Assignment 2 - DS4Biz Y63

TextScraping_Classification

Team Detail

Team Name: BBGUN

Student 1

Student ID: 61070297

Student Full Name: นายธนคล สร้อยสน

Student 2

Student ID: 61070300

Student Full Name: นางสาวัญลักษณ์ โชติรัตนเดชเดโช

URL : https://quotes.toscrape.com/

In [2]:
```
import pandas as pd
```

Requests

In [3]:
```
import requests
url = "https://quotes.toscrape.com/"
data = requests.get(url)
print(data.text)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Quotes to Scrape</title>
    <link rel="stylesheet" href="/static/bootstrap.min.css">
    <link rel="stylesheet" href="/static/main.css">
</head>
<body>
    <div class="container">
        <div class="row header-box">
            <div class="col-md-8">
                <h1>
                    <a href="/" style="text-decoration: none">Quotes to Scrape</a>
                </h1>
            </div>
            <div class="col-md-4">
                <p>
```

Beautiful Soup

In [4]:
```
from bs4 import BeautifulSoup
soup = BeautifulSoup(data.text,'html.parser')
print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   Quotes to Scrape
  </title>
  <link href="/static/bootstrap.min.css" rel="stylesheet"/>
  <link href="/static/main.css" rel="stylesheet"/>
 </head>
 <body>
  <div class="container">
   <div class="row header-box">
    <div class="col-md-8">
     <h1>
      <a href="/" style="text-decoration: none">
       Quotes to Scrape
      </a>
     </h1>
```

Finding Tags and Scrape

In [5]:
```
url = "http://quotes.toscrape.com/page/%i/" # url format to follow
page = 1 # page numbers for the url
stop = False # parameter to stop our crawl when necessary
lst_page = [] # list to append scraped info to
scrape = []
while stop == False:
    req = requests.get("http://quotes.toscrape.com/page/%i/" % page)
    soup = BeautifulSoup(req.content)
    if soup.find("div", {"class":"quote"}) == None:
        stop = True
    else:
        for quote in soup.find_all("div", {"class":"quote"}):
            d = {} # dictionary for our scraped information
            d['quote'] = quote.find("span", {"class":"text"}).text
            tags = []
            for tag in quote.find_all("a", {"class":"tag"}):
                tags.append(tag.text)
            d['Tags'] = tags
            lst_page.append(d)
            scrape.append(d)
        page += 1
```

In [6]:
```
lst_page
```

Out[6]:
```
[{'quote': '“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinkin
g.”',
  'Tags': ['change', 'deep-thoughts', 'thinking', 'world']},
 {'quote': '“It is our choices, Harry, that show what we truly are, far more than our abilities.”',
  'Tags': ['abilities', 'choices']},
 {'quote': '“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everyth
ing is a miracle.”',
  'Tags': ['inspirational', 'life', 'live', 'miracle', 'miracles']},
 {'quote': '“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”',
  'Tags': ['aliteracy', 'books', 'classic', 'humor']},
 {'quote': '“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”',
  'Tags': ['be-yourself', 'inspirational']},
 {'quote': '“Try not to become a man of success. Rather become a man of value.”',
  'Tags': ['adulthood', 'success', 'value']},
 {'quote': '“It is better to be hated for what you are than to be loved for what you are not.”',
  'Tags': ['life', 'love']},
 {'quote': '“I have not failed. I've just found 10,000 ways that won't work.”',
  'Tags': ['edison', 'failure', 'inspirational', 'paraphrased']},
 {'quote': '“A woman is like a tea bag; you never know how strong it is until it's in hot water.”',
```

In [110]:
```
#วน loop เพื่อดึง page
```

```
lstpage = [] #สร้าง list มาเพื่อเก็บ link ของแต่ละหน้า
for i in range(1,11): #เก็บ link แต่ละเลื่อนเข้า list
    reqm = "http://quotes.toscrape.com/page/"+ str(i)
    lstpage.append(reqm)
    i += 1
```

In [111]:
```
lstpage
```

Out[111]:
```
['http://quotes.toscrape.com/page/1',
 'http://quotes.toscrape.com/page/2',
 'http://quotes.toscrape.com/page/3',
 'http://quotes.toscrape.com/page/4',
 'http://quotes.toscrape.com/page/5',
 'http://quotes.toscrape.com/page/6',
 'http://quotes.toscrape.com/page/7',
 'http://quotes.toscrape.com/page/8',
 'http://quotes.toscrape.com/page/9',
 'http://quotes.toscrape.com/page/10']
```

