

# Assignment 2 - DS4Biz Y63

## TextScraping\_Classification

### Team Detail

Team Name: sompinandsomshine

### Student 1

Student ID: 61070278

Student Full Name: นายกิตติภณ สุรุ่งเรืองสกุล

### Student 2

Student ID: 61070330

Student Full Name: นางสาวอิงฟ้า ภูมิวนารถ

import package ที่ต้องใช้

- requests, bs4 ใช้ scrape ข้อมูลจากหน้าเว็บ
- pandas(pd), numpy(np) ใช้จัดการข้อมูลที่ scrape มา เช่น ทำเป็น DataFrame, แปลงเป็น csv/txt
- warnings ใช้ปิด warning ที่แจ้งต่อหน้า
- CountVectorizer, TfidfTransformer, TfIdfVectorizer ใช้เตรียมข้อมูลเช่น ทำ term weighting
- nltk ใช้ในการประมวลผลภาษาธรรมชาติ เช่น กำจัด stopwords (คำที่เกิดขึ้นบ่อยและไม่มีอิทธิพลในการจำแนก)
- Pipeline ทำให้การทำงานเป็นลำดับและนาโนมาใช้กับ GridSearchCV
- train\_test\_split, cross\_val\_score, StratifiedKFold, GridSearchCV นำมาทำ model selection และหา best hyperparameter
- neighbors(KNeighborsClassifier), xgboost(xgb), ensemble(RandomForestClassifier) model ที่จะนำมาใช้ในการท่านายว่าแต่ ข้าเป็นประเภทไหน

```
In [1]: import requests
import bs4

import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfIdfVectorizer

import nltk
from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

from sklearn.neighbors import KNeighborsClassifier
import xgboost as xgb
from sklearn.ensemble import RandomForestClassifier

import matplotlib.pyplot as plt
```

## Data Collection

link: [http://www.it.kmitl.ac.th/~teerapong/news\\_archive/index.html](http://www.it.kmitl.ac.th/~teerapong/news_archive/index.html)

ใช้ requests.get ตรวจสอบว่าลิงก์ที่ต้องการ scrape ข้อมูลสามารถเข้าได้มั้ย

```
In [2]: response = requests.get('http://www.it.kmitl.ac.th/~teerapong/news_archive/index.html') #ใช้สังก์ชื่อเราต้อง
    #scrape
print(response) # response 200 คือสามารถเข้าถึงได้ พร้อมดึงข้อมูล

<Response [200]>
```

ใช้ bs4.BeautifulSoup ใน การอ่าน code ของลิงก์ที่เราใส่ไป

```
In [3]: html_page = bs4.BeautifulSoup(response.content, 'html.parser') #เก็บ code ที่ได้ไว้ในตัวแปร html_page
print(html_page)

<!DOCTYPE html>
<html lang="en">
```

```

<html lang="en">
<head>
<title>Online News Archive</title>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<meta content="noindex" name="robots"/>
<meta content="news,articles,news" name="keywords">
<meta content="Breaking News | International Headlines" property="og:title"/>
<meta content="News Archive" property="og:site_name"/>
<meta content="Latest news and more from the definitive brand of quality news." property="og:description"/>
<link href="css/bootstrap.min.css" rel="stylesheet"/>
<script src="js/jquery-3.2.1.slim.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/tether.min.js"></script>
<script src="js/jquery-3.2.1.slim.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/tether.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<style>
    .main{ padding: 0; text-align: center; }
    .footer{ padding: 6px;text-align: center; margin-top: 1em; }

    h1{
        font-size: 180%;
        margin-top: 15px;
        margin-bottom: 15px;
    }
    ul {list-style-type: none;}
    li { margin-top: 5px; }

</style>
</meta></head>
<body>
<div class="container" style="margin-top: 2em;">
<div class="main">

<h1>News Article Archive</h1>
<p>Archive of all news headlines and stories, organised per month.</p>
<ul>
<li>Articles - <a href="month-jan-2017.html">January</a> [118]</li>
<li>Articles - <a href="month-feb-2017.html">February</a> [124]</li>
<li>Articles - <a href="month-mar-2017.html">March</a> [116]</li>
<li>Articles - <a href="month-apr-2017.html">April</a> [118]</li>
<li>Articles - <a href="month-may-2017.html">May</a> [115]</li>
<li>Articles - <a href="month-jun-2017.html">June</a> [115]</li>
<li>Articles - <a href="month-jul-2017.html">July</a> [122]</li>
<li>Articles - <a href="month-aug-2017.html">August</a> [116]</li>
<li>Articles - <a href="month-sep-2017.html">September</a> [113]</li>
<li>Articles - <a href="month-oct-2017.html">October</a> [124]</li>
<li>Articles - <a href="month-nov-2017.html">November</a> [122]</li>
<li>Articles - <a href="month-dec-2017.html">December</a> [115]</li>
</ul>
</div>
<div class="footer">
<span><a href="#">Terms & Conditions</a> | <a href="#">Privacy Policy</a> | <a href="#">Cookie Information</a> </span><br/>
<span>© <span class="thisyear">2019-2020</span> – Original rights holders</span>
</div>
</div>
</body>
</html>

```

In [4]: #ด้องการดึงลิงก์ชื่อของแต่ละเดือน  
 selector = 'body > div > div.main > ul > li > a' #ลิงก์ชื่อของแต่ละเดือนอยู่ใน tags ul>li>a จึงกำหนดให้ไปดึง code ที่อยู่ในส่วนนี้ออกมานะ  
 tags = html\_page.select(selector) #โดยที่ return code ที่ตรงกับ selector ที่กำหนดไว้ว่าที่นี่มันคือจะเก็บไว้ในตัวแปร tags  
 tags #เรียกคุณลักษณะที่ต้องการนั้น

Out[4]: [[January](month-jan-2017.html),  
[February](month-feb-2017.html),  
[March](month-mar-2017.html),  
[April](month-apr-2017.html),  
[May](month-may-2017.html),  
[June](month-jun-2017.html),  
[July](month-jul-2017.html),  
[August](month-aug-2017.html),  
[September](month-sep-2017.html),  
[October](month-oct-2017.html),  
[November](month-nov-2017.html),  
[December](month-dec-2017.html)]

ได้ code ที่เป็นลิงก์มาหมดแล้วแต่ยังไม่พร้อมใช้งานเพรา ลิงก์ที่ด้องการมีหน้าตาแบบนี้  
[http://www.it.kmit.ac.th/~teerapong/news\\_archive/month-jan-2017.html](http://www.it.kmit.ac.th/~teerapong/news_archive/month-jan-2017.html) จึงจะทำการสร้าง list\_archive เพื่อรอดึงลิงก์ที่พร้อมใช้งาน

In [5]: list\_archive = [] #List หรือเก็บลิงก์ที่จะนำไปใช้งานต่อ  
 for tag in tags: #ทำการวน for Loop จากค่าใน tags ที่เก็บ code ที่มีลิงก์อยู่
 list\_archive.append('http://www.it.kmit.ac.th/~teerapong/news\_archive/' + tag['href'])
 '''นำ tag['href'] ซึ่งคือส่วนที่อยู่ใน href เม้ามัน (ex. month-jan-2017.html) มาต่อท้าย http://www.it.kmit.ac.th/~teerapong/news\_archive/
 เพื่อให้ได้ลิงก์ตามที่เราต้องการ และลิงก์นั้นไปเก็บไว้ใน list\_archive ที่สร้างเอาไว้'''
 list\_archive #เรียกคุณลักษณะที่ต้องการนั้น

Out[5]: ['http://www.it.kmit.ac.th/~teerapong/news\_archive/month-jan-2017.html',
 'http://www.it.kmit.ac.th/~teerapong/news\_archive/month-feb-2017.html',
 'http://www.it.kmit.ac.th/~teerapong/news\_archive/month-mar-2017.html',
 ...]

```
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-apr-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-may-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-jun-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-jul-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-aug-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-sep-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-oct-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-nov-2017.html',
'http://www.it.kmit.ac.th/~teerapong/news_archive/month-dec-2017.html']
```

