມ", "ວ່າ", "ຕັ້ງ", "ແນບ", "ວັນ", "ຕ້ານ", "ວ່ອງ", "ກົກ", "ສູງ",
"ແຕ່", "ຈົງ", "ຫຼວຍ", "ເອງ", "ວ່ານ", "ຕັ້ງ", "ເຫັນ", "ຮາຍ", "ສົ່ງ", "ເລຍ", "ວັນ", "ຫຼວງ", "ເຈັນ", "ຮອກວ່າງ", "ແຄ່", "ຄົນ",
"ຈິງ", "ເຮົາ", "ຮ່ວມ", "ຈຳກັດ", "ນີ້ອີງ", "ຍົງ", "ຈົດ", "ເພື່ອ", "ມີ", "ຈະ", "ພຣະ", "ນາກ", "ຕົວ", "ເປັນການ", "ຄວາມ",
"ເປັນ", "ພ້ອມ", "ຕ້ອງ", "ເປີດເພື່ອ", "ພບ", "ຄົວ", "ເປີຕ", "ຜ່ານ", "ຫຸ້ນ", "ເນື້ອງຈາກ", "ຜລ", "ຂອງ", "ເຕືອກັນ", "ນາງ", "ຂອງ",
"ເຕືອງ", "ນໍາ", "ຂະບົນ", "ເຫັນ", "ນີ້", "ກ່ອນ", "ເຈົາວ", "ນໍາ", "ກີ", "ເຂຍ", "ນັ້ນ", "ກາງ", "ນັ້ນ", "ນະບະ", "ດີ",
"ເຫັນ", "ນັ້ນ", "ກັບ", "ເຫຼົາ", "ນອກຈາກ", "ກັນ", "ອີກ", "ທຸກ", "ກວ່າ", "ອາຈຸ", "ທີ່ສຸດ", "ກລ່າວ", "ຫິນ", "ບັນ", "ວະ", "ຕະ",
"ອະໄຮ", "ທີ", "ອອກ", "ທ່າໃຫ້", "ອໝາຍ", "ທ່າ", "ອູ້", "ທາງ", "ອຢາກ", "ທັງນີ້", "ຖຸ", "ສະ", "ຂະ", "ນະ", "ຈາ"])
```

ภาพที่ 9 นำเข้า library ต่าง ๆ ที่เกี่ยวกับการทำ word cloud และ thai stopwords

และนำเข้า import copy เพื่อคัดลอก thai stopwords ไปใช้ต่อไป

- นำเข้า library ต่าง ๆ ที่เกี่ยวกับการทำ word cloud
 - แสดง thai stopwords ที่กำหนดไว้
 - นำเข้า import copy เพื่อคัดลอก thai stopwords ที่กำหนดไว้ ไปใช้งานในการแสดง word cloud

3.5.2 word cloud ของ keywords เศร้า

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[df.keywords == 'ເຕີມ'].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 10 word cloud ของ keywords เศร้า

- คัดลอก thai stopwords เก็บเข้าดัวแพร teststop เพื่อใช้งาน thai stopwords ที่กำหนดไว้ก่อนหน้า
 - เรียกใช้ font_path ที่ติดตั้งไว้
 - กำหนดพื้นหลัง = สีดำ
 - กำหนดขนาด ความกว้าง = 100

- กำหนดขนาด ความสูง = 100
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud= 100 คำ
- ลบ stopwords และ Generate เชื่อมกับคอมลัมນ์ keywords เศร้า เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงผลมาเป็นรูปภาพ

3.5.3 word cloud ของ keywords เศร้า ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'เศร้า') & (df.label == 1)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 11 word cloud ของ keywords เศร้า ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

- คัดลอก thai stopwords เก็บเข้าตัวแปร teststop เพื่อใช้งาน thai stopwords ที่กำหนดไว้ก่อนหน้า
- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 100
- กำหนดขนาด ความสูง = 100
- กำหนด regexp ให้สามารถวิเคราะห์ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud= 100 คำ
- ลบ stopwords และ generate เชื่อมกับคอมลัมນ์ keywords เศร้า ที่ label เป็น 1 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงผลมาเป็นรูปภาพ

3.5.4 word cloud ของ keywords เศร้า ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop,).generate(str(' '.join(df[(df.keywords == 'เศร้า') & (df.label == 0)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 12 word cloud ของ keywords เศร้า ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

- คัดลอก thai stopwords เก็บเข้าตัวแปร teststop เพื่อใช้งาน thai stopwords ที่กำหนดไว้ก่อนหน้า
- เรียกใช้ font_path ที่ติดตั้งไว้
- กำหนดพื้นหลัง = สีดำ
- กำหนดขนาด ความกว้าง = 100
- กำหนดขนาด ความสูง = 100
- กำหนด regexp ให้สามารถจับคำที่ข้อความเป็นภาษาไทยได้
- กำหนดจำนวนคำที่แสดงใน word cloud= 100 คำ
- ลบ stopwords และ generate เชื่อมกับคอมล้มน์ keywords เศร้า ที่ label เป็น 0 เพื่อสร้าง word cloud
- เรียกใช้คำสั่งเพื่อแสดงอุปมาเป็นรูปภาพ

3.5.5 word cloud ของ keywords อยาқตای

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[df.keywords == 'อยาқตัย'].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 13 word cloud ของ keywords อยาқตัย

3.5.6 word cloud ของ keywords อยาқตัย ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'อยาқตัย') & (df.label == 1)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 14 word cloud ของ keywords อยาқตัย ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.7 word cloud ของ keywords อยาқตัย ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words= 50,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'อยาқตัย') & (df.label == 0)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 15 word cloud ของ keywords อยาқตัย ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.8 word cloud ของ keywords เกลี่ยดตัวเอง

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words= 100,
    stopwords = teststop).generate(str(' '.join(df[df.keywords == 'เกลี่ยดตัวเอง'].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 16 word cloud ของ keywords เกลี่ยดตัวเอง

3.5.9 word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'เกลี่ยดตัวเอง') & (df.label == 1)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 17 word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.10 word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'เกลี่ยดตัวเอง') & (df.label == 0)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 18 word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.11 word cloud ของ keywords ท้อ

