

Assignment 2 - DS4Biz Y63

TextScraping_Classification

Team Detail

Team Name: sompinandsomshine

Student 1

Student ID: 61070278

Student Full Name: นายกิตติภณ สรุ่งเรืองสกุล

Student 2

Student ID: 61070330

Student Full Name: นางสาวอิงฟ้า ภูติวนารถ

import package ที่ต้องใช้

- requests, bs4 ใช้ scrape ข้อมูลจากหน้าเว็บ
- pandas(pd), numpy(np) ใช้จัดการข้อมูลที่ scrape มา เช่น ทำเป็น DataFrame, แปลงเป็น csv/txt
- warnings ใช้ปิด warning ที่แจ้งเตือน
- CountVectorizer, TfidfTransformer, TfidfVectorizer ใช้เครื่องข้อมูลเช่น คำ term weighting
- nltk ใช้ในการประมวลผลภาษาธรรมชาติ เช่น กำจัด stopwords (คำที่เกิดขึ้นบ่อยและไม่มีอิทธิพลในการจำแนก)
- Pipeline ทำให้การทำงานเป็นลำดับและนำมายัง GridSearchCV
- train_test_split, cross_val_score, StratifiedKFold, GridSearchCV นำมายัง model selection และหา best hyperparameter
- neighbors(KNeighborsClassifier), xgboost(xgb), ensemble(RandomForestClassifier) model ที่จะนำมาใช้ในการท่านายว่าแต่ข่าวเป็นประเภทไหน

```
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.pipeline import Pipeline

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

ใช้ `requests.get` ตรวจสอบว่าลิงก์ที่ต้องการ scrape ข้อมูลสามารถเข้าได้มั้ย

```
In [3]: response = requests.get('http://www.it.kmitl.ac.th/~teerapong/news_archive/index.html') #ใช้ลิงก์ที่เราต้องการ scrape
print(response) # response 200 คือสามารถเข้าถึงได้ พร้อมดึงข้อมูล

<Response [200]>
```

ใช้ `bs4.BeautifulSoup` ในการอ่าน code ของลิงก์ที่เราได้มา

```
In [4]: html_page = bs4.BeautifulSoup(response.content, 'html.parser') #เก็บ code ที่ได้ไว้ในตัวแปร html_page
print(html_page)
```

```
<!DOCTYPE html>

<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="https://www.it.kmitl.ac.th/~teerapong/news_archive/" property="og:url">
<meta content="Online News Archive" property="og:title">
<meta content="summary" property="og:type">
<meta content="https://www.it.kmitl.ac.th/~teerapong/news_archive/icon.png" property="og:image">
<meta content="https://www.it.kmitl.ac.th/~teerapong/news_archive/icon.png" property="og:image:secure_url">
<meta content="https://www.it.kmitl.ac.th/~teerapong/news_archive/icon.png" property="og:image:width">
<meta content="https://www.it.kmitl.ac.th/~teerapong/news_archive/icon.png" property="og:image:height">
```

```
<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>
```

ค่าสั่ง select ใช้ในการแสดง code ที่ตรงตาม selector ที่เรากำหนดไว้ โดยจะ return ค่าที่ตรงกับที่เรากำหนดไว้กลับมาทั้งหมด

```
In [5]: #ต้องการดึงก็ข่าวของแต่ละเดือน
selector = 'body > div > div.main > ul > li> a' #ลึกลึกลึกของแต่ละเดือนอยู่ใน tags ul>li>a จึงกำหนดให้ไปดึง code ที่อยู่ในส่วนนี้ออกมานะ
tags = html_page.select(selector) #โดยให้ return code ที่ตรงกับ selector ที่กำหนดไว้ทั้งหมดและเก็บไว้ในตัวแปร tags
tags #เรียกดูผลลัพธ์ว่าตรงกับที่ต้องการมั้ย
```

```
Out[5]: [<a href="month-jan-2017.html">January</a>,
<a href="month-feb-2017.html">February</a>,
<a href="month-mar-2017.html">March</a>,
<a href="month-apr-2017.html">April</a>,
<a href="month-may-2017.html">May</a>,
<a href="month-jun-2017.html">June</a>,
<a href="month-jul-2017.html">July</a>,
<a href="month-aug-2017.html">August</a>,
<a href="month-sep-2017.html">September</a>,
<a href="month-oct-2017.html">October</a>,
<a href="month-nov-2017.html">November</a>,
<a href="month-dec-2017.html">December</a>]
```

ได้ code ที่เป็นลิงก์มาหมดแล้วแต่ยังไม่พร้อมใช้งานเพรำะ ลิงก์ที่ต้องการมีหน้าตาแบบนี้ http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jan-2017.html จึงจะทำการสร้าง list_archive เพื่อรอเก็บลิงก์ที่พร้อมใช้งาน

```
In [6]: list_archive = [] #List ที่รอเก็บลิงก์ที่จะนำไปใช้งานต่อ
for tag in tags: #ทำการวน for Loop จากตัวแปร tags ที่เก็บ code ที่มีลิงก์อยู่
    list_archive.append('http://www.it.kmitl.ac.th/~teerapong/news_archive/' + tag['href'])
    '''นำ tag['href'] ชิ้นคือส่วนที่อยู่ใน href เท่านั้น (ex. month-jan-2017.html) มาต่อ กับ http://www.it.kmitl.ac.th/~teerapong/news_archive/
เพื่อให้ได้ลิงก์ตามที่เราต้องการ และวิ่งกันนั้นไปเก็บไว้ใน list_archive ที่สร้างเอาไว้'''
list_archive #เรียกดูผลลัพธ์ว่าตรงกับที่ต้องการมั้ย
```

```
Out[6]: ['http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jan-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-feb-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-mar-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-apr-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-may-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jun-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jul-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-aug-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-sep-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-oct-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-nov-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-dec-2017.html']
```

Create function

สร้าง Function ที่ใช้ในการ scrape

- category ประเภทข่าว -> extract_category()
- title หัวข้อข่าว -> extract_title()
- link ลิงก์ข่าว -> extract_link_title()