In [9]:
```
df = pd.DataFrame(scrape)
df #เรียกดูข้อมูลที่เก็บไว้ใน scrape
```

Out[9]:

	quote	Tags
0	“The world as we have created it is a process ...	[change, deep-thoughts, thinking, world]
1	“It is our choices, Harry, that show what we t...	[abilities, choices]
2	“There are only two ways to live your life. On...	[inspirational, life, live, miracle, miracles]
3	“The person, be it gentleman or lady, who has ...	[aliteracy, books, classic, humor]
4	“Imperfection is beauty, madness is genius and...	[be-yourself, inspirational]
...	...	...
95	“You never really understand a person until yo...	[better-life-empathy]
96	“You have to write the book that wants to be w...	[books, children, difficult, grown-ups, write,...
97	“Never tell the truth to people who are not wo...	[truth]
98	“A person's a person, no matter how small.”	[inspirational]
99	“... a mind needs books as a sword needs a whe...	[books, mind]

100 rows × 2 columns

In [10]:
```
quote = '\n'.join(df.quote)
with open("quote.txt", "w", encoding="utf-8") as f:
    f.write(quote)
```

In [11]:
```
with open('quote.txt', 'r', encoding='utf-8') as f:
    line = f.read()
    line = line.split('\n') #ลบ /n ออก
```

In [12]:
```
line #เรียกดู text quote ที่เก็บไว้ใน line
```

Out[12]:
```
['“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”',
 '“It is our choices, Harry, that show what we truly are, far more than our abilities.”',
 '“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.”',
 '“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”',
 '“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”',
 '“Try not to become a man of success. Rather become a man of value.”',
 '“It is better to be hated for what you are than to be loved for what you are not.”',
 '“I have not failed. I've just found 10,000 ways that won't work.”',
 '“A woman is like a tea bag; you never know how strong it is until it's in hot water.”',
 '“A day without sunshine is like, you know, night.”',
 '“This life is what you make it. No matter what, you're going to mess up sometimes, it's a universal truth. But the good part is you get to decide how you're going to mess it up. Girls will be your friends - they'll act like it anyway. But just remember, some come, some go. The ones that stay with you through everything - they're your true best friends. Don't let go of them. Also remember, sisters make the best friends in the world. As for lovers, well, they'll come and go too. And baby, I hate to say it, most of them - actually pretty much all of them are going to break your heart, but you can't give up because if you give up, you'll never find your soulmate. You'll never find that half who makes you whole and that goes for everything. Just because you fail once, doesn't mean you're gonna fail at everything. Keep trying, hold on, and always, always, always believe in yourself, because if you don't, then who will, sweetie? So keep your head high, keep your chin up, and most importantl
```

In [13]:
```
with open("tag.txt", "w", encoding="utf-8") as f:
    for i in df['Tags']:
        text = ''
        for j in i:
            text += j + ","
        #print(text)
        f.write(text+'\n')
```

In [14]:
```
fin = open('tag.txt', 'r', encoding='UTF-8') #ทำการเปิดไฟล์ และอ่านไฟล์
raw = fin.readlines() #อ่านทีละบรรทัด
rawdata =''
for i in raw:
    text = i.rstrip()
    rawdata += text
print(rawdata)
```

```
change,deep-thoughts,thinking,world,abilities,choices,inspirational,life,live,miracle,miracles,aliteracy,books,classic,humor,be-yourself,inspirational,adulthood,success,value,life,love,edison,failure,inspirational,paraphrased,misattributed-eleanor-roosevelt,humor,obvious,simile,friends,heartbreak,inspirational,life,love,sisters,courage,friends,simplicity,understand,love,fantasy,life,navigation,activism,apathy,hate,indifference,inspirational,love,opposite,philosophy,friendship,lack-of-friendship,lack-of-love,love,marriage,unhappy-marriage,books,contentment,friends,friendship,life,fate,life,misattributed-john-lennon,planning,plans,love,poetry,happiness,attributed-no-source,humor,religion,humor,comedy,life,yourself,children,fairy-tales,imagination,music,learning,reading,seuss,dumbledore,friendship,misattributed-to-mother-teresa,paraphrased,death,inspirational,chocolate,food,humor,misattributed-to-c-s-lewis,reading,knowledge,learning,understanding,wisdom,books,library,inspirational,read,readers,reading,reading-books,books,inspirational,reading,tea,girls,love,life,simile,love,attributed-no-source,hope,inspirational,dumbledore,love,friendship,love,attributed,fear,inspiration,attributed-no-source,music,books,thought,misattributed-to-einstein,drug,romance,simile,books,friends,novelist-quotes,inspirational,inspirational,life,yourself,alcohol,the-hunger-games,humor,love,bilbo,journey,lost,quest,travel,wander,live-death-love,good,writing,life,regrets,education,troubles,friendship,love,humor,humor,open-mind,thinking,humor,philosophy,authors,books,literature,reading,writing,humor,insanity,lies,lying,self-indulgence,truth,beatles,connection,dreamers,dreaming,dreams,hope,inspirational,peace,humor,sinister,books,classic,reading,mistakes,humor,love,romantic,women,integrity,books,library,reading,elizabeth-bennet,jane-austen,age,fairytales,growing-up,god,death,life,misattributed-mark-twain,truth,christianity,faith,religion,sun,truth,death,life,adventure,love,courage,life,better-life-empathy,books,children,difficult,grown-ups,write,writers,writing,truth,inspirational,books,mind,
```