```
▶ teststop = copy.deepcopy(thai_stopwords)
# teststop.add('เกลียด')

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words= 100,
    stopwords = teststop).generate(str(' '.join(df[df.keywords == 'ท้อ'].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 19 word cloud ของ keywords ท้อ

3.5.12 word cloud ของ keywords ท้อ ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'ท้อ') & (df.label == 1)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 20 word cloud ของ keywords เกลียดตัวเอง ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.13 word cloud ของ keywords ท้อ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'ท้อ') & (df.label == 0)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 21 word cloud ของ keywords เกลียดตัวเอง ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.14 word cloud ของ keywords หมวดหัวข้อ

```
▶ teststop = copy.deepcopy(thai_stopwords)
teststop.add('แคปชัน')

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[df.keywords == 'หมวดหัวข้อ'].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 22 word cloud ของ keywords หมวดหัวข้อ

3.5.15 word cloud ของ keywords หมวดหัวข้อ ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'หมวดหัวข้อ') & (df.label == 1)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 23 word cloud ของ keywords เกี่ยดด้วยเรื่อง ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.16 word cloud ของ keywords หมวดหัวข้อ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'หมวดหัวข้อ') & (df.label == 0)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 24 word cloud ของ keywords หมวดหัวข้อ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.17 word cloud ของ keywords เกิดมาใหม่

```
▶ teststop = copy.deepcopy(thai_stopwords)
# teststop.add('เกิดมาใหม่')

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[df.keywords == 'เกิดมาใหม่'].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 25 word cloud ของ keywords เกิดมาใหม่

3.5.18 word cloud ของ keywords เกิดมาใหม่ ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'เกิดมาใหม่') & (df.label == 1)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 26 word cloud ของ keywords เกิดมาใหม่ ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.5.19 word cloud ของ keywords เกิดมาใหม่ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

```
▶ teststop = copy.deepcopy(thai_stopwords)

wordcloud = WordCloud(
    font_path=font_path,
    background_color = 'black',
    width = 1000,
    height = 500,
    regexp=regexp,
    max_words=100,
    stopwords = teststop).generate(str(' '.join(df[(df.keywords == 'เกิดมาใหม่') & (df.label == 0)].token.values)))

fig, ax = plt.subplots(1, 1, figsize=(16, 12))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis("off")
fig.show()
```

ภาพที่ 27 word cloud ของ keywords เกิดมาใหม่ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

3.6 การทำแบ่ง Train set และ Test set

```
[97] data = df.token.values  
y = df.label.values
```

ภาพที่ 28 แสดงการเก็บค่าต่างๆไว้ในตัวแปร

- นำคอลัมน์ token ไปเก็บไว้ในตัวแปร data
- นำคอลัมน์ label ไปเก็บไว้ในตัวแปร y
-

```
[99] 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)
```

ภาพที่ 29 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

3.7.1 การทำ feature extraction ด้วย CountVectorizer

```
▶ 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 30 แสดงการทำ feature extraction ด้วย CountVectorizer

นำเข้าไลบรารีที่สำคัญในการสกัด CountVectorizer ดังนี้

from sklearn.feature_extraction.text import CountVectorizer

เรียกใช้ CountVectorizer เก็บค่าเข้าตัวแปร Vectorizer

และนำมา vectorizer.fit_transform(data) ข้อมูลที่มีอยู่

3.7.2 การทำ feature extraction ด้วย TfidfVectorizer

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

ภาพที่ 31 แสดงการทำ feature extraction ด้วย TfidfVectorizer

นำเข้าไลบรารีที่สำคัญในการสร้าง 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

3.8.1 Logistic Regression

```
▶ 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 32 แสดงการเทรนโมเดล Logistic Regression ด้วยการสกัด CountVectorizer

3.8.2 Random Forest

```
▶ 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 33 แสดงการトレนโมเดล Random Forest ด้วยการสกัด CountVectorizer

3.8.3 Decision tree

```
▶ 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 34 แสดงการเทรนโมเดล Decision tree ด้วยการสัด CountVectorizer

3.8.3 K-Nearest Neighbors

```
▶ 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 35 แสดงการเทรนโมเดล K-Nearest Neighbors ด้วยการสัด CountVectorizer

3.9 การ Train Model ด้วยการสกัด TfidfVectorizer

3.9.1 Logistic Regression

```
▶ 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 36 แสดงการเทรนโมเดล Logistic Regression ด้วยการสัด TfidfVectorizer