```
In [7]: # สร้าง Function ที่ใช้ในการ scrape category
def extract_category(html_page):
    selector = 'body > div > div.main > table > tbody > tr > td.category' #กำหนดให้ดึงค่าที่อยู่ในตารางในส่วนของคอลัมน์ Article Category
    tags = html_page.select(selector)
    category = [] #สร้าง List ชื่อ category ไว้รับค่าที่ scrape ได้

    for tag in tags: #ทำการวน for Loop จากตัวแปร tags
        if tag.text.strip() != 'N/A': #จะเอาข้อมูล category ใส่ใน List ชื่อ category ถ้าข้อมูลที่ดึงมาไม่เป็น N/A (ค่าว่าง)
            category.append(tag.text.strip()) #เพิ่มข้อมูลที่ scrape ได้ใส่ไว้ใน List ชื่อ category

    return category
```

```
In [8]: # สร้าง Function ที่ใช้ในการ scrape title
def extract_title(html_page):
    selector = 'body > div > div.main > table > tbody > tr > td.title > a' #กำหนดให้ดึงค่าที่อยู่ในตารางในส่วนของคอลัมน์ Article Title และดึงไปที่ tag a เพราะแต่ที่มีข้อมูลจะเป็น tag a แต่ถ้าไม่มีข้อมูลแล้วจะเป็น tag i
    tags = html_page.select(selector)
    title = [] #สร้าง List ชื่อ title ไว้รับค่าที่ scrape ได้

    for tag in tags: #ทำการวน for Loop จากตัวแปร tags
        title.append(tag.text.strip()) #เพิ่มข้อมูลที่ scrape ได้ใส่ไว้ใน List ชื่อ title

    return title
```

```
In [9]: # สร้าง Function ที่ใช้ในการ scrape Link
def extract_link_title(html_page):
    selector = 'body > div > div.main > table > tbody > tr > td.title > a' #กำหนดให้ดึงค่าที่อยู่ในตารางในส่วนของคอลัมน์ Article Title และดึงไปที่ tag a เพราะลิงก์ของข่าวอยู่ tag a
    tags = html_page.select(selector)
    link_title = [] #สร้าง List ชื่อ link_title ไว้รับค่าที่ scrape ได้

    for tag in tags: #ทำการวน for Loop จากตัวแปร tags
        link_title.append('http://www.it.kmitl.ac.th/~teerapong/news_archive/' + tag['href'])
        #ทำการแก้ไขข้อมูลให้เป็นลิงก์ที่พร้อมใช้งาน และค่อยเพิ่มข้อมูลใส่ไว้ใน List ชื่อ link_title

    return link_title
```

```
In [10]: #ตัวอย่าง
url = 'http://www.it.kmitl.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.kmitl.ac.th/~teerapong/news_archive/article-jan-0418.html

สร้าง Function (`extract_news_archive_info()`) สำหรับเรียกใช้งาน Function ในการ scrape `category` ประเภทข่าว, `title` หัวข้อข่าว, `link` ลิงก์ข่าว ทั้งหมด โดยใส่แค่ลิงก์เดือนที่จะดึง `category` ประเภทข่าว, `title` หัวข้อข่าว, `link` ลิงก์ข่าว ในเดือนนั้นออกมายังหน้าต่างและจะ return กลับมาในรูปของ DataFrame

```
In [11]: def extract_news_archive_info(url): #รับลิงก์ของเดือนที่ต้องการดึงข้อมูล
    response = requests.get(url)
    html_page = bs4.BeautifulSoup(response.content, 'html.parser') #ดึงหน้าเว็บของลิงก์นั้น

    #เอาหน้าเว็บที่ดึงออกมายไปใส่ใน argument แต่ละ Function ที่ได้สร้างไว้เพื่อถึงข้อมูล ผลลัพธ์ที่ได้ก็ลับมาจะอยู่ในรูปของ List
    category = extract_category(html_page)
    title = extract_title(html_page)
    link_title = extract_link_title(html_page)

    #เอาข้อมูลที่อยู่ในรูปของ List ที่ได้ก็ลับมามาแปลงให้อยู่ในรูปของ pandas Series เพื่อที่จะได้อาช้อมูลทั้ง 3 ตัวมา concat กันให้เป็น DataFrame
    category = pd.Series(category)
    title = pd.Series(title)
    link_title = pd.Series(link_title)

    #เอาข้อมูลทั้ง 3 ตัวที่อยู่ในรูปของ pandas Series และมา concat กันในแนวคอลัมน์ (เอากลั่นมาต่อๆกัน)
    result = pd.concat([category, title, link_title], axis=1)
    #rename ชื่อ column ให้ตรงกับที่ต้องการใช้งาน โดยจะเปลี่ยนชื่อเป็น category-ประเภทข่าว, title-หัวข้อข่าว, Link-ลิงก์ข่าว
    result = result.rename(columns={0: "category", 1: "title", 2: "link"})
    return result #return DataFrame
```

```
In [12]: #ตัวอย่าง อันนี้คือ DataFrame ที่ Function extract_news_archive_info() return กลับมา
#ข้อมูลที่ return กลับมาจะเป็นข่าวทั้งหมดในเดือนที่เราส่งไป
url = 'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jan-2017.html'
extract_news_archive_info(url)
```

Out[12]:

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

118 rows × 3 columns

Create DataFrame

สร้าง DataFrame เพื่อรับข้อมูลที่ scrape ได้

ชิ้นข้อมูลที่เราจะ scrape คือลิงก์ข่าวในแต่ละเดือนใน list_archive โดยเราจะ scrape category ประเภทข่าว, title หัวข้อข่าว, link ลิงก์ข่าว

```
In [13]: #สร้าง DataFrame ที่รับข้อมูลที่ scrape ได้และกำหนดชื่อ column ตามสิ่งที่จะ scrape
df = pd.DataFrame(columns=['category', 'title', 'link'])
df #ได้ df เป็นล่ามที่รับข้อมูลที่ scrape ได้
```

Out[13]:

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

เอา list_archive (ลิงก์เดือนทั้งหมด) ไปเข้า Function extract_news_archive_info()

ขั้นตอนนี้ทำเพื่อให้ได้ข่าวทั้งหมดในทุกๆเดือนมารวมกันเป็น DataFrame อันเดียว

```
In [14]: list_archive
```

```
Out[14]: ['http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jan-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-feb-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-mar-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-apr-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-may-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jun-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-jul-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-aug-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-sep-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-oct-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-nov-2017.html',
'http://www.it.kmitl.ac.th/~teerapong/news_archive/month-dec-2017.html']
```

เอาแต่ละลิงก์ใน list_archive ไปเข้า function extract_news_archive_info() และเพิ่มเข้าไปใน DataFrame ที่เราเตรียมไว้

```
In [15]: for url in list_archive: #วน for Loop เพื่อเอาลิงก์เดือนที่ละอันไปเข้า function
    result = extract_news_archive_info(url) #เอาลิงก์เดือนไปเข้า function และเก็บ DataFrame ที่ return กลับมาในตัวแปร result
    df = df.append(result, ignore_index=True) #เอา result ที่ได้จาก function ไปเพิ่มใน DataFrame ที่สร้างไว้
```

```
In [16]: df #หน้าตาของ DataFrame ที่มีข้อมูล category ประเภทข่าว, title หัวข้อข่าว, link ลิงก์ข่าว ทั้งหมด
```