In [15]:
```
from sklearn.feature_extraction.text import CountVectorizer
tokenize = CountVectorizer().build_tokenizer() #เครื่องมือในการsplitคำโดยลัดตัวแปลแปลกๆออก
# convert to lowercase, then tokenize
tokens1 = tokenize(rawdata.lower())
print(tokens1)
```

```
['change', 'deep', 'thoughts', 'thinking', 'world', 'abilities', 'choices', 'inspirational', 'life', 'live', 'miracle', 'miracles', 'aliteracy', 'books', 'classic', 'humor', 'be', 'yourself', 'inspirational', 'adulthood', 'success', 'value', 'life', 'love', 'edison', 'failure', 'inspirational', 'paraphrased', 'misattributed', 'eleanor', 'roosevelt', 'humor', 'obvious', 'simile', 'friends', 'heartbreak', 'inspirational', 'life', 'love', 'sisters', 'courage', 'friends', 'simplicity', 'understand', 'love', 'fantasy', 'life', 'navigation', 'activism', 'apathy', 'hate', 'indifference', 'inspirational', 'love', 'opposite', 'philosophy', 'friendship', 'lack', 'of', 'friendship', 'lack', 'of', 'love', 'love', 'marriage', 'unhappy', 'marriage', 'books', 'contentment', 'friends', 'friendship', 'life', 'fate', 'life', 'misattributed', 'john', 'lennon', 'planning', 'plans', 'love', 'poetry', 'happiness', 'attributed', 'no', 'source', 'humor', 'religion', 'humor', 'comedy', 'life', 'yourself', 'children', 'fairy', 'tales', 'imagination', 'music', 'learning', 'reading', 'seuss', 'dumbledore', 'friendship', 'misattributed', 'to', 'mother', 'teresa', 'paraphrased', 'death', 'inspirational', 'chocolate', 'food', 'humor', 'misattributed', 'to', 'lewis', 'reading', 'knowledge', 'learning', 'understanding', 'wisdom', 'books', 'library', 'inspirational', 'read', 'readers', 'reading', 'reading', 'books', 'books', 'inspirational', 'reading', 'tea', 'girls', 'love', 'life', 'simile', 'love', 'attributed', 'no', 'source', 'hope', 'inspirational', 'dumbledore', 'love', 'friendship', 'love', 'attributed', 'fear', 'inspiration', 'attributed', 'no', 'source', 'music', 'books', 'thought', 'misattributed', 'to', 'einstein', 'drug', 'romance', 'simile', 'books', 'friends', 'novelist', 'quotes', 'inspirational', 'inspirational', 'life', 'yourself', 'alcohol', 'the', 'hunger', 'games', 'humor', 'love', 'bilbo', 'journey', 'lost', 'quest', 'travel', 'wander', 'live', 'death', 'love', 'good', 'writing', 'life', 'regrets', 'education', 'troubles', 'friendship', 'love', 'humor', 'humor', 'open', 'mind', 'thinking', 'humor', 'philosophy', 'authors', 'books', 'literature', 'reading', 'writing', 'humor', 'insanity', 'lies', 'lying', 'self', 'indulgence', 'truth', 'beatles', 'connection', 'dreamers', 'dreaming', 'dreams', 'hope', 'inspirational', 'peace', 'humor', 'sinister', 'books', 'classic', 'reading', 'mistakes', 'humor', 'love', 'romantic', 'women', 'integrity', 'books', 'library', 'reading', 'elizabeth', 'bennet', 'jane', 'austen', 'age', 'fairytales', 'growing', 'up', 'god', 'death', 'life', 'misattributed', 'mark', 'twain', 'truth', 'christianity', 'faith', 'religion', 'sun', 'truth', 'death', 'life', 'adventure', 'love', 'courage', 'life', 'better', 'life', 'empathy', 'books', 'children', 'difficult', 'grown', 'ups', 'write', 'writers', 'writing', 'truth', 'inspirational', 'books', 'mind']
```

