Getting dataset from Kaggle

Install Kaggle library

In [1]:

```
! pip install -q kaggle
```

Uplaoding kaggle credential API key

In [2]:

```
from google.colab import files

files.upload()
```

Choose files No file chosen

Upload widget is only available when the cell has been executed in the current browser session.
Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[2]:

```
{'kaggle.json': b'{"username":"adityap10","key":"e525fd5175445ca2d0801a24b
968c10e"}'}
```

Make a directory named ".kaggle"

In [3]:

```
! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/
```

Allocate the required permission for this file.

In [4]:

```
! chmod 600 ~/.kaggle/kaggle.json
```

Downloading Competitions dataset

In [5]:

```
!kaggle datasets download -d stackoverflow/stacksample
```

```
Downloading stacksample.zip to /content
 98% 1.09G/1.11G [00:06<00:00, 227MB/s]
100% 1.11G/1.11G [00:06<00:00, 191MB/s]
```

In [6]:

```
!mkdir stacksample
!unzip stacksample.zip -d stacksample
```

```
Archive:  stacksample.zip
  inflating: stacksample/Answers.csv  Y

  inflating: stacksample/Questions.csv  Y

  inflating: stacksample/Tags.csv
```

Installing necessary libraries

In [7]:

```
!pip install -q scikit-multilearn
```

```
     |████████████████████████████████| 89 kB 5.2 MB/s
```

Importing necessary libraries

In [8]:

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

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

Out[8]:

```
True
```

In [9]:

```
import pickle
import pandas as pd
import warnings

warnings.filterwarnings("ignore")

from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from scipy.sparse import hstack

from bs4 import BeautifulSoup
import lxml
import re

from sklearn.multioutput import ClassifierChain
import numpy as np

from nltk.corpus import stopwords
from nltk.tokenize import ToktokTokenizer
from nltk.stem.wordnet import WordNetLemmatizer

from sklearn.metrics import accuracy_score
from sklearn.svm import LinearSVC
from sklearn.metrics import hamming_loss
from sklearn.metrics import f1_score
from skmultilearn.problem_transform import LabelPowerset
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn import model_selection
from sklearn.metrics import make_scorer
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score

from skmultilearn.problem_transform import BinaryRelevance
from sklearn.naive_bayes import GaussianNB
```

Read CSV files to get questions and tags

In [10]:

```
df_questions = pd.read_csv("/content/stacksample/Questions.csv", encoding="ISO-8859-1")
df_tags = pd.read_csv("/content/stacksample/Tags.csv", encoding="ISO-8859-1", dtype={'T
ag': str})
df_questions.head()
```

Out[10]:

	Id	OwnerUserId	CreationDate	ClosedDate	Score	Title	Body
0	80	26.0	2008-08-01T13:57:07Z	NaN	26	SQLStatement.execute() - multiple queries in o...	<p>I've written a database generation script i...
1	90	58.0	2008-08-01T14:41:24Z	2012-12-26T03:45:49Z	144	Good branching and merging tutorials for Torto...	<p>Are there any really good tutorials explain...
2	120	83.0	2008-08-01T15:50:08Z	NaN	21	ASP.NET Site Maps	<p>Has anyone got experience creating <strong>...
3	180	2089740.0	2008-08-01T18:42:19Z	NaN	53	Function for creating color wheels	<p>This is something I've pseudo-solved many t...
4	260	91.0	2008-08-01T23:22:08Z	NaN	49	Adding scripting functionality to .NET applica...	<p>I have a little game written in C#. It uses...