3.9.2 Random Forest

```
▶ 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 37 แสดงการเทรนโมเดล Random Forest ด้วยการสัด TfidfVectorizer

3.9.3 Decision tree

```
▶ from sklearn import tree
  from sklearn import metrics

  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 38 แสดงการトレนโมเดล Decision tree ด้วยการสัด TfidfVectorizer

3.9.3 K-Nearest Neighbors

```
▶ from sklearn import neighbors
  from sklearn import metrics

  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 39 แสดงการトレนโมเดล K-Nearest Neighbors ด้วยการสัด TfidfVectorizer

ผลการทดลอง และการอภิปรายผล

สภาพแวดล้อมในการทดลอง

ข้อมูลที่ใช้คือข้อมูลที่เป็นข้อความจากการทวีต รวบรวมมาจาก แอปพลิเคชัน “ทวิตเตอร์” จำนวน 2400 ชุดข้อมูล โดยมี 6 คีย์เวิร์ด ได้แก่ เศร้า อยากตาย เกลียดตัวเอง ห้อ หมดหัวง และเกิดมาทำไม่ และแยกประเภท ของข้อความเป็น 2 ประเภท ได้แก่ 0 และ 1 “0 หมายถึง ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า” และ “1 หมายถึง ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า” ประเมินโดยอาศัยค่าความถูกต้อง

ผลการทดสอบที่ได้รับ

4.1 การแสดง Word Cloud

4.1.1 ภาพ word cloud ของ keywords เศร้า



ภาพที่ 40 ผลลัพธ์ word cloud ของ keywords เครื่อง

4.1.2 ภาพ word cloud ของ keywords เศร้า ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)



ภาพที่ 41 ผลลัพธ์ word cloud ของ keywords เศร้า ที่มีความเน้น 1 (1=ผู้ที่มีความเสี่ยงเจ็บป่วยมาก) 6

4.1.3 ภาพ word cloud ของ keywords เศร้า ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรค

ចិនក្រោម



ภาพที่ 42 ผลลัพธ์ word cloud ของ keywords เครื่องที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรค)

จากการทดลอง การแสดง word cloud ของ keywords เศร้า พบร่วมกับคำที่เกิดขึ้นบ่อย ได้แก่ ตัวเอง, เศร้า, รู้สึก, ชีมเศร้า, ตาย, ชอบ, รู้สึก, เพลง และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าและผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าพบว่า ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าคำที่แสดงใน Word Cloud ได้แก่ เศร้า, ชีมเศร้า, กลัว, ชีวิต, รู้สึก, แม่ และอื่น ๆ และผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าพบว่า คำที่แสดงใน Word Cloud ได้แก่ เศร้า, ร้องไห้, เพลง, ชอบ, อ่าน, ตอน, เรื่อง, พระเอก และอื่น ๆ เป็นต้น เห็นได้ชัดว่าผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นปัญหาด้านครอบครัว เพื่อน หรือปัญหานิชีวิตต่าง ๆ แต่ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำเหล่านี้ไปทางประโยคบอกเล่าหรือบ่น เช่น “หนังเรื่องนี้เศร้ามาก ดูแล้วร้องไห้”

