```python
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import sqlite3
import csv
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from wordcloud import WordCloud
import re
import os
from sqlalchemy import create_engine # database connection
import datetime as dt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn import metrics
from sklearn.metrics import f1_score, precision_score, recall_score
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from skmultilearn.adapt import mlknn
from skmultilearn.problem_transform import ClassifierChain
from skmultilearn.problem_transform import BinaryRelevance
from skmultilearn.problem_transform import LabelPowerset
from sklearn.naive_bayes import GaussianNB
from datetime import datetime
import nltk
```

StackOverflow: Tag Prediction

Real World / Business Objectives and Constraints

1. Predict as many tags as possible with high precision and recall.
2. Incorrect tags would impact customer experience on StackOverflow. No strict latency constraints.

Exploratory Data Analysis

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to load the data

```python
# Creating db file from csv
if not os.path.isfile('train.db'):
    start = datetime.now()
```

```
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 180000
    j = 0
    index_start = 1
    for df in pd.read_csv('D:/3D Objects/Applied AI/Case 
study/Stackoverflow/Train.csv', names=['Id', 'Title', 'Body', 'Tags'], 
chunksize=chunksize, iterator=True, encoding='utf-8', ):
        df.index += index_start
        j+=1
        print('{} rows'.format(j*chunksize))
        df.to_sql('data', disk_engine, if_exists='append')
        index_start = df.index[-1] + 1
    print("Time taken to run this cell :", datetime.now() - start)
```

3.1.2 Counting the number of rows

```
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    num_rows = pd.read_sql_query("""SELECT count(*) FROM data""", con)
    #Always remember to close the database
    print("Number of rows in the database :","\
n",num_rows['count(*)'].values[0])
    con.close()
    print("Time taken to count the number of rows :", datetime.now() - 
start)
else:
    print("Please download the train.db file from drive or run the 
above cell to genarate train.db file")
```

```
Number of rows in the database :
 6034196
Time taken to count the number of rows : 0:00:02.498884
```

3.1.3 Checking for duplicates

```
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    df_no_dup = pd.read_sql_query('SELECT Title, Body, Tags, COUNT(*) 
as cnt_dup FROM data GROUP BY Title, Body, Tags', con)
    con.close()
    print("Time taken to run this cell :", datetime.now() - start)
else:
    print("Please download the train.db file from drive or run the 
first to genarate train.db file")
```

```
Time taken to run this cell : 0:01:47.396097
```

```
df_no_dup.head()
```

```
                                               Title  \
0       Implementing Boundary Value Analysis of S...
1           Dynamic Datagrid Binding in Silverlight?
2           Dynamic Datagrid Binding in Silverlight?
3      java.lang.NoClassDefFoundError: javax/serv...
4      java.sql.SQLException:[Microsoft][ODBC Dri...

                                                Body  \
0  <pre><code>#include&lt;iostream&gt;\n#include&...
1  <p>I should do binding for datagrid dynamicall...
2  <p>I should do binding for datagrid dynamicall...
3  <p>I followed the guide in <a href="http://sta...
4  <p>I use the following code</p>\n\n<pre><code>...

                                 Tags  cnt_dup
0                               c++ c        1
1          c# silverlight data-binding        1
2  c# silverlight data-binding columns        1
3                             jsp jstl        1
4                            java jdbc        2
```

```
print("Number of duplicate questions:", num_rows['count(*)'].values[0]
- df_no_dup.shape[0], "(",(1-
((df_no_dup.shape[0])/(num_rows['count(*)'].values[0])))*100, "%)")
```

```
Number of duplicate questions: 1827881 ( 30.292038906260256 %)
```

```
# Number of times each question appeared in our database
df_no_dup.cnt_dup.value_counts()
```

```
1    2656284
2    1272336
3     277575
4         90
5         25
6          5
Name: cnt_dup, dtype: int64
```

```
df_no_dup.dropna(how='any',axis=0,inplace=True)
```

```
# dropping rows with empty values in Tags
start = datetime.now()
df_no_dup["tag_count"] = df_no_dup["Tags"].apply(lambda text:
len(text.split(' ') if text is not None else '0' ))
# adding a new feature number of tags per question
print("Time taken to run this cell :", datetime.now() - start)
df_no_dup.head()
```

```
Time taken to run this cell : 0:00:01.959057
```

```
                                               Title  \
0       Implementing Boundary Value Analysis of S...
1           Dynamic Datagrid Binding in Silverlight?
2           Dynamic Datagrid Binding in Silverlight?
3      java.lang.NoClassDefFoundError: javax/serv...
4      java.sql.SQLException:[Microsoft][ODBC Dri...

                                                Body  \
0  <pre><code>#include&lt;iostream&gt;\n#include&...
1  <p>I should do binding for datagrid dynamicall...
2  <p>I should do binding for datagrid dynamicall...
3  <p>I followed the guide in <a href="http://sta...
4  <p>I use the following code</p>\n\n<pre><code>...

                                  Tags  cnt_dup  tag_count
0                                c++ c        1          2
1          c# silverlight data-binding        1          3
2  c# silverlight data-binding columns        1          4
3                             jsp jstl        1          2
4                            java jdbc        2          2
```

```python
# distribution of number of tags per question
df_no_dup.tag_count.value_counts()
```

```
3    1206157
2    1111706
4     814996
1     568291
5     505158
Name: tag_count, dtype: int64
```

```python
# creating a new database with no duplicates
if not os.path.isfile('train_no_dup.db'):
    disk_dup = create_engine("sqlite:///train_no_dup.db")
    no_dup = pd.DataFrame(df_no_dup, columns=['Title', 'Body',
'Tags'])
    no_dup.to_sql('no_dup_train', disk_dup)

conn=sqlite3.connect('train_no_dup.db')
no_dup=pd.read_sql_query('Select Title,Body,Tags from
no_dup_train',conn)
conn.close()
no_dup.head()
```

```
                                               Title  \
0       Implementing Boundary Value Analysis of S...
1           Dynamic Datagrid Binding in Silverlight?
2           Dynamic Datagrid Binding in Silverlight?
3      java.lang.NoClassDefFoundError: javax/serv...
4      java.sql.SQLException:[Microsoft][ODBC Dri...
```

```
                                                Body  \
0  <pre><code>#include&lt;iostream&gt;\n#include&...
1  <p>I should do binding for datagrid dynamicall...
2  <p>I should do binding for datagrid dynamicall...
3  <p>I followed the guide in <a href="http://sta...
4  <p>I use the following code</p>\n\n<pre><code>...

                                  Tags
0                                c++ c
1          c# silverlight data-binding
2  c# silverlight data-binding columns
3                             jsp jstl
4                            java jdbc
```

```
tag_data=pd.DataFrame(no_dup,columns=['Tags'])
tag_data.head()
```

```
                                  Tags
0                                c++ c
1          c# silverlight data-binding
2  c# silverlight data-binding columns
3                             jsp jstl
4                            java jdbc
```

```
# This methods seems to work more appropriately with this much data
# creating a connection with database file
# if os.path.isfile('train_no_dup.db'):
#     start = datetime.now()
#     con = sqlite3.connect('train_no_dup.db')
#     tag_data = pd.read_sql_query("""SELECT Tags FROM
no_dup_train""", con)
#     # Always close the connection
#     con.close()

#     #let's now drop unwanted column
#     tag_data.drop(tag_data.index[0], inplace=True)
#     #Printing first 5 columns from our dataframe
#     tag_data.head()
#     print("Time taken to run this cell :", datetime.now() - start)
# else:
#     print("Please download the train.db file from drive or run the
above cells to genarate train.db file")
```

3.2 Analysis of Tags

3.2.1 Total number of unique tags

```
# Importing and Initializing the "CountVectorizer" object, which
#is scikit-learn's bag of words tool.
```

```
#by default 'split()' will tokenize each tag using space.
vectorizer = CountVectorizer(tokenizer = lambda x: x.split())
# fit_transform() does two functions: First, it fits the model
# and learns the vocabulary; second, it transforms our training data
# into feature vectors. The input to fit_transform should be a list of
strings.
tag_dtm = vectorizer.fit_transform(tag_data['Tags'])

print('Number of data points:', tag_dtm.shape[0])
print('Number of unique tags:', tag_dtm.shape[1])
#print(tag_dtm)
```

```
Number of data points: 4206308
Number of unique tags: 42048
```

```
#get_feature_name() gives us the vocabulary
tags = vectorizer.get_feature_names()
# lets look at tags we have
print('Some of the tags we have', tags[:10])
```

```
Some of the tags we have ['.a', '.app', '.asp.net-mvc', '.aspxauth',
'.bash-profile', '.class-file', '.cs-file', '.doc', '.drv', '.ds-
store']
```

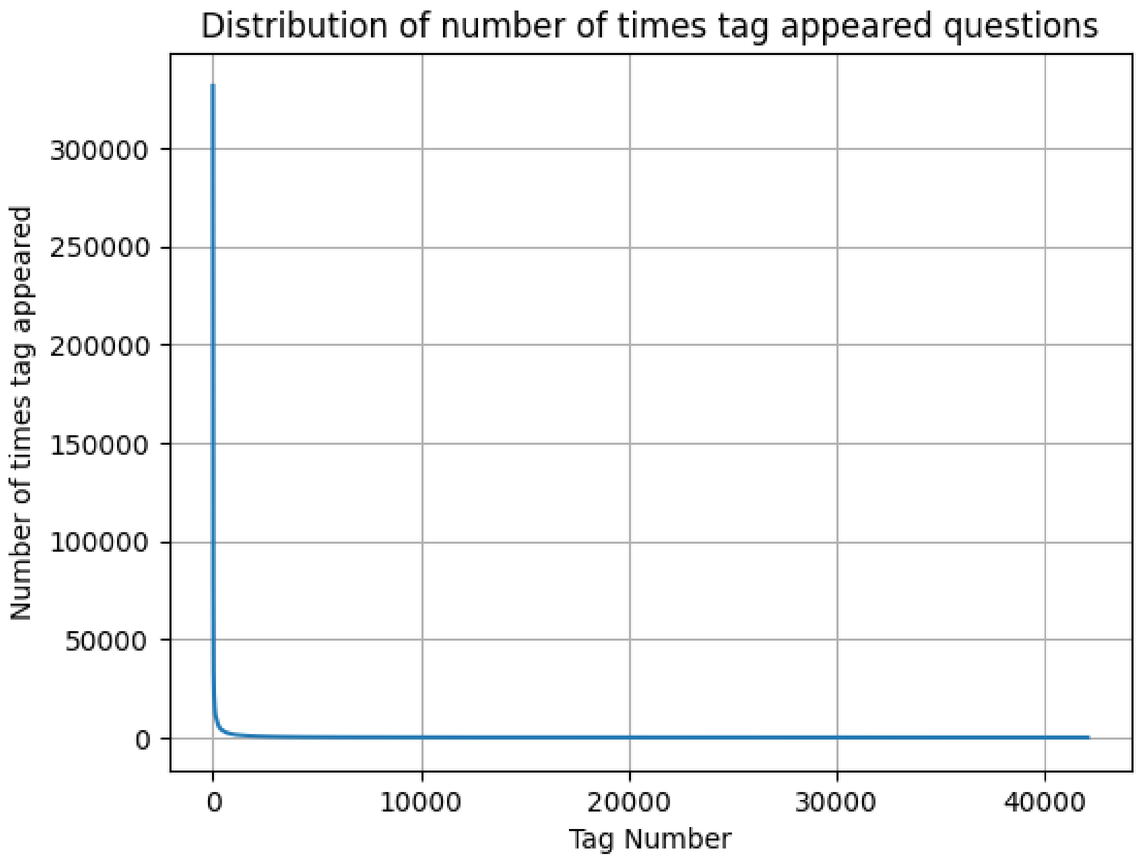
3.2.3 Number of times a tag appeared

```
# https://stackoverflow.com/questions/15115765/how-to-access-sparse-
matrix-elements
#Store the document term matrix in a dictionary.
freqs = tag_dtm.sum(axis=0).A1
result = dict(zip(tags, freqs))

# Saving this dictionary to csv file
lst=[]
for key, value in result.items():
    lst.append([key,value])

tag_df = pd.DataFrame(lst,columns=['Tags','Counts'])
tag_df.head()
```