In [11]:

```
df_tags.head()
```

Out[11]:

	Id	Tag
0	80	flex
1	80	actionscript-3
2	80	air
3	90	svn
4	90	tortoisesvn

Process tags

Process them tags into something nice to query

Group tags by id and join them

In [12]:

```
df_tags['Tag'] = df_tags['Tag'].astype(str)
grouped_tags = df_tags.groupby("Id")['Tag'].apply(lambda tags: ' '.join(tags))
grouped_tags.head(5)
```

Out[12]:

```
Id
80                              flex actionscript-3 air
90         svn tortoisesvn branch branching-and-merging
120                                 sql asp.net sitemap
180      algorithm language-agnostic colors color-space
260            c# .net scripting compiler-construction
Name: Tag, dtype: object
```

Reset index for making simpler dataframe

In [13]:

```
grouped_tags.reset_index()
grouped_tags_final = pd.DataFrame({'Id':grouped_tags.index, 'Tags':grouped_tags.values
})
grouped_tags_final.head(5)
```

Out[13]:

	Id	Tags
0	80	flex actionscript-3 air
1	90	svn tortoisesvn branch branching-and-merging
2	120	sql asp.net sitemap
3	180	algorithm language-agnostic colors color-space
4	260	c# .net scripting compiler-construction

Process Questions

Drop unnecessary columns

In [14]:

```
df_questions.drop(columns=['OwnerUserId', 'CreationDate', 'ClosedDate'], inplace=True)
```

Merge questions and tags into one dataframe

In [15]:

```
df = df_questions.merge(grouped_tags_final, on='Id')
df.head(5)
```

Out[15]:

	Id	Score	Title	Body	Tags
0	80	26	SQLStatement.execute() - multiple queries in o...	<p>I've written a database generation script i...	flex actionscript-3 air
1	90	144	Good branching and merging tutorials for Torto...	<p>Are there any really good tutorials explain...	svn tortoisesvn branch branching-and-merging
2	120	21	ASP.NET Site Maps	<p>Has anyone got experience creating <strong>...	sql asp.net sitemap
3	180	53	Function for creating color wheels	<p>This is something I've pseudo-solved many t...	algorithm language-agnostic colors color-space
4	260	49	Adding scripting functionality to .NET applica...	<p>I have a little game written in C#. It uses...	c# .net scripting compiler-construction

Filter out questions with a score lower than 5

In [16]:

```
new_df = df[df['Score']>5]
```

Split tags in order to get a list of tags

In [17]:

```
new_df['Tags'] = new_df['Tags'].apply(lambda x: x.split())
all_tags = [item for sublist in new_df['Tags'].values for item in sublist]
flat_list = [item for sublist in new_df['Tags'].values for item in sublist]

keywords = nltk.FreqDist(flat_list)
keywords = nltk.FreqDist(keywords)
```

Get most frequent tags

In [18]:

```
frequencies_words = keywords.most_common(25)
tags_features = [word[0] for word in frequencies_words]
print(tags_features)
```

```
['c#', 'java', 'javascript', 'android', 'python', 'c++', 'php', 'jquery',
'.net', 'ios', 'html', 'css', 'c', 'iphone', 'objective-c', 'ruby-on-rail
s', 'sql', 'asp.net', 'mysql', 'ruby', 'r', 'git', 'asp.net-mvc', 'linux',
'sql-server']
```

Drop unnecessary columns at this point

In [19]:

```
new_df.drop(columns=['Id', 'Score'], inplace=True)
```

Change Tags column into None for questions that don't have a most common tag

In [20]:

```
def most_common(tags):
    """Function to check if tag is in most common tag list"""
    tags_filtered = []
    for i in range(0, len(tags)):
        if tags[i] in tags_features:
            tags_filtered.append(tags[i])
    return tags_filtered

new_df['Tags'] = new_df['Tags'].apply(lambda x: most_common(x))
new_df['Tags'] = new_df['Tags'].apply(lambda x: x if len(x)>0 else None)
```

In [21]:

```
# fig, ax = plt.subplots(figsize=(15, 10))
# keywords.plot(100, cumulative=False)

# TODO
```

Drop rows that contain None in Tags column

In [22]:

```
new_df.dropna(subset=['Tags'], inplace=True)
new_df.shape
```

Out[22]:

```
(52418, 3)
```

Preprocess Data

- Remove special characters from title and body
- Remove stop words
- Remove HTML tags
- Convert characters to lowercase
- Lemmatize the words

Converting to String

In [23]:

```
new_df['Title'] = new_df['Title'].apply(lambda x: str(x))
```

Filter out HTML

In [24]:

```
new_df['Body'] = new_df['Body'].apply(lambda x: BeautifulSoup(x, "lxml").get_text())
```