4.1.4 ภาพ word cloud ของ keywords อย่างตัวอย่าง



ภาพที่ 43 ผลลัพธ์ word cloud ของ keywords อย่างตัว

4.1.5 ภาพ word cloud ของ keywords อย่างตัวที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็น

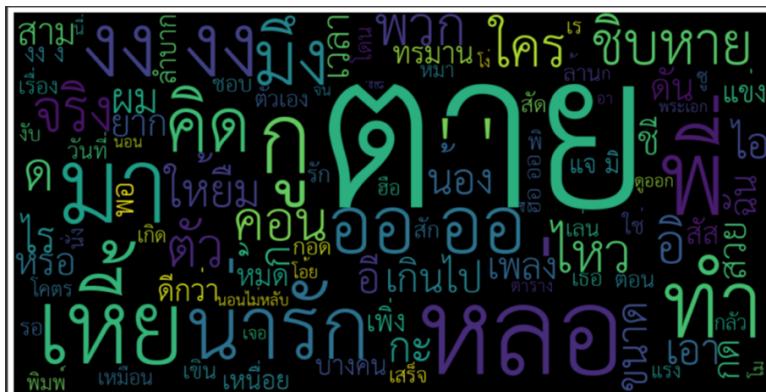
ໂຮມໝໍາເສົ້າ



ภาพที่ 44 ผลลัพธ์ word cloud ของ keywords อย่างตัวที่มีความเป็น 1 (ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

4.1.6 ภาพ word cloud ของ keywords อย่างตัวที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยง)

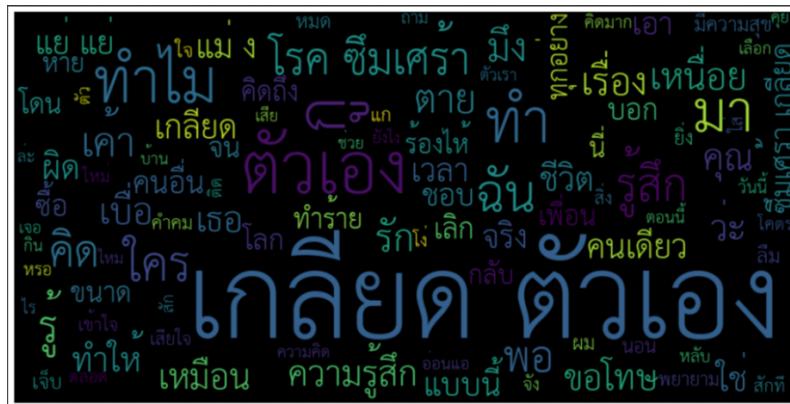
ເປັນໂຮຄສົມເສຣ້າ



ภาพที่ 45 ผลลัพธ์ word cloud ของ keywords อย่างติดย ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

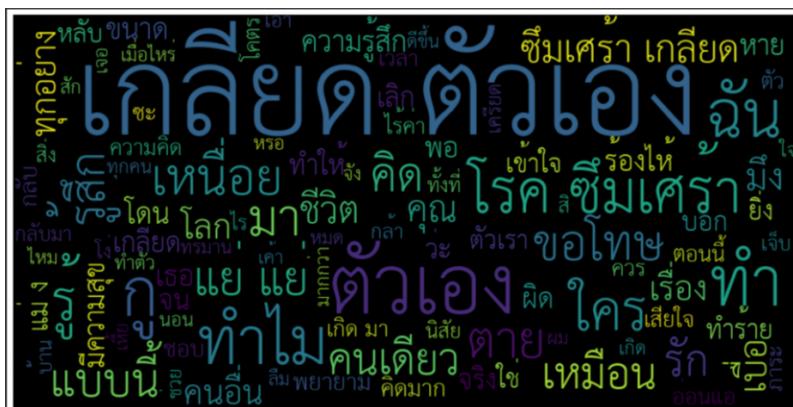
จากผลการทดลอง การแสดง word cloud ของ keywords อย่างตаяy พบว่าคำที่เกิดขึ้นบ่อย ได้แก่ ตาย, โรค, ชีมศร้า, เหนือย, ฆ่าตัวตาย, หล่อ, ชีวิต และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ผู้ที่มีความเสี่ยงเป็นโรคชีมศร้าและผู้ที่ไม่มีความเสี่ยงเป็นโรคชีมศร้าพบว่า ผู้ที่มีความเสี่ยงเป็นโรคชีมศร้าคำที่แสดงใน Word Cloud ได้แก่ ตาย, ชีมศร้า, โรค, เหนือย, ครอบครัว และอื่น ๆ และผู้ที่ไม่มีความเสี่ยงเป็นโรคชีมศร้าพบว่า คำที่แสดงใน Word Cloud ได้แก่ ตาย, หล่อ, น่ารัก, ชีบทาย, พี่ และอื่น ๆ เป็นต้น เห็นได้ชัดว่าผู้ที่มีความเสี่ยงเป็นโรคชีมศร้าจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นปัญหาด้านครอบครัว เพื่อน หรือปัญหานิชีวิตต่าง ๆ แต่ผู้ที่ไม่มีความเสี่ยงเป็นโรคชีมศร้าจะใช้คำเหล่านี้ไปในทางประโยชน์ออกเล่าหรือบ่น เช่น “พี่คนนี้หล่อมากกูอยากร้าย”

4.1.7 ภาพ word cloud ของ keywords เกลี่ยดตัวเอง



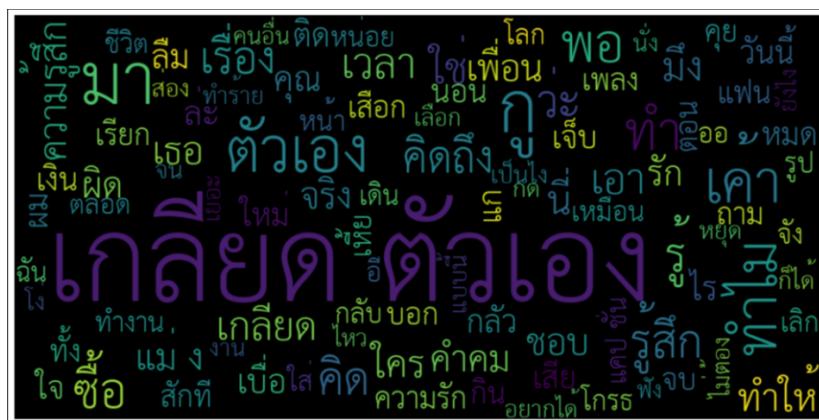
ภาพที่ 46 ผลลัพธ์ word cloud ของ keywords เกลือยดตัวเอง

4.1.8 ภาพ word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยง เป็นโรคซึมเศร้า)



ภาพที่ 47 ผลลัพธ์ word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 1 (1=ผู้มีความเสี่ยงเป็นโรคซึมเศร้า)

4.1.9 ภาพ word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยง เป็นโรคซึมเศร้า)



ภาพที่ 48 ผลลัพธ์ word cloud ของ keywords เกลี่ยดตัวเอง ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