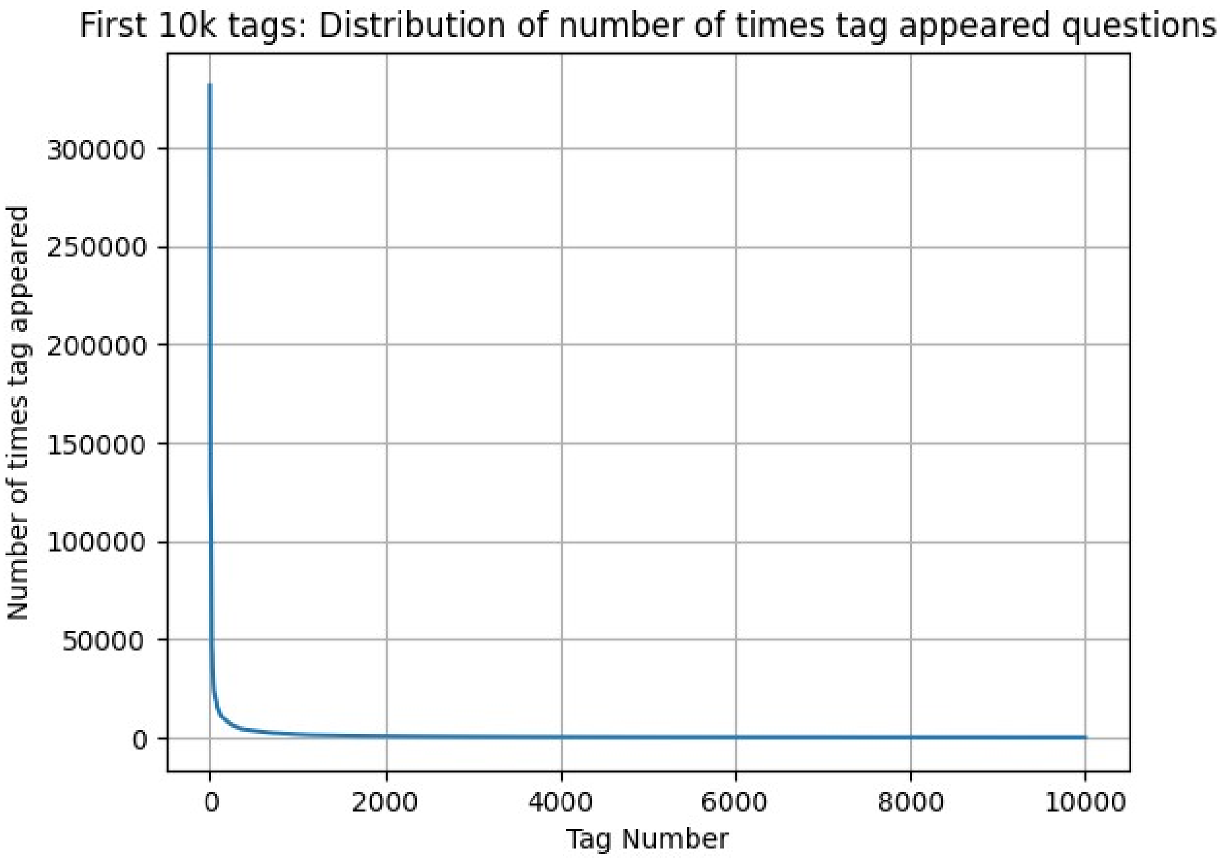
	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138

```
tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
tag_counts = tag_df_sorted['Counts'].values
```

```
plt.plot(tag_counts)
plt.title('Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel('Tag Number')
plt.ylabel('Number of times tag appeared')
plt.show()
```



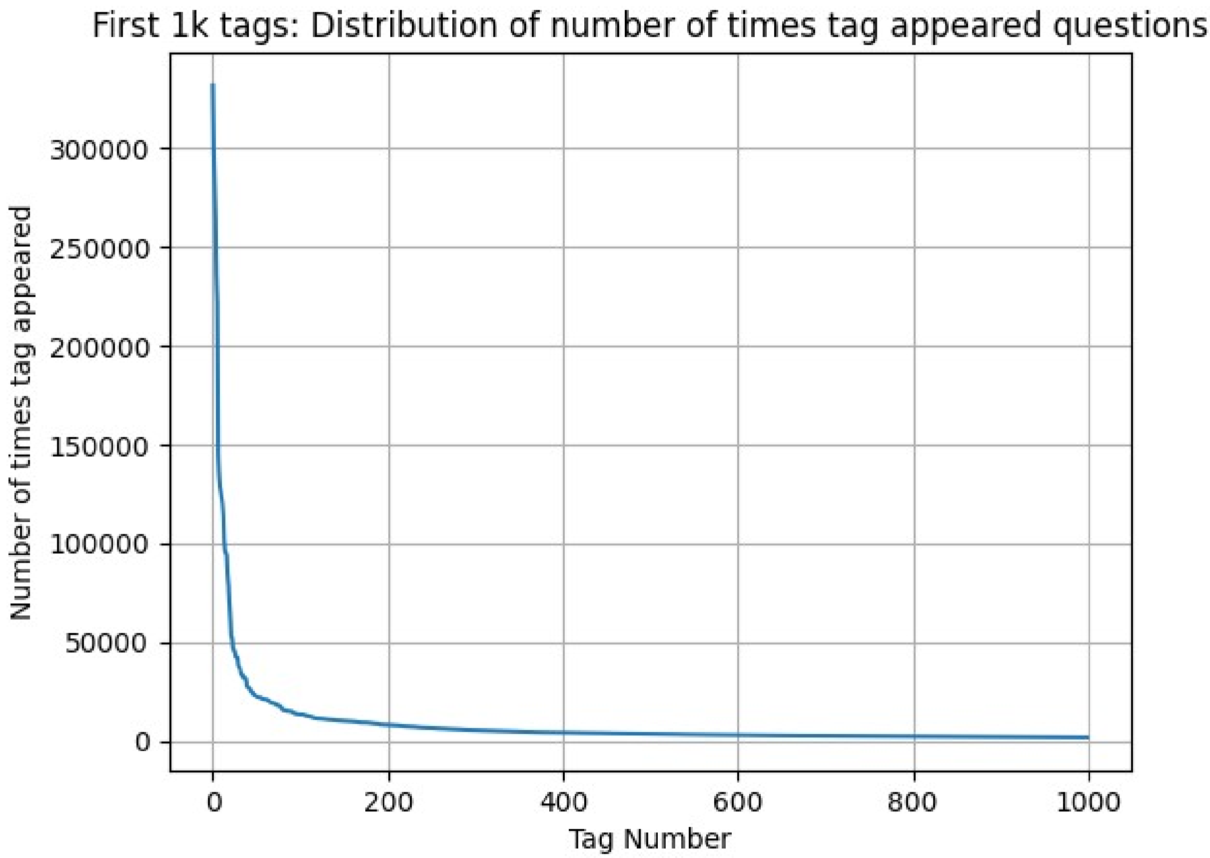

```
plt.plot(tag_counts[:10000])
plt.title('First 10k tags: Distribution of number of times tag
appeared questions')
plt.grid()
plt.xlabel('Tag Number')
plt.ylabel('Number of times tag appeared')
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```




```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054
7151
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2986   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   1574   1532   1479   1448   1406   1365   1328   1300   1266
   1245   1222   1197   1181   1158   1139   1121   1101   1076   1056
   1038   1023   1006    983    966    952    938    926    911    891
    882    869    856    841    830    816    804    789    779    770
    752    743    733    725    712    702    688    678    671    658
    650    643    634    627    616    607    598    589    583    577
    568    559    552    545    540    533    526    518    512    506
    500    495    490    485    480    477    469    465    457    450
    447    442    437    432    426    422    418    413    408    403
    398    393    388    385    381    378    374    370    367    365
    361    357    354    350    347    344    342    339    336    332
    330    326    323    319    315    312    309    307    304    301
    299    296    293    291    289    286    284    281    278    276
    275    272    270    268    265    262    260    258    256    254
    252    250    249    247    245    243    241    239    238    236
    234    233    232    230    228    226    224    222    220    219
    217    215    214    212    210    209    207    205    204    203
    201    200    199    198    196    194    193    192    191    189
```

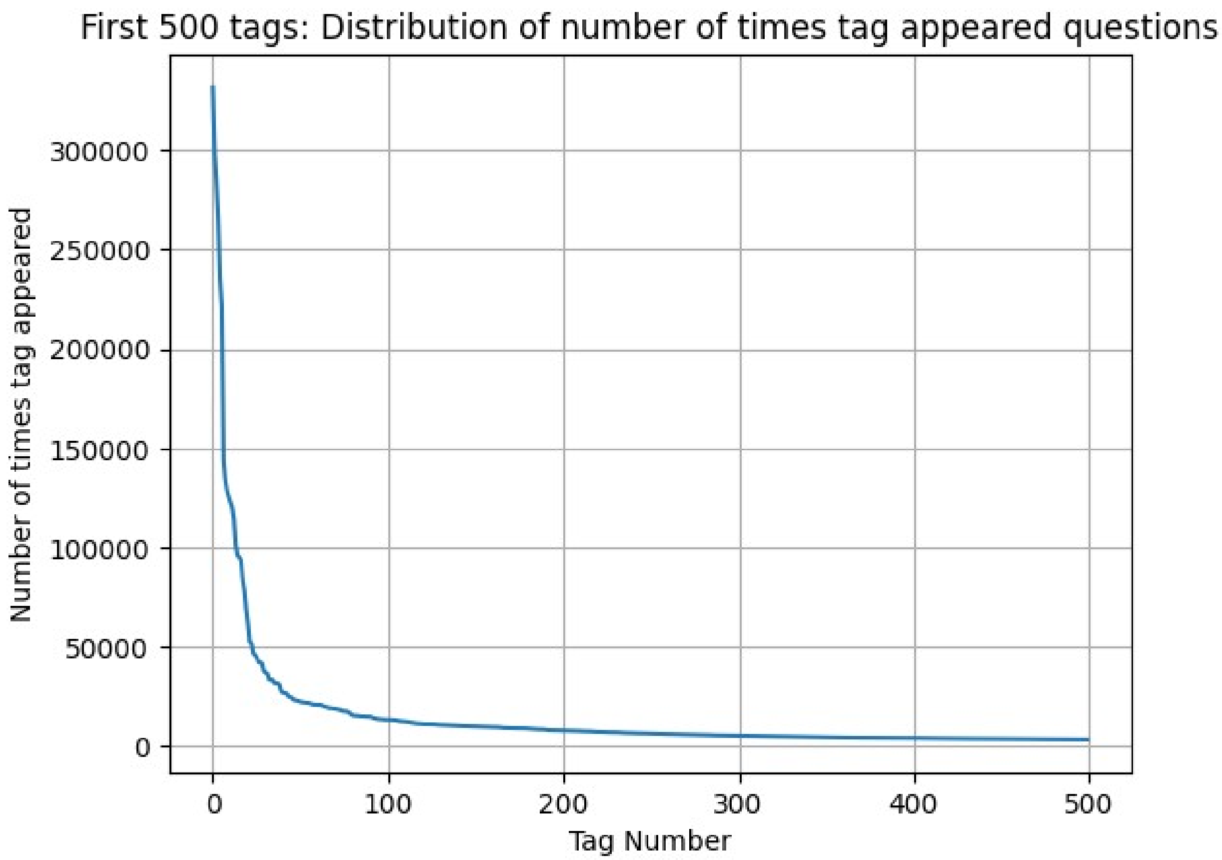
```
     188      186      185      183      182      181      180      179      178      177
     175      174      172      171      170      169      168      167      166      165
     164      162      161      160      159      158      157      156      156      155
     154      153      152      151      150      149      149      148      147      146
     145      144      143      142      142      141      140      139      138      137
     137      136      135      134      134      133      132      131      130      130
     129      128      128      127      126      126      125      124      124      123
     123      122      122      121      120      120      119      118      118      117
     117      116      116      115      115      114      113      113      112      111
     111      110      109      109      108      108      107      106      106      106
     105      105      104      104      103      103      102      102      101      101
     100      100       99       99       98       98       97       97       96       96
      95       95       94       94       93       93       93       92       92       91
      91       90       90       89       89       88       88       87       87       86
      86       86       85       85       84       84       83       83       83       82
      82       82       81       81       80       80       80       79       79       78
      78       78       78       77       77       76       76       76       75       75
      75       74       74       74       73       73       73       73       72
72]
```