## Create function

สร้าง Function ที่ใช้ในการ scrape

- category ประเภทข่าว -> extract\_category()
- title หัวข้อข่าว -> extract\_title()
- link ลิงค์ข่าว -> extract\_link\_title()

```
In [6]: # สร้าง Function ที่ใช้ในการ scrape category
def extract_category(html_page):
    selector = 'body > div > div.main > table > tbody > tr > td.category' #กำหนดให้ดึงค่าที่อยู่ในตารางในส่วนของ column Article Category
    tags = html_page.select(selector)
    category = [] #สร้าง List ที่อัพเดต category ไว้รองเก็บค่าที่ scrape ได้
    for tag in tags: #ทำการวน for Loop จากตัวแปร tags
        if tag.text.strip() != 'N/A': #จะมาอ้อม category ใส่ใน List ที่อัพเดต category ถ้าข้อมูลที่ดึงมาไม่เป็น N/A (ค่าว่าง)
            category.append(tag.text.strip()) #เพิ่มข้อมูลที่ scrape ได้ใส่ไว้ใน List ที่อัพเดต category
    return category
```

```
In [7]: # สร้าง Function ที่ใช้ในการ scrape title
def extract_title(html_page):
    selector = 'body > div > div.main > table > tbody > tr > td.title > a'
    #กำหนดให้ดึงค่าที่อยู่ในตารางในส่วนของ column Article Title และตั้งไว้ที่ tag a เพราะแท็กที่มีข้อมูลจะเป็น tag a และตัวมีข้อมูลและจะเป็น tag i
    tags = html_page.select(selector)
    title = [] #สร้าง List ที่อัพเดต title ไว้รองเก็บค่าที่ scrape ได้
    for tag in tags: #ทำการวน for Loop จากตัวแปร tags
        title.append(tag.text.strip()) #เพิ่มข้อมูลที่ scrape ได้ใส่ไว้ใน List ที่อัพเดต title
    return title
```

```
In [8]: # สร้าง Function ที่ใช้ในการ scrape Link
def extract_link_title(html_page):
    selector = 'body > div > div.main > table > tbody > tr > td.title > a'
    #กำหนดให้ดึงค่าที่อยู่ในตารางในส่วนของ column Article Title และตั้งไว้ที่ tag a เพราะลิงก์ของข่าวอยู่ tag a
    tags = html_page.select(selector)
    link_title = [] #สร้าง List ที่อัพเดต link_title ไว้รองเก็บค่าที่ scrape ได้
    for tag in tags: #ทำการวน for Loop จากตัวแปร tags
        link_title.append('http://www.it.kmit.ac.th/~teerapong/news_archive/' + tag['href'])
    #ทำการแก้ไขข้อมูลให้เป็นสิ่งที่พร้อมใช้งาน แล้วค่อยเพิ่มข้อมูลใส่ไว้ใน List ที่อัพเดต link_title
    return link_title
```

```
In [9]: #ตัวอย่าง
url = 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-jan-2017.html'
response = requests.get(url)
html_page = bs4.BeautifulSoup(response.content, 'html.parser')
print('example:\n', extract_category(html_page)[0], '\n', extract_title(html_page)[0], '\n', extract_link_title(html_page)[0])

example:
technology
21st-Century Sports: How Digital Technology Is Changing the Face Of The Sporting Industry
http://www.it.kmit.ac.th/~teerapong/news_archive/article-jan-0418.html
```

สร้าง Function extract\_news\_archive\_info() สำหรับเรียกใช้งาน Function ในการ scrape category ประเภทข่าว, title หัวข้อข่าว, link ลิงค์ข่าว ทั้งหมด โดยใช้สีเคนกเดือนก็จะดึง category ประเภทข่าว, title หัวข้อข่าว, link ลิงค์ข่าว ในเดือนนั้นออกมากทั้งหมด และจะ return กลับมาในรูปของ DataFrame

```
In [10]: def extract_news_archive_info(url): #รับลิงก์ของเดือนที่ต้องการดึงข้อมูล
    response = requests.get(url)
    html_page = bs4.BeautifulSoup(response.content, 'html.parser') #ดึงหน้าเว็บของลิงก์นั้น

    #ເອานັນໄວ້ທີ່ລົງອາມາໄປໃສໃນ argument ມີລະ Function ທີ່ໄດ້ສ້າງໄວ້ພໍ່ດີ່ດັ່ງข່ອມູນ ພລັບພົກທີ່ໄດ້ກັບລັນມາຈະອູ່ໃນຮູບຂອງ List
    category = extract_category(html_page)
    title = extract_title(html_page)
    link_title = extract_link_title(html_page)

    #ເອົາຂ່ອມູນທີ່ຕື່ມໃນຮູບຂອງ List ທີ່ໄດ້ກັບລັນມາແບ່ງໃຫ້ອູ່ໃນຮູບຂອງ pandas Series ເພື່ອທີ່ຈະໄດ້ອ້າຂ່ອມູນທີ່ 3 ຕົວນາ concat ກົນໃໝ່ເປັນ DataFrame
    category = pd.Series(category)
    title = pd.Series(title)
    link_title = pd.Series(link_title)

    #ເອົາຂ່ອມູນທີ່ 3 ຕົວນາໃນຮູບຂອງ pandas Series ແລ້ວນາ concat ກົນໃນແນວຄອສົມນີ້ (ເອົາຄອສົມນີ້ມາຕອງກົນ)
    result = pd.concat([category, title, link_title], axis=1)
    #rename ສືບ column ໃຫ້ຄຽງກົນທີ່ຕົ້ນການໃຊ້ຈຳນາ ໂດຍຈະເປັນສິນ້ນີ້ເປັນ category-ປະເທດຂ່າງ, title-ຫ້າຂໍ້ຂ່າງ, Link-ລິ້ງກົດ
    result = result.rename(columns={0: "category", 1: "title", 2: "link"})
```

```
return result #return DataFrame
```

```
In [11]: #ดูว่าอย่าง ลืมที่คือ DataFrame ที่ Function extract_news_archive_info() return กลับมา  
#ข้อมูลที่ return กลับมายังเป็นข่าวทั้งหมดในเดือนที่เราส่งไป  
url = 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-jan-2017.html'  
extract_news_archive_info(url)
```

```
Out[11]:
```

category	title	link
0 technology	21st-Century Sports: How Digital Technology Is...	http://www.it.kmit.ac.th/~teerapong/news_arch...
1 business	Asian quake hits European shares	http://www.it.kmit.ac.th/~teerapong/news_arch...
2 technology	BT offers free net phone calls	http://www.it.kmit.ac.th/~teerapong/news_arch...
3 business	Barclays shares up on merger talk	http://www.it.kmit.ac.th/~teerapong/news_arch...
4 sport	Barkley fit for match in Ireland	http://www.it.kmit.ac.th/~teerapong/news_arch...
...	...	...
113 business	Worldcom boss 'left books alone'	http://www.it.kmit.ac.th/~teerapong/news_arch...
114 technology	Xbox 2 may be unveiled in summer	http://www.it.kmit.ac.th/~teerapong/news_arch...
115 technology	Xbox power cable 'fire fear'	http://www.it.kmit.ac.th/~teerapong/news_arch...
116 business	Yangtze Electric's profits double	http://www.it.kmit.ac.th/~teerapong/news_arch...
117 business	Yukos loses US bankruptcy battle	http://www.it.kmit.ac.th/~teerapong/news_arch...

118 rows × 3 columns

## Create DataFrame

สร้าง DataFrame เพื่อรับข้อมูลที่ scrape ได้

ซึ่งข้อมูลที่เรา scrape คือลิงก์ข่าวในแต่ละเดือนใน list\_archive โดยเราจะ scrape category ประเภทข่าว, title หัวข้อข่าว, link ลิงก์ข่าว

```
In [12]: #สร้าง DataFrame ที่อ่านข้อมูลที่ scrape ได้และกำหนดชื่อ column ตามสิ่งที่จะ scrape  
df = pd.DataFrame(columns=['category', 'title', 'link'])  
df #ได้ df เป็นล่ามาที่รับข้อมูลที่ scrape ได้
```

```
Out[12]:
```

category	title	link
----------	-------	------

## เอา list\_archive (ลิงก์เดือนทั้งหมด) ไปเข้า Function extract\_news\_archive\_info()

ขั้นตอนนี้ทำเพื่อให้ได้ข่าวทั้งหมดในทุกๆเดือนมารวมกันเป็น DataFrame อีกแล้ว

```
In [13]: list_archive
```

```
Out[13]: ['http://www.it.kmit.ac.th/~teerapong/news_archive/month-jan-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-feb-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-mar-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-apr-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-may-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-jun-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-jul-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-aug-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-sep-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-oct-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-nov-2017.html',  
 'http://www.it.kmit.ac.th/~teerapong/news_archive/month-dec-2017.html']
```

เอาแต่ละลิงก์ใน list\_archive ไปเข้า function extract\_news\_archive\_info() และเพิ่มเข้าไปใน DataFrame ที่เราเตรียมไว้

```
In [14]: for url in list_archive: #วน for Loop เพื่อเอาลิงก์เดือนที่เหลือไปเข้า function  
    result = extract_news_archive_info(url) #เอาลิงก์เดือนไปเข้า function และเก็บ DataFrame ที่ return กลับมาไว้  
    df = df.append(result, ignore_index=True) #เอา result ที่ได้จาก function ไปเพิ่มใน DataFrame ที่สร้างไว้
```

```
In [15]: df #หน้าตาของ DataFrame ที่มีข้อมูล category ประเภทข่าว, title หัวข้อข่าว, link ลิงก์ข่าว ทั้งหมด
```

```
Out[15]:
```

category	title	link
0 technology	21st-Century Sports: How Digital Technology Is...	http://www.it.kmit.ac.th/~teerapong/news_arch...
1 business	Asian quake hits European shares	http://www.it.kmit.ac.th/~teerapong/news_arch...
2 technology	BT offers free net phone calls	http://www.it.kmit.ac.th/~teerapong/news_arch...
3 business	Barclays shares up on merger talk	http://www.it.kmit.ac.th/~teerapong/news_arch...
4 sport	Barkley fit for match in Ireland	http://www.it.kmit.ac.th/~teerapong/news_arch...
...	...	...
1403 sport	Woodward eyes Brennan for Lions	http://www.it.kmit.ac.th/~teerapong/news_arch...
1404 business	WorldCom trial starts in New York	http://www.it.kmit.ac.th/~teerapong/news_arch...
1405 business	Yukos accused of lying to court	http://www.it.kmit.ac.th/~teerapong/news_arch...
1406 business	Yukos drops banks from court bid	http://www.it.kmit.ac.th/~teerapong/news_arch...
1407 sport	Zambia confident and cautious	http://www.it.kmit.ac.th/~teerapong/news_arch...

1408 rows × 3 columns

ลองเช็คค่า null

```
In [16]: df.isna().sum()  
#ไม่มีค่า null เพราะ clean ไปตอน scrape ข้อมูลแล้ว
```

```
Out[16]: category    0  
title      0  
link       0  
dtype: int64
```

## Add id column

แต่ละข่าวมี id ของรั้นเองจะมาทำการเก็บแยกเพิ่มอีก colum หนึ่ง

```
In [17]: #ในแต่ละสิ่งที่ข่าวจะมี id ของข่าว放ลงตัวอักษร  
df['link'].iloc[0]
```

```
Out[17]: 'http://www.it.kmit.ac.th/~teerapong/news_archive/article-jan-0418.html'
```

```
In [18]: #ใช้ index slicing ในการหา id ข่าว โดยจะเริ่มนับตั้งแต่ตัวที่ 50 ถึงตัวที่ 6 จากหลังสุด  
df['id'] = df['link'].str[50:-5]  
df
```

```
Out[18]:
```

	category	title	link	id
0	technology	21st-Century Sports: How Digital Technology Is...	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0418
1	business	Asian quake hits European shares	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0027
2	technology	BT offers free net phone calls	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0631
3	business	Barclays shares up on merger talk	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-2105
4	sport	Barkley fit for match in Ireland	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-3300
...	...	...	...	...
1403	sport	Woodward eyes Brennan for Lions	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-2238
1404	business	WorldCom trial starts in New York	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-2334
1405	business	Yukos accused of lying to court	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-2095
1406	business	Yukos drops banks from court bid	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-1351
1407	sport	Zambia confident and cautious	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-0068

1408 rows × 4 columns

## Get Content

ดึงข้อมูลในสิ่งที่ข่าว

```
In [19]: list_content = [] #สร้าง List ชื่อ list_content ไว้เก็บข้อมูล content ทั้งหมด
```

สร้าง Function ที่ใช้ในการตั้งข้อมูลข่าวในสิ่งที่ข่าวที่ใส่เข้าไป

```
In [20]: def get_content(url): #รับลิงก์ข่าวและเก็บไว้ในชื่อ url  
    response = requests.get(url)  
    html_page = bs4.BeautifulSoup(response.content, 'html.parser')  
  
    selector = 'body > div > div.main > p' #เลือกหัวข่าวอยู่ในส่วน tag p  
    #เมื่อจากเนื้อหาข่าวมีหลัก喻อยู่หน้า (1 ยังหน้า : 1 tag p) จึงใช้ select ในการตั้ง ค่าที่ได้จะอยู่ในรูปของ List  
    tags = html_page.select(selector)  
  
    content = '' #สร้างตัวแปร content รอบเก็บที่หัวข่าวในรูปของ string  
    for tag in tags: #วน for loop ในตัวแปร tags  
        if tag.text.strip() != '': #ในเมื่อหัวข่าวจะมีส่วนที่เป็นเว้นวรูปหน้าบล็อก ซึ่งเราไม่ต้องการเก็บส่วนนั้น จึงจะทำการเก็บ  
        เดพาระส่วนที่ไม่ได้เป็นวรูป  
        if '.html' in str(tag): #ในส่วนล่างสุดในบางช่วงเวลาจะมีลิงก์ให้กดคลิกไปที่หน้าหลัก ซึ่งเราจะไม่ต้องการเก็บ  
        ซึ่งจะทำการข้ามไป/  
        pass  
        elif 'Comments are closed for this article' in str(tag): #ในส่วนล่างสุดในบางช่วงเวลาจะมีข้อความว่าปิดให้ไม่สามารถcomment ได้ ซึ่งเรา  
        ก็ไม่ต้องการเก็บข้อมูลนี้ จึงจะทำการข้ามไป/  
        pass  
        else:  
            content += tag.text.strip() + ' ' #หลังจากข้ามส่วนที่ไม่ต้องการเก็บไปหนึ่งครั้งจะเพิ่มส่วนต่อไปของการเก็บ  
            ไว้ในตัวแปร content  
    list_content.append(content.strip()) #หลังจากจบ Loop ก็จะทำการเพิ่มเนื้อหาข่าวไปในตัวแปร list_content
```

```
In [21]: #df['link'] ต้องสิ่งที่ทั้งหมด  
print('example:', df['link'][0])
```

```
example: http://www.it.kmit.ac.th/~teerapong/news_archive/article-jan-0418.html
```

```
In [22]: for i in range(len(df)): #วน for loop ตามความยาวทั้งหมดของ df  
    get_content(df['link'].iloc[i]) #เอาลิงก์ข่าวใส่ไปใน function get_content()
```

หลังจากวน for loop เพื่อสิ่งที่ข่าวใส่ไปใน function get\_content() ครบหมดแล้ว ในตัวแปร list\_content จะมีเนื้อหาข่าวทั้งหมด เราจะแปลง list\_content จาก list ให้เป็น pandas Series เพื่อที่จะได้เข้าไป concat กับ df ในสิ่งที่ข้อมูล

```
In [23]: content = pd.Series(list_content) #แปลง list_content จาก List ให้เป็น pandas Series และเก็บไว้ในชื่อ content  
df = pd.concat([df, content], axis=1) #เอา content ที่เป็น pandas Series มารวมกับ df ที่มีอยู่แล้วในแนวคอลัมน์ (หากคอลัมน์มีชื่อเดียวกัน)
```

```
df = df.rename(columns={0: "content"}) #rename ชื่อ column เพื่อให้สะดวกต่อการใช้งาน โดยจะเปลี่ยนชื่อเป็น content-  
ນີ້ຂ່າຍ້າ  
df
```

Out[23]:

	category	title	link	id	content
0	technology	21st-Century Sports: How Digital Technology Is...	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0418	The sporting industry has come a long way sinc...
1	business	Asian quake hits European shares	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0027	Asian quake hits European shares Shares in Eur...
2	technology	BT offers free net phone calls	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0631	BT is offering customers free internet telepho...
3	business	Barclays shares up on merger talk	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-2105	Barclays shares up on merger talk Shares in UK...
4	sport	Barkley fit for match in Ireland	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-3300	England centre Olly Barkley has been passed fi...
...	...	...	...	...	...
1403	sport	Woodward eyes Brennan for Lions	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-2238	Woodward eyes Brennan for Lions Toulouse's for...
1404	business	WorldCom trial starts in New York	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-2334	The trial of Bernie Ebbers, former chief execu...
1405	business	Yukos accused of lying to court	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-2095	Yukos accused of lying to court Russian oil fi...
1406	business	Yukos drops banks from court bid	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-1351	Russian oil company Yukos has dropped the thre...
1407	sport	Zambia confident and cautious	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-dec-0068	Zambia's technical director, Kalusha Bwalya is...

1408 rows × 5 columns

## save DataFarame to csv/txt

เพื่อให้ง่ายต่อการใช้ในภายหลัง

In [24]: `df.to_csv('datastore/data.csv', index=False)`

In [25]: `df = pd.read_csv('datastore/data.csv')`  
`df.head()`

Out[25]:

	category	title	link	id	content
0	technology	21st-Century Sports: How Digital Technology Is...	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0418	The sporting industry has come a long way sinc...
1	business	Asian quake hits European shares	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0027	Asian quake hits European shares Shares in Eur...
2	technology	BT offers free net phone calls	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-0631	BT is offering customers free internet telepho...
3	business	Barclays shares up on merger talk	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-2105	Barclays shares up on merger talk Shares in UK...
4	sport	Barkley fit for match in Ireland	http://www.it.kmit.ac.th/~teerapong/news_arch...	article-jan-3300	England centre Olly Barkley has been passed fi...

## คัดให้เนื้อเฉพาะที่ใช้ในการวิเคราะห์

- AllArticles\_OnlyHeading
- AllArticles\_OnlyContent
- AllArticles\_HeadingPlusContent
- Category

In [26]: `AllArticles_OnlyHeading = df['title']`  
`AllArticles_OnlyContent = df['content']`  
`AllArticles_HeadingPlusContent = df[['title','content']]`  
`Category = df['category']`

CSV

save เป็นไฟล์ประเภท csv

In [27]: `AllArticles_OnlyHeading.to_csv('datastore/AllArticles_OnlyHeading.csv', index=False)`

In [28]: `AllArticles_OnlyContent.to_csv('datastore/AllArticles_OnlyContent.csv', index=False)`

```
In [29]: AllArticles_HeadingPlusContent.to_csv('datastore/AllArticles_HeadingPlusContent.csv', index=False)
```

```
In [30]: Category.to_csv('target/category.csv', index=False)
```

txt

save เป็นไฟล์ประเภท txt

```
In [31]: AllArticles_OnlyHeading.to_csv(r'datastore/AllArticles_OnlyHeading.txt', header=None, index=None, sep=' ', mode='a')
```

```
In [32]: AllArticles_OnlyContent.to_csv(r'datastore/AllArticles_OnlyContent.txt', header=None, index=None, sep=' ', mode='a')
```

```
In [33]: AllArticles_HeadingPlusContent.to_csv(r'datastore/AllArticles_HeadingPlusContent.txt', header=None, index=None, sep=' ', mode='a')
```

```
In [34]: Category.to_csv(r'target/category.txt', header=None, index=None, sep=' ', mode='a')
```

```
In [ ]:
```

## Modeling

ทำการ Tuning hyperparameter และหา Model ที่ดีที่สุดในการทำงาน

- K-Nearest Neighbors
- XGBoost
- Random forest

โหลดข้อมูลที่เคย save เก็บไว้

- AllArticles\_OnlyContent.txt ศิลป์ข้อมูลที่มีเฉพาะเนื้อหาข่าวโดยจะเก็บข้อมูลในตัวแปร raw\_documents
- target/category.txt ศิลป์ส่วนของ target หรือเฉลยว่าป้าดังกล่าวอยู่ในประเภทอะไร

```
In [2]: with open("datastore/AllArticles_OnlyContent.txt","r", encoding="utf8") as f: #ตรวจสอบจำนวนบรรทัด
    raw_documents = f.read().splitlines() #ใช้เมธode string ที่รูดชื่นบรรทัดใหม่คือ จุดที่มี '\n' และคืนค่ากลับมาเป็นชื่อ
    # ข้อมูลประเภท List
f.close() #ปิดไฟล์หลังใช้เสร็จ
print("Read %d raw text documents" % len(raw_documents)) #ตรวจสอบจำนวนบรรทัด
```

Read 1408 raw text documents

```
In [3]: raw_documents[0] #เรียกดูข้อมูลตัวแรกที่อ่านมา
```

```
Out[3]: ''The sporting industry has come a long way since the '60s. It has carved out for itself a niche with its roots so deep that I cannot fathom the sports industry showing any sign of decline any time soon - or later. The reason can be found in this seemingly subtle difference - other industries have customers; the sporting industry has fans. Vivek Ranadivé, leader of the ownership group of the NBA's Sacramento Kings, explained it beautifully, "Fans will paint their face purple, fans will evangelize. ... Every other CEO in every business is dying to be in our position – they're dying to have fans." While fan passion alone could almost certainly keep the industry going, leagues and sporting franchises have decided not to rest on their laurels. The last few years have seen the steady introduction of technology into the world of sports - amplifying fans' appreciation of games, enhancing athletes' public profiles and informing their training methods, even influencing how contests are waged. Also, digital technology in particular has helped to create an alternative source of revenue, besides the games themselves - corporate sponsorship. They achieved this by capitalizing on the ardor of their customer base - sorry, fan base.''
```

```
In [4]: with open("target/category.txt","r", encoding="utf8") as f: #ตรวจสอบจำนวนบรรทัด
    y = f.read().splitlines() #ใช้เมธode string ที่รูดชื่นบรรทัดใหม่คือ จุดที่มี '\n' และคืนค่ากลับมาเป็นชื่อ
    # ข้อมูลประเภท List
f.close() #ปิดไฟล์หลังใช้เสร็จ
print("Read %d" % len(y)) #ตรวจสอบจำนวนบรรทัด
```

Read 1408

## เข้าใจข้อมูล imbalance ใหม่

```
In [5]: #สร้าง dataframe จากข้อมูลที่ดึงเข้ามา
all_data = pd.DataFrame(list(zip(raw_documents, y)),
                        columns=['news', 'target'])
all_data
```

Out[5]:

	news	target
0	"The sporting industry has come a long way sin...	technology
1	"Asian quake hits European shares Shares in Eu...	business
2	"BT is offering customers free internet teleph...	technology
3	"Barclays shares up on merger talk Shares in U...	business
4	"England centre Olly Barkley has been passed f...	sport
...	...	...
1403	"Woodward eyes Brennan for Lions Toulouse's fo...	sport
1404	"The trial of Bernie Ebbers, former chief exec...	business
1405	"Yukos accused of lying to court Russian oil f...	business
1406	"Russian oil company Yukos has dropped the thr...	business

1408 rows × 2 columns

```
In [6]: #นับจำนวนแต่ละ target และเก็บไว้ที่ตัวแปร y_axis
y_axis = all_data['target'].value_counts()
y_axis
```

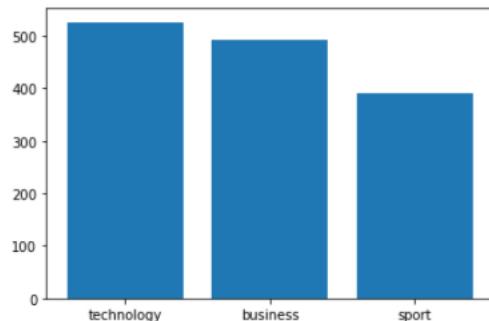
```
Out[6]: sport      526
business    491
technology  391
Name: target, dtype: int64
```

```
In [7]: #เก็บ target ไว้ที่ตัวแปร x_axis
x_axis = all_data['target'].unique()
x_axis
```

```
Out[7]: array(['technology', 'business', 'sport'], dtype=object)
```

```
In [8]: #plot กราฟเพื่อดูว่ามีปัญหา imbalance ไหม
plt.bar(x_axis, y_axis)
```

```
Out[8]: <BarContainer object of 3 artists>
```



ปัญหานี้ใน imbalance สามารถเอาไปใช้ได้ปกติ ไม่ต้องทำ upsampling หรือ downsampling

## Text preprocessing

สร้าง function ที่ต้องใช้ในการทำ preprocessing

```
In [9]: # X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
#แบ่งข้อมูลเป็น 2 ส่วน train 80% test 20%
```

```
In [10]: # Tokenizer and WordNet Lemmatizer with POS (part of speech)
def lemma_tokenizer_w_pos_tag(text):

    def convert_tags(tag):
        #แปลง tag ที่มี vbd vbg vbz ให้มีเป็น v
        if tag == 'vbd' or tag == 'vbg' or tag == 'vbz':
            return 'v'
        #แยกหนีดีออกจากคำอื่นที่ซ้ำกันให้มีเป็น n
        else:
            return 'n'

    standard_tokenizer = CountVectorizer().build_tokenizer() #สร้างตัวที่ใช้แปลง text (string) ที่เข้ามาออกเป็น tokens
    tokens = standard_tokenizer(text) #แปลง text (string) ที่เข้ามาออกเป็น tokens
    tokens_with_pos_tag = nltk.pos_tag(tokens) #ติด pos_tag ให้กับแต่ละ token

    lemmatizer = nltk.WordNetLemmatizer() #สร้างตัวที่ใช้แปลงรูปของคำให้อยู่รูปฟอร์มพื้นฐาน (รากค่า)
    lemma_tokens = [] #สร้าง list ที่จะ放 lemma_tokens เพื่อรอเก็บคำหัวลงจากแปลงด้วย pos_tag อันใหม่
    for token in tokens_with_pos_tag: #วนลูป ในตัวแปร tokens_with_pos_tag
        new_tag = convert_tags(token[1].lower())
        '''token[1] คือ pos_tag ที่เพิ่มเข้ามาเมื่อผ่านการทำ tokenization (nltk.pos_tag(tokens)) ใช้ .lower() เพื่อปรับให้เป็นตัวเล็ก
        และจึงท้ากรสูงไปที่ function convert_tags() เพื่อแปลงให้เกิดข้อต่อ 2 แบบคือถ้าเป็น
        VBD verb: past tense took,
        VBG verb: gerund/present participle taking,
        VBZ verb: 3rd person sing. present takes
        จะแปลงให้เหลือแค่ v นอกจากนั้นจะแปลงเป็น n ทั้งหมด
        เช่น pos_tag ที่ได้มาใหม่ก็จะเป็น new_tag'''

        lemma_tokens.append(lemmatizer.lemmatize(token[0], new_tag))
        '''จะนำ pos_tag ใหม่ที่ได้มาใช้กับ token[0] หรือคำหัวตอนนั้นที่ต้องการทำ token[0] ให้แปลงให้อยู่ในรูปของ pos_tag อันใหม่
        เช่น token[0] = amplifying, new_tag = v ผลลัพธ์ที่ได้คือ amplify และจะเอาผลลัพธ์ที่เพิ่มเข้าไปใน list ชื่อ Lemma_tokens''''

    return lemma_tokens
```

ด้วยภาษาเมืองเอกสารเข้าไปใน function lemma\_tokenizer\_w\_pos\_tag()

```
In [11]: lemma_tokenizer_w_pos_tag('amplifying')
```

```
Out[11]: ['amplify']
```

```
# ตัวอย่าง vectorizer / คำนวณ document-term matrix และคำนวณ tf-idf
#default ของ analyzer คือ word, กำจัด stop_words
non_weight = TfidfVectorizer(stop_words="english", min_df = 0, tokenizer=lemma_tokenizer_w_pos_tag)
```

```
In [13]: #Term weighting ใน Scikit-Learn เรากำหนดร่างที่ TF-IDF weighted document-term matrix โดยใช้ TfidfVectorizer() แทน CountVectorizer()
#default ของ analyzer คือ word, กำจัด stop_words
weight = TfidfVectorizer(analyzer='word', ngram_range=(1,3),
                         min_df = 0, stop_words = 'english', sublinear_tf=True, lowercase=True)
```

สร้าง dict ขึ้น accuracy\_dict เพื่อเก็บค่า accuracy ของแต่ละ model

```
In [14]: accuracy_dict = dict() #สร้าง dict ใหม่ accuracy_dict เพื่อเก็บค่า accuracy
```

```
In [ ]:
```

ทำการ transform raw documents โดยจะแบ่งเป็น 2 แบบคือ non\_weight และ weight

- non\_weight จะเก็บไว้ที่ตัวแปร non\_weight\_raw\_documents
- weight จะเก็บไว้ที่ตัวแปร weight\_raw\_documents

```
In [15]: # transform raw documents เมื่อไว้ที่ตัวแปร non_weight_raw_documents
non_weight_raw_documents = non_weight.fit_transform(raw_documents)
```

```
In [16]: # transform raw documents เมื่อไว้ที่ตัวแปร weight_raw_documents
weight_raw_documents = weight.fit_transform(raw_documents)
```

## GridSearchCV Function

สร้าง Function ที่นำไปในการทำ hyperparameter tuning

```
In [184]: def tuning_params(model, params): #รับค่าเป็น model และ params ที่จะทำการ tune
    folds = 3 #กำหนด fold ที่จะใช้ในการทำ K-fold
    skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001) #ทำการก้าวนดเครื่องมือ K-fold จำนวน 3 folds โดย StratifiedKFold เป็นการแบ่งช่วงลุบแบบหนึ่ง
    #อ้าง Classifier, params ไม่ลงไว้และ ก้าวนด cv ด้วย k-fold ที่เราก้าวนดไว้ในตัวแปร skf
    grid = GridSearchCV(model, params, verbose = 3, cv=skf.split(X_train,y_train), n_jobs = -1)
    grid.fit(X_train, y_train) #อ้าง GridSearchCV ที่ fit สำหรับ train set
    return(grid.best_params_) #สืบค่า hyperparameter ที่ดีที่สุดกลับไป
```

## K-Nearest Neighbors

ในการทำ Tuning parameter จะต้องก้าวนด parameter ที่เราจะทำการวิเคราะห์เช่น knn จะต้องมีการก้าวนดจำนวน k

```
In [240]: #ทำการก้าวนด parameter ที่จะเอาไปใช้เครื่องเรียน โดยค่าที่ก้าวนดไว้คือ parameter ของ knn
params = {
    'n_neighbors' : list(np.arange(1,50,2)), #ให้ทำการทดสอบหา k ที่ดีที่สุดตั้ง 1-50 โดยขยายตั้ง 2 เช่น 1, 3, 5, ...
    'weights' : ['uniform', 'distance'], #weight function ที่ใช้ในการทำนาย
    'metric' : ['euclidean', 'manhattan'] #คิดระยะทางด้วยวิธีไหน
}
```

## KNN Tuning parameter แบบไม่ weight

ทำการแบ่ง data ออกเป็น 2 ส่วนคือ **train set** (X\_train, y\_train) กับ **test set** (X\_test, y\_test) โดยส่วน train set จะใช้ในการทำ Tuning parameter, Model selection และ Train model โดยการทำ Tuning parameter จะใช้ **GridSearchCV**, การทำ Model selection จะใช้ **cross\_val\_score** โดยเลือก model ที่มีค่า accuracy สูงที่สุด

```
In [246]: X_train, X_test, y_train, y_test = train_test_split(non_weight_raw_documents, y, test_size=0.2, random_state=42)
#แบ่งช่วงลุบเป็น 2 ส่วน train 80% test 20%
```

เอาค่าทั้งหมดที่เราก้าวนดไว้มาใส่ GridSearchCV

```
In [187]: knn_non_weight=KNeighborsClassifier() #เรียกใช้ KNN
```

```
In [188]: knn_non_weight_best_params = tuning_params(knn_non_weight, params) #ทำ hyperparameter tuning
knn_non_weight_best_params #แสดงผลที่ได้ก้าวนด
```

Fitting 3 folds for each of 100 candidates, totalling 300 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    1.0s
[Parallel(n_jobs=-1)]: Done 152 tasks      | elapsed:   2.7s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:   8.6s finished
```

```
Out[188]: {'metric': 'euclidean', 'n_neighbors': 37, 'weights': 'uniform'}
```

## KNN แบบไม่ weight Best params

นำ best parameter ที่หาได้มาใส่ที่ classifier

ทำการ fit model กับ parameter ที่ดีที่สุดที่เราระบุมาได้

```
In [247]: # กำหนด parameter ตามที่เราได้
knn_non_weight.set_params(**knn_non_weight_best_params) #ทำการ set hyperparameter ที่ได้จากการท่า gridsearchcv
knn_non_weight.fit(X_train, y_train) #ทำการ fit model
```

```
Out[247]: KNeighborsClassifier(metric='euclidean', n_neighbors=37)
```

หา accuracy ด้วย cross\_val\_score

```
In [248]: #cross_val_score จะ return accuracy ออกมากเป็น List โดยจำนวน accuracy ที่ return กลับมามากเท่ากับ cv ที่เรากำหนด
# คำนวณ accuracy ของทั้งหมด
scores = cross_val_score(knn_non_weight, X_train, y_train, scoring="accuracy", cv=10)
print("knn_non_weight model: ", scores.mean()) #เอ้า List ของ accuracy มาหา mean
knn_non_weight model:  0.9733723135271809
```

```
In [249]: #เพิ่ม accuracy ที่ผลลัพธ์ที่ได้ใน accuracy_dict ที่สร้างไว้ โดยให้ key เป็นชื่อ model และ value เป็น accuracy
accuracy_dict['knn_non_weight'] = scores.mean()
```

### Test KNN แบบ weight

เอา model ที่ใช้ parameter ที่ดีที่สุดมา test กับ test set ที่แบ่งไว้ดอนแรก

```
In [250]: #นำ X_test มาห้ามัย
predicted = knn_non_weight.predict(X_test)
```

```
In [251]: # หา accuracy โดยเอา เดลตของ X_test มาเทียบกับ predicted หรือค่าที่ model เรานำมา
accuracy_score(y_test, predicted)
```

```
Out[251]: 0.9468085106382979
```

```
In [ ]:
```

### KNN Tunning parameter แบบ weight

```
In [252]: X_train, X_test, y_train, y_test = train_test_split(weight_raw_documents, y, test_size=0.2, random_state=42)
#แบ่งข้อมูลเป็น 2 ส่วน train 80% test 20%
```

เอาค่าทั้งหมดที่เราห้ามไว้มาใส่ GridSearchCV

```
In [195]: knn_weight=KNeighborsClassifier() #เรียกใช้ KNN
```

```
In [196]: knn_weight_best_params = tuning_params(knn_weight, params) #ท่า hyperparameter tuning
knn_weight_best_params #แสดงผลที่ได้กลับมา
```

Fitting 3 folds for each of 100 candidates, totalling 300 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 184 tasks      | elapsed:   6.7s
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:  17.7s finished
```

```
Out[196]: {'metric': 'euclidean', 'n_neighbors': 41, 'weights': 'uniform'}
```

### KNN แบบweight Best params

นำ best parameter ที่หาได้มาใส่ที่ classifier

ทำการ fit model กับ parameter ที่ดีที่สุดที่เราระบุมาได้

```
In [253]: # กำหนด parameter ตามที่เราได้
knn_weight.set_params(**knn_weight_best_params) #ทำการ set hyperparameter ที่ได้จากการท่า gridsearchcv
knn_weight.fit(X_train, y_train) #ทำการ fit model
```

```
Out[253]: KNeighborsClassifier(metric='euclidean', n_neighbors=41)
```

หา accuracy ด้วย cross\_val\_score

```
In [254]: #cross_val_score จะ return accuracy ออกมากเป็น List โดยจำนวน accuracy ที่ return กลับมามากเท่ากับ cv ที่เรากำหนด
# คำนวณ accuracy ของทั้งหมด
scores = cross_val_score(knn_weight, X_train, y_train, scoring="accuracy", cv=10)
print("knn_weight model: ", scores.mean()) #เอ้า List ของ accuracy มาหา mean
knn_weight model:  0.9751422250316055
```

```
In [255]: #เพิ่ม accuracy ที่ผลลัพธ์ที่ได้ใน accuracy_dict ที่สร้างไว้ โดยให้ key เป็นชื่อ model และ value เป็น accuracy
accuracy_dict['knn_weight'] = scores.mean()
```

### Test KNN แบบ weight

เอา model ที่ใช้ parameter ที่ดีที่สุดมา test กับ test set ที่แบ่งไว้ดอนแรก

```
In [256]: #นำ X_test มาทำนาย
predicted = knn_weight.predict(X_test)

In [257]: # หา accuracy โดยเอา เดลต์ของ X_test มาเทียบกับ predicted หรือค่าที่ model เรากำหนด
accuracy_score(y_test, predicted)

Out[257]: 0.9680851063829787
```

## XGBoost

```
In [202]: #ค่าที่กำหนดไว้ซึ่ง parameter ของ XGBoost
params = {
    'min_child_weight': [1, 5],
    'gamma': [0.5, 1, 1.5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'max_depth': [3, 4, 5]
}
```

### XGBoost Tunning parameter แบบไม่ weight

```
In [258]: X_train, X_test, y_train, y_test = train_test_split(non_weight_raw_documents, y, test_size=0.2, random_state=42)
#แบ่งชื่ออยู่เป็น 2 ส่วน train 80% test 20%
```

เราค่าทั้งหมดที่เรากำหนดไว้มาใส่ GridSearchCV

```
In [204]: import xgboost as xgb
xgb_non_weight = xgb.XGBClassifier() #เรียกใช้ XGBoost

In [205]: xgb_non_weight_best_params = tuning_params(xgb_non_weight, params) #หา hyperparameter tuning
xgb_non_weight_best_params #แสดงผลที่ได้ก้าวมา

Fitting 3 folds for each of 162 candidates, totalling 486 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:   8.8s
[Parallel(n_jobs=-1)]: Done 104 tasks      | elapsed:  51.0s
[Parallel(n_jobs=-1)]: Done 264 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done 486 out of 486 | elapsed:  4.6min finished

Out[205]: {'colsample_bytree': 0.6,
 'gamma': 0.5,
 'max_depth': 5,
 'min_child_weight': 1,
 'subsample': 0.6}
```

### XGBoost แบบไม่ weight Best params

นำ best parameter ที่หาได้มาใส่ที่ classifier

ทำการ fit model กับ parameter ที่ดีที่สุดที่วิเคราะห์มาได้

```
In [259]: # กำหนด parameter ตามที่หาได้
xgb_non_weight.set_params(**xgb_non_weight_best_params) #ทำการ set hyperparameter ที่ได้จากการท่า gridsearchcv
xgb_non_weight.fit(X_train, y_train) #ทำการ fit model

Out[259]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=0.6, gamma=0.5, gpu_id=-1,
                      importance_type='gain', interaction_constraints='',
                      learning_rate=0.300000012, max_delta_step=0, max_depth=5,
                      min_child_weight=1, missing=nan, monotone_constraints='('),
                      n_estimators=100, n_jobs=0, num_parallel_tree=1,
                      objective='multi:softprob', random_state=0, reg_alpha=0,
                      reg_lambda=1, scale_pos_weight=None, subsample=0.6,
                      tree_method='exact', validate_parameters=1, verbosity=None)
```

หา accuracy ด้วย cross\_val\_score

```
In [260]: #cross_val_score จะ return accuracy ออกมาเป็น List โดยจำนวน accuracy ที่ return ก็จำนวนเท่ากับ cv ที่เรากำหนด
scores = cross_val_score(xgb_non_weight, X_train, y_train, scoring="accuracy", cv=10)
print("xgb_non_weight model: ", scores.mean()) #เอาม List ของ accuracy มาหา mean
```

xgb\_non\_weight model: 0.9671381163084705

```
In [261]: #เพิ่ม accuracy ที่ได้รับมาลงใน accuracy_dict ที่สร้างไว้ โดยให้ key เป็นชื่อ model และ value เป็น accuracy
accuracy_dict['xgb_non_weight'] = scores.mean()
```

### Test XGBoost แบบไม่ weight

เอา model ที่ใช้ parameter ที่ดีที่สุดมา test กับ test set ที่แบ่งไว้ดอนแกรก

```
In [262]: #นำ X_test มาหานาย  
predicted = xgb_non_weight.predict(X_test)
```

```
In [263]: # หา accuracy โดยเอา เอลยของ X_test มาเทียบกับ predicted หรือค่าที่ model เรากำหนด  
accuracy_score(y_test, predicted)
```

```
Out[263]: 0.9397163120567376
```

## XGBoost Tuning parameter แบบ weight

```
In [264]: X_train, X_test, y_train, y_test = train_test_split(weight_raw_documents, y, test_size=0.2, random_state=42)  
#แบ่งข้อมูลเป็น 2 ส่วน train 80% test 20%
```

ເລືອກຕົ້ນຫຼາຍທີ່ເຮັດວຽກໃຊ້ GridSearchCV

```
In [213]: import xgboost as xgb  
xgb_weight = xgb.XGBClassifier() #ເລືອກໃຊ້ XGBoost
```

```
In [214]: xgb_weight_best_params = tuning_params(xgb_weight, params) #ຫຳ hyperparameter tuning  
xgb_weight_best_params #ສະຄຸງພົມທີ່ໄດ້ກຳລັງນີ້
```

Fitting 3 folds for each of 162 candidates, totalling 486 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.  
[Parallel(n_jobs=-1)]: Done 8 tasks | elapsed: 1.3min  
[Parallel(n_jobs=-1)]: Done 104 tasks | elapsed: 13.2min  
[Parallel(n_jobs=-1)]: Done 264 tasks | elapsed: 34.2min  
[Parallel(n_jobs=-1)]: Done 486 out of 486 | elapsed: 63.9min finished
```

```
Out[214]: {'colsample_bytree': 0.6,  
'gamma': 1.5,  
'max_depth': 3,  
'min_child_weight': 1,  
'subsample': 0.6}
```

## XGBoost แบบ weight Best params

ນີ້ແມ່ນ best parameter ທີ່ໄດ້ນຳໃຫ້ໃຫ້ classifier

ທ່ານາກ fit model ກັບ parameter ທີ່ຕີ້ວ່າສຸດທ່ຽວເຄራະໜໍາໄດ້

```
In [265]: # กໍານົດ parameter ດາວໂຫຼດໄດ້  
xgb_weight.set_params(**xgb_weight_best_params) #ທ່ານາກ set hyperparameter ທີ່ໄດ້ຈຳກັດທ່າງ gridsearchcv  
xgb_weight.fit(X_train, y_train) #ທ່ານາກ fit model
```

```
Out[265]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,  
colsample_bynode=1, colsample_bytree=0.6, gamma=1.5, gpu_id=-1,  
importance_type='gain', interaction_constraints='',  
learning_rate=0.300000012, max_delta_step=0, max_depth=3,  
min_child_weight=1, missing=nan, monotone_constraints='()',  
n_estimators=100, n_jobs=0, num_parallel_tree=1,  
objective='multi:softprob', random_state=0, reg_alpha=0,  
reg_lambda=1, scale_pos_weight=None, subsample=0.6,  
tree_method='exact', validate_parameters=1, verbosity=None)
```

ຫ຾ accuracy ດ້ວຍ cross\_val\_score

```
In [266]: #cross_val_score ຈະ return accuracy ອອກນາມເປັນ List ໂດຍຈຳນວນ accuracy ທີ່ return ກລັນມາຈະເຫັນ cv ທີ່ເຮັດວຽກ  
scores = cross_val_score(xgb_weight, X_train, y_train, scoring="accuracy", cv=10)  
print("xgb_weight model: ", scores.mean()) #ເຂົ້າ List ຂອງ accuracy ມານີ້ mean  
xgb_weight model: 0.9626738305941845
```

```
In [267]: #ເພີ່ມ accuracy ທີ່ໄດ້ຈຳກັດລວມໄປໃນ accuracy_dict ທີ່ສ່ວນອ່າວີ່ ໂດຍໃຫ້ key ເປັນນີ້ຂອງ model ແລະ value ເປັນ accuracy  
accuracy_dict['xgb_weight'] = scores.mean()
```

## Test XGBoost แบบ weight

ເລືອກຕົ້ນຫຼາຍທີ່ໃຫ້ parameter ທີ່ຕີ້ວ່າສຸດມາ test กັບ test set ທີ່ແນບໄວ້ດອນແຮກ

```
In [268]: #ນີ້ແມ່ນ X_test ມາຫຳນາຍ  
predicted = xgb_weight.predict(X_test)
```

```
In [269]: # ຫ຾ accuracy ໂດຍເລືອຍຂອງ X_test ມາເທິ່ນກັນ predicted หรือค่าທີ່ model ເຮັດວຽກ  
accuracy_score(y_test, predicted)
```

```
Out[269]: 0.9468085106382979
```

---

## Random forest

---

```
In [271]: #ค่าที่กำหนดไว้สำหรับ parameter ของ Random forest
params = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']
}
```

## Random forest Tuning parameter แบบไม่ weight

```
In [222]: X_train, X_test, y_train, y_test = train_test_split(non_weight_raw_documents, y, test_size=0.2, random_state=42)
#แบ่งข้อมูลเป็น 2 ส่วน train 80% test 20%
```

เราคำนึงหมวดที่เรากำหนดไว้มาใช้ GridSearchCV

```
In [223]: from sklearn.ensemble import RandomForestClassifier
rf_non_weight = RandomForestClassifier() #เรียกใช้ Random Forest
```

```
In [224]: rf_non_weight_best_params = tuning_params(rf_non_weight, params) #ทำการ hyperparameter tuning
rf_non_weight_best_params #แสดงผลที่ได้กันบ้าง
```

Fitting 3 folds for each of 60 candidates, totalling 180 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done 104 tasks      | elapsed:   10.6s
[Parallel(n_jobs=-1)]: Done 180 out of 180 | elapsed:   17.9s finished
```

```
Out[224]: {'criterion': 'gini',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 500}
```

## Random forest แบบไม่ weight Best params

นำ best parameter ที่หาได้มานำมาใช้ที่ classifier

ทำการ fit model กับ parameter ที่ดีที่สุดที่เราระบุมาได้

```
In [272]: # กำหนด parameter ตามที่หาได้
rf_non_weight.set_params(**rf_non_weight_best_params) #ทำการ set hyperparameter ที่ได้จากการทำ gridsearch cv
rf_non_weight.fit(X_train, y_train) #ทำการ fit model
```

```
Out[272]: RandomForestClassifier(max_depth=8, n_estimators=500)
```

หา accuracy ด้วย cross\_val\_score

```
In [273]: #cross_val_score จะ return accuracy ออกมากเป็น List โดยจำนวน accuracy ที่ return คลึ่งมาจาก cv ที่เรากำหนด
scores = cross_val_score(rf_non_weight, X_train, y_train, scoring="accuracy", cv=10)
print("rf_non_weight model: ", scores.mean()) #เอ้า List ของ accuracy มาหา mean
rf_non_weight model:  0.9405420353982301
```

```
In [274]: #เพิ่ม accuracy ที่เหลือกับ剩ไว้ใน accuracy_dict ที่สร้างไว้ โดยให้ key เป็นชื่อ model และ value เป็น accuracy
accuracy_dict['rf_non_weight'] = scores.mean()
```

## Test Random forest แบบไม่ weight

นำ model ที่ใช้ parameter ที่ดีที่สุดมา test กับ test set ที่แบ่งไว้ตอนแรก

```
In [275]: #นำ X_test มาหานาย
predicted = rf_non_weight.predict(X_test)
```

```
In [276]: # หา accuracy โดยเอา เอ้อยของ X_test มาเทียบกับ predicted หรือค่าที่ model เรากำหนด
accuracy_score(y_test, predicted)
```

```
Out[276]: 0.9148936170212766
```

## Random forest Tuning parameter แบบ weight

```
In [277]: X_train, X_test, y_train, y_test = train_test_split(weight_raw_documents, y, test_size=0.2, random_state=42)
#แบ่งข้อมูลเป็น 2 ส่วน train 80% test 20%
```

เราคำนึงหมวดที่เรากำหนดไว้มาใช้ GridSearchCV

```
In [231]: from sklearn.ensemble import RandomForestClassifier
rf_weight = RandomForestClassifier() #เรียกใช้ Random Forest
```

```
In [232]: rf_weight_best_params = tuning_params(rf_weight, params) #ทำการ hyperparameter tuning
rf_weight_best_params #แสดงผลที่ได้กันบ้าง
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
```

```
Fitting 3 folds for each of 60 candidates, totalling 180 fits
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:    7.7s
[Parallel(n_jobs=-1)]: Done 104 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 180 out of 180 | elapsed:  2.2min finished

Out[232]: {'criterion': 'gini',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 200}
```

### Random forest แบบ weight Best params

นำ best parameter ที่หาได้มาใส่ที่ classifier

ทำการ fit model กับ parameter ที่ดีที่สุดที่เราหาได้

```
In [278]: # กำหนด parameter ตามที่หาได้
rf_weight.set_params(**rf_weight_best_params) #ทำการ set hyperparameter ที่ได้จากการท่า gridsearchcv
rf_weight.fit(X_train, y_train) #ทำการ fit model

Out[278]: RandomForestClassifier(max_depth=8, n_estimators=200)
```

หา accuracy ด้วย cross\_val\_score

```
In [279]: #cross_val_score จะ return accuracy ออกมาเป็น list โดยจำนวน accuracy ที่ return กลับมาระหว่าง cv ที่เรากำหนด
# scores = cross_val_score(rf_weight, X_train, y_train, scoring="accuracy", cv=10)
print("rf_weight model: ", scores.mean()) #เอา list ของ accuracy มาหา mean

rf_weight model:  0.9298830594184577
```

```
In [280]: #เพิ่ม accuracy ที่เดียวกันแล้วไปใน accuracy_dict ที่สร้างไว้ โดยให้ key เป็นชื่อ model และ value เป็น accuracy
accuracy_dict['rf_weight'] = scores.mean()
```

### Test Random forest แบบ weight

เอา model ที่ใช้ parameter ที่ดีที่สุดมา test กับ test set ที่แบ่งไว้ตอนแรก

```
In [281]: #นำ X_test มาพิจารณา
predicted = xgb_weight.predict(X_test)

In [282]: # นำ accuracy ที่เคยเอา เอสัยของ X_test มาเทียบกับ predicted หรือค่าที่ model เราทำนาย
accuracy_score(y_test, predicted)

Out[282]: 0.9468085106382979
```

## Conclusion

หลังจากทำการทดสอบ model ทุกแบบแล้ว จึงนำ accuracy ที่ได้จาก cross\_val\_score มาเปรียบเทียบกัน

```
In [283]: accuracy_dict

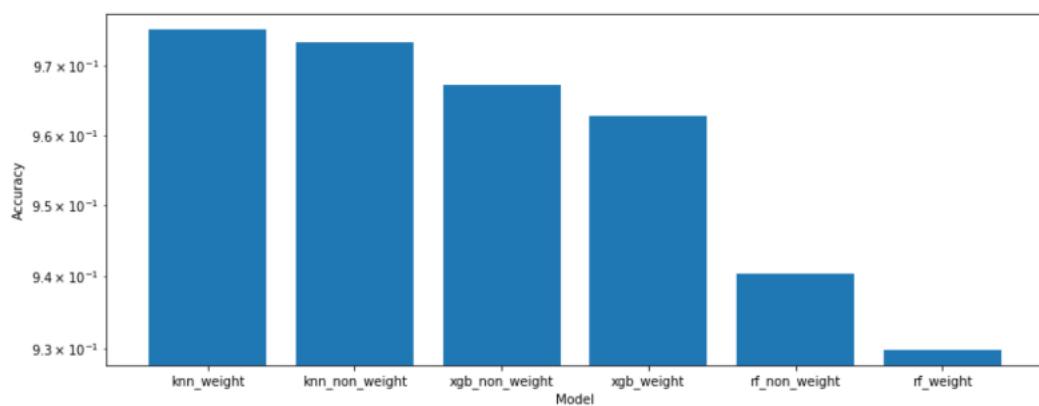
Out[283]: {'knn_non_weight': 0.9733723135271809,
 'knn_weight': 0.9751422250316055,
 'xgb_non_weight': 0.9671381163084705,
 'xgb_weight': 0.9626738305941845,
 'rf_non_weight': 0.9405420353982301,
 'rf_weight': 0.9298830594184577}

In [284]: # นำ accuracy_dict นามบัญญัติเป็น DataFrame เพื่อให้ง่ายต่อการอ่านและเบริยนเทียน
accuracy_df = pd.DataFrame(list(accuracy_dict.items()),columns = ['Model', 'Accuracy'])

Out[284]:
   Model  Accuracy
0  knn_non_weight  0.973372
1      knn_weight  0.975142
2  xgb_non_weight  0.967138
3      xgb_weight  0.962674
4  rf_non_weight  0.940542
5      rf_weight  0.929883
```

```
In [285]: #ทำการเรียงลำดับตามค่า Accuracy โดยเรียงจากมากไปน้อย
accuracy_df = accuracy_df.sort_values(by=['Accuracy'], ascending=False)

In [304]: #ทำการ plot bar chart เพื่อให้ง่ายต่อการอ่านค่าและเบริยนเทียน โดยกำหนดให้ take log เพื่อใช้ในการแสดงค่า
plt.rcParams["figure.figsize"] = (13,5)
plt.bar(accuracy_df['Model'], accuracy_df['Accuracy'].round(4), log=True)
plt.ylabel('Accuracy')
plt.xlabel('Model');
```



สรุปผล: Model KNN ที่ใช้วิธี Term weighting ให้ accuracy สูงที่สุด

In [ ]: