Assignment 2 - DS4Biz Y63

TextScraping_Classification

Team Detail

Team Name: Platinum bread

Student 1

Student ID: 61070315
Student Full Name: นายภูธเนศ สุวรรณดาม

Student 2

Student ID: 61070326
Student Full Name: น.ส. สุวิภา นันชัย

In [1]:
```
#import library
import pandas as pd
import numpy as np
import bs4
import requests
from sklearn import preprocessing
import nltk
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
import re
from sklearn.model_selection import train_test_split
```

Data Collection

In [2]:
```
#ฟังก์ชั่นดึง quotes จากเว็บ
def extract_quotes():
    selector='body> div > div > div.col-md-8 > div > span.text'
    tags = html_page.select(selector)
    for t in tags:
        quotes.append(t.text)
```

In [3]:
```
#ฟังก์ชั่นดึงชื่อผู้เขียน
def extract_author():
    selector='body> div > div > div.col-md-8 > div > span > small'
    tags = html_page.select(selector)
    for t in tags:
        authors.append(t.text)
```

In [4]:
```
#ฟังก์ชั่นดึงรายละเอียดผู้เขียน
def extract_detail():
    selector='body> div > div > div.col-md-8 > div > span > a'
    tags = html_page.select(selector)
    for t in tags:
        response = requests.get('https://quotes.toscrape.com'+t['href'])
        soup = bs4.BeautifulSoup(response.content, 'lxml')
        selector = 'body > div > div.author-details > div'
        tags2 = soup.select(selector)
        for t in tags2:
            detail = t.text.replace('\n',' ')
            authordetail.append(detail)
```

In [5]:
```
#ฟังก์ชั่นดึง tags
def extract_tags():
    selector='body> div > div > div.col-md-8 > div > div > meta'
    tags = html_page.select(selector)
    for t in tags:
        con = t['content'].split(',')
        tag.append(con)
```

In [6]:
```
#ลูปดึงข้อมูลจากทุกหน้า
quotes = []
authors = []
authordetail = []
tag = []
for i in range(1,11):
    response = requests.get('https://quotes.toscrape.com/page/'+str(i)+'/')
    html_page = bs4.BeautifulSoup(response.content, 'lxml')
    extract_quotes()
    extract_author()
    extract_tags()
    extract_detail()
```

In [7]:
```
#เปลี่ยนข้อมูลลิสต์ให้อยู่ในรูปแบบ dataframe
df = pd.DataFrame()
df['Quotes'] = quotes
df['Author'] = authors
df['Author Detail'] = authordetail
df['Tags'] = tag
df
```

Out[7]:

	Quotes	Author	Author Detail	Tags
0	“The world as we have created it is a process ...	Albert Einstein	In 1879, Albert Einstein was born in ...	[change, deep-thoughts, thinking, world]
1	“It is our choices, Harry, that show what we t...	J.K. Rowling	See also: Robert GalbraithAlthough sh...	[abilities, choices]
2	“There are only two ways to live your life. On...	Albert Einstein	In 1879, Albert Einstein was born in ...	[inspirational, life, live, miracle, miracles]
3	“The person, be it gentleman or lady, who has ...	Jane Austen	Jane Austen was an English novelist w...	[aliteracy, books, classic, humor]

4	“Imperfection is beauty, madness is genius and...	Marilyn Monroe	Marilyn Monroe (born Norma Jeane Mort...	[be-yourself, inspirational]
...	...	...	...	...
95	"You never really understand a person until yo...	Harper Lee	Harper Lee, known as Nelle, was born ...	[better-life-empathy]
96	“You have to write the book that wants to be w...	Madeleine L'Engle	Madeleine L'Engle was an American wri...	[books, children, difficult, grown-ups, write,...
97	“Never tell the truth to people who are not wo...	Mark Twain	Samuel Langhorne Clemens, better know...	[truth]
98	“A person's a person, no matter how small.”	Dr. Seuss	Theodor Seuss Geisel was born 2 March...	[inspirational]
99	“... a mind needs books as a sword needs a whe...	George R.R. Martin	George R. R. Martin was born Septembe...	[books, mind]