```
plt.plot(tag_counts[:1000])
plt.title('First 1k tags: Distribution of number of times tag appeared
questions')
plt.grid()
plt.xlabel('Tag Number')
plt.ylabel('Number of times tag appeared')
plt.show()
print(len(tag_counts[0:1000:5]), tag_counts[0:1000:5])
```



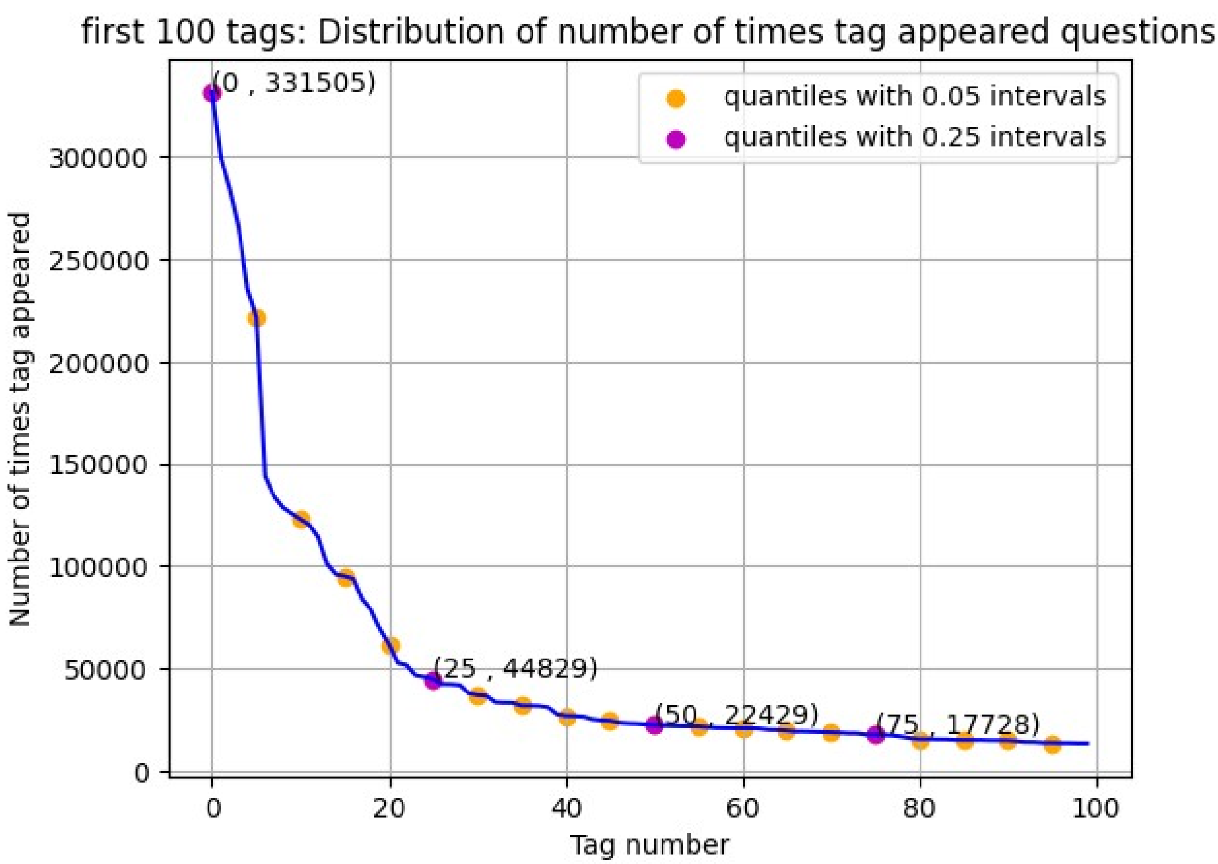

```
200 [331505 221533 122769  95160  62023  44829  37170  31897  26925
24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   7867   7702   7564   7274   7151   7052   6847   6656   6553
   6466   6291   6183   6093   5971   5865   5760   5577   5490   5411
   5370   5283   5207   5107   5066   4983   4891   4785   4658   4549
   4526   4487   4429   4335   4310   4281   4239   4228   4195   4159
   4144   4088   4050   4002   3957   3929   3874   3849   3818   3797
   3750   3703   3685   3658   3615   3593   3564   3521   3505   3483
   3453   3427   3396   3363   3326   3299   3272   3232   3196   3168
   3123   3094   3073   3050   3012   2986   2983   2953   2934   2903
   2891   2844   2819   2784   2754   2738   2726   2708   2681   2669
   2647   2621   2604   2594   2556   2527   2510   2482   2460   2444
   2431   2409   2395   2380   2363   2331   2312   2297   2290   2281
   2259   2246   2222   2211   2198   2186   2162   2142   2132   2107
   2097   2078   2057   2045   2036   2020   2011   1994   1971   1965
   1959   1952   1940   1932   1912   1900   1879   1865   1855   1841
   1828   1821   1813   1801   1782   1770   1760   1747   1741   1734
   1723   1707   1697   1688   1683   1673   1665   1656   1646
1639]
```

```
plt.plot(tag_counts[:500])
plt.title('First 500 tags: Distribution of number of times tag
appeared questions')
plt.grid()
plt.xlabel('Tag Number')
plt.ylabel('Number of times tag appeared')
plt.show()
print(len(tag_counts[0:500:5]), tag_counts[0:500:5])
```




```
100 [331505 221533 122769  95160  62023  44829  37170  31897  26925
24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   7867   7702   7564   7274   7151   7052   6847   6656   6553
   6466   6291   6183   6093   5971   5865   5760   5577   5490   5411
   5370   5283   5207   5107   5066   4983   4891   4785   4658   4549
   4526   4487   4429   4335   4310   4281   4239   4228   4195   4159
   4144   4088   4050   4002   3957   3929   3874   3849   3818   3797
   3750   3703   3685   3658   3615   3593   3564   3521   3505
3483]
```

```
plt.plot(tag_counts[0:100], c='b')
plt.scatter(x=list(range(0,100,5)), y=tag_counts[0:100:5], c='orange',
label="quantiles with 0.05 intervals")
# quantiles with 0.25 difference
plt.scatter(x=list(range(0,100,25)), y=tag_counts[0:100:25], c='m',
label = "quantiles with 0.25 intervals")

for x,y in zip(list(range(0,100,25)), tag_counts[0:100:25]):
    plt.annotate("({} , {})".format(x,y), xy=(x,y), xytext=(x-0.05,
y+500))

plt.title('first 100 tags: Distribution of number of times tag
appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.legend()
plt.show()
print(len(tag_counts[0:100:5]), tag_counts[0:100:5])
```




```
20 [331505 221533 122769  95160  62023  44829  37170  31897  26925
24537
```

```
 22429  21820  20957  19758  18905  17728  15533  15097  14884
13703]
```

```
# store tags greater than 10K in one list
lst_tags_gt_10k = tag_df[tag_df.Counts>10000].Tags
#print the length of the list
print('{} Tags are used more than 10000
times'.format(len(lst_tags_gt_10k)))
# Store tags greater than 100K in one list
lst_tags_gt_100k = tag_df[tag_df.Counts>100000].Tags
#Print the length of the list.
print ('{} Tags are used more than 100000
times'.format(len(lst_tags_gt_100k)))
```

```
153 Tags are used more than 10000 times
14 Tags are used more than 100000 times
```

Observations:

1. There are total 153 tags which are used more than 10000 times.
2. There are total 14 tags which are used more than 100000 times.
3. Most frequent tag (C#) is used 331505 times.
4. Since some tags occur more frequently than others, Micro-averaged F1-score is the appropriate metric for this problem.

3.2.4 Tags per Question