จากการทดลอง การแสดง word cloud ของ keywords เกลี่ยดตัวเอง พบร่วมคำที่เกิดขึ้นบ่อย ได้แก่ เกลี่ยดตัวเอง, ชีมศร้า, เปื้อ, แย่, ร้องไห้, ขอโทษ และอื่น ๆ เป็นต้น เปรียบเทียบระหว่างผู้ที่มีความเสี่ยงเป็นโรคชีมศร้าและผู้ที่ไม่มีความเสี่ยงเป็นโรคชีมศร้าพบว่า ผู้ที่มีความเสี่ยงเป็นโรคชีมศร้าคำที่แสดงใน Word Cloud ได้แก่ เกลี่ยดตัวเอง, ชีมศร้า, ขอโทษ, แย่ และอื่น ๆ และผู้ที่ไม่มีความเสี่ยงเป็นโรคชีมศร้าพบว่า คำที่แสดงใน Word Cloud ได้แก่ เกลี่ยดตัวเอง, คิดถึง, ความรัก, คำคม, ชื้อ และอื่น ๆ เป็นต้น เห็นได้ชัดว่าผู้ที่มีความเสี่ยงเป็นโรคชีมศร้าจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นปัญหาด้านครอบครัว เพื่อน หรือปัญหานิสิตต่าง ๆ แต่ผู้ที่ไม่มีความเสี่ยงเป็นโรคชีมศร้าจะใช้คำเหล่านี้ไปในทางประโยคบอกรเล่าหรือบ่น เช่น “เกลี่ยดตัวเองที่เงินอกรากไรใช้หมดตลอด”

4.1.10 ภาพ word cloud ของ keywords ที่อ



ภาพที่ 49 ผลลัพธ์ word cloud ของ keywords ห้อง

4.1.11 ภาพ word cloud ของ keywords ท้อ ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)



ภาพที่ 50 ผลลัพธ์ word cloud ของ keywords ท้อ ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

4.1.12 ภาพ word cloud ของ keywords ท้อ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)



ภาพที่ 51 ผลลัพธ์ word cloud ของ keywords ท้อ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)

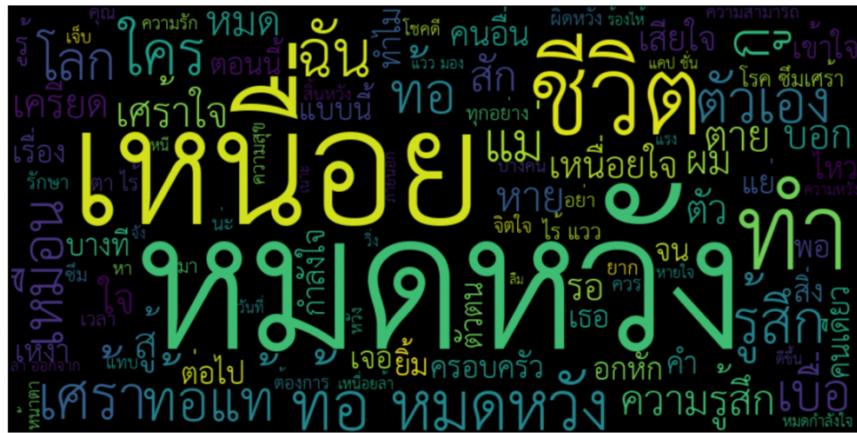
จากการทดลอง การแสดง word cloud ของ keywords ท้อ พบร่วมคำที่เกิดขึ้นบ่อย ได้แก่ เหนือย, ห้อ, ร้องให้, รู้สึก, ชีวิต, คนเดียว และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าและผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าพบว่า ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าคำที่แสดงใน Word Cloud ได้แก่ ห้อ, เหนือย, ร้องให้, รู้สึก, ทำไม่ และอื่น ๆ และผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าพบว่า คำที่แสดงใน Word Cloud ได้แก่ ห้อ, เหนือย, สู้, อย่า, เวลา และอื่น ๆ เป็นต้น เห็นได้ชัดว่าผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นปัญหาด้านครอบครัว เพื่อน หรือปัญหานิสิตต่าง ๆ แต่ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำเหล่านี้ไปในทางประโยชน์ของเล่าหรือให้กำลังใจ เช่น “อย่าเพิ่งห้อนะ สู้ๆ”

4.1.13 ภาพ word cloud ของ keywords หมวดหัว



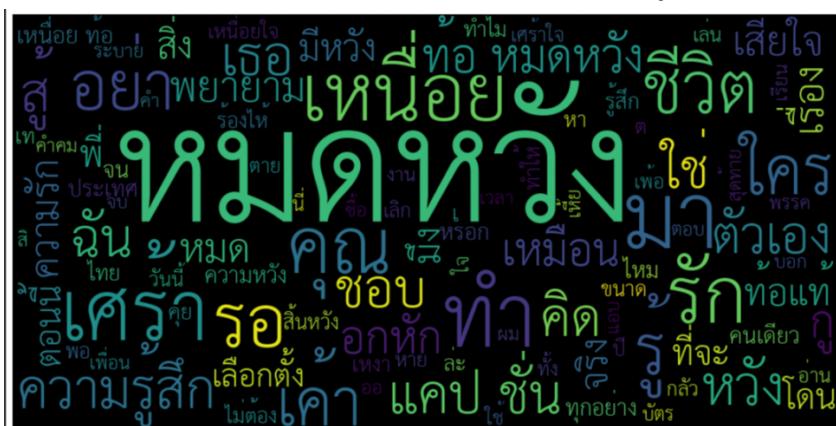
ภาพที่ 52 ผลลัพธ์ word cloud ของ keywords หมวดหัวง

4.1.14 ภาพ word cloud ของ keywords หมวดหัวัง ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)



ภาพที่ 53 ผลลัพธ์ word cloud ของ keywords หมวดหัวข้อความเป็น 1 (ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

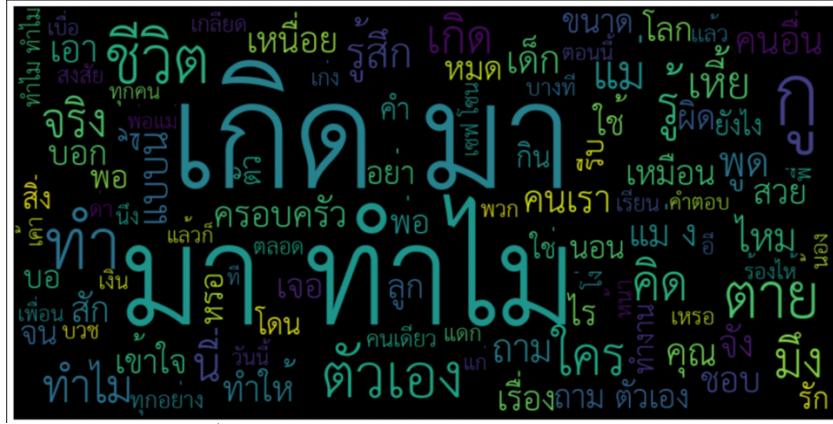
4.1.15 ภาพ word cloud ของ keywords หมวดหัวงที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า)



ภาพที่ 54 ผลลัพธ์ word cloud ของ keywords หมวดหัวที่ข้อความเป็น 0 (0=ที่ไม่มีความเลื่อนเป็นโรคซึมเศร้า)