Remove stopwords

In [25]:

```
def removeStopWords(text):
    words = ToktokTokenizer().tokenize(text)
    stop_words = set(stopwords.words("english"))
    filtered = [w for w in words if not w in stop_words]
    return ' '.join(map(str, filtered))
```

In [26]:

```
new_df['Body'] = new_df['Body'].apply(lambda x: removeStopWords(x))
new_df['Title'] = new_df['Title'].apply(lambda x: removeStopWords(x))
```

Remove punctuation

In [27]:

```
def strip_list_noempty(mylist):
    newlist = (item.strip() if hasattr(item, 'strip') else item for item in mylist)
    return [item for item in newlist if item != '']

def removePunctuation(text):
    punct = '!"$%&\'()*,./:;<=>?@[\\]^_`{|}~'
    words=ToktokTokenizer().tokenize(text)
    punctuation_filtered = []
    regex = re.compile('[%s]' % re.escape(punct))
    remove_punctuation = str.maketrans(' ', ' ', punct)
    for w in words:
        if w in tags_features:
            punctuation_filtered.append(w)
        else:
            punctuation_filtered.append(regex.sub('', w))

    filtered_list = strip_list_noempty(punctuation_filtered)

    return ' '.join(map(str, filtered_list))
```

In [28]:

```
new_df['Body'] = new_df['Body'].apply(lambda x: removePunctuation(x))
new_df['Title'] = new_df['Title'].apply(lambda x: removePunctuation(x))
```

Lemmatization

In [29]:

```
def lemmatizeWords(text):
    words=ToktokTokenizer().tokenize(text)
    listLemma=[]
    for w in words:
        x=WordNetLemmatizer().lemmatize(w, pos="v")
        listLemma.append(x.lower())
    return ' '.join(map(str, listLemma))
```

In [30]:

```
new_df['Body'] = new_df['Body'].apply(lambda x: lemmatizeWords(x))
new_df['Title'] = new_df['Title'].apply(lambda x: lemmatizeWords(x))
```

In [31]:

```
new_df['Title'] = new_df['Title'].apply(lambda x: ' '.join(x.split()*3))
```

In [32]:

```
new_df['Title'].head()
```

Out[32]:

```
2    aspnet site maps aspnet site maps aspnet site ...
4    adding script functionality net applications a...
5    should i use nest class case should i use nest...
6    homegrown consumption web service homegrown co...
7    deploying sql server databases test live deplo...
Name: Title, dtype: object
```

In [33]:

```
new_df['Body'].head()
```

Out[33]:

```
2    has anyone get experience create sql-based asp...
4    i little game write c# it use database back-en...
5    i work collection class use video playback rec...
6    i write web service .net app i ready consume t...
7    i wonder guy manage deployment database 2 sql ...
Name: Body, dtype: object
```

Splitting Data

Define X, y

In [34]:

```
X1 = new_df['Body']
X2 = new_df['Title']
y = new_df['Tags']
print(len(X1), len(X2), len(y))
```

```
52418 52418 52418
```

Define multilabel binarizer

In [35]:

```
multilabel_binarizer = MultiLabelBinarizer()
y_bin = multilabel_binarizer.fit_transform(y)

vectorizer_X1 = TfidfVectorizer(analyzer = 'word',
                                       min_df=0.0005,
                                       max_df = 1.0,
                                       strip_accents = None,
                                       encoding = 'utf-8',
                                       ngram_range = (1, 3),
                                       preprocessor=None,
                                       token_pattern=r"(?u)\S\S+",
                                       max_features=35000)

vectorizer_X2 = TfidfVectorizer(analyzer = 'word',
                                       min_df=0.0,
                                       max_df = 1.0,
                                       strip_accents = None,
                                       encoding = 'utf-8',
                                       ngram_range = (1, 3),
                                       preprocessor=None,
                                       token_pattern=r"(?u)\S\S+",
                                       max_features=35000)

X1_tfidf = vectorizer_X1.fit_transform(X1)
X2_tfidf = vectorizer_X2.fit_transform(X2)
```

Stack X1 and X2 into X_tfidf

In [36]:

```
X_tfidf = hstack([X1_tfidf,X2_tfidf])
```

Split training and test data

In [37]:

```
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y_bin, test_size = 0.2, ra
ndom_state = 0)
```

Classifier implementation

Evaluation Metric

In [38]:

```
def print_score(y_pred, clf):
    print("Clf: ", clf.__class__.__name__)
    print("Accuracy score: {}".format(accuracy_score(y_test, y_pred)))
    print("Recall score: {}".format(recall_score(y_true=y_test, y_pred=y_pred, average=
'weighted')))
    print("Precision score: {}".format(precision_score(y_true=y_test, y_pred=y_pred, av
erage='weighted')))
    print("Hamming loss: {}".format(hamming_loss(y_pred, y_test)*100))
    print("F1 score: {}".format(f1_score(y_pred, y_test, average='weighted')))
    print("---")
```

Using Label Powerset

In [39]:

```
svc = LinearSVC()
sgd = SGDClassifier(n_jobs=-1)

clf = LabelPowerset(svc)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print_score(y_pred, clf)

kfold = KFold(n_splits=5)
X_sparse = X_tfidf.tocsr()

scores = []

for train_indices, test_indices in kfold.split(X_sparse, y_bin):
    clf.fit(X_sparse[train_indices], y_bin[train_indices])
    print(clf.score(X_sparse[test_indices], y_bin[test_indices]))
    scores.append(clf.score(X_sparse[test_indices], y_bin[test_indices]))

print(sum(scores)/len(scores))
```

```
Clf:  LabelPowerset
Accuracy score: 0.6237123235406333
Recall score: 0.6885059216519891
Precision score: 0.7651771625345002
Hamming loss: 2.5730637161388783
F1 score: 0.7374719315043108
---
0.5901373521556658
0.6115986264784433
0.6109309423884014
0.6320709720499857
0.6204330821329772
0.6130341950410946
```

Saving Model

In [40]:

```
with open('label_powerset_model.pkl','wb') as f:
    pickle.dump(clf,f)
```

Using Classifier Chains

In [41]:

```
chains = [ClassifierChain(svc, order='random', random_state=i)
          for i in range(10)]

for chain in chains:
    chain.fit(X_train, y_train)

Y_pred_chains = np.array([chain.predict(X_test) for chain in
                          chains])

Y_pred_ensemble = Y_pred_chains.mean(axis=0)
ensemble_accuracy_score = accuracy_score(y_test, Y_pred_ensemble >= .5)
ensemble_recall_score = recall_score(y_test, Y_pred_ensemble >= .5, average='weighted')
ensemble_precision_score = precision_score(y_test, Y_pred_ensemble >= .5, average='weig
hted')
ensemble_f1_score = f1_score(y_pred, Y_pred_ensemble >= .5, average='weighted')
hamm = hamming_loss(Y_pred_ensemble >= .5, y_test)*100
print(ensemble_accuracy_score, ensemble_recall_score, ensemble_precision_score, ensembl
e_f1_score, hamm)
```

```
0.5946203739030904 0.6818250835104768 0.8311283015789716 0.820683699907695
2.2499046165585654
```

Saving Model

In [42]:

```
with open("classifier_chain_model.pkl", "wb") as f:
    for chain in chains:
        pickle.dump(chain, f)
```

Using Binary Relevance

In [43]:

```
# initialize binary relevance multi-label classifier
# with a gaussian naive bayes base classifier
classifier = BinaryRelevance(svc)

# train
classifier.fit(X_train, y_train)

# predict
predictions = classifier.predict(X_test)

print_score(predictions, classifier)
```

```
Clf:  BinaryRelevance
Accuracy score: 0.5628576879053796
Recall score: 0.6471302763437595
Precision score: 0.8452195939969764
Hamming loss: 2.3170545593285006
F1 score: 0.747106233552806
---
```

Saving Model

In [44]:

```
with open('binary_relevance_model.pkl','wb') as f:
    pickle.dump(classifier,f)
```