```
# Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting list of lists into single list, we will get [[3], [4],
[2], [2], [3]] and we are converting this to [3, 4, 2, 2, 3]]
tag_quest_count = [int(j) for i in tag_quest_count for j in i]
print('We have total {} datapoints.'.format(len(tag_quest_count)))
print(tag_quest_count[:5])
```

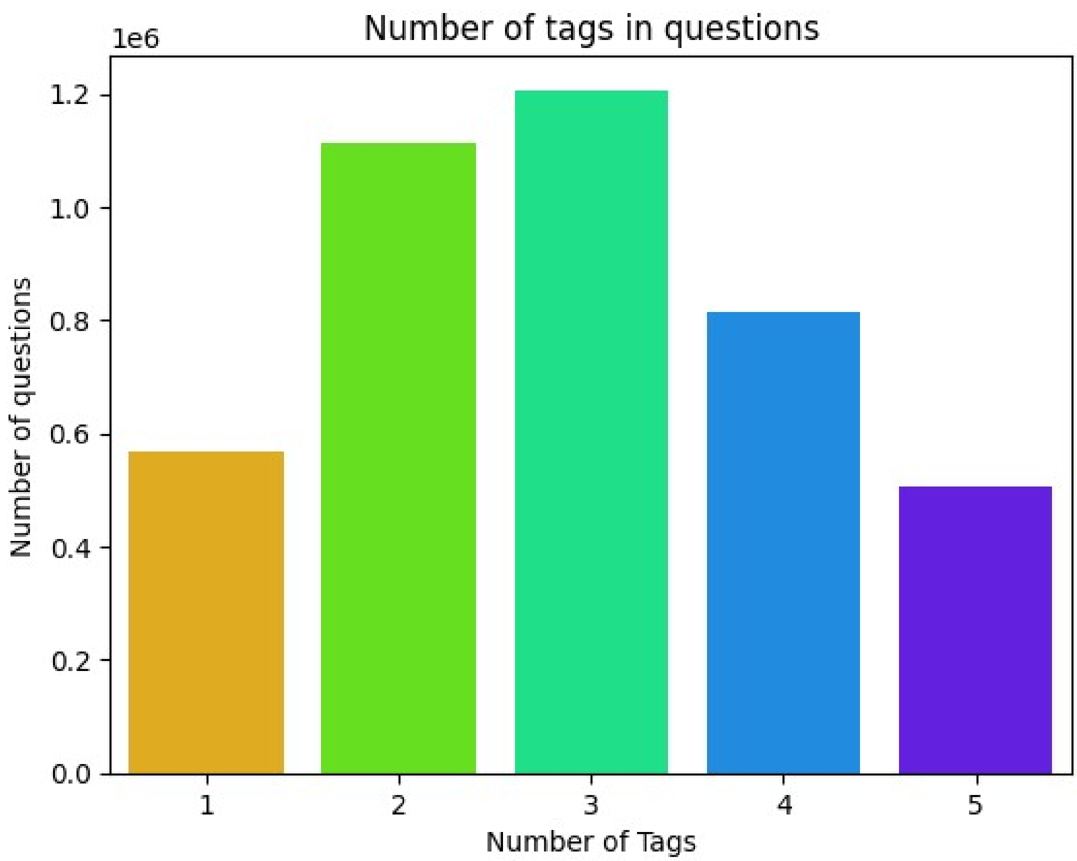
```
We have total 4206308 datapoints.
[2, 3, 4, 2, 2]
```

```
print("Maximum number of tags per question: %d"%max(tag_quest_count))
print("Minimum number of tags per question: %d"%min(tag_quest_count))
print("Average number of tags per question: %f"%
((sum(tag_quest_count)*1.0)/len(tag_quest_count)))
```

```
Maximum number of tags per question: 5
Minimum number of tags per question: 1
Average number of tags per question: 2.899442
```

```
# plotting the number of tags per question
sns.countplot(tag_quest_count, palette='gist_rainbow')
plt.title('Number of tags in questions')
plt.xlabel('Number of Tags')
```

```
plt.ylabel('Number of questions')
plt.show()
```



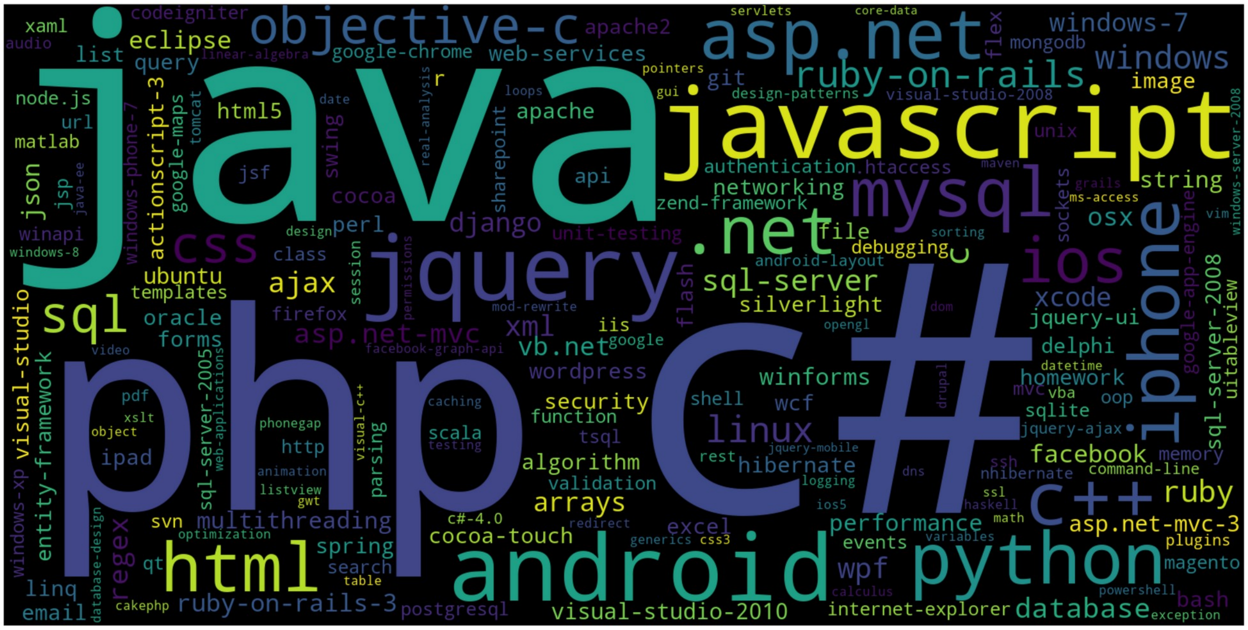
Obervations:

1. Maximum number of tags per question: 5
2. Minimum number of tags per question: 1
3. Average number of tags per question: 2.899
4. Most of the questions have either 2 or 3 tags.

3.2.5 Most Frequent Tags

```
# Plotting Wordcloud
start = datetime.now()

# Let's first convert the 'result' dictionary to 'list of tuples'
tup = dict(result.items())
# Initializing Wordcloud using frequencies of tags.
wordcloud = WordCloud(background_color='black',
                      width=1600,
                      height=800,
                     ).generate_from_frequencies(tup)
```

```
fig=plt.figure(figsize=(30,20))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
fig.savefig('tag.png')
plt.show()
print("Time taken to run this cell :", datetime.now() - start)
```



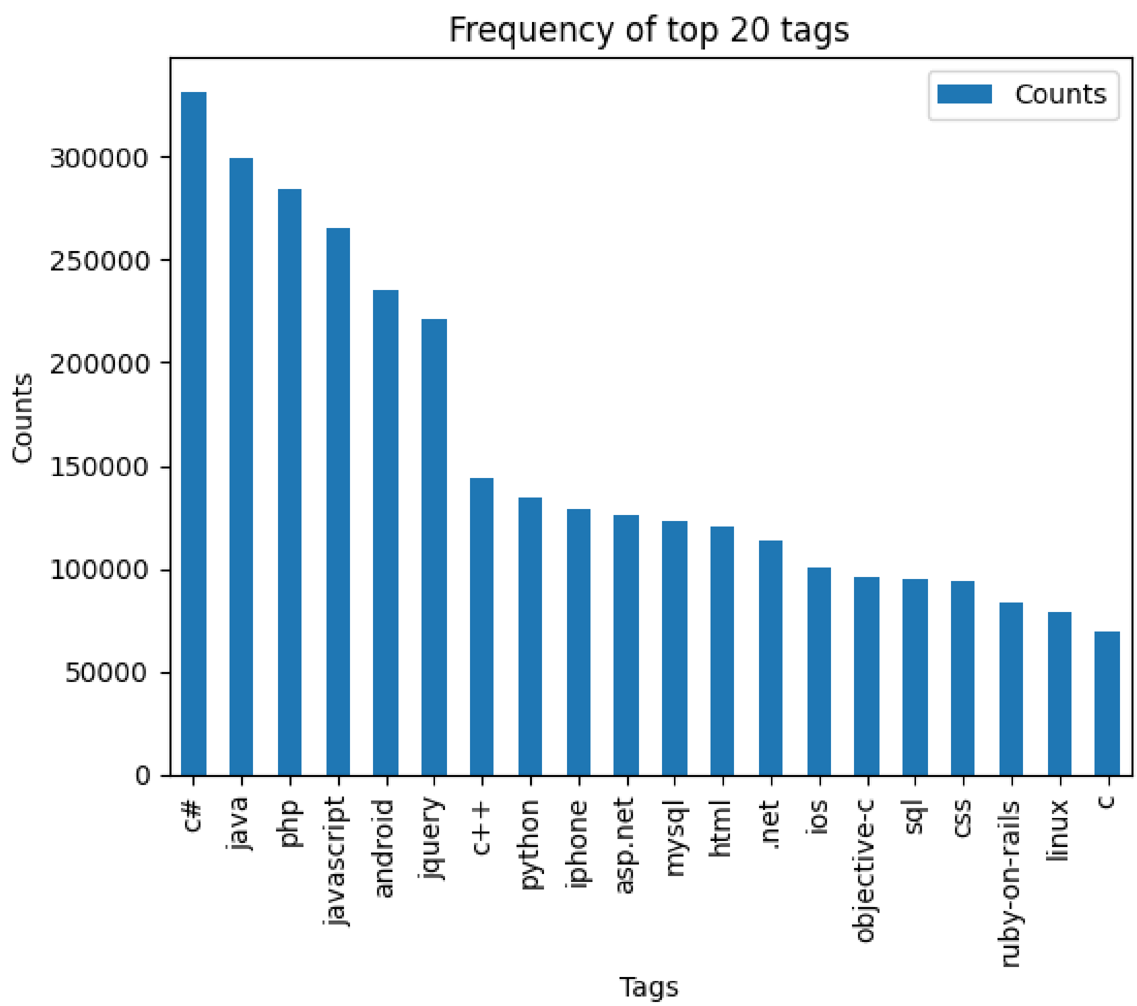

```
Time taken to run this cell : 0:00:03.916737
```

Observations:

From the above image we can see "c#", "Java", "php", "android", "javascript" are the most frequent tags.

3.2.6 Top 20 tags

```
i=np.arange(20)
tag_df_sorted.head(20).plot(kind='bar')
plt.title('Frequency of top 20 tags')
plt.xticks(i, tag_df_sorted['Tags'].head(20))
plt.xlabel('Tags')
plt.ylabel('Counts')
plt.show()
```



Observations:

1. Majority of the most frequent tags are programming languages.
2. C# is the most frequent tag.
3. Android, IOS, Linux and windows are among the top most frequent operating systems

3.3 Cleaning and preprocessing of Questions

3.3.1 Preprocessing

1. Sample 1M data points(easy computation)
2. Separate out code-snippets from Body
3. Remove Spatial characters from Question title and description (not in code)
4. Remove stop words (Except "C")
5. Remove HTML Tags
6. Convert all the characters into small letters
7. Use Snowball Stemmer to stem the words

```python
# nltk.download('stopwords')
def striphtml(data):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', str(data))
    return cleantext
stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer("english")

#http://www.sqlitetutorial.net/sqlite-python/create-tables/
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)

def checkTableExists(dbcon):
    cursr = dbcon.cursor()
    str = "select name from sqlite_master where type='table'"
    table_names = cursr.execute(str)
    print("Tables in the databse:")
    tables =table_names.fetchall()
    print(tables[0][0])
    return(len(tables))

def create_database_table(database, query):
    conn = create_connection(database)
    if conn is not None:
        create_table(conn, query)
        checkTableExists(conn)
    else:
```