Out[16]:

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

```
Out[17]: category      0
title        0
link         0
dtype: int64
```

Add id column

แต่ละข่าวมี id ของมันเองเลยจะทำการเก็บแยกเพิ่มอีก columnn

```
In [18]: # ในแต่ละลิงก์ข่าวจะมี id ของข่าวอยู่
df['link'].iloc[0]
```

```
Out[18]: 'http://www.it.kmit.ac.th/~teerapong/news_archive/article-jan-0418.html'
```

```
In [19]: #ใช้ index slicing ในการหา id ข่าว โดยจะเริ่มเก็บตั้งแต่ตัวที่ 50 ถึงตัวที่ 6 จากหลังสุด
df['id'] = df['link'].str[50:-5]
df
```

```
Out[19]:
```

	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-ian-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

ดึงข้อมูลในลิงก์ข่าว

```
In [20]: list_content = [] #สร้าง List ชื่อ list_content ไว้เก็บข้อมูล content ทั้งหมด
```

สร้าง Function ที่ใช้ในการดึงข้อมูลข่าวในลิงก์ข่าวที่ใส่เข้าไป

```
In [21]: def get_content(url): #รับลิงก์ข่าวและเก็บไว้ในชื่อ url
    response = requests.get(url)
    html_page = bs4.BeautifulSoup(response.content, 'html.parser')

    selector = 'body > div > div.main > p' #เมื่อหาข่าวอยู่ในส่วน tag p
    #เป็นองค์ประกอบของข่าวมีหลายอย่างมา (1 อย่างมา : 1 tag p) จึงใช้ select ในการถือ ค่าที่ได้จะอยู่ในรูปของ List
    tags = html_page.select(selector)

    content = '' #สร้างตัวแปร content รอเก็บเนื้อหาข่าวในรูปของ string
    for tag in tags: #วน for Loop ในตัวแปร tags
        if tag.text.strip() != '': #ในเมื่อหาข่าวจะมีส่วนที่เป็นเว้นวรรคหน้าเปล่าๆ ซึ่งเราไม่ต้องการเก็บส่วนนั้น จึงจะทำการเก็บเฉพาะส่วนที่ไม่ได้เว้นวรรคไว้
            if '.html' in str(tag): #ในส่วนล่างสุดในบางข่าวจะมีลิงก์ให้กดคลิ๊กไปที่หน้าหลัก ซึ่งเราจะไม่ต้องการเก็บเช่นกัน จึงจะทำการข้ามไป
                pass
            elif 'Comments are closed for this article' in str(tag): #ในส่วนล่างสุดในบางข่าวจะมีบล็อกไม่สามารถcomment ได้ จึงจะทำการข้ามไป
                pass
            else:
                content += tag.text.strip() + ' ' #หลังจากข้ามส่วนที่ไม่ต้องการเก็บไปหนดแล้วก็จะเพิ่มส่วนต้องการเก็บไว้ในตัวแปร content

    list_content.append(content.strip()) #หลังจากจบ Loop ก็จะทำการเพิ่มน้ำหน้าข่าวไปในตัวแปร list_content
```

```
In [22]: #df['Link'] คือลิงก์ข่าวทั้งหมด
print('example:', df['link'][0])
```

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

```
In [23]: for i in range(len(df)): #วน for Loop ตามความยาวทั้งหมดของ df
    get_content(df['link'].iloc[i]) #เอาลิงก์ข่าวใส่ไปใน function get_content()
```

หลังจากวน for loop เพื่อเลิ�ก์ข่าวใส่ไปใน function `get_content()` ครบหมดแล้ว ในตัวแปร `list_content` จะมีเนื้อหาข่าวทั้งหมด เราจะแปลง `list_content` จาก `list` ให้เป็น `pandas Series` เพื่อที่จะได้อusaไป concat กับ `df` ให้สมบูรณ์

```
In [24]: 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 (เอากลั้นนำมาต่อกัน)
df = df.rename(columns={0: "content"}) #rename ชื่อ column เพื่อให้สะดวกต่อการใช้งาน โดยจะเปลี่ยนชื่อเป็น content - เนื้อหา/ข่าว
df
```

Out[24]:

	category	title	link	id	content
0	technology	21st-Century Sports: How Digital Technology Is...	http://www.it.kmitl.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.kmitl.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.kmitl.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.kmitl.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.kmitl.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.kmitl.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.kmitl.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.kmitl.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.kmitl.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.kmitl.ac.th/~teerapong/news_arch...	article-dec-0068	Zambia's technical director, Kalusha Bwalya is...

1408 rows × 5 columns

save DataFarame to csv/txt

เพื่อให้ง่ายต่อการนำไปวิเคราะห์

```
In [17]: df.to_csv('datastore/data.csv', index=False)
```

```
In [18]: df = pd.read_csv('datastore/data.csv')
df.head()
```

Out[18]:

	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...

คัดให้เหลือเฉพาะที่ใช้ในการวิเคราะห์

- AllArticles_OnlyHeading
- AllArticles_OnlyContent
- AllArticles_HeadingPlusContent
- Category

```
In [3]: AllArticles_OnlyHeading = df['title']
AllArticles_OnlyContent = df['content']
AllArticles_HeadingPlusContent = df[['title','content']]
Category = df['category']
```

csv

```
In [4]: AllArticles_OnlyHeading.to_csv('datastore/AllArticles_OnlyHeading.csv', index=False)
```

```
In [5]: AllArticles_OnlyContent.to_csv('datastore/AllArticles_OnlyContent.csv', index=False)
```

```
In [6]: AllArticles_HeadingPlusContent.to_csv('datastore/AllArticles_HeadingPlusContent.csv', index=False)
```

```
In [7]: Category.to_csv('target/category.csv', index=False)
```

txt

```
In [8]: AllArticles_OnlyHeading.to_csv(r'datastore/AllArticles_OnlyHeading.txt', header=None, index=None, sep=' ', mode='a')
```

```
In [9]: AllArticles_OnlyContent.to_csv(r'datastore/AllArticles_OnlyContent.txt', header=None, index=None, sep=' ', mode='a')

In [10]: AllArticles_HeadingPlusContent.to_csv(r'datastore/AllArticles_HeadingPlusContent.txt', header=None, index=None, sep=' ', mode='a')

In [15]: Category.to_csv(r'target/category.txt', header=None, index=None, sep=' ', mode='a')

In [ ]:
```