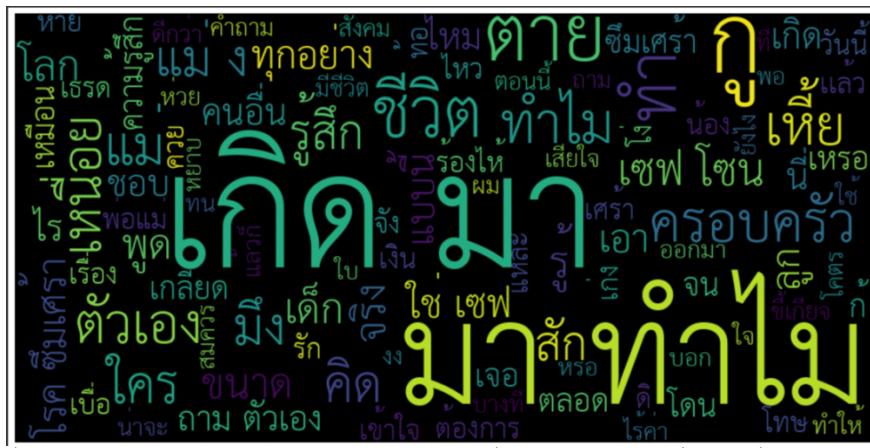
จากผลการทดลอง การแสดง word cloud ของ keywords หมวดหัวง พบว่าคำที่เกิดขึ้นบ่อยได้แก่ เหนือย, หมวดหัวง, ความรู้สึก, ตัวเอง, รัก และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าและผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าพบว่า ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าคำที่แสดงใน Word Cloud ได้แก่ เศร้า, เหนือย, หมวดหัวง, ห้อ, รู้สึก และอื่น ๆ และผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าพบว่า คำที่แสดงใน Word Cloud ได้แก่ หมวดหัวง, เหนือย, รอ, อกหัก, พยายาม และอื่น ๆ เป็นต้น เห็นได้ชัดว่าผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นปัญหาด้านครอบครัว เพื่อน หรือปัญหาในชีวิตต่าง ๆ แต่ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำเหล่านี้ไปในทางประโยคบอกเล่าหรือบ่น เช่น “ไม่มีทางเลยที่คนจะชอบเรา หมวดหัวง”

4.1.16 ภาพ word cloud ของ keywords เกิดมาทำไม



ภาพที่ 55 ผลลัพธ์ word cloud ของ keywords เกิดมาทำไม้

4.1.17 ภาพ word cloud ของ keywords เกิดมาทำไม่ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยง เป็นโรคซึมเศร้า)



ภาพที่ 56 ผลลัพธ์ word cloud ของ keywords เกิดมาทำไม่ที่ข้อความเป็น 1 (1=ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า)

4.1.17 ภาพ word cloud ของ keywords เกิดมาทำไม ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยง เป็นโรคซึมเศร้า)



ภาพที่ 57 ผลลัพธ์ word cloud ของ keywords กิจกรรมทำไม้ ที่ข้อความเป็น 0 (0=ผู้ที่ไม่มีความเสี่ยงเป็นโรคชิมเครร์)

จากผลการทดลอง การแสดง word cloud ของ keywords เกิดมาทำไม่พบว่าคำที่เกิดขึ้นบ่อยได้แก่ เกิด, มา, ทำไม่, ตัวเอง, ชีวิต, ตาย, และอื่น ๆ เป็นต้น เปรียบเทียบระหว่าง ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าและผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าพบว่า ผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าคำที่แสดงใน Word Cloud ได้แก่ เกิด, มา, ทำไม่, ชีวิต, ครอบครัว และอื่น ๆ และผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าคำที่แสดงใน Word Cloud ได้แก่ ถาม, ตัวเอง, เกิด, มา, ทำไม่ และอื่น ๆ เป็นต้น เห็นได้ชัดว่าผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำที่แสดงให้เห็นปัญหาที่เกิดขึ้น ไม่ว่าจะเป็นปัญหาด้านครอบครัว เพื่อน หรือปัญหานิชีวิตต่าง ๆ แต่ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้าจะใช้คำเหล่านี้ไปในทางประโยคบอกเล่าหรือบ่น เช่น “บางทีก็สังสัยว่าคนเรา” (โรงพยาบาลบางปะกอก3, 2019)

4.2 ผลการทดลองการ Train Model ด้วยการสักดิ์ CountVectorizer

4.2.1 Logistic Regression