```
        print("Error! cannot create the database connection.")
    conn.close()

sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed
(question text NOT NULL, code text, tags text, words_pre integer,
words_post integer, is_code integer);"""
create_database_table("Processed.db", sql_create_table)
```

```
Tables in the databse:
QuestionsProcessed
```

```
# http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-
a-sqlite-table
start = datetime.now()
read_db = 'train_no_dup.db'
write_db = 'Processed.db'
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        reader.execute("SELECT Title, Body, Tags From no_dup_train
ORDER BY RANDOM() LIMIT 1000000;")

if os.path.isfile(write_db):
    conn_w = create_connection(write_db)
    if conn_w is not None:
        tables = checkTableExists(conn_w)
        writer =conn_w.cursor()
        if tables != 0:
            writer.execute("DELETE FROM QuestionsProcessed WHERE 1")
            print("Cleared All the rows")
print("Time taken to run this cell :", datetime.now() - start)
```

```
Tables in the databse:
QuestionsProcessed
Cleared All the rows
Time taken to run this cell : 0:04:42.191270
```

We create a new database to store the sampled and preprocessed questions

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\verma\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
#http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sqlite-table/

start = datetime.now()
preprocessed_data_list=[]
reader.fetchone()
questions_with_code=0
len_pre=0
len_post=0
questions_proccesed = 0
for row in reader:

    is_code = 0

    title, question, tags = row[0], row[1], row[2]

    if '<code>' in question:
        questions_with_code+=1
        is_code = 1
    x = len(question)+len(title)
    len_pre+=x

    code = str(re.findall(r'<code>(.*?)</code>', question, flags=re.DOTALL))

    question=re.sub('<code>(.*?)</code>', '', question, flags=re.MULTILINE|re.DOTALL)
    question=striphtml(question.encode('utf-8'))

    title=title.encode('utf-8')

    question=str(title)+" "+str(question)
    question=re.sub(r'[^A-Za-z]+',' ',question)
    words=word_tokenize(str(question.lower()))

    #Removing all single letter and and stopwords from question exceptt for the letter 'c'
    question=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words and (len(j)!=1 or j=='c'))

    len_post+=len(question)
    tup = (question,code,tags,x,len(question),is_code)
    questions_proccesed += 1
    writer.execute("insert into QuestionsProcessed(question,code,tags,words_pre,words_post,is_code) values (?,?,?,?,?,?)",tup)
    if (questions_proccesed%60000==0):
        print("number of questions completed=",questions_proccesed)

no_dup_avg_len_pre=(len_pre*1.0)/questions_proccesed
no_dup_avg_len_post=(len_post*1.0)/questions_proccesed
```

```
print( "Avg. length of questions(Title+Body) before processing:
%d"%no_dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) after processing:
%d"%no_dup_avg_len_post)
print ("Percent of questions containing code: %d"%
((questions_with_code*100.0)/questions_proccesed))

print("Time taken to run this cell :", datetime.now() - start)
```

```
number of questions completed= 60000
number of questions completed= 120000
number of questions completed= 180000
number of questions completed= 240000
number of questions completed= 300000
number of questions completed= 360000
number of questions completed= 420000
number of questions completed= 480000
number of questions completed= 540000
number of questions completed= 600000
number of questions completed= 660000
number of questions completed= 720000
number of questions completed= 780000
number of questions completed= 840000
number of questions completed= 900000
number of questions completed= 960000
Avg. length of questions(Title+Body) before processing: 1170
Avg. length of questions(Title+Body) after processing: 326
Percent of questions containing code: 57
Time taken to run this cell : 0:15:42.729691
```

```
# don't forget to close the connections, or else you will end up with
locks
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()

if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        reader = conn_r.cursor()
        reader.execute("SELECT question from QuestionsProcessed LIMIT
10")
        print('Questions after preprocessed')
        print('='*100)
        reader.fetchone()
        for row in reader:
            print(row)
            print('-'*100)
```

```
conn_r.commit()
conn_r.close()

Questions after preprocessed
==============================================================================================================
('insert hashmap chang valu store hashmap ok stump one idea caus
problem problem occur class given text file find number instanc one
letter come anoth hashmapcharact hashmap store key hash charact
contain text charact correspond inner hash contain number case charact
inner hash key outerhash key inner hash key would contain number time
come number time c come number time come forth charact come
muteableint hold int valu allow increment method error occur comput
total count contain inner hash find total insert key total correct
insert pull total later find everi inner hash total take valu last
total enter analyz war peac last total enter total count pull total
instanc count charact also hope make sens offend code',)
--------------------------------------------------------------------------------------------------------------
('use string control object current project made mdiform menustrip
coupl toolstripmenuitem na coupl button devexpress navbarcontrol
intent user log userid nthe applic get datarow specif control nin row
bool true item must visibl otherwis item must invis datarow also
contain name item use item string name item use hide menustrip work
nif tri menustipitem give null refer except control item insid
menustip name item code',)
--------------------------------------------------------------------------------------------------------------
('date pars javascript differ safari chrome follow code chrome correct
print date consol safari nit fail correct import best way nto handl',)
--------------------------------------------------------------------------------------------------------------
('chang locat backup file left fail git mergetool use git merg tool
cancel ctrl c follow file left repositori chang locat file written
repositori even temporarili exampl mergefil',)
--------------------------------------------------------------------------------------------------------------
('reason ntpd sync local server use time sourc run cento releas time
drift issu server fix sync hwclock reboot ntp would second never sync
time investig problem notic ntpd synchron local regular reason ntpd
configur sync local server never go use time sourc answer need use
undisciplin local clock unless want use server local time server
connect time server fail log messag ntpd ntpd conf go disabl local
sync still curious local sync happen put temporari time server place
sub net ntpd still sync local time possibl answer ntpdc c sysinfo stat
startum ntp server wors told ntpd use local time go look sourc ntpd',)
--------------------------------------------------------------------------------------------------------------
('pivot tabl calcul across total pivot tabl excel two field total area
say calcul countof possibl calcul total instead display side side
```

```
total area display countofa countofb countofa countofb',)
--------------------------------------------------------------------------------------------------------
('master page menus hi want creat master page alreadi develop project
sinc project contain mani form quit difficult includ master page form
possibl includ master page simplest way pleas give suggest thank
advanc',)
--------------------------------------------------------------------------------------------------------
('flow control asp net form hi tri develop form user control basic
dropdownlist load static list pick list valu master page written code
page behind bind valu non pick list field find flow control like
runtim first code behind page run bind data control master page code
run user control code behind run bind pick list valu drop list would
thought flow normal anyon explain',)
--------------------------------------------------------------------------------------------------------
('next prev post link render current post thumbnail use code render
next previous post thumbnail show current post thumbnail custom post
type singl page solv',)
--------------------------------------------------------------------------------------------------------
```

```python
# Taking 1M entries to a DataFrame
write_db = 'Processed.db'
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        preprocessed_data = pd.read_sql_query("""SELECT question, Tags
FROM QuestionsProcessed""", conn_r)
conn_r.commit()
conn_r.close()

preprocessed_data.head()
```

```
                                            question  \
0  name function like pseudocod like use tree str...
1  insert hashmap chang valu store hashmap ok stu...
2  use string control object current project made...
3  date pars javascript differ safari chrome foll...
4  chang locat backup file left fail git mergetoo...

                                           tags
0    python function data-structures tree nodes
1                                          java
2                              c# .net winforms
3  javascript parsing date google-chrome safari
4                                  git mergetool
```

```
print("number of data points in sample :", preprocessed_data.shape[0])
print("number of dimensions :", preprocessed_data.shape[1])

number of data points in sample : 999999
number of dimensions : 2
```

4. Machine Learning Models

4.1 Converting tags for multilabel problems

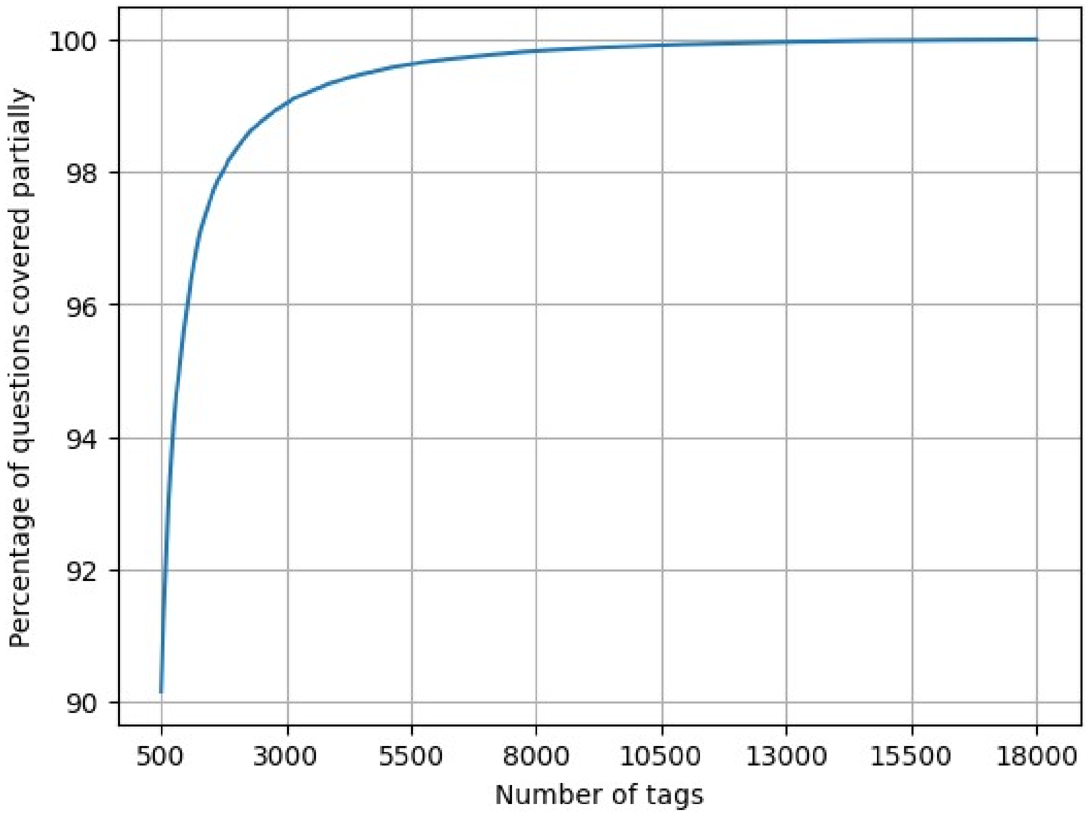
Modeling with less data points (0.5M data point) and more weight to title and 500 tags.

```
# binary=true will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(),
binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

We will sample the number of tags instead of considering all of them(due to limitation of computinf power)