In [113]:
```
lst_tags = [] #สร้าง list มาเพื่อเก็บ tag
for i in lstpage:
    reqq = requests.get(i)
    soupq = BeautifulSoup(reqq.content, 'html.parser')
    quotes = soupq.find_all('div', class_='quote')
    for q in quotes:
        print(q.find_all('a'))
        tag = []
```

```
        for j in q.find_all('a',class_='tag'):
            tag.append(j.text)
        tags.append(tag) #เก็บ tag ในแต่ละ quote
```

```
[<a href="/author/Albert-Einstein">(about)</a>, <a class="tag" href="/tag/change/page/1/">change</a>, <a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a>, <a class="tag" href="/tag/thinking/page/1/">thinking</a>, <a class="tag" href="/tag/world/page/1/">world</a>]
[<a href="/author/J-K-Rowling">(about)</a>, <a class="tag" href="/tag/abilities/page/1/">abilities</a>, <a class="tag" href="/tag/choices/page/1/">choices</a>]
[<a href="/author/Albert-Einstein">(about)</a>, <a class="tag" href="/tag/inspirational/page/1/">inspirational</a>, <a class="tag" href="/tag/life/page/1/">life</a>, <a class="tag" href="/tag/live/page/1/">live</a>, <a class="tag" href="/tag/miracle/page/1/">miracle</a>, <a class="tag" href="/tag/miracles/page/1/">miracles</a>]
[<a href="/author/Jane-Austen">(about)</a>, <a class="tag" href="/tag/aliteracy/page/1/">aliteracy</a>, <a class="tag" href="/tag/books/page/1/">books</a>, <a class="tag" href="/tag/classic/page/1/">classic</a>, <a class="tag" href="/tag/humor/page/1/">humor</a>]
[<a href="/author/Marilyn-Monroe">(about)</a>, <a class="tag" href="/tag/be-yourself/page/1/">be-yourself</a>, <a class="tag" href="/tag/inspirational/page/1/">inspirational</a>]
[<a href="/author/Albert-Einstein">(about)</a>, <a class="tag" href="/tag/adulthood/page/1/">adulthood</a>, <a class="tag" href="/tag/success/page/1/">success</a>, <a class="tag" href="/tag/value/page/1/">value</a>]
[<a href="/author/Andre-Gide">(about)</a>, <a class="tag" href="/tag/life/page/1/">life</a>, <a class="tag" href="/tag/love/page/1/">love</a>]
[<a href="/author/Thomas-A-Edison">(about)</a>, <a class="tag" href="/tag/edison/page/1/">edison</a>, <a class="tag" href="/tag/failure/page/1/">failure</a>, <a class="tag" href="/tag/inspirational/page/1/">inspirational</a>, <a class="tag" href="/ta
```

In [106]:
```
file = open('C:/Users/thunyaluck/Desktop/quote/datastore/quotes.txt','w',encoding="utf-8")
for i in line:
    file.writelines(i+'\n')
file.close()
```

In [107]:
```
fin = open('C:/Users/thunyaluck/Downloads/ds4biz-assign2/datastore/quotes.txt', 'r', encoding='UTF-8') #ทำการเปิดไฟล์ และอ่านไฟล์
raw = fin.readlines() #อ่านทีละบรรทัด
fin.close()
print("Read %d raw text documents" % len(raw))
```

```
Read 100 raw text documents
```

In [108]:
```
file = open('C:/Users/thunyaluck/Desktop/quote/target/tag.txt','w',encoding="utf-8")
for i in tags:
    x = ','.join(str(x) for x in i)
    file.writelines(x+'\n')
file.close()
```

In [109]:
```
fin = open('C:/Users/thunyaluck/Downloads/ds4biz-assign2/target/tag.txt', 'r', encoding='UTF-8') #ทำการเปิดไฟล์ และอ่านไฟล์
raw = fin.readlines() #อ่านทีละบรรทัด
fin.close()
print("Read %d raw text documents" % len(raw))
```

```
Read 100 raw text documents
```