Modeling

ทำการ Tunning hyperparameter และหา Model ที่ดีที่สุดในการทำนาย

- K-Nearest Neighbors
- XGBoost
- Random forest

โหลดข้อมูลที่เคย save เก็บไว้

- AllArticles_OnlyContent.txt ต้องข้อมูลที่มีเฉพาะเนื้อหาป่าวโดยจะเก็บข้อมูลในตัวแปร raw_documents
- target/category.txt ต้องส่วนของ target หรือเฉลยว่าข่าวตั้งกล้าอยู่ในประเภทอะไร

```
In [20]: with open("datastore/AllArticles_OnlyContent.txt","r", encoding="utf8") as f:  
    raw_documents = f.read().splitlines() #ใช้แยก string ที่จดเขียนบรรทัดใหม่คือ จดที่มี '\n' และศึนค่ากลับมาเป็นข้อมูลประเภท List  
f.close()  
print("Read %d raw text documents" % len(raw_documents))
```

Read 1408 raw text documents

```
In [21]: raw_documents[0]
```

Out[21]: "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 [22]: with open("target/category.txt","r", encoding="utf8") as f:  
    y = f.read().splitlines() #ใช้แยก string ที่จดเขียนบรรทัดใหม่คือ จดที่มี '\n' และศึนค่ากลับมาเป็นข้อมูลประเภท List  
f.close()
```

```
print("Read %d" % len(y))
```

Read 1408

In []:

สร้าง dict ชื่อ accuracy_dict เพื่อใช้เก็บค่า accuracy ของแต่ละ model

```
In [62]: accuracy_dict = dict() #สร้าง dict ชื่อ accuracy_dict เพื่อรับค่า accuracy
```

Text preprocessing

สร้าง function ที่ต้องใช้ในการทำ preprocessing

```
In [63]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

```
In [64]: # Tokenizer and WordNet Lemmatizer with POS (part of speech)
def lemma_tokenizer_w_pos_tag(text):

    def convert_tags(tag):
        if tag == 'vbd' or tag == 'vbg' or tag == 'vbz':
            return 'v'
        else:
            return 'n'

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

    lemmatizer = nltk.WordNetLemmatizer() #สร้างตัวที่ใช้แปลงรูปของคำให้อยู่รูปเดิมพื้นฐาน (รากคำ)
    lemma_tokens = [] #สร้าง List ชื่อ Lemma_tokens เพื่อรับค่าหลังจากแปลงด้วย pos_tag อันใหม่
    for token in tokens_with_pos_tag: #วน for Loop ในตัวแปร tokens_with_pos_tag
        new_tag = convert_tags(token[1].lower())
        '''token[1] คือ pos_tag ที่เพิ่มเข้ามาเมื่อผ่านบรรทัดที่ 12 (nltk.pos_tag(tokens)) ใช้ .Lower() เพื่อบรรทับให้เป็นตัวเล็ก
        และจึงทำการส่งไปที่ function convert_tags() เพื่อแปลงให้เหลือแค่ 2 แบบคือคำเป็น
        VBD verb: past tense took,
        VBG verb: gerund/present participle taking,
        VBZ verb: 3rd person sing. present takes
        จะแปลงให้เหลือแค่ v นอกเหนือจากนั้นจะแปลงเป็น n ทั้งหมด
        ซึ่ง pos_tag ที่ได้มานาทีจะเก็บไว้ในตัวแปร new_tag'''

        lemma_tokens.append(lemmatizer.lemmatize(token[0], new_tag))
        '''จะนำ pos_tag ใหม่ที่ได้มานาทีกับ token[0] หรือค่าที่ต่อนบรรทัดที่ 12 เพื่อแปลงให้อยู่ในรูปของ pos_tag อันใหม่
        เช่น token[0] = amplifying, new_tag = v ผลลัพธ์ที่ได้คือ amplify และจะเอาผลลัพธ์เพิ่มเข้าไปใน List ชื่อ lemma_tokens'''

    return lemma_tokens
```

ตัวอย่างเมื่อเอาค่าเข้าไปใน function lemma_tokenizer_w_pos_tag()

```
In [65]: lemma_tokenizer_w_pos_tag('amplifying')
```

```
Out[65]: ['amplify']
```

```
In [66]: #สร้าง pipeline สำหรับทำ CountVectorizer และ TfidfTransformer
pipe = Pipeline([
    #ทำการตัด stop_words, เอา text ไปใส่ function Lemma_tokenizer_w_pos_tag โดยผลลัพธ์ที่ได้หลังจากทำ CountVectorizer คือ Term-Document Matrix
    ('vect', CountVectorizer(stop_words="english", min_df = 0, tokenizer=lemma_tokenizer_w_pos_tag)),
    #แปลงเมทริกซ์เป็น tf-idf
    ('tfidf', TfidfTransformer()),
]).fit(X_train)
```

```
In [67]: #Term weighting ใน Scikit-Learn เราสามารถสร้างที่ TF-IDF weighted document-term matrix โดยใช้ TfidfVectorizer() แทน CountVectorizer()
#default ของ analyzer คือ word, ก้าวตัด stop_words
vectorizer = TfidfVectorizer(analyzer='word', ngram_range=(1,3),
                            min_df = 0, stop_words = 'english', sublinear_tf=True, lowercase=True)
```

K-Nearest Neighbors

ในการทำ Tuning parameter จะต้องกำหนด parameter ที่เราจะทำการวิเคราะห์ เช่น knn จะต้องมีการกำหนดจำนวน k

```
In [68]: #ทำการกำหนด parameter ที่จะเอาไปวิเคราะห์ โดยค่าที่กำหนดไว้คือ parameter ของ knn
params = {
    'n_neighbors' : list(np.arange(1,50,2)), #ให้ทำการทดลองหา k ที่ดีที่สุดตั้ง 1-50 โดยนับทีละ 2 เช่น 1, 3, 5
    'weights' : ['uniform', 'distance'], #weight function ที่ใช้ในการท่านาย
    'metric' : ['euclidean', 'manhattan'] #คือระยะทางด้วยวิธีไหน
}
```

KNN Tuning parameter (CountVectorizer & TfidfTransformer)

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 [39]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

ทุกครั้งก่อนน้ำ train set ไปวิเคราะห์จะต้องทำการ transform ข้อมูลให้อยู่ในรูปแบบที่จะใช้ก่อน

```
In [40]: X_train = pipe.transform(X_train) #ทำการแปลง X_train โดยใช้ pipeline ที่เราสร้างไว้
```

```
In [41]: X_train
```

```
Out[41]: <1126x17450 sparse matrix of type '<class 'numpy.float64'>'  
with 154876 stored elements in Compressed Sparse Row format>
```

ເລືອດຕ່າງໆທັງໝົດທີ່ເຮົາກ່ານດໄວ້ມາໃສ່ GridSearchCV

```
In [127]: folds = 5 #ກ່ານດັບ fold ທີ່ຈະໃຊ້ໃນການທ່າ K-fold  
knn=KNeighborsClassifier() #ເລືອດຕ່າງໆ KNN  
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001) #ທ່າກ່ານດັບຕຽ່ອງນີ້ຂອງ K-fold ວ່າຈະໃຊ້ 5 folds ໂດຍ Stratifi  
edKFold ເປັນການນັບຂອ້ມຄວບປະນຍໍ້  
#ເລືອດຕ່າງໆ Classifier, params ໄລັງໄປແລະ ກ່ານດັບ cv ດ້ວຍ k-fold ທີ່ເຮົາກ່ານດັວງໃນດ້ວຍແມ່ນ skf  
knn_grid = GridSearchCV(knn, params, verbose = 3, cv=skf.split(X_train,y_train), n_jobs = -1)  
knn_grid.fit(X_train, y_train) #ເລືອດຕ່າງໆ GridSearchCV ມາ fit ສັນ train set  
#tatalling 500 fit ເກີດຈາກ params ທີ່ເປັນ n_neighbors 25 ຕົວ * weights 2 ຕົວ * metric 2 ຕົວ * folds 5 = 25*2*2*5 = 500
```

Fitting 5 folds for each of 100 candidates, totalling 500 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.  
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:   1.1s  
[Parallel(n_jobs=-1)]: Done 188 tasks      | elapsed:   1.8s  
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed:   9.1s finished
```

```
Out[127]: GridSearchCV(cv=<generator object _BaseKFold.split at 0x0000021A83CA8200>,  
estimator=KNeighborsClassifier(), n_jobs=-1,  
param_grid={'metric': ['euclidean', 'manhattan'],  
           'n_neighbors': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21,  
                           23, 25, 27, 29, 31, 33, 35, 37, 39, 41,  
                           43, 45, 47, 49],  
           'weights': ['uniform', 'distance']},  
verbose=3)
```

```
In [128]: #ແສດງຄ່າ parameter ທີ່ດີທີ່ສຸດທ່ານໄດ້  
print(knn_grid.best_params_)
```

```
{'metric': 'euclidean', 'n_neighbors': 41, 'weights': 'uniform'}
```

CountVectorizer & TfidfTransformer Best params

ນີ້ແມ່ນ best parameter ທີ່ທາງໄດ້ມາໃສ່ທີ່ classifier

```
In [69]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

```
In [70]: #ກ່ອນເລືອດຕ່າງໆ X ໃນໄຟຕ້ອງ transform ກ່ອນທຸກຄົ່ນ  
X_train = pipe.transform(X_train)
```

ทำการ fit model กับ parameter ที่ดีที่สุดที่วิเคราะห์มาได้

```
In [71]: knn=KNeighborsClassifier(metric='euclidean', n_neighbors=41, weights='uniform') #เรียกใช้ knn และกำหนด parameter  
knn.fit(X_train, y_train)
```

```
Out[71]: KNeighborsClassifier(metric='euclidean', n_neighbors=41)
```

หา accuracy ด้วย cross_val_score

```
In [72]: #cross_val_score จะ return accuracy ออกมากเป็น List โดยจำนวน accuracy ที่ return กลับมาระหว่างกัน cv ที่เรากำหนดไว้  
scores = cross_val_score(knn, X_train, y_train, scoring="accuracy", cv=10)  
print("knn model: ", scores.mean()) #เอา List ของ accuracy มาหา mean
```

knn model: 0.9733723135271809

```
In [73]: #เพิ่ม accuracy ที่เหลือกันแล้วไว้ใน accuracy_dict ที่สร้างไว้ โดยให้ key เป็นชื่อ model และ value เป็น accuracy  
accuracy_dict['knn'] = scores.mean()
```

Test knn model (CountVectorizer & TfidfTransformer)

เอา model ที่ใช้ parameter ที่ดีที่สุดมา test กับ test set ที่แบ่งไว้ตอนแรก

```
In [74]: # transform X_test ให้เหมือนกับ X_train ก่อน เพื่อที่จะได้ใช้กับ model ที่ train ไว้ได้แล้วเก็บไว้ในตัวแปร X_new_tfidf  
X_new_tfidf = pipe.transform(X_test)  
#นำ X_new_tfidf มาทำนาย  
predicted = knn.predict(X_new_tfidf)
```

```
In [75]: # นับ accuracy โดยเอา เฉลยของ X_test มาเทียบกับ predicted หรือค่าที่ model เราทำนาย  
accuracy_score(y_test, predicted)
```

```
Out[75]: 0.9539007092198581
```

```
In [ ]:
```

KNN Tuning parameter (weighting)

ในส่วนของ Tuning parameter โดย X ที่เราจะใช้จัดทำ Term weighting ซึ่งจะคล้ายกับตอนที่เรา Tuning parameter ตอนที่ X ใช้ tfidf แต่จะต่างกันแค่ส่วนของการ transform X

```
In [36]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

Transform X ก่อนนำไปวิเคราะห์ ตรงนี้คือส่วนที่ต่างจาก tfidf

```
In [37]: X_train = vectorizer.fit_transform(X_train)
```

```
In [38]: X_train
```

```
Out[38]: <1126x399935 sparse matrix of type '<class 'numpy.float64'>'  
with 611630 stored elements in Compressed Sparse Row format>
```

ในการทำ Tuning parameter จะต้องกำหนด parameter ที่เราจะทำการวิเคราะห์ ซึ่งตรงนี้ก็จะเหมือนกับข้างบนเพราะเป็น classifier ด้วยกัน

```
In [139]: #code ในส่วนนี้ก็จะเหมือนกับข้างบนแต่จะต่างแค่ X_train ที่เราใช้ Term weighting  
folds = 5 #กำหนด fold ที่จะใช้ในการทำ K-fold  
knn=KNeighborsClassifier()  
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)  
knn_grid_weight = GridSearchCV(knn, params, verbose = 3, cv=skf.split(X_train,y_train), n_jobs = -1)  
knn_grid_weight.fit(X_train, y_train)
```

Fitting 5 folds for each of 100 candidates, totalling 500 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: 2.1s  
[Parallel(n_jobs=-1)]: Done 420 tasks | elapsed: 16.1s  
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed: 21.9s finished
```

```
Out[139]: GridSearchCV(cv=<generator object _BaseKFold.split at 0x0000021A839EC510>,  
estimator=KNeighborsClassifier(), n_jobs=-1,  
param_grid={'metric': ['euclidean', 'manhattan'],  
'n_neighbors': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21,  
23, 25, 27, 29, 31, 33, 35, 37, 39, 41,  
43, 45, 47, 49],  
'weights': ['uniform', 'distance']},  
verbose=3)
```

```
In [140]: #แสดงค่า parameter ที่ดีที่สุด ในส่วนนี้จะต่างกับข้างบนเพราะว่าเราได้ใช้ Term weighting ในส่วนนี้จึงทำให้ parameter ที่ดีที่สุดนั้นต่างกัน  
print(knn_grid_weight.best_params_)
```

```
{'metric': 'euclidean', 'n_neighbors': 17, 'weights': 'uniform'}
```

Term weighting Best hyperparameter

น่า best parameter ที่หาได้มาใส่ที่ classifier

```
In [127]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

```
In [128]: #ก่อนเอา X ไปใช้ต้อง transform ก่อนทุกครั้ง  
X_train = vectorizer.fit_transform(X_train)
```

```
In [129]: #กำหนด parameter ตามที่หาได้  
knn_weight = KNeighborsClassifier(metric='euclidean', n_neighbors=17, weights='uniform')  
knn_weight.fit(X_train, y_train)
```

```
Out[129]: KNeighborsClassifier(metric='euclidean', n_neighbors=17)
```

```
In [130]: #นับ accuracy ด้วย cross_val_score  
scores = cross_val_score(knn_weight, X_train, y_train, cv=10)  
print('KNN model (weight):', scores.mean())
```

```
KNN model (weight): 0.9769121365360304
```

```
In [80]: accuracy_dict['knn_weight'] = scores.mean()
```

Test knn model (weight)

นำ model ที่ลุน parameter เรียบร้อยแล้วมา test กับ test set

```
In [131]: X_new_tfidf = vectorizer.transform(X_test)  
predicted = knn_weight.predict(X_new_tfidf)
```

```
In [132]: accuracy_score(y_test, predicted)
```

```
Out[132]: 0.975177304964539
```

XGBoost

ทำการกำหนด parameter ที่เราจะทำการวิเคราะห์

```
In [83]: #ค่าที่กำหนดไว้คือ 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 (CountVectorizer & TfidfTransformer)

```
In [266]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

ก่อนเอา X ไปใช้ต้อง transform ก่อนทุกครั้ง

```
In [149]: X_train = pipe.transform(X_train)
```

เอาค่าทั้งหมดที่เรากำหนดไว้มาใส่ GridSearchCV

```
In [151]: import xgboost as xgb
folds = 5 #จำนวน fold ที่จะใช้ในการทำ K-fold
xgb = xgb.XGBClassifier()
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)
xgb_grid = GridSearchCV(xgb, params, verbose = 3, cv=skf.split(X_train,y_train), n_jobs = -1)
xgb_grid.fit(X_train, y_train)
```

Fitting 5 folds for each of 162 candidates, totalling 810 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks    | elapsed:   6.3s
[Parallel(n_jobs=-1)]: Done 104 tasks    | elapsed:  33.7s
[Parallel(n_jobs=-1)]: Done 264 tasks    | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 488 tasks    | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done 776 tasks    | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done 810 out of 810 | elapsed:  5.7min finished
```

```
Out[151]: GridSearchCV(cv=<generator object _BaseKFold.split at 0x0000021A83E2C4A0>,
estimator=XGBClassifier(base_score=None, booster=None,
                        colsample_bylevel=None,
                        colsample_bynode=None,
                        colsample_bytree=None, gamma=None,
                        gpu_id=None, importance_type='gain',
                        interaction_constraints=None,
                        learning_rate=None, max_delta_step=None,
                        max_depth=None, min_child_weight=None,
                        missing=n...
                        n_estimators=100, n_jobs=None,
                        num_parallel_tree=None, random_state=None,
                        reg_alpha=None, reg_lambda=None,
                        scale_pos_weight=None, subsample=None,
                        tree_method=None, validate_parameters=None,
                        verbosity=None),
n_jobs=-1,
param_grid={'colsample_bytree': [0.6, 0.8, 1.0],
            'gamma': [0.5, 1, 1.5], 'max_depth': [3, 4, 5],
            'min_child_weight': [1, 5],
            'subsample': [0.6, 0.8, 1.0]},
verbose=3)
```

```
In [152]: #แสดงค่า parameter ที่ดีที่สุด
```

```
print(xgb_grid.best_params_)

{'colsample_bytree': 0.6, 'gamma': 0.5, 'max_depth': 4, 'min_child_weight': 1, 'subsample': 0.8}
```

CountVectorizer & TfidfTransformer Best params

```
In [84]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

ก่อนเอา X ไปใช้ต้อง transform

```
In [85]: X_train = pipe.transform(X_train)
```

```
In [86]: #กำหนด parameter ที่ดีที่สุดตามที่หาได้
import xgboost as xgb
xgb = xgb.XGBClassifier(colsample_bytree=0.6, gamma=0.5, max_depth=4, min_child_weight=1, subsample=0.8)
xgb.fit(X_train, y_train)
```

```
Out[86]: 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=4,
                      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=0.8,
                      tree_method='exact', validate_parameters=1, verbosity=None)
```

```
In [87]: #หา accuracy ด้วย cross_val_score
scores = cross_val_score(xgb, X_train, y_train, scoring="accuracy", cv=10)
print("XGB model: ", scores.mean())
```

XGB model: 0.9689159292035396

```
In [88]: accuracy_dict['xgb'] = scores.mean()
```

Test xgb model (CountVectorizer & TfidfTransformer)

นำ model ที่จุน parameter เรียบร้อยแล้วมา test กับ test set

```
In [157]: X_new_tfidf = pipe.transform(X_test)

predicted = xgb.predict(X_new_tfidf)
```

```
In [158]: from sklearn.metrics import accuracy_score
accuracy_score(y_test, predicted)
```

```
Out[158]: 0.9468085106382979
```

```
In [ ]:
```

XGBoost Tuning parameter (weighting)

```
In [272]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

ก่อนเรา X ไปใช้ต้อง transform

```
In [161]: X_train = vectorizer.fit_transform(X_train)
```

```
In [163]: import xgboost as xgb
folds = 4 #กำหนด fold ที่จะใช้ในการทำ K-fold กำหนด folds = 4 เพราะว่าข้อมูลมี 4 ช่วงๆ ละ 1 หน่วย
xgb = xgb.XGBClassifier()
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)
xgb_grid_weight = GridSearchCV(xgb, params, verbose = 3, cv=skf.split(X_train,y_train), n_jobs = -1)
xgb_grid_weight.fit(X_train, y_train)
```

Fitting 4 folds for each of 162 candidates, totalling 648 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 104 tasks      | elapsed: 10.2min
[Parallel(n_jobs=-1)]: Done 264 tasks      | elapsed: 26.5min
[Parallel(n_jobs=-1)]: Done 488 tasks      | elapsed: 52.3min
[Parallel(n_jobs=-1)]: Done 648 out of 648 | elapsed: 69.4min finished
```

```
Out[163]: GridSearchCV(cv=<generator object _BaseKFold.split at 0x0000021A839414A0>,
estimator=XGBClassifier(base_score=None, booster=None,
colsample_bylevel=None,
colsample_bynode=None,
colsample_bytree=None, gamma=None,
gpu_id=None, importance_type='gain',
interaction_constraints=None,
learning_rate=None, max_delta_step=None,
max_depth=None, min_child_weight=None,
missing=n...
n_estimators=100, n_jobs=None,
num_parallel_tree=None, random_state=None,
reg_alpha=None, reg_lambda=None,
scale_pos_weight=None, subsample=None,
tree_method=None, validate_parameters=None,
verbosity=None),
n_jobs=-1,
param_grid={'colsample_bytree': [0.6, 0.8, 1.0],
'gamma': [0.5, 1, 1.5], 'max_depth': [3, 4, 5],
'min_child_weight': [1, 5],
'. . . . .': [1, 2, 3, 4, 5]})
```

```
    'subsample': [0.6, 0.8, 1.0]),
    verbose=3)
```

```
In [164]: #แสดงค่า parameter ที่ดีที่สุด
print(xgb_grid_weight.best_params_)

{'colsample_bytree': 0.6, 'gamma': 1, 'max_depth': 5, 'min_child_weight': 1, 'subsample': 1.0}
```

Term weighting Best hyperparameter

```
In [89]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

ก่อนเรา X ไปใช้ต้อง transform

```
In [90]: X_train = vectorizer.fit_transform(X_train)
```

```
In [91]: import xgboost as xgb
xgb_weight = xgb.XGBClassifier(colsample_bytree=0.6, gamma=1, max_depth=5, min_child_weight=1, subsample=1.0)
xgb_weight.fit(X_train, y_train)
```

```
Out[91]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=0.6, gamma=1, 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=1.0,
                      tree_method='exact', validate_parameters=1, verbosity=None)
```

```
In [92]: #หา accuracy ด้วย cross_val_score
scores = cross_val_score(xgb_weight, X_train, y_train, cv=10)
print('XGB model (weight):', scores.mean())

XGB model (weight): 0.9662531605562579
```

```
In [93]: accuracy_dict['xgb_weight'] = scores.mean()
```

Test xgb model (weight)

```
In [94]: X_new_tfidf = vectorizer.transform(X_test)

predicted = xgb_weight.predict(X_new_tfidf)
```

```
In [95]: from sklearn.metrics import accuracy_score
accuracy_score(y_test, predicted)
```

```
Out[95]: 0.9326241134751773
```

Random forest

```
In [96]: params = {  
    'n_estimators': [200, 500],  
    'max_features': ['auto', 'sqrt', 'log2'],  
    'max_depth' : [4,5,6,7,8],  
    'criterion' :['gini', 'entropy']  
}
```

Random forest Tunning parameter (CountVectorizer & TfidfTransformer)

```
In [278]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

```
In [173]: X_train = pipe.transform(X_train)
```

```
In [175]: from sklearn.ensemble import RandomForestClassifier  
folds = 5  
rf = RandomForestClassifier()  
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)  
rf_grid = GridSearchCV(rf, params, verbose = 3, cv=skf.split(X_train,y_train), n_jobs = -1)  
rf_grid.fit(X_train, y_train)
```

Fitting 5 folds for each of 60 candidates, totalling 300 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.  
[Parallel(n_jobs=-1)]: Done 8 tasks      | elapsed:  2.1s  
[Parallel(n_jobs=-1)]: Done 104 tasks     | elapsed:  9.5s  
[Parallel(n_jobs=-1)]: Done 264 tasks     | elapsed: 24.8s  
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 28.6s finished
```

```
Out[175]: GridSearchCV(cv=<generator object _BaseKFold.split at 0x0000021A83F41740>,  
                      estimator=RandomForestClassifier(), n_jobs=-1,  
                      param_grid={'criterion': ['gini', 'entropy'],  
                                  'max_depth': [4, 5, 6, 7, 8],  
                                  'max_features': ['auto', 'sqrt', 'log2'],  
                                  'n_estimators': [200, 500]},  
                      verbose=3)
```

```
In [176]: print(rf_grid.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'n_estimators': 500}
```

CountVectorizer & TfIdfTransformer Best params

```
In [97]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

```
In [98]: X_train = pipe.transform(X_train)
```

```
In [99]: rf = RandomForestClassifier(criterion='gini', max_depth=8, max_features='auto', n_estimators=500)
rf.fit(X_train, y_train)
```

```
Out[99]: RandomForestClassifier(max_depth=8, n_estimators=500)
```

```
In [100]: scores = cross_val_score(rf, X_train, y_train, scoring="accuracy", cv=10)
print("RF model: ", scores.mean())
```

```
RF model:  0.9689238305941845
```

```
In [101]: accuracy_dict['rf'] = scores.mean()
```

Test rf model (CountVectorizer & TfIdfTransformer)

```
In [102]: X_new_tfidf = pipe.transform(X_test)

predicted = rf.predict(X_new_tfidf)
```

```
In [103]: from sklearn.metrics import accuracy_score
accuracy_score(y_test, predicted)
```

```
Out[103]: 0.9574468085106383
```

```
In [ ]:
```

Random forest Tunning parameter (weighting)

```
In [284]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)
```

```
In [185]: X_train = vectorizer.fit_transform(X_train)
```

```
In [187]: folds = 5
rf = RandomForestClassifier()
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)
rf_grid_weight = GridSearchCV(rf, params, verbose = 3, cv=skf.split(X_train,y_train), n_jobs = -1)
```

```
rf_grid_weight.fit(X_train, y_train)

Fitting 5 folds for each of 60 candidates, totalling 300 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 tasks    | elapsed:   6.9s
[Parallel(n_jobs=-1)]: Done 104 tasks    | elapsed:  58.8s
[Parallel(n_jobs=-1)]: Done 264 tasks    | elapsed: 2.5min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 2.9min finished
```

```
Out[187]: GridSearchCV(cv=<generator object _BaseKFold.split at 0x0000021A8DE51F20>,
estimator=RandomForestClassifier(), n_jobs=-1,
param_grid={'criterion': ['gini', 'entropy'],
'max_depth': [4, 5, 6, 7, 8],
'max_features': ['auto', 'sqrt', 'log2'],
'n_estimators': [200, 500]},
verbose=3)
```

```
In [188]: print(rf_grid_weight.best_params_)

{'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto', 'n_estimators': 500}
```

Term weighting Best hyperparameter

```
In [104]: X_train, X_test, y_train, y_test = train_test_split(raw_documents, y, test_size=0.2, random_state=42)

In [105]: X_train = vectorizer.fit_transform(X_train)

In [106]: rf_weight = RandomForestClassifier(criterion='entropy', max_depth=8, max_features='auto', n_estimators=500)
rf_weight.fit(X_train, y_train)

Out[106]: RandomForestClassifier(criterion='entropy', max_depth=8, n_estimators=500)
```

```
In [107]: scores = cross_val_score(rf_weight, X_train, y_train, cv=10)
print('RF model (weight):', scores.mean())

RF model (weight): 0.9431890012642224
```

```
In [108]: accuracy_dict['rf_weight'] = scores.mean()
```

Test rf model (weight)

```
In [109]: X_new_tfidf = vectorizer.transform(X_test)

predicted = rf_weight.predict(X_new_tfidf)

In [110]: accuracy score(v test. predicted)
```

```
Out[110]: 0.9326241134751773
```

Conclusion

ผลลัพธ์จากการทดสอบ model ทุกแบบแล้ว จึงนำ accuracy ที่ได้จาก cross_val_score มาเปรียบเทียบกัน

```
In [111]: accuracy_dict
```

```
Out[111]: {'knn': 0.9733723135271809,
 'knn_weight': 0.9769121365360304,
 'xgb': 0.9689159292035396,
 'xgb_weight': 0.9662531605562579,
 'rf': 0.9689238305941845,
 'rf_weight': 0.9431890012642224}
```

```
In [120]: # นำ accuracy_dict มาแปลงเป็น DataFrame เพื่อให้ง่ายต่อการอ่านและเปรียบเทียบ
accuracy_df = pd.DataFrame(list(accuracy_dict.items()),columns = ['Model','Accuracy'])
accuracy_df
```

```
Out[120]:
```

	Model	Accuracy
0	knn	0.973372
1	knn_weight	0.976912
2	xgb	0.968916
3	xgb_weight	0.966253
4	rf	0.968924
5	rf_weight	0.943189

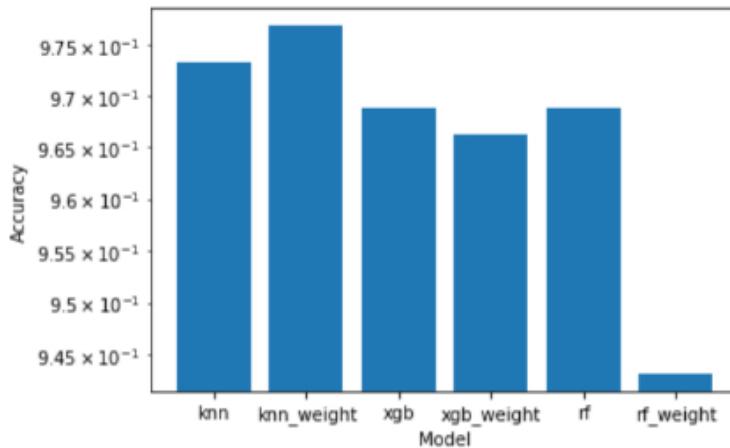
```
In [121]: #ทำการเรียงลำดับตามค่า Accuracy โดยเรียงจากมากไปน้อย
accuracy_df.sort_values(by=[ 'Accuracy' ], ascending=False)
```

```
Out[121]:
```

	Model	Accuracy
1	knn_weight	0.976912
0	knn	0.973372
4	rf	0.968924
2	xgb	0.968916
3	xgb_weight	0.966253

```
5    rf_weight  0.943189
```

```
In [113]: #ทำการ plot bar chart เพื่อให้ง่ายต่อการอ่านค่าและเปรียบเทียบ โดยกำหนดให้ take Log เพื่อใช้ในการแสดงค่า  
plt.bar(accuracy_df['Model'], accuracy_df['Accuracy'], log=True)  
plt.ylabel('Accuracy')  
plt.xlabel('Model');
```



สรุปผล: Model KNN ที่ใช้วิธี Term weighting มี accuracy สูงที่สุด

```
In [ ]:
```

Test with unseen data

ลองนำข่าวจากลิงค์ด้านล่างมาวิเคราะห์

- <https://www.60secondsnow.com/sports/>
- <https://www.60secondsnow.com/business/>
- <https://www.60secondsnow.com/technology/>

knn model (weight)

```
In [123]: doc_cat = ['sport', 'business', 'technology', 'sport', 'business', 'technology']  
docs_new = ["Ajinkya Rahane remains vice-captain in India's revised squad for the Australia Tests but Irfan Pathan feels a much  
more experienced Rohit Sharma has to lead the side in Virat Kohli's absence following the series opener. Kohli has been granted  
paternity leave after the first Test and Pathan said it is bound to have a huge impact on the team.",  
          "The S&P 500 and Dow were set to open at record highs on Monday, as news of the first successful late stage COVID-19  
vaccine trials stirred hopes of the economy emerging from a year of pandemic-driven crisis.",  
          "Apple One More Thing event brought in a couple of new products, including the new M1 processor. Based on the 5nm ar  
chitecture. The company also launched three new products based on the new Apple M1 processor, namely the MacBook Air, MacBook Pr
```

o, and the Apple Mini. Here's everything you need to know about the new offerings.",

"The FIGC has rejected appeals from Napoli and Roma against 3-0 defeats imposed on them earlier this season. Gennaro Gattuso's men did not turn up for the game in Turin last month and were also docked a point by Lega Serie A for failing to fulfil their fixture obligations. Napoli argued they were banned from travelling by local health authorities.",

"On Wednesday, global index provider MSCI (Morgan Stanley Capital International) announced changes to its Global Standard Indexes as part of its semi-annual index review. As part of the changes, two Indian stocks, that is Bosch and LIC Housing Finance, will be removed from the index. On the other hand, 12 stocks will be added to the index",

"In the latest incident, a Nokia phone was charred in flames. The victim called Chandra Babu who hails from Kerala was sleeping when this mishap took place. The device that blasted was a Nokia feature phone and not any smartphone. The device is said to be not charging when this incident happen. The victim got burn injuries on the left arm and shoulder."

]

แปลงข่าวใหม่ก่อนนำไปใช้ model

```
In [124]: X_new_tfidf = vectorizer.transform(docs_new)

predicted = knn_weight.predict(X_new_tfidf)
```

```
In [125]: # sport, business, technology, sport, business, technology

print("predicted:",predicted)

predicted: ['sport' 'business' 'technology' 'sport' 'business' 'technology']
```

```
In [126]: accuracy_score(doc_cat, predicted)

Out[126]: 1.0
```