```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6

Enter your authorization code:
··········
Mounted at /content/drive
```

```
#Importing Libraries
import pandas as pd
import numpy as np
```

1. Load the dataset (5 points)

   Tip: As the dataset is large, use fewer rows. Check what is working well on your machine and decide ac

```
# Reading the data
corpus_df = pd.read_csv("/content/drive/My Drive/blogtext.csv")
corpus_df.head(10)
```

	id	gender	age	topic	sign	date	te
0	2059027	male	15	Student	Leo	14,May,2004	Info has been found (+/- 100 pages
1	2059027	male	15	Student	Leo	13,May,2004	These are the team members: Drewe
2	2059027	male	15	Student	Leo	12,May,2004	In het kader van kernfusie op aarde
3	2059027	male	15	Student	Leo	12,May,2004	testing!!! testing
4	3581210	male	33	InvestmentBanking	Aquarius	11,June,2004	Thanks to Yahoo!'s Toolbar I can
5	3581210	male	33	InvestmentBanking	Aquarius	10,June,2004	I had an interesting conversation
6	3581210	male	33	InvestmentBanking	Aquarius	10,June,2004	Somehow Coca-Cola has a way of su
7	3581210	male	33	InvestmentBanking	Aquarius	10,June,2004	If anything, Korea is a country c
8	3581210	male	33	InvestmentBanking	Aquarius	10,June,2004	Take a read of this news article
9	3581210	male	33	InvestmentBanking	Aquarius	09,June,2004	I surf the English news sites a

```
corpus_df.shape
```

```
(681284, 7)
```

```
# Taking only initial 3k rows to initial pre processing & training
corpus_df_sample = corpus_df[:3000]
print(corpus_df_sample.shape)
corpus_df_sample["text"].loc[0]
```

```
(3000, 7)
'           Info has been found (+/- 100 pages, and 4.5 MB of .pdf files) Now i have to wait un
```

2. Preprocess rows of the “text” column (7.5 points)

a. Remove unwanted characters

b. Convert text to lowercase

c. Remove unwanted spaces

d. Remove stopwords

```
#Removing unwanted / special characters
corpus_df_sample['text'] = corpus_df_sample['text'].str.replace('[^A-Za-z]',' ')
corpus_df_sample["text"].loc[0]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """Entry point for launching an IPython kernel.
'           Info has been found          pages  and     MB of  pdf files  Now i have to wait un
```

```
# Coverting to lower case
corpus_df_sample['text'] = corpus_df_sample['text'].str.lower()
corpus_df_sample["text"].loc[0]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """Entry point for launching an IPython kernel.
'           info has been found          pages  and     mb of  pdf files  now i have to wait un
```

```
#Removing spaces
corpus_df_sample["text"] = corpus_df_sample["text"].str.strip()
corpus_df_sample["text"].loc[0]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """Entry point for launching an IPython kernel.
'info has been found          pages  and     mb of  pdf files  now i have to wait untill our te
```

```
# splitting each row of text data into individual words.
# So it can be iterated through to remove only stopwords in next steps.
corpus_df_sample["text"] = corpus_df_sample["text"].str.split()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """Entry point for launching an IPython kernel.
```

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
import re
import nltk
from nltk.corpus import stopwords
```

```
#Removing stop words
stop = stopwords.words('english')
def removestopwords(y):   # Function definition
 stopwordremoved = [w for w in y if w not in stop]
 return(" ".join(stopwordremoved))
```

```
text_column_size = corpus_df_sample["text"].size
print("text column size :", text_column_size)

# Initialize an empty list to hold the text after stop word removal
cleaner_corpus_df_sample_text = []

# Loop over each text
for i in range( 0, text_column_size):
    cleaner_corpus_df_sample_text.append(removestopwords(corpus_df_sample["text"][i]))
```

```
text column size : 3000
```

```
cleaner_corpus_df_sample_text[10]
```

```
'ah korean language looks difficult first figure read hanguel korea surprisingly easy learn alp
```

```
#Replace text column with cleaner_corpus_df_sample_text
corpus_df_sample["text"] = cleaner_corpus_df_sample_text
corpus_df_sample["text"][10]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """Entry point for launching an IPython kernel.
'ah korean language looks difficult first figure read hanguel korea surprisingly easy learn alp
```

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
True
```

```
from nltk.stem import WordNetLemmatizer
```

```
#Lemmatization
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_text(text):
    lemm = [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]
    return(" ".join(lemm))

corpus_df_sample["text"] = corpus_df_sample.text.apply(lemmatize_text)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
```

3. As we want to make this into a multi-label classification problem, you are required to merge all the label all the labels together for a particular sentence (7.5 points)

   a. Label columns to merge: “gender”, “age”, “topic”, “sign”

```
corpus_df_sample.head(2)
```

	id	gender	age	topic	sign	date	text
0	2059027	male	15	Student	Leo	14,May,2004	info found page mb pdf file wait untill team l...
1	2059027	male	15	Student	Leo	13,May,2004	team member drewes van der laag urllink mail r...

```
# mergeing 'gender', 'age', 'topic', 'sign'
corpus_df_sample['age'] = corpus_df_sample['age'].astype(str)
corpus_df_sample['labels'] = corpus_df_sample[['gender','age','topic','sign']].apply(lambda x: ','.j
corpus_df_sample_merged = corpus_df_sample.drop(labels = ['date','gender', 'age','topic','sign','id'
corpus_df_sample_merged.head()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
```

	text	labels
0	info found page mb pdf file wait untill team l...	male,15,Student,Leo
1	team member drewes van der laag urllink mail r...	male,15,Student,Leo
2	het kader van kernfusie op aarde maak je eigen...	male,15,Student,Leo
3	testing testing	male,15,Student,Leo
4	thanks yahoo toolbar capture url popups mean s...	male,33,InvestmentBanking,Aquarius

```
corpus_df_sample_merged.shape
```

```
(3000, 2)
```

4. Separate features and labels, and split the data into training and testing (5 points)

```
from sklearn.model_selection import train_test_split
```

```
feature = corpus_df_sample_merged['text']
corpus_df_sample_merged['labels'] = corpus_df_sample_merged['labels'].str.lower()
labels = corpus_df_sample_merged['labels']
X_train, X_test, Y_train, Y_test = train_test_split(feature,labels, test_size = 0.33, random_state =
Y_train.shape
```

```
(2010,)
```

5. Vectorize the features (5 points)

a. Create a Bag of Words using count vectorizer

i. Use ngram_range=(1, 2)

ii. Vectorize training and testing features

b. Print the term-document matrix

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
# Creating Bag of words
vectorizer = CountVectorizer(min_df = 2,ngram_range = (1,2),stop_words = "english")
X_train = vectorizer.fit_transform(X_train)
```

```
X_test = vectorizer.transform(X_test)
print("X_train shape & sample",X_train.shape)
X_train[0]
```

```
X_train shape & sample (2010, 16018)
<1x16018 sparse matrix of type '<class 'numpy.int64'>'
        with 27 stored elements in Compressed Sparse Row format>
```

6. Create a dictionary to get the count of every label i.e. the key will be label name and value will be the total count of the

```
vectorizer_labels = CountVectorizer(min_df = 1,ngram_range = (1,1),stop_words = "english")
labels_vector = vectorizer_labels.fit_transform(labels)
vectorizer_labels.vocabulary_
```

```
{'14': 0,
 '15': 1,
 '16': 2,
 '17': 3,
 '23': 4,
 '24': 5,
 '25': 6,
 '26': 7,
 '27': 8,
 '33': 9,
 '34': 10,
 '35': 11,
 '37': 12,
 '39': 13,
 '41': 14,
 '44': 15,
 '45': 16,
 'accounting': 17,
 'aquarius': 18,
 'aries': 19,
 'arts': 20,
 'banking': 21,
 'businessservices': 22,
 'cancer': 23,
 'capricorn': 24,
 'communications': 25,
 'education': 26,
 'engineering': 27,
 'female': 28,
 'gemini': 29,
 'indunk': 30,
 'internet': 31,
 'investmentbanking': 32,
 'leo': 33,
 'libra': 34,
 'libraries': 35,
 'male': 36,
 'media': 37,
 'museums': 38,
 'non': 39,
 'pisces': 40,
 'profit': 41,
 'recreation': 42,
 'sagittarius': 43,
 'science': 44,
 'scorpio': 45,
 'sports': 46,
 'student': 47,
 'taurus': 48,
 'technology': 49,
 'virgo': 50}
```

```
# Extracing only key value from above dictionary, which contains unique labels. These set of labels
label_classes = []
for key in vectorizer_labels.vocabulary_.keys():
    label_classes.append(key)

print(sorted(label_classes))
```

7. Transform the labels - (7.5 points) As we have noticed before, in this task each example can have multiple tags. To de
to transform labels in a binary form and the prediction will be a mask of 0s and 1s. For this purpose, it is convenient to

a. Convert your train and test labels using MultiLabelBinarizer

```
from sklearn.preprocessing import MultiLabelBinarizer
```

```
 # initialising multilabelbinariser with all unique possible classes
 mlb = MultiLabelBinarizer(classes = label_classes)
```

```
# Converting entire se of labels into format required by mlb
labels = [["".join(re.findall("\w",f)) for f in lst] for lst in [s.split(",") for s in labels]]
labels[30]
```

```
['male', '33', 'investmentbanking', 'aquarius']
```

```
labels_trans = mlb.fit(labels) # transforming entire set of lables
labels_trans
```

```
MultiLabelBinarizer(classes=['male', '15', 'student', 'leo', '33',
                             'investmentbanking', 'aquarius', 'female', '14',
                             'indunk', 'aries', '25', 'capricorn', '17',
                             'gemini', '23', 'non', 'profit', 'cancer',
                             'banking', '37', 'sagittarius', '26', '24',
                             'scorpio', '27', 'education', '45', 'engineering',
                             'libra', ...],
                    sparse_output=False)
```

```
#Convert Y_train into a format as required by mlb
Y_train = [["".join(re.findall("\w",f)) for f in lst] for lst in [s.split(",") for s in Y_train]]
Y_train[30]
```

```
['male', '24', 'engineering', 'libra']
```

```
Y_train_trans = mlb.transform(Y_train) # transforming Train lables using mlb which is trained on all
Y_train_trans[30]
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/_label.py:987: UserWarning: unknow
  .format(sorted(unknown, key=str)))
array([1, 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, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0])
```

```
#Convert Y_test into a format as required by mlb
Y_test = [["".join(re.findall("\w",f)) for f in lst] for lst in [s.split(",") for s in Y_test]]
Y_test_trans = mlb.transform(Y_test) # transforming test labels.
print(Y_test[30])
```

```
['male', '35', 'technology', 'aries']
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/_label.py:987: UserWarning: unknow
  .format(sorted(unknown, key=str)))
```

8. Choose a classifier - (5 points) In this task, we suggest using the One-vs-Rest approach, which is implemented in OneV
k classifiers (= number of tags) are trained. As a basic classifier, use LogisticRegression . It is one of the simplest met
enough in text classification tasks. It might take some time because the number of classifiers to train is large.

a. Use a linear classifier of your choice, wrap it up in OneVsRestClassifier to train it on every label

(b. As One-vs-Rest approach might not have been discussed in the sessions)

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression
```

```
clf = LogisticRegression(solver = 'lbfgs',max_iter = 1000)  # initiating the classifier
#from sklearn.svm import SVC
#clf = SVC(kernel = "linear")
clf = OneVsRestClassifier(clf)
```

9. Fit the classifier, make predictions and get the accuracy (5 points)

a. Print the following

i. Accuracy score

ii. F1 score

iii. Average precision score

iv. Average recall score

v. Tip: Make sure you are familiar with all of them. How would you expect the things to work for the multi-labe
micro/macro/weighted averaging

```
from sklearn.metrics import confusion_matrix, classification_report,f1_score, accuracy_score, recall
```

```
clf.fit(X_train,Y_train_trans) # Fitting on  train data
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 16 is p
  str(classes[c]))
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 17 is p
  str(classes[c]))
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 33 is p
  str(classes[c]))
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 34 is p
  str(classes[c]))
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 36 is p
  str(classes[c]))
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 37 is p
  str(classes[c]))
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 45 is p
  str(classes[c]))
/usr/local/lib/python3.6/dist-packages/sklearn/multiclass.py:75: UserWarning: Label not 46 is p
  str(classes[c]))
OneVsRestClassifier(estimator=LogisticRegression(C=1.0, class_weight=None,
                                                 dual=False, fit_intercept=True,
                                                 intercept_scaling=1,
                                                 l1_ratio=None, max_iter=1000,
                                                 multi_class='auto',
                                                 n_jobs=None, penalty='l2',
                                                 random_state=None,
                                                 solver='lbfgs', tol=0.0001,
                                                 verbose=0, warm_start=False),
                    n_jobs=None)
```

```
print("Train Accuracy:",clf.score(X_train,Y_train_trans))
```

```
Train Accuracy: 0.972139303482587
```

```
Y_pred = clf.predict(X_test)
```

```
print("Test Accuracy:" + str(accuracy_score(Y_test_trans, Y_pred)))
print("F1: " + str(f1_score(Y_test_trans, Y_pred, average='micro')))
print("F1_macro: " + str(f1_score(Y_test_trans, Y_pred, average='macro')))
print("Precision: " + str(precision_score(Y_test_trans, Y_pred, average='micro')))
print("Precision_macro: " + str(precision_score(Y_test_trans, Y_pred, average='macro')))
print("Recall: " + str(recall_score(Y_test_trans, Y_pred, average='micro')))
print("Recall_macro: " + str(recall_score(Y_test_trans, Y_pred, average='macro')))
```

```
Test Accuracy:0.5686868686868687
F1: 0.7584394023242943
F1_macro: 0.2777544085541704
Precision: 0.8270971635485818
Precision_macro: 0.42504160995159523
Recall: 0.7003065917220235
Recall_macro: 0.2290073113591835
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1515: UndefinedMetric
  average, "true nor predicted", 'F-score is', len(true_sum)
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetric
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetric
  _warn_prf(average, modifier, msg_start, len(result))
```

10. Print true label and predicted label for any five examples (7.5 points)

```
Y_pred_inv = mlb.inverse_transform(Y_pred)   # inverse transforming predited label data
Y_test_trans_inv =  mlb.inverse_transform(Y_test_trans) # inverse transforming original test label d
```

```
print(" predicted :",Y_pred_inv[25])
print(" Actual :",Y_test_trans_inv[25])
```