100 rows × 4 columns

In [8]:
```
#เซฟข้อมูล Quotes เป็นไฟล์ txt
np.savetxt('datastore/AllQuote_Quotes.txt',df['Quotes'].values,fmt='%s',encoding='utf-8')
```

In [9]:
```
#เซฟข้อมูล Quote + Author Detail เป็นไฟล์ txt
np.savetxt('datastore/AllQuote_Quotes+AuthorDetail.txt',(df['Quotes']+df['Author Detail']).values,fmt='%s',encoding='utf-8')
```

In [10]:
```
#ลบ Bad Character
for index, row in df.iterrows():
    row['Quotes'] = re.sub('[^a-zA-Z0-9\s]', '', row['Quotes'])
df
```

Out[10]:

	Quotes	Author	Author Detail	Tags
0	The world as we have created it is a process o...	Albert Einstein	In 1879, Albert Einstein was born in ...	[change, deep-thoughts, thinking, world]
1	It is our choices Harry that show what we trul...	J.K. Rowling	See also: Robert GalbraithAlthough sh...	[abilities, choices]
2	There are only two ways to live your life One ...	Albert Einstein	In 1879, Albert Einstein was born in ...	[inspirational, life, live, miracle, miracles]
3	The person be it gentleman or lady who has not...	Jane Austen	Jane Austen was an English novelist w...	[aliteracy, books, classic, humor]
4	Imperfection is beauty madness is genius and i...	Marilyn Monroe	Marilyn Monroe (born Norma Jeane Mort...	[be-yourself, inspirational]
...	...	...	...	...
95	You never really understand a person until you...	Harper Lee	Harper Lee, known as Nelle, was born ...	[better-life-empathy]
96	You have to write the book that wants to be wr...	Madeleine L'Engle	Madeleine L'Engle was an American wri...	[books, children, difficult, grown-ups, write,...
97	Never tell the truth to people who are not wor...	Mark Twain	Samuel Langhorne Clemens, better know...	[truth]
98	A persons a person no matter how small	Dr. Seuss	Theodor Seuss Geisel was born 2 March...	[inspirational]
99	a mind needs books as a sword needs a whetsto...	George R.R. Martin	George R. R. Martin was born Septembe...	[books, mind]

100 rows × 4 columns

In [11]:
```
#เก็บ tags ทั้งหมดให้อยู่ในลิสต์เดียว
alltags = []
for i in df.Tags:
    for j in i:
        if j != '':
            alltags.append(j)
        else:
            pass
```

In [12]:
```
#เลือก tag ที่มีจำนวนสูงสุด 15 อันดับ
counts = pd.Series(alltags)
top = counts.value_counts(sort=True).nlargest(15)
toptags = top.index.tolist()
```

In [13]:
```
toptags
#toptags = ['love', 'life', 'inspirational', 'humor', 'books', 'reading', 'friendship', 'truth', 'friends', 'writing', 'attribute
```

Out[13]:
```
['love',
 'inspirational',
 'life',
 'humor',
 'books',
 'reading',
 'friendship',
 'friends',
 'truth',
 'writing',
 'death',
 'simile',
 'attributed-no-source',
 'classic',
 'learning']
```

In [14]:
```
#สร้างลิสต์ tag ใหม่ที่มี tag 15 อันดับ
new = []
for i in df.Tags:
    tag = []
    for j in i:
        if j in toptags:
            tag.append(j)
    new.append(tag)
```

In [15]:
```
#เตรียมข้อมูลให้เป็น Binary
from sklearn import preprocessing
mlb = preprocessing.MultiLabelBinarizer()
mlb.fit(new)
```

Out[15]:
```
MultiLabelBinarizer()
```

In [16]:
```
labels =mlb.transform(new)
labels
```

Out[16]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 1, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 1, 0, ..., 0, 0, 0]])
```

In [17]:
```
#นำข้อมูลที่ทำเป็น Binary ให้อยู่ในรูปแบบ Dataframe
dftags = pd.DataFrame(labels,columns=list(mlb.classes_))
dftags
```

Out[17]:

	attributed-no-source	books	classic	death	friends	friendship	humor	inspirational	learning	life	love	reading	simile	truth	writing
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

3	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
95	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
96	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	
97	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
98	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
99	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	

100 rows × 15 columns

In [18]:
```
#เชื่อม 2 dataframe
df_all = pd.concat([df,dftags], axis=1)
df_all
```

Out[18]:

	Quotes	Author	Author Detail	Tags	attributed-no-source	books	classic	death	friends	friendship	humor	inspirational	learning	life	love	rea
0	The world as we have created it is a process o...	Albert Einstein	In 1879, Albert Einstein was born in ...	[change, deep-thoughts, thinking, world]	0	0	0	0	0	0	0	0	0	0	0	
1	It is our choices Harry that show what we trul...	J.K. Rowling	See also: Robert GalbraithAlthough sh...	[abilities, choices]	0	0	0	0	0	0	0	0	0	0	0	
2	There are only two ways to live your life One ...	Albert Einstein	In 1879, Albert Einstein was born in ...	[inspirational, life, live, miracle, miracles]	0	0	0	0	0	0	0	1	0	1	0	
3	The person be it gentleman or lady who has not...	Jane Austen	Jane Austen was an English novelist w...	[aliteracy, books, classic, humor]	0	1	1	0	0	0	1	0	0	0	0	
4	Imperfection is beauty madness is genius and i...	Marilyn Monroe	Marilyn Monroe (born Norma Jeane Mort...	[be-yourself, inspirational]	0	0	0	0	0	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
95	You never really understand a person until you...	Harper Lee	Harper Lee, known as Nelle, was born ...	[better-life-empathy]	0	0	0	0	0	0	0	0	0	0	0	
96	You have to write the book that wants to be wr...	Madeleine L'Engle	Madeleine L'Engle was an American wri...	[books, children, difficult, grown-ups, write,...	0	1	0	0	0	0	0	0	0	0	0	
97	Never tell the truth to people who are not wor...	Mark Twain	Samuel Langhorne Clemens, better know...	[truth]	0	0	0	0	0	0	0	0	0	0	0	
98	A persons a person no matter how small	Dr. Seuss	Theodor Seuss Geisel was born 2 March...	[inspirational]	0	0	0	0	0	0	0	1	0	0	0	
99	a mind needs books as a sword needs a whetsto...	George R.R. Martin	George R. R. Martin was born Septembe...	[books, mind]	0	1	0	0	0	0	0	0	0	0	0	

100 rows × 19 columns

In [19]:
```
#เซฟข้อมูล tags เป็นไฟล์ txt
np.savetxt('target/tags.txt',df['Tags'].values,fmt='%s',encoding='utf-8')
```

In [20]:
```
#เซฟข้อมูล tags ที่เป็น Binary เป็นไฟล์ csv
dftags.to_csv('target/tags.csv')
```

Text Classification

Data preprocessing

In [21]:
```
#ดึงข้อมูล quotes จากไฟล์ที่เซฟไว้
fin = open("datastore/AllQuote_Quotes.txt","r", encoding='utf-8')
raw_documents = fin.readlines()
fin.close()
print("Read %d raw text documents" % len(raw_documents))
```

```
Read 100 raw text documents
```

In [22]:
```
#นำข้อมูลที่ดึงมาให้อยู่ในรูปแบบลิสต์ และเป็นพิมพ์เล็กทั้งหมด
data = []
for i in raw_documents:
    data.append(i.lower())
data
```

Out[22]:
```
['“the world as we have created it is a process of our thinking. it cannot be changed without changing our thinking.”\n',
 '“it is our choices, harry, that show what we truly are, far more than our abilities.”\n',
 '“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.”\n',
 '“the person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”\n',
 "“imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.”\n",
 '“try not to become a man of success. rather become a man of value.”\n',
 '“it is better to be hated for what you are than to be loved for what you are not.”\n',
 "“i have not failed. i've just found 10,000 ways that won't work.”\n",
 "“a woman is like a tea bag; you never know how strong it is until it's in hot water.”\n",
 '“a day without sunshine is like, you know, night.”\n',
 "“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 belie
```

```
ve in yourself, because if you don't, then who will, sweetie? so keep your head high, keep your chin up, and most importantl
y, keep smiling, because life's a beautiful thing and there's so much to smile about."\n"
```

In [23]:
```
#ดึง stopword จาก library
from sklearn.feature_extraction import text
stopwords = text.ENGLISH_STOP_WORDS
print(stopwords)
```

```
frozenset({'serious', 'most', 'fifty', 'but', 'to', 'after', 'them', 'thru', 'hereupon', 'herself', 'its', 'him', 'nevertheles
s', 'hereby', 'under', 'often', 'become', 'through', 'hence', 'whereupon', 'please', 'beside', 'with', 'such', 'mostly', 'see
m', 'anything', 'two', 'between', 'wherever', 'find', 'during', 'therefore', 'another', 'she', 'myself', 'down', 'inc', 'con',
'anyhow', 'meanwhile', 'against', 'a', 'here', 'give', 'front', 'further', 'interest', 'together', 'somehow', 'whatever', 'whil
e', 'nine', 'sixty', 'that', 'eg', 'before', 'around', 'himself', 'namely', 'seems', 'nowhere', 'we', 'latter', 'although', 'do
ne', 'or', 'why', 'ltd', 'one', 'too', 'formerly', 'there', 'moreover', 'some', 'almost', 'they', 'for', 'my', 'have', 'withou
t', 'her', 'anyone', 'fire', 'cry', 'whereas', 'by', 'sometimes', 'thereupon', 'couldnt', 'yet', 'amongst', 'same', 'i', 'towar
d', 'then', 'thereafter', 'third', 'yourself', 'off', 'until', 'towards', 'put', 'per', 'except', 'least', 'co', 'about', 'ma
y', 'became', 'empty', 'even', 'should', 'our', 'across', 'found', 'thus', 'every', 'onto', 'both', 'however', 'someone', 'he',
'seemed', 'ever', 'twenty', 'own', 'of', 'was', 'either', 'beyond', 'hers', 'though', 'itself', 'becoming', 'over', 'via', 'no
t', 'out', 'few', 'fifteen', 'eleven', 'yourselves', 'ours', 'it', 'de', 'keep', 'would', 'describe', 'thin', 'nothing', 'whos
e', 'wherein', 'besides', 'six', 'upon', 'still', 'first', 'thereby', 'top', 'as', 'everything', 'take', 'twelve', 'more', 'hun
dred', 'never', 'because', 'cannot', 'since', 'afterwards', 'thick', 'system', 'nobody', 'everywhere', 'hereafter', 'detail',
'many', 'eight', 'none', 'latterly', 'where', 'being', 'side', 'whereby', 'whom', 'only', 'next', 'part', 'less', 'be', 'therei
n', 'forty', 'on', 'above', 'us', 'whence', 'go', 'rather', 'elsewhere', 'bill', 'move', 'your', 'bottom', 'cant', 'throughou
t', 'nor', 'whole', 'ten', 'get', 'been', 'beforehand', 'call', 'who', 'from', 'already', 'once', 'those', 'amoungst', 'back',
'within', 'something', 'now', 'must', 'somewhere', 'below', 'thence', 'this', 'everyone', 'very', 'whoever', 'will', 'if', 'wer
e', 'three', 'all', 'perhaps', 'behind', 'each', 'ie', 'much', 'noone', 'what', 'indeed', 'had', 'are', 'so', 'alone', 'wheneve
r', 'whereafter', 'themselves', 'else', 'seeming', 'etc', 'their', 'has', 'becomes', 'sometime', 'other', 'un', 'whether', 'alw
ays', 'at', 'along', 'herein', 'see', 'ourselves', 'might', 'others', 'name', 'again', 'anywhere', 'due', 'anyway', 'last', 'fu
ll', 'which', 'enough', 'me', 'former', 'his', 'whither', 'no', 'any', 'and', 'am', 'can', 'amount', 'these', 'into', 'mill',
're', 'an', 'in', 'mine', 'yours', 'than', 'how', 'well', 'neither', 'among', 'sincere', 'otherwise', 'several', 'could', 'mad
e', 'five', 'do', 'hasnt', 'you', 'also', 'show', 'up', 'the', 'is', 'when', 'fill', 'four'})
```

In [24]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [25]:
```
#ฟังก์ชั่นการตัดคำฟุ่มเฟือย หรือ tokenizer
def lemma_tokenizer(text):
    # use the standard scikit-learn tokenizer first
    standard_tokenizer = CountVectorizer().build_tokenizer()
    tokens = standard_tokenizer(text)
    # then use NLTK to perform lemmatisation on each token
    lemmatizer = nltk.stem.WordNetLemmatizer()
    lemma_tokens = []
    for token in tokens:
        #ตัด stopwords
        if not token in stopwords:
            lemma_tokens.append(lemmatizer.lemmatize(token))
    return lemma_tokens
```

In [26]:
```
raw_documents[0]
```

Out[26]:
```
'"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."\n'
```

In [27]:
```
#ทดสอบฟังก์ชั่น lemma_tokenizer
lemma_tokenizer(data[0])
```

Out[27]:
```
['world', 'created', 'process', 'thinking', 'changed', 'changing', 'thinking']
```

In [28]:
```
vectorizer = CountVectorizer(stop_words="english", min_df=3, tokenizer=lemma_tokenizer)
X = vectorizer.fit_transform(data)
print(X.shape)
```

```
(100, 57)
```

Term Weighting

As well as including/excluding terms, we can also modify or weight the frequency values themselves. We can improve the usefulness of the document-term matrix by giving more weight to the more "important" terms.

The most common normalisation is term frequency–inverse document frequency (TF-IDF). In Scikit-learn, we can generate at TF-IDF weighted document-term matrix by using TfidfVectorizer() in place of CountVectorizer().

In [29]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# we can pass in the same preprocessing parameters
vectorizer = TfidfVectorizer(stop_words="english",min_df=5,  tokenizer=lemma_tokenizer)
X = vectorizer.fit_transform(data)
# display some sample weighted values
print(X)
```

```
  (0, 21)       1.0
  (2, 8)        0.5067473898903182
  (2, 10)       0.6095929309134702
  (2, 20)       0.6095929309134702
  (3, 4)        1.0
  (7, 5)        0.7071067811865476
  (7, 20)       0.7071067811865476
  (8, 6)        0.7297599850512536
  (8, 9)        0.6837034183167392
  (9, 1)        0.6037284973895782
  (9, 6)        0.5817573649270407
  (9, 9)        0.5450415303377056
  (10, 16)      0.1558368136413415
  (10, 7)       0.1798683868468555
  (10, 2)       0.359736773693711
  (10, 3)       0.5178493019387345
  (10, 19)      0.17261643397957818
  (10, 12)      0.4823803622466967
  (10, 9)       0.1558368136413415
  (10, 5)       0.34523286795915636
  (10, 4)       0.16633450666858549
  (10, 8)       0.28698799784389945
  (10, 21)      0.17261643397957818
  (11, 3)       0.7071067811865476
  (11, 5)       0.7071067811865476
  :     :
  (84, 17)      0.3978406030577204
  (84, 11)      0.33086863479487993
  (84, 1)       0.42709341984150484
  (84, 21)      0.42709341984150484
  (85, 1)       1.0
  (87, 18)      0.5154255685643149
  (87, 8)       0.8569343517855686
  (88, 19)      0.5773502691896257
  (88, 20)      0.5773502691896257
  (88, 21)      0.5773502691896257
  (90, 16)      0.6701090937974693
  (90, 19)      0.7422626236110333
```

```
(91, 18)      0.4575411467459101
(91, 7)       0.47676334219129807
(91, 8)       0.3803485410066615
(91, 10)      0.4575411467459101
(91, 20)      0.4575411467459101
(93, 15)      1.0
(94, 19)      1.0
(95, 15)      0.755791228244953
(95, 16)      0.6548126597035103
(96, 0)       1.0
(97, 13)      0.6938824991939403
(97, 19)      0.720088242726106
(99, 0)       1.0
```

Classifier Evaluation

In [30]:
```
#train test ข้อมูล
train_x, test_x, train_y, test_y = train_test_split(X, dftags, random_state=0, \
                                   train_size = 0.5)
```

In [31]:
```
from sklearn.metrics import classification_report
import warnings; warnings.filterwarnings('ignore')
```

In [32]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.multiclass import OneVsRestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [33]:
```
#กำหนดโมเดลที่จะทำการทดสอบ
models = [
    ('Decision Tree', OneVsRestClassifier(DecisionTreeClassifier(criterion='gini', max_depth=12))),
    ('K Nearest Neighbor', OneVsRestClassifier(KNeighborsClassifier(n_neighbors=3))),
    ('Neural Network', OneVsRestClassifier(MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)))
]
```

In [34]:
```
train_y.head()
```

Out[34]:

	attributed-no-source	books	classic	death	friends	friendship	humor	inspirational	learning	life	love	reading	simile	truth	writing
91	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
59	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In [35]:
```
train_y.columns
```

Out[35]:
```
Index(['attributed-no-source', 'books', 'classic', 'death', 'friends',
       'friendship', 'humor', 'inspirational', 'learning', 'life', 'love',
       'reading', 'simile', 'truth', 'writing'],
      dtype='object')
```

In [36]:
```
#ฟังก์ชั่นหาค่าเฉลี่ยของ scores ทุก tags
def dict_mean(dict_list):
    mean_dict = {}
    for key in dict_list[0].keys():
        mean_dict[key] = sum(d[key] for d in dict_list) / len(dict_list)
    return mean_dict
```

In [38]:
```
#ลูปทดสอบข้อมูล 3 โมเดล
names = []
reports = []
for name, clf in models:
    scores = []
    #predict แต่ละ tags
    for col in train_y.columns:
        clf.fit(train_x.toarray(), train_y[col])
        answers = clf.predict(test_x.toarray())
        #classification report แต่ละโมเดล
        report = classification_report(test_y[col], answers, output_dict=True)
        #เลือก macro avg
        macro_avgs = report['macro avg']
        macro_avgs['accuracy'] = report['accuracy']

        del macro_avgs['support']

        scores.append(macro_avgs)
    x = dict_mean(scores)
    names.append(name)
    reports.append(x)

#         print("Algo: `%s` for `%s` class:" %(name, col))
#         print(macro_avgs)
#แสดงผล precision ,recall, f1-score , accuracy
    print('Algo: %s' %(name))
    print(x)
```

```
Algo: Decision Tree
{'precision': 0.6378527328159829, 'recall': 0.6513803742841009, 'f1-score': 0.6402401668151229, 'accuracy': 0.9199999999999999}
Algo: K Nearest Neighbor
{'precision': 0.5242811214525898, 'recall': 0.5256163498337412, 'f1-score': 0.5215365044346388, 'accuracy': 0.9346666666666666}
Algo: Neural Network
{'precision': 0.5430977026744503, 'recall': 0.5338357599956979, 'f1-score': 0.5347383099557065, 'accuracy': 0.9026666666666667}
```

In [41]:
```
evaluate = pd.DataFrame(reports)
evaluate.insert(loc=0, column='Model', value=names)
evaluate
```

Out[41]:

	Model	precision	recall	f1-score	accuracy
0	Decision Tree	0.637853	0.651380	0.640240	0.920000
1	K Nearest Neighbor	0.524281	0.525616	0.521537	0.934667
2	Neural Network	0.543098	0.533836	0.534738	0.902667

โมเดลที่ดีที่สุดคือ Decision Tree โดยดูที่ค่า f1-score คือค่าเฉลี่ยของ precision ความแม่นยำ และ recall ความถูกต้องของโมเดล โดยไม่ค่า accuracy เนื่องจาก ค่า accuracy เป็นค่าที่จะนับจำนวนถูกทั้งหมดโดยไม่ได้คำนวณว่าที่ทำนายนั้นตรงหรือไม่ ซึ่งไม่เหมาะกับ data ที่ unbalanced โดยค่า f1-score ของ Decision Tree มีค่า 0.651380 ซึ่งมากสุดใน 3 โมเดลจึงดีที่สุด