```
[4] 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	0.81	0.83	0.82	427
1	0.74	0.71	0.73	293
accuracy			0.78	720
macro avg	0.78	0.77	0.77	720
weighted avg	0.78	0.78	0.78	720

ภาพที่ 58 แสดงผลการเทรนโมเดล Logistic Regression ด้วยการสักดิ์ CountVectorizer

4.2.2 Random Forest

```
[95] 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	0.77	0.84	0.80	427
1	0.73	0.64	0.68	293
accuracy			0.76	720
macro avg	0.75	0.74	0.74	720
weighted avg	0.76	0.76	0.76	720

ภาพที่ 59 แสดงผลการเทรนโมเดล Random Forest ด้วยการสักดิ์ 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	0.72	0.92	0.81	427
1	0.81	0.48	0.60	293
accuracy			0.74	720
macro avg	0.77	0.70	0.71	720
weighted avg	0.76	0.74	0.73	720

ภาพที่ 60 แสดงผลการเรนโมเดล Decision Tree ด้วยการสัด 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	0.68	0.77	0.72	427
1	0.59	0.48	0.53	293
accuracy			0.65	720
macro avg	0.64	0.63	0.63	720
weighted avg	0.64	0.65	0.64	720

ภาพที่ 61 แสดงผลการเรนโมเดล K-Nearest Neighbors ด้วยการสัด CountVectorizer

4.3 ผลการทดลองการTrain Model ด้วยการสกัด TfifdVectorizer

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	0.81	0.83	0.82	427
1	0.74	0.71	0.73	293
accuracy			0.78	720
macro avg	0.78	0.77	0.77	720
weighted avg	0.78	0.78	0.78	720

ภาพที่ 62 แสดงผลการเรนโมเดล Logistic Regression ด้วยการสัด TfifdVectorizer

4.3.2 Random Forest

```
[68] 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	0.77	0.83	0.80	427
1	0.72	0.63	0.68	293
accuracy			0.75	720
macro avg	0.75	0.73	0.74	720
weighted avg	0.75	0.75	0.75	720

ภาพที่ 63 แสดงผลการเทรนโมเดล Random Forest ด้วยการสัด 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	0.72	0.92	0.81	427
1	0.81	0.48	0.60	293
accuracy			0.74	720
macro avg	0.76	0.70	0.70	720
weighted avg	0.76	0.74	0.72	720

ภาพที่ 64 แสดงผลการเทรนโมเดล Decision Tree ด้วยการสัด 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	0.68	0.77	0.72	427
1	0.59	0.48	0.53	293
accuracy			0.65	720
macro avg	0.64	0.63	0.63	720
weighted avg	0.64	0.65	0.64	720

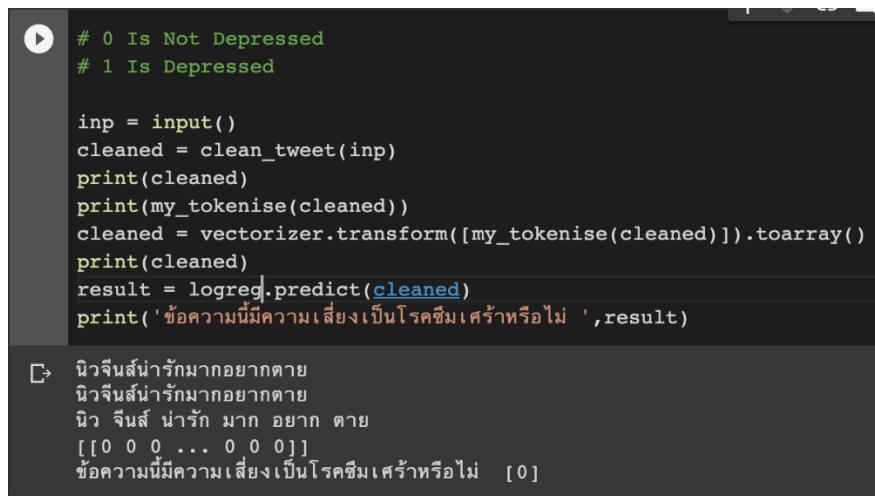
ภาพที่ 65 แสดงผลการเทรนโมเดล K-Nearest Neighbors ด้วยการสัด TfidfVectorizer

4.4 ผลการทดลองค่าประสิทธิภาพการ Train Model ในรูปแบบตาราง

การสกัด	โมเดล	ค่าประสิทธิภาพโมเดล			
		Accuracy	Precision	Recall	f1-score
CountVectorizer	Logistic Regression	0.78	0.78	0.78	0.78
	Random Forest	0.76	0.76	0.76	0.76
	Decision Tree	0.74	0.76	0.74	0.73
	K-Nearest Neighbors	0.65	0.64	0.65	0.64
TfidfVectorizer	Logistic Regression	0.78	0.78	0.77	0.77
	Random Forest	0.75	0.75	0.75	0.75
	Decision Tree	0.74	0.76	0.74	0.72
	K-Nearest Neighbors	0.65	0.64	0.65	0.64

ตารางที่ 1 ผลการทดลองค่าประสิทธิภาพการ Train Model

4.5 ผลการทดสอบ Model



```
# 0 Is Not Depressed
# 1 Is Depressed

inp = input()
cleaned = clean_tweet(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned = vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)
result = logreg.predict(cleaned)
print('ข้อความนี้มีความเสี่ยงเป็นโรคซึมเศร้าหรือไม่ ',result)

→ นิวเจ็นส์น่ารักมากอยากรถ
นิวเจ็นส์น่ารักมากอยากรถ
นิว เจ็นส์ น่ารัก มาก อยาก ด้วย
[[0 0 0 ... 0 0 0]]
ข้อความนี้มีความเสี่ยงเป็นโรคซึมเศร้าหรือไม่ [0]
```

ภาพที่ 66 ผลการทดสอบโมเดล

```

# 0 Is Not Depressed
# 1 Is Depressed

inp = input()
cleaned = clean_tweet(inp)
print(cleaned)
print(my_tokenise(cleaned))
cleaned = vectorizer.transform([my_tokenise(cleaned)]).toarray()
print(cleaned)
result = logreg.predict(cleaned)
print('ข้อความนี้มีความเสี่ยงเป็นโรคซึมเศร้าหรือไม่ ',result)

⇒ เหนื่อยแล้วอยากตาย
    เหนื่อยแล้วอยากตาย
    เหนื่อย แล้ว อยาก ตาย
    [[0 0 0 ... 0 0 0]]
    ข้อความนี้มีความเสี่ยงเป็นโรคซึมเศร้าหรือไม่ [1]

```

ภาพที่ 67 ผลการทดสอบโมเดล

จากการทดลอง การทดสอบโมเดลที่มีค่า Accuracy สูงที่สุด คือโมเดล Logistic Regression ที่สกัดด้วยวิธี CountVectorizer มีค่า Accuracy 78% พบว่าเมื่อป้อนข้อความ “นิวจีนส์น่ารักมากอยากร้าย” ผลลัพธ์ = 0 หมายถึง ผู้ที่ไม่มีความเสี่ยงเป็นโรคซึมเศร้า และเมื่อป้อนข้อความ “เหนื่อยแล้วอยากตาย” ผลลัพธ์ = 1 คือผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้า

การอภิปรายผล

จากการทดลองด้วยข้อมูลข้อความจากการโพสต์ลงบนทวิตเตอร์ จำนวน 2,400 ชุดข้อมูล โดยใช้โมเดล Logistic Regression, Random Forest, Decision Tree และ K-Nearest Neighbors ทำการเปรียบเทียบและประเมินประสิทธิภาพของโมเดลทั้งนี้ พบว่า ผลจากการทำงานจากทั้ง 4 โมเดลนั้น โมเดล Logistic Regression มีค่าความถูกต้องสูงที่สุด โมเดล Random Forest และ Decision Tree มีค่าความถูกต้องใกล้เคียงกัน และโมเดล K-Nearest Neighbors มีค่าความถูกต้องต่ำที่สุด ในส่วนของการทำงานผู้วิจัยจึงได้นำโมเดล Random Forest ที่มีความถูกต้องสูงที่สุดมาใช้ในการทดสอบการหาผลลัพธ์ จากข้อความในทวิตเตอร์ว่าข้อความนั้นมีความเสี่ยงต่อการเป็นโรคซึมเศร้าหรือไม่ โดยโมเดลอาจจะทำงานโดยยึดจากคีย์เวิร์ดเป็นหลัก ซึ่งในความเป็นจริงนั้นภาษาไทยมีรูปแบบประโยคและการใช้คำที่หลากหลายมากกว่าคีย์เวิร์ดที่ผู้วิจัยได้กำหนดและนำมาใช้ในงานวิจัยนี้อาจเป็นคีย์เวิร์ดเพียงส่วนเล็ก ๆ ของผู้ที่มีความเสี่ยงเป็นโรคซึมเศร้าจำนวนมากใช้การโพสต์ลงทวิตเตอร์ ในอนาคตควรจะหาและรวบรวมข้อมูลเกี่ยวกับผู้ที่เป็นโรคซึมเศร้า เพื่อเพิ่มประสิทธิภาพของโมเดลการทำงาน

เอกสารอ้างอิง

กานุจันทร์เบจร ชูชีพ. (2018). เข้าถึงได้จาก

<https://forestadmin.forest.ku.ac.th/304xxx/?q=system/files/book/5%282018%29%20Logistic%20Regression.pdf>

มหาวิทยาลัยนเรศวร. (ม.ป.ป.). เข้าถึงได้จาก <https://csit.nu.ac.th/kraisak/ds/ds/chapter05/Chapter05.pdf>

โรงพยาบาลรามงาดกอก3. (2019). เข้าถึงได้จาก https://www.bangpakkok3.com/care_blog/view/201