Term Weighting

In [17]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# we can pass in the same preprocessing parameters
vectorizer = TfidfVectorizer(stop_words="english",min_df = 5) #ทำการถ่วงน้ำหนักและstop wordเพื่อคัดคำที่ไม่จำเป็นออกจากเนื้อหาของข่าว
X = vectorizer.fit_transform(line)
# display some sample weighted values
print(X)
```

```
  (0, 19)       1.0
  (2, 7)        0.6592779466037926
  (2, 9)        0.7518993211340775
  (3, 3)        1.0
  (7, 4)        1.0
  (8, 5)        0.7297599850512536
  (8, 8)        0.6837034183167392
  (9, 1)        0.6037284973895782
  (9, 5)        0.5817573649270407
  (9, 8)        0.5450415303377056
  (10, 15)      0.22712786304351262
  (10, 6)       0.22712786304351262
  (10, 2)       0.4542557260870252 5
  (10, 18)      0.21797049755805145
  (10, 11)      0.42007605352238897
  (10, 8)       0.19678211989527714
  (10, 4)       0.4359409951161029
  (10, 3)       0.21003802676119449
  (10, 7)       0.3822403824850753
```

สร้างโมเดล โดยใช้วิธีการ Holdout Splitting

In [18]:
```
#เอา tag แต่ละอันมาทำเป็น 0, 1
from sklearn.preprocessing import MultiLabelBinarizer
mlb = MultiLabelBinarizer()
binary_labels=pd.DataFrame(mlb.fit_transform(tags),columns=mlb.classes_)
df = df.merge(binary_labels, how='inner', left_index=True, right_index=True)
df
```

Out[18]:

	quote	Tags	abilities	activism	adulthood	adventure	age	alcohol	aliteracy	apathy	...	unhappy-marriage	value	wander	wisdom	women	wo
0	“The world as we have created it is a process ...	[change, deep-thoughts, thinking, world]	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	“It is our choices, Harry, that show what we t...	[abilities, choices]	1	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	“There are only two ways to live your life. On...	[inspirational, life, live, miracle, miracles]	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
	“The person,	[aliteracy,															

Top 15 Tags

In [19]:
```
#หา tag top 15
df.iloc[:,12:].sum().nlargest(15)
```

Out[19]:
```
love             14
inspirational    13
life             13
humor            12
books            11
reading           7
friendship        5
friends           4
truth             4
death             3
simile            3
writing           3
children          2
classic           2
courage           2
dtype: int64
```

ทำการทำนายค่า x กับค่า y ของ tags แต่ละตัวว่ามี accuracy เท่าไหร่บ้าง

In [190]:
```
Y = df['love']
```

In [191]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [192]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 76.67 percent
```

Naive Bayes

In [193]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 73.33 percent
```

SVM

In [194]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 80.00 percent
```

Decision Tree

In [195]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 83.33 percent
```

In [20]:
```
Y1 = df['inspirational']
```

In [21]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y1, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [22]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 83.33 percent
```

Naive Bayes

In [23]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 70.00 percent
```

SVM

In [24]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 83.33 percent
```

Decision Tree

In [25]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 70.00 percent
```

In [27]:
```
Y2 = df['life']
```

In [28]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y2, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 83.33 percent
```

Naive Bayes

In [30]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 86.67 percent
```

SVM

In [31]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 80.00 percent
```

Decision Tree

In [32]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 86.67 percent
```

In [33]:
```
Y3 = df['humor']
```

In [34]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y3, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [35]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

Naive Bayes

In [36]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 90.00 percent
```

SVM

In [37]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 96.67 percent
```

Decision Tree

In [38]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 83.33 percent
```

In [39]:
```
Y4 = df['books']
```

In [40]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y4, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [41]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 93.33 percent
```

Naive Bayes

In [42]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 93.33 percent
```

SVM

In [43]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 93.33 percent
```

Decision Tree

In [44]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 96.67 percent
```

In [45]:
```
Y5 = df['reading']
```

In [46]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y5, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [47]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

Naive Bayes

In [48]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

SVM

In [49]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 96.67 percent
```

Decision Tree

In [50]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 96.67 percent
```

In [51]:
```
Y6 = df['friendship']
```

In [52]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y6, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [53]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 93.33 percent
```

Naive Bayes

In [54]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 86.67 percent
```

SVM

In [55]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 93.33 percent
```

Decision Tree

In [56]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 90.00 percent
```

In [58]:
```
Y7 = df['friends']
```

In [59]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y7, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [60]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

Naive Bayes

In [61]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 93.33 percent
```

SVM

In [62]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 96.67 percent
```

Decision Tree

In [63]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 80.00 percent
```

In [64]:
```
Y8 = df['truth']
```

In [65]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y8, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [66]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

Naive Bayes

In [67]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

SVM

In [68]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 96.67 percent
```

Decision Tree

In [69]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 96.67 percent
```

In [70]:
```
Y9 = df['death']
```

In [71]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y9, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [72]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

Naive Bayes

In [73]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

SVM

In [74]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 96.67 percent
```

Decision Tree

In [75]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 96.67 percent
```

In [76]:
```
Y10 = df['simile']
```

In [77]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y10, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [78]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

Naive Bayes

In [79]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

SVM

In [80]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 100.00 percent
```

Decision Tree

In [81]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 100.00 percent
```

In [82]:
```
Y11 = df['writing']
```

In [83]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y11, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [84]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 93.33 percent
```

Naive Bayes

In [85]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

SVM

In [86]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 100.00 percent
```

Decision Tree

In [87]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
```

```
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 93.33 percent
```

In [88]:
```
Y12 = df['children']
```

In [89]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y12, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [90]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

Naive Bayes

In [91]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

SVM

In [92]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 100.00 percent
```

Decision Tree

In [93]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 93.33 percent
```

In [94]:
```
Y13 = df['classic']
```

In [95]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y13, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [96]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

Naive Bayes

In [97]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 96.67 percent
```

SVM

In [98]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 96.67 percent
```

Decision Tree

In [99]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 96.67 percent
```

In [100]:
```
Y14 = df['courage']
```

In [101]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, Y14, test_size=0.3, random_state=101) #แบ่งชุดข้อมูลเป็นแบบ train,test
```

K-Nearest Neighbors

In [102]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
classifier_pred = classifier.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,classifier_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

Naive Bayes

In [103]:
```
from sklearn.naive_bayes import BernoulliNB
naive = BernoulliNB(alpha=0.2)
naive.fit(X_train, y_train)
naive_pred = naive.predict(X_test)
print("Accuracy of KNN = %.2f percent"% (metrics.accuracy_score(y_test,naive_pred)*100))
```

```
Accuracy of KNN = 100.00 percent
```

SVM

In [104]:
```
from sklearn import svm
svm = svm.SVC(C=1, kernel='linear')
svm.fit(X_train, y_train)
svm_pred = svm.predict(X_test)
print("Accuracy of SVM = %0.2f percent" % (metrics.accuracy_score(y_test,svm_pred)*100))
```

```
Accuracy of SVM = 100.00 percent
```

Decision Tree

In [105]:
```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
decision_tree_pred = decision_tree.predict(X_test)
print("Accuracy of Decision tree = %0.2f percent" % (metrics.accuracy_score(y_test,decision_tree_pred)*100))
```

```
Accuracy of Decision tree = 100.00 percent
```

สรุปผล

Target	K-Nearest-Neighbors	Naive Bayes	SVM	Decision Tree
love	76.67 %	73.33 %	80.00 %	83.33 %
inspirational	83.33 %	70.00 %	83.33 %	70.00 %
life	83.33 %	86.67 %	80.00 %	86.67 %
humor	96.67 %	90.00 %	96.67 %	83.33 %
books	93.33 %	93.33 %	93.33 %	96.67 %
reading	96.67 %	96.67 %	96.67 %	96.67 %
friendship	93.33 %	86.67 %	93.33 %	90.00 %
friends	96.67 %	93.33 %	96.67 %	80.00 %
truth	96.67 %	96.67 %	96.67 %	96.67 %
death	96.67 %	100.00 %	96.67 %	96.67 %
simile	100.00 %	100.00 %	100.00 %	100.00 %
writing	93.33 %	100.00 %	93.33 %	93.33 %
children	100.00 %	100.00 %	100.00 %	93.33 %
classic	96.67 %	96.67 %	96.67 %	96.67 %
courage	100.00 %	100.00 %	100.00 %	100.00 %
Total	93.56 %	92.22 %	93.56 %	90.89 %

จากผลการทดลอง model ได้นำ model ว่าวัดประสิทธิภาพด้วยค่า accuracy หรือเปอร์เซนต์ความถูกต้อง จะเห็นได้ว่า K-Nearest-Neighbors และ SVM มีค่า accuracy อยู่ที่ 93.56% ซึ่งมากกว่า Naive Bayes และ Decision Tree

In [ ]: