Social Media and Text Analytics - Industry Assignment 1

Topic Modelling - To build an API Pipeline using Flask in Python and Deploy it on Heroku

Import Required Libraries

In [1]:

```
import re
import nltk
import pickle
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.multioutput import MultiOutputClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer, TfidfTransformer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from nltk import sent_tokenize, word_tokenize
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords

import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

In [2]:

```
train = pd.read_csv("train.csv")

test = pd.read_csv("test.csv")

train.head(5)
```

Out[2]:

	ID	TITLE	ABSTRACT	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology	Quantitative Finance
0	1	Reconstructing Subject-Specific Effect Maps	Predictive models allow subject-specific inf...	1	0	0	0	0	0
1	2	Rotation Invariance Neural Network	Rotation invariance and translation invarian...	1	0	0	0	0	0
2	3	Spherical polyharmonics and Poisson kernels fo...	We introduce and develop the notion of spher...	0	0	1	0	0	0
3	4	A finite element approximation for the stochas...	The stochastic Landau--Lifshitz--Gilbert (LL...	0	0	1	0	0	0
4	5	Comparative study of Discrete Wavelet Transfor...	Fourier-transform infra-red (FTIR) spectra o...	1	0	0	1	0	0

In [3]:

```
print("Training Data: ",train.shape)

print("Testing Data: ",test.shape)
```

```
Training Data:  (20972, 9)
Testing Data:  (8989, 3)
```

In [4]:

```
col = ['Computer Science','Physics','Mathematics','Statistics','Quantitative Biology','Quantitative Finance']

test = test.drop(['ID'],axis=1)

X = train.loc[:,['TITLE','ABSTRACT']]

y = train.loc[:, col]
```

Training and Testing the Data

In [5]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42, shuffle=True)

print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)

y_test.reset_index(drop=True,inplace=True)
X_test.reset_index(drop=True,inplace=True)

y1 = np.array(y_train)
```

```
y2 = np.array(y_test)
```

```
(18874, 2) (2098, 2)
(18874, 6) (2098, 6)
```

Removing the Punctuations

In [6]:

```
X_train.replace('[^a-zA-Z]',' ', regex=True, inplace=True)
X_test.replace('[^a-zA-Z]',' ', regex=True, inplace=True)

test.replace('[^a-zA-Z]',' ', regex=True, inplace=True)
```

Converting into Lower Case Characters

In [7]:

```
for index in X_train.columns:
  X_train[index] = X_train[index].str.lower()

for index in X_test.columns:
  X_test[index] = X_test[index].str.lower()

for index in test.columns:
  test[index] = test[index].str.lower()
```

Removing One Letter Words

In [8]:

```
X_train['ABSTRACT'] = X_train['ABSTRACT'].str.replace(r'\b\w\b', '').str.replace(r'\s+', ' ')
X_test['ABSTRACT'] = X_test['ABSTRACT'].str.replace(r'\b\w\b', '').str.replace(r'\s+', ' ')

test['ABSTRACT'] = test['ABSTRACT'].str.replace(r'\b\w\b', '').str.replace(r'\s+', ' ')
```

Removing Multiple Blank Spaces

In [9]:

```
X_train = X_train.replace(r's+', ' ', regex=True)
X_test = X_test.replace(r's+', ' ', regex=True)

test = test.replace(r's+', ' ', regex=True)
```

Lowercase the Text, Tokenization, Lemmatization and Stop Words

In [10]:

```
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('averaged_preceptron_tagger')

def preprocess_text(text):
    # Lowercase the text
    text = text.lower()

    # Remove non-alphabetic characters
    text = re.sub('[^a-zA-Z]', ' ', text)

    # Tokenize the text
    tokens = nltk.word_tokenize(text)

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]

    # Lemmatize the tokens
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Join tokens back to form a preprocessed text
    processed_text = ' '.join(tokens)

    return processed_text
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\khush\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\khush\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\khush\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Error loading averaged_preceptron_tagger: Package
[nltk_data]     'averaged_preceptron_tagger' not found in index
```

In [11]:

```
def convert_to_lines(data):
    lines = []
    for row in range(data.shape[0]):
        lines.append(' '.join(str(x) for x in data.iloc[row, :]))
    return lines
```

```
stop_words = set(stopwords.words('english'))
X_train['combined'] = X_train['TITLE']+' '+X_train['ABSTRACT']
X_test['combined'] = X_test['TITLE']+' '+X_test['ABSTRACT']

test['combined'] = test['TITLE']+' '+test['ABSTRACT']

X_train = X_train.drop(['TITLE','ABSTRACT'],axis=1)
X_test = X_test.drop(['TITLE','ABSTRACT'],axis=1)

test = test.drop(['TITLE','ABSTRACT'],axis=1)

X_train.head()
```

Out[11]:

	combined
13275	clu tering in hilbert pace of a quantum optim...
19273	graph heat mixture model learning graph infer...
6427	fa t and un upervi ed method for multilingual...
19168	nata ha fa ter non convex tocha tic optimiza...
14148	ku taanheimo tiefel tran formation with an ar...

In [12]:

```
X_lines = []
for row in range(0,X.shape[0]):
  X_lines.append(' '.join(str(x) for x in X.iloc[row,:]))

train_lines = []
for row in range(0,X_train.shape[0]):
  train_lines.append(' '.join(str(x) for x in X_train.iloc[row,:]))

test_lines = []
for row in range(0,X_test.shape[0]):
  test_lines.append(' '.join(str(x) for x in X_test.iloc[row,:]))

predtest_lines = []
for row in range(0,test.shape[0]):
  predtest_lines.append(' '.join(str(x) for x in test.iloc[row,:]))
```

In [13]:

```
countvector = CountVectorizer(ngram_range=(1,2))
X_train_cv = countvector.fit_transform(train_lines)
X_test_cv = countvector.transform(test_lines)

test_cv = countvector.transform(predtest_lines)
```

TF-IDF Vectorizer

In [14]:

```
tfidfvector = TfidfTransformer()
X_train_tf = tfidfvector.fit_transform(X_train_cv)
X_test_tf = tfidfvector.fit_transform(X_test_cv)

test_tf = tfidfvector.fit_transform(test_cv)

X_cv = countvector.transform(X_lines)

X_tf = tfidfvector.fit_transform(X_cv) #x_tf,y
```

In [15]:

```
model = LinearSVC(C=0.5, class_weight='balanced', random_state=42)
models = MultiOutputClassifier(model)

models.fit(X_train_tf, y1)
preds = models.predict(X_test_tf)
preds
```

Out[15]:

```
array([[1, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0],
       ...,
       [0, 1, 0, 0, 1, 0],
       [1, 0, 0, 1, 0, 0],
       [0, 1, 0, 0, 0, 0]], dtype=int64)
```

Confusion Matrix

In [16]:

```
print(classification_report(y2,preds))
print(accuracy_score(y2,preds))
predssv = models.predict(test_tf)
predssv
test = pd.read_csv('test.csv')

submit = pd.DataFrame({'ID': test.ID, 'Computer Science': predssv[:,0],'Physics':predssv[:,1],
                       'Mathematics':predssv[:,2],'Statistics':predssv[:,3],'Quantitative Biology':predssv[:,4]
                       'Quantitative Finance':predssv[:,5]})
submit.head()
submit.to_csv('Khushi_Submission.csv', index=False)
```

```
              precision    recall  f1-score   support

           0       0.80      0.90      0.85       853
           1       0.89      0.88      0.89       623
           2       0.83      0.83      0.83       580
           3       0.73      0.85      0.78       516
           4       0.49      0.40      0.44        58
           5       0.81      0.65      0.72        26

   micro avg       0.80      0.86      0.83      2656
   macro avg       0.76      0.75      0.75      2656
weighted avg       0.81      0.86      0.83      2656
 samples avg       0.84      0.88      0.84      2656

0.6601525262154433
```

Saving the Model

In [17]:

```python
# Save the trained MultiOutputClassifier model to a file
with open('multi_output_classifier_model.pkl', 'wb') as file:
    pickle.dump(models, file)
# Loading the MultiOutputClassifier model
with open('multi_output_classifier_model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)

# Save the CountVectorizer
with open('countvectorizer.pkl', 'wb') as file:
    pickle.dump(countvector, file)
# Loading the CountVectorizer
with open('countvectorizer.pkl', 'rb') as file:
    loaded_countvectorizer = pickle.load(file)

# Save the TfidfTransformer
with open('tfidftransformer.pkl', 'wb') as file:
    pickle.dump(tfidfvector, file)
# Loading the TfidfTransformer
with open('tfidftransformer.pkl', 'rb') as file:
    loaded_tfidftransformer = pickle.load(file)
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js