```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i],
reverse=True)
    multilabel_yn = multilabel_y[:,sorted_tags_i[:n]]
    return multilabel_yn

def questions_explained_fn(n):
    multilabel_yn = tags_to_choose(n)
    x = multilabel_yn.sum(axis=1)
    return (np.count_nonzero(x==0))

questions_explained=[]
total_tags=multilabel_y.shape[1]
total_qs=preprocessed_data.shape[0]
for i in range(500, total_tags, 100):
    questions_explained.append(np.round(((total_qs-
questions_explained_fn(i))/total_qs)*100,3))

fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel('Number of tags')
plt.ylabel("Percentage of questions covered partially")
plt.grid()
plt.show()
# you can choose any number of tags based on your computing power,
minimun is 50(it covers 90% of the tags)
```

```
print("with ",5500,"tags we are covering ",questions_explained[50],"% of questions")
```




```
with  5500 tags we are covering  99.032 % of questions
```

```
multilabel_yx=tags_to_choose(5500)
print("number of questions that are not covered :", questions_explained_fn(5500),"out of ", total_qs)
```

```
number of questions that are not covered : 9683 out of  999999
```

```
print('Number of tags in sample:', multilabel_y.shape[1])
print('Number of tags taken:', multilabel_yx.shape[1], "(", (multilabel_yx.shape[1]/multilabel_y.shape[1])*100, "%)")
```

```
Number of tags in sample: 35574
Number of tags taken: 5500 ( 15.460729746444033 %)
```

We consider top 15% of tags which covers 99% of the questions.

4.2 Split the data into test and train (80:20)

```
total_size=preprocessed_data.shape[0]
train_size=int(0.8*total_size)
```

```
x_train=preprocessed_data.head(train_size)
x_test=preprocessed_data.tail(total_size - train_size)

y_train=multilabel_yx[0:train_size,:]
y_test=multilabel_yx[train_size:total_size,:]

print("Number of data points in train data :", y_train.shape)
print("Number of data points in test data :", y_test.shape)
```

```
Number of data points in train data : (799999, 5500)
Number of data points in test data : (200000, 5500)
```

4.3 Featurizing data

```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=30000,
smooth_idf=True, norm="l2", \
                             tokenizer=lambda x: x.split(),
sublinear_tf=False, ngram_range=(1,3))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)

print("Dimensions of train data X:", x_train_multilabel.shape, "Y:",
y_train.shape)
print("Dimensions of test data X:", x_test_multilabel.shape, "Y:",
y_test.shape)

# https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-
label-classification/
#https://stats.stackexchange.com/questions/117796/scikit-multi-label-
classification
# classifier = LabelPowerset(GaussianNB())
"""
from skmultilearn.adapt import MLkNN
classifier = MLkNN(k=21)

# train
classifier.fit(x_train_multilabel, y_train)

# predict
predictions = classifier.predict(x_test_multilabel)
print(accuracy_score(y_test,predictions))
print(metrics.f1_score(y_test, predictions, average = 'macro'))
print(metrics.f1_score(y_test, predictions, average = 'micro'))
print(metrics.hamming_loss(y_test,predictions))

"""
# we are getting memory error because the multilearn package
```

```
# is trying to convert the data into dense matrix
#
------------------------------------------------------------------------
-----
#MemoryError                               Traceback (most recent call
last)
#<ipython-input-170-f0e7c7f3e0be> in <module>()
#----> classifier.fit(x_train_multilabel, y_train)
```

4.4 Applying Logistic Regression with OneVsRest Classifier

```
# This takes about 6-7 hours to run try not to run it, download the
lr_with_equal_weight.pkl file and use to predict


classifier = OneVsRestClassifier(SGDClassifier(loss='log',
alpha=0.00001, penalty='l1'), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict(x_test_multilabel)

print("accuracy :",metrics.accuracy_score(y_test,predictions))
print("macro f1 score :",metrics.f1_score(y_test, predictions, average
= 'macro'))
print("micro f1 scoore :",metrics.f1_score(y_test, predictions,
average = 'micro'))
print("hamming loss :",metrics.hamming_loss(y_test,predictions))
print("Precision recall report :\
n",metrics.classification_report(y_test, predictions))

from sklearn.externals import joblib
joblib.dump(classifier, 'lr_with_equal_weight.pkl')
```

4.5 Modeling with less data points(0.5M data points) and more weight to title and 500 tags only

```
sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed
(question text NOT NULL, code text, tags text, words_pre integer,
words_post integer, is_code integer);"""
create_database_table("Titlemoreweight.db", sql_create_table)
```

```
Tables in the databse:
QuestionsProcessed
```

```
# http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-
a-sqlite-table

read_db = "train_no_dup.db"
write_db = 'Titlemoreweight.db'
```

```
train_datasize = 400000
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader = conn_r.cursor()
        # for selecting first 0.5M rows
        reader.execute("SELECT Title, Body, Tags from no_dup_train
LIMIT 500001;")
        # for selecting random points
        #reader.execute("SELECT Title, Body, Tags From no_dup_train
ORDER BY RANDOM() LIMIT 1000001;")

if os.path.isfile(write_db):
    conn_w = create_connection(write_db)
    if conn_w is not None:
        tables = checkTableExists(conn_w)
        writer = conn_w.cursor()
        if tables != 0:
            writer.execute("DELETE FROM QuestionsProcessed WHERE 1")
            print('Cleared all the rows')
```

```
Tables in the databse:
QuestionsProcessed
Cleared all the rows
```

4.5.1 Preprocessing of questions

```
start = datetime.now()
preprocessed_data_list = []
reader.fetchone()
questions_with_code=0
len_pre=0
len_post=0
questions_proccesed=0
for row in reader:
    is_code=0
    title, question, tags = row[0], row[1], row[2]

    if '<code>' in question:
        questions_with_code+=1
        is_code = 1
    x = len(question)+len(title)
    len_pre+=x

    code = str(re.findall(r'<code>(.*?)</code>', question,
flags=re.DOTALL))

    question=re.sub('<code>(.*?)</code>', '', question,
flags=re.MULTILINE|re.DOTALL)
    question=striphtml(question.encode('utf-8'))
```

```
    title=title.encode('utf-8')

    question=str(title)+" "+str(title)+" "+str(title)+"
"+str(question)
    question=re.sub(r'[^A-Za-z]+',' ',question)
    words=word_tokenize(str(question.lower()))

    #Removing all single letter and and stopwords from question
exceptt for the letter 'c'
    question=' '.join(str(stemmer.stem(j)) for j in words if j not in
stop_words and (len(j)!=1 or j=='c'))

    len_post+=len(question)
    tup = (question,code,tags,x,len(question),is_code)
    questions_proccesed += 1
    writer.execute("insert into
QuestionsProcessed(question,code,tags,words_pre,words_post,is_code)
values (?,?,?,?,?,?)",tup)
    if (questions_proccesed%100000==0):
        print("number of questions completed=",questions_proccesed)

no_dup_avg_len_pre=(len_pre*1.0)/questions_proccesed
no_dup_avg_len_post=(len_post*1.0)/questions_proccesed

print( "Avg. length of questions(Title+Body) before processing:
%d"%no_dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) after processing:
%d"%no_dup_avg_len_post)
print ("Percent of questions containing code: %d"%
((questions_with_code*100.0)/questions_proccesed))

print("Time taken to run this cell :", datetime.now() - start)
```

```
number of questions completed= 100000
number of questions completed= 200000
number of questions completed= 300000
number of questions completed= 400000
number of questions completed= 500000
Avg. length of questions(Title+Body) before processing: 1239
Avg. length of questions(Title+Body) after processing: 412
Percent of questions containing code: 57
Time taken to run this cell : 0:12:38.485003
```

```
# never forget to close the conections or else we will end up with
database locks
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()
```

__ Sample questions after preprocessing data__

```
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        reader.execute("SELECT question From QuestionsProcessed LIMIT 10")
        print("Questions after preprocessed")
        print('='*100)
        reader.fetchone()
        for row in reader:
            print(row)
            print('-'*100)
conn_r.commit()
conn_r.close()
```

```
Questions after preprocessed
======================================================================
==============================
('dynam datagrid bind silverlight dynam datagrid bind silverlight
dynam datagrid bind silverlight bind datagrid dynam code wrote code
debug code block seem bind correct grid come column form come grid
column although necessari bind nthank repli advanc',)
----------------------------------------------------------------------
------------------------------
('java lang noclassdeffounderror javax servlet jsp tagext
taglibraryvalid java lang noclassdeffounderror javax servlet jsp
tagext taglibraryvalid java lang noclassdeffounderror javax servlet
jsp tagext taglibraryvalid follow guid link instal jstl got follow
error tri launch jsp page java lang noclassdeffounderror javax servlet
jsp tagext taglibraryvalid taglib declar instal jstl tomcat webapp tri
project work also tri version jstl still messag caus solv',)
----------------------------------------------------------------------
------------------------------
('java sql sqlexcept microsoft odbc driver manag invalid descriptor
index java sql sqlexcept microsoft odbc driver manag invalid
descriptor index java sql sqlexcept microsoft odbc driver manag
invalid descriptor index use follow code display caus solv',)
----------------------------------------------------------------------
------------------------------
('better way updat feed fb php sdk better way updat feed fb php sdk
better way updat feed fb php sdk novic facebook api read mani tutori
still confus find post feed api method like correct second way use
curl someth like way better',)
----------------------------------------------------------------------
------------------------------
('btnadd click event open two window record ad btnadd click event open
two window record ad btnadd click event open two window record ad open
window search aspx use code hav add button search aspx nwhen insert
```

```
record btnadd click event open anoth window nafter insert record close
window',)
----------------------------------------------------------------------
------------------------------
('sql inject issu prevent correct form submiss php sql inject issu
prevent correct form submiss php sql inject issu prevent correct form
submiss php check everyth think make sure input field safe type sql
inject good news safe bad news one tag mess form submiss place even
touch life figur exact html use templat file forgiv okay entir php
script get execut see data post none forum field post problem use
someth titl field none data get post current use print post see submit
noth work flawless statement though also mention script work flawless
local machin use host come across problem state list input test
mess',)
----------------------------------------------------------------------
------------------------------
('countabl subaddit lebesgu measur countabl subaddit lebesgu measur
countabl subaddit lebesgu measur let lbrace rbrace sequenc set sigma
algebra mathcal want show left bigcup right leq sum left right
countabl addit measur defin set sigma algebra mathcal think use
monoton properti somewher proof start appreci littl help nthank ad han
answer make follow addit construct given han answer clear bigcup
bigcup cap emptyset neq left bigcup right left bigcup right sum left
right also construct subset monoton left right leq left right final
would sum leq sum result follow',)
----------------------------------------------------------------------
------------------------------
('hql equival sql queri hql equival sql queri hql equival sql queri
hql queri replac name class properti name error occur hql error',)
----------------------------------------------------------------------
------------------------------
('undefin symbol architectur objc class skpsmtpmessag referenc error
undefin symbol architectur objc class skpsmtpmessag referenc error
undefin symbol architectur objc class skpsmtpmessag referenc error
import framework send email applic background import framework
skpsmtpmessag somebodi suggest get error collect ld return exit status
import framework correct sorc taken framework follow
mfmailcomposeviewcontrol question lock field updat answer drag drop
folder project click copi nthat',)
----------------------------------------------------------------------
------------------------------
```

__ Saving Preprocessed data to a Database__

```
#Taking 0.5 Millioon entries to a dataframe
write_db = 'Titlemoreweight.db'
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
```

```
        preprocessed_data = pd.read_sql_query('''SELECT question, Tags
from QuestionsProcessed''', conn_r)
conn_r.commit()
conn_r.close()
```

```
preprocessed_data.head()
```

```
                                            question  \
0  dynam datagrid bind silverlight dynam datagrid...
1  dynam datagrid bind silverlight dynam datagrid...
2  java lang noclassdeffounderror javax servlet j...
3  java sql sqlexcept microsoft odbc driver manag...
4  better way updat feed fb php sdk better way up...

                                  tags
0          c# silverlight data-binding
1  c# silverlight data-binding columns
2                             jsp jstl
3                            java jdbc
4        facebook api facebook-php-sdk
```

```
print("number of data points in sample :", preprocessed_data.shape[0])
print("number of dimensions :", preprocessed_data.shape[1])
```

```
number of data points in sample : 500000
number of dimensions : 2
```

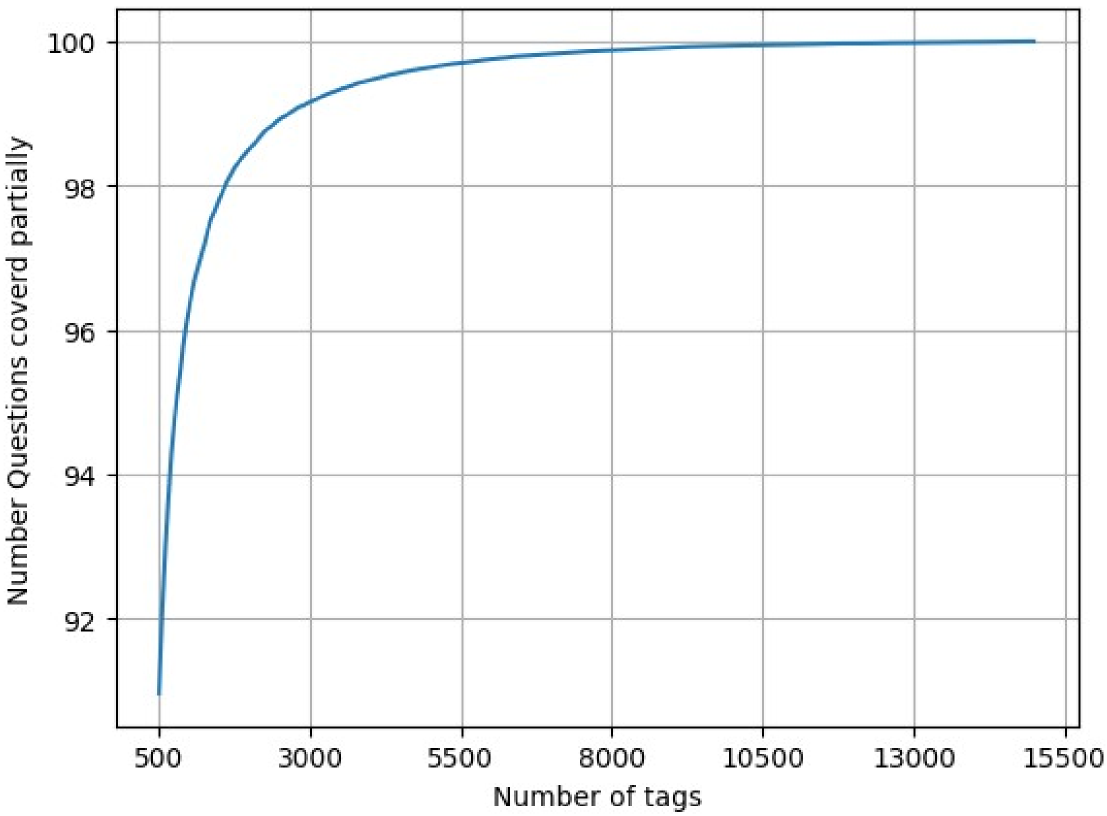
Converting tags into multilabel output variables

```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(),
binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

Selecting 500 tags

```
questions_explained = []
total_tags = multilabel_y.shape[1]
total_qs=preprocessed_data.shape[0]
for i in range(500, total_tags, 100):
    questions_explained.append(np.round(((total_qs-
questions_explained_fn(i))/total_qs)*100, 3))

fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Number Questions coverd partially")
plt.grid()
plt.show()
```

```
# you can choose any number of tags based on your computing power,
minimun is 500(it covers 90% of the tags)
print("with ",5500,"tags we are covering ",questions_explained[50],"%
of questions")
print("with ",500,"tags we are covering ",questions_explained[0],"% of
questions")
```




```
with  5500 tags we are covering  99.157 % of questions
with  500 tags we are covering  90.956 % of questions
```

```
# we will be taking 500 tags
multilabel_yx = tags_to_choose(500)
print("number of questions that are not covered :",
questions_explained_fn(500),"out of ", total_qs)
```

```
number of questions that are not covered : 45221 out of  500000
```

```
x_train=preprocessed_data.head(train_datasize)
x_test=preprocessed_data.tail(preprocessed_data.shape[0] - 400000)

y_train = multilabel_yx[0:train_datasize,:]
y_test = multilabel_yx[train_datasize:preprocessed_data.shape[0],:]
```

```
print("Number of data points in train data :", y_train.shape)
print("Number of data points in test data :", y_test.shape)
```

```
Number of data points in train data : (400000, 500)
Number of data points in test data : (100000, 500)
```

4.5.2 Featurizing data with Tfidf Vectorizer

```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000,
smooth_idf=True, norm='l2', \
                             tokenizer = lambda x: x.split(),
sublinear_tf=False, ngram_range=(1,3))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel= vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:04:22.901639
```

```
print("Dimensions of train data X:",x_train_multilabel.shape,
"Y :",y_train.shape)
print("Dimensions of test data
X:",x_test_multilabel.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (400000, 98846) Y : (400000, 500)
Dimensions of test data X: (100000, 98846) Y: (100000, 500)
```

4.5.3 Applying Logistic Regression with OneVsRest Classifier

```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log',
alpha=0.00001, penalty='l1'), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


print("Accuracy :",metrics.accuracy_score(y_test, predictions))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions))


precision = precision_score(y_test, predictions, average='micro')
recall = recall_score(y_test, predictions, average='micro')
f1 = f1_score(y_test, predictions, average='micro')

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')
```

```
print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

print (metrics.classification_report(y_test, predictions))
print("Time taken to run this cell :", datetime.now() - start)
```

```
Accuracy : 0.23152
Hamming loss  0.00282472
Micro-average quality numbers
Precision: 0.7066, Recall: 0.3206, F1-measure: 0.4410
Macro-average quality numbers
Precision: 0.5217, Recall: 0.2541, F1-measure: 0.3261
              precision    recall  f1-score   support

           0       0.95      0.65      0.77      5519
           1       0.65      0.25      0.36      8190
           2       0.80      0.37      0.51      6529
           3       0.70      0.35      0.46      3231
           4       0.80      0.41      0.54      6430
           5       0.82      0.35      0.49      2879
           6       0.85      0.50      0.63      5086
           7       0.87      0.55      0.68      4533
           8       0.54      0.13      0.21      3000
           9       0.81      0.53      0.64      2765
          10       0.57      0.16      0.25      3051
          11       0.70      0.33      0.45      3009
          12       0.62      0.23      0.33      2630
          13       0.65      0.16      0.26      1426
          14       0.89      0.55      0.68      2548
          15       0.66      0.18      0.29      2371
          16       0.64      0.23      0.34       873
          17       0.88      0.61      0.72      2151
          18       0.61      0.23      0.33      2204
          19       0.70      0.40      0.51       831
          20       0.75      0.42      0.54      1860
          21       0.26      0.07      0.12      2023
          22       0.49      0.22      0.31      1513
          23       0.91      0.49      0.64      1207
          24       0.56      0.28      0.37       506
          25       0.48      0.26      0.33       425
          26       0.62      0.39      0.48       793
          27       0.58      0.33      0.42      1291
          28       0.73      0.36      0.48      1208
          29       0.39      0.09      0.15       406
          30       0.38      0.05      0.09       504
          31       0.28      0.10      0.14       732
          32       0.59      0.25      0.35       441
          33       0.52      0.16      0.24      1645
```

34	0.71	0.24	0.36	1058
35	0.82	0.54	0.65	946
36	0.65	0.22	0.33	644
37	0.97	0.70	0.81	136
38	0.66	0.36	0.46	570
39	0.82	0.27	0.40	766
40	0.62	0.28	0.39	1132
41	0.44	0.18	0.25	174
42	0.77	0.55	0.64	210
43	0.80	0.40	0.53	433
44	0.66	0.51	0.58	626
45	0.35	0.08	0.13	852
46	0.72	0.46	0.56	534
47	0.34	0.13	0.19	350
48	0.71	0.51	0.59	496
49	0.79	0.62	0.69	785
50	0.19	0.05	0.08	475
51	0.34	0.10	0.16	305
52	0.35	0.02	0.04	251
53	0.68	0.40	0.50	914
54	0.41	0.14	0.21	728
55	0.15	0.01	0.01	258
56	0.47	0.19	0.28	821
57	0.48	0.09	0.15	541
58	0.61	0.12	0.20	748
59	0.94	0.66	0.78	724
60	0.33	0.07	0.11	660
61	0.64	0.13	0.21	235
62	0.91	0.71	0.80	718
63	0.84	0.64	0.72	468
64	0.53	0.28	0.36	191
65	0.35	0.11	0.17	429
66	0.30	0.05	0.09	415
67	0.71	0.50	0.59	274
68	0.83	0.52	0.64	510
69	0.66	0.43	0.52	466
70	0.27	0.06	0.09	305
71	0.50	0.16	0.24	247
72	0.75	0.50	0.60	401
73	0.99	0.78	0.87	86
74	0.76	0.40	0.52	120
75	0.89	0.66	0.76	129
76	0.67	0.01	0.02	473
77	0.42	0.30	0.35	143
78	0.78	0.47	0.58	347
79	0.72	0.24	0.36	479
80	0.52	0.34	0.41	279
81	0.76	0.18	0.29	461
82	0.29	0.02	0.04	298

83	0.74	0.46	0.56	396
84	0.57	0.33	0.42	184
85	0.39	0.06	0.10	573
86	0.53	0.06	0.11	325
87	0.42	0.21	0.28	273
88	0.40	0.17	0.24	135
89	0.31	0.08	0.12	232
90	0.53	0.31	0.39	409
91	0.55	0.20	0.29	420
92	0.75	0.53	0.62	408
93	0.63	0.48	0.55	241
94	0.36	0.04	0.08	211
95	0.32	0.07	0.12	277
96	0.27	0.03	0.06	410
97	0.88	0.32	0.47	501
98	0.75	0.65	0.70	136
99	0.54	0.29	0.38	239
100	0.57	0.17	0.26	324
101	0.92	0.64	0.75	277
102	0.92	0.73	0.82	613
103	0.54	0.18	0.27	157
104	0.21	0.05	0.09	295
105	0.82	0.32	0.46	334
106	0.85	0.17	0.28	335
107	0.77	0.50	0.61	389
108	0.52	0.23	0.32	251
109	0.53	0.41	0.46	317
110	0.42	0.04	0.08	187
111	0.29	0.01	0.03	140
112	0.64	0.33	0.44	154
113	0.56	0.17	0.27	332
114	0.44	0.28	0.34	323
115	0.46	0.22	0.29	344
116	0.76	0.52	0.62	370
117	0.57	0.23	0.32	313
118	0.78	0.69	0.73	874
119	0.46	0.20	0.28	293
120	0.00	0.00	0.00	200
121	0.77	0.48	0.59	463
122	0.42	0.09	0.15	119
123	0.00	0.00	0.00	256
124	0.89	0.68	0.77	195
125	0.41	0.14	0.21	138
126	0.78	0.49	0.60	376
127	0.14	0.03	0.05	122
128	0.13	0.03	0.05	252
129	0.49	0.12	0.19	144
130	0.44	0.10	0.16	150
131	0.28	0.02	0.04	210

132	0.30	0.02	0.04	361
133	0.94	0.55	0.69	453
134	0.89	0.74	0.81	124
135	0.17	0.02	0.04	91
136	0.65	0.26	0.37	128
137	0.60	0.37	0.46	218
138	0.72	0.15	0.25	243
139	0.42	0.21	0.28	149
140	0.65	0.33	0.44	318
141	0.29	0.11	0.16	159
142	0.64	0.35	0.45	274
143	0.85	0.74	0.79	362
144	0.62	0.18	0.28	118
145	0.67	0.37	0.48	164
146	0.59	0.27	0.37	461
147	0.66	0.41	0.51	159
148	0.33	0.13	0.19	166
149	0.99	0.46	0.63	346
150	0.43	0.05	0.09	350
151	0.91	0.75	0.82	55
152	0.80	0.46	0.59	387
153	0.47	0.11	0.18	150
154	0.57	0.11	0.18	281
155	0.17	0.03	0.06	202
156	0.76	0.63	0.69	130
157	0.24	0.07	0.10	245
158	0.87	0.63	0.73	177
159	0.47	0.26	0.34	130
160	0.49	0.12	0.19	336
161	0.91	0.56	0.69	220
162	0.19	0.03	0.05	229
163	0.89	0.43	0.58	316
164	0.76	0.41	0.53	283
165	0.62	0.32	0.42	197
166	0.49	0.26	0.34	101
167	0.47	0.19	0.28	231
168	0.42	0.12	0.18	370
169	0.39	0.17	0.23	258
170	0.27	0.06	0.10	101
171	0.38	0.21	0.27	89
172	0.51	0.35	0.42	193
173	0.42	0.21	0.28	309
174	0.54	0.15	0.23	172
175	0.95	0.75	0.84	95
176	0.92	0.60	0.73	346
177	0.93	0.46	0.61	322
178	0.62	0.45	0.52	232
179	0.45	0.07	0.12	125
180	0.51	0.26	0.35	145

181	0.45	0.12	0.19	77
182	0.12	0.02	0.03	182
183	0.55	0.29	0.38	257
184	0.07	0.01	0.02	216
185	0.35	0.05	0.09	242
186	0.34	0.13	0.19	165
187	0.75	0.57	0.65	263
188	0.32	0.09	0.14	174
189	0.71	0.32	0.44	136
190	0.90	0.50	0.64	202
191	0.44	0.14	0.21	134
192	0.71	0.40	0.51	230
193	0.47	0.19	0.27	90
194	0.59	0.46	0.52	185
195	0.17	0.04	0.06	156
196	0.41	0.07	0.13	160
197	0.33	0.02	0.03	266
198	0.40	0.07	0.11	284
199	0.30	0.05	0.08	145
200	0.94	0.71	0.81	212
201	0.12	0.01	0.02	317
202	0.77	0.56	0.65	427
203	0.26	0.08	0.12	232
204	0.45	0.25	0.32	217
205	0.51	0.45	0.48	527
206	0.13	0.02	0.03	124
207	0.57	0.13	0.21	103
208	0.88	0.48	0.62	287
209	0.36	0.09	0.14	193
210	0.69	0.33	0.44	220
211	0.38	0.04	0.07	140
212	0.13	0.02	0.03	161
213	0.53	0.25	0.34	72
214	0.59	0.43	0.50	396
215	0.84	0.31	0.45	134
216	0.48	0.06	0.11	400
217	0.57	0.23	0.32	75
218	0.96	0.75	0.85	219
219	0.75	0.33	0.46	210
220	0.90	0.61	0.72	298
221	0.96	0.61	0.75	266
222	0.76	0.40	0.52	290
223	0.08	0.01	0.01	128
224	0.71	0.28	0.40	159
225	0.30	0.12	0.17	164
226	0.66	0.36	0.47	144
227	0.55	0.30	0.38	276
228	0.11	0.01	0.02	235
229	0.14	0.01	0.02	216

230	0.34	0.17	0.22	228
231	0.71	0.47	0.57	64
232	0.47	0.09	0.15	103
233	0.73	0.33	0.45	216
234	0.69	0.09	0.17	116
235	0.56	0.36	0.44	77
236	0.95	0.63	0.76	67
237	0.56	0.10	0.17	218
238	0.34	0.08	0.13	139
239	0.20	0.01	0.02	94
240	0.52	0.29	0.37	77
241	0.00	0.00	0.00	167
242	0.86	0.36	0.51	86
243	0.43	0.16	0.23	58
244	0.57	0.18	0.27	269
245	0.18	0.05	0.08	112
246	0.94	0.73	0.82	255
247	0.00	0.00	0.00	58
248	0.43	0.04	0.07	81
249	0.08	0.01	0.01	131
250	0.36	0.16	0.22	93
251	0.71	0.27	0.39	154
252	0.25	0.02	0.04	129
253	0.58	0.36	0.44	83
254	0.38	0.07	0.12	191
255	0.11	0.02	0.03	219
256	0.22	0.03	0.05	130
257	0.47	0.29	0.36	93
258	0.64	0.42	0.51	217
259	0.30	0.10	0.15	141
260	0.40	0.01	0.03	143
261	0.52	0.13	0.21	219
262	0.54	0.28	0.37	107
263	0.42	0.25	0.31	236
264	0.24	0.14	0.18	119
265	0.34	0.14	0.20	72
266	0.00	0.00	0.00	70
267	0.25	0.12	0.16	107
268	0.69	0.45	0.54	169
269	0.29	0.10	0.15	129
270	0.74	0.53	0.62	159
271	0.84	0.45	0.58	190
272	0.26	0.04	0.06	248
273	0.91	0.70	0.79	264
274	0.60	0.23	0.33	105
275	0.53	0.08	0.13	104
276	0.11	0.02	0.03	115
277	0.85	0.62	0.71	170
278	0.64	0.26	0.37	145

279	0.92	0.63	0.75	230
280	0.57	0.41	0.48	80
281	0.67	0.54	0.60	217
282	0.75	0.50	0.60	175
283	0.31	0.05	0.08	269
284	0.64	0.28	0.39	74
285	0.83	0.51	0.63	206
286	0.88	0.59	0.71	227
287	0.87	0.30	0.45	130
288	0.38	0.07	0.12	129
289	0.33	0.03	0.05	80
290	0.18	0.07	0.10	99
291	0.79	0.33	0.46	208
292	0.22	0.03	0.05	67
293	0.91	0.46	0.61	109
294	0.39	0.22	0.28	140
295	0.22	0.07	0.10	241
296	0.22	0.10	0.13	72
297	0.17	0.03	0.05	107
298	0.86	0.39	0.54	61
299	0.96	0.34	0.50	77
300	0.19	0.07	0.10	111
301	0.00	0.00	0.00	126
302	0.00	0.00	0.00	73
303	0.58	0.36	0.45	176
304	0.96	0.75	0.84	230
305	0.95	0.61	0.74	156
306	0.52	0.37	0.43	146
307	0.33	0.10	0.16	98
308	0.00	0.00	0.00	78
309	0.67	0.06	0.12	94
310	0.73	0.38	0.50	162
311	0.81	0.53	0.64	116
312	0.48	0.21	0.29	57
313	0.75	0.05	0.09	65
314	0.51	0.36	0.42	138
315	0.49	0.19	0.28	195
316	0.51	0.30	0.38	69
317	0.34	0.10	0.16	134
318	0.55	0.39	0.45	148
319	0.84	0.44	0.58	161
320	0.18	0.12	0.15	104
321	0.82	0.52	0.64	156
322	0.60	0.33	0.43	134
323	0.59	0.38	0.46	232
324	0.48	0.17	0.26	92
325	0.45	0.27	0.34	197
326	0.11	0.02	0.03	126
327	0.00	0.00	0.00	115

328	0.97	0.67	0.79	198
329	0.58	0.30	0.40	125
330	0.82	0.22	0.35	81
331	0.50	0.09	0.15	94
332	1.00	0.02	0.04	56
333	0.15	0.03	0.05	260
334	0.17	0.03	0.06	60
335	0.37	0.10	0.16	110
336	0.59	0.42	0.49	71
337	0.27	0.06	0.10	66
338	0.46	0.34	0.39	150
339	0.00	0.00	0.00	54
340	0.84	0.55	0.67	195
341	0.86	0.24	0.38	79
342	0.47	0.18	0.26	38
343	0.62	0.35	0.45	43
344	0.50	0.21	0.29	68
345	0.68	0.41	0.51	73
346	0.23	0.03	0.05	116
347	0.88	0.33	0.48	111
348	0.23	0.10	0.13	63
349	0.85	0.61	0.71	104
350	0.62	0.45	0.53	44
351	0.56	0.12	0.20	40
352	0.98	0.44	0.61	136
353	0.45	0.24	0.31	54
354	0.45	0.04	0.07	134
355	0.51	0.27	0.35	120
356	0.54	0.23	0.32	228
357	0.70	0.30	0.42	269
358	0.76	0.44	0.56	80
359	0.86	0.44	0.58	140
360	0.32	0.13	0.18	125
361	0.90	0.64	0.75	169
362	0.11	0.04	0.05	56
363	0.95	0.68	0.79	154
364	0.29	0.03	0.06	58
365	0.31	0.15	0.21	71
366	1.00	0.63	0.77	54
367	0.31	0.03	0.06	116
368	0.50	0.02	0.04	54
369	0.00	0.00	0.00	71
370	0.18	0.03	0.06	61
371	0.55	0.08	0.15	71
372	0.62	0.44	0.52	52
373	0.78	0.39	0.52	150
374	0.29	0.11	0.16	93
375	0.14	0.03	0.05	67
376	0.00	0.00	0.00	76

377	0.70	0.22	0.33	106
378	0.11	0.01	0.02	86
379	0.25	0.07	0.11	14
380	0.98	0.48	0.64	122
381	0.17	0.03	0.05	104
382	0.29	0.08	0.12	66
383	0.47	0.31	0.37	110
384	0.00	0.00	0.00	155
385	0.45	0.10	0.16	50
386	0.29	0.14	0.19	64
387	0.27	0.06	0.10	93
388	0.56	0.28	0.38	102
389	0.07	0.01	0.02	108
390	0.97	0.66	0.79	178
391	0.64	0.18	0.28	115
392	0.81	0.40	0.54	42
393	0.00	0.00	0.00	134
394	0.50	0.02	0.03	112
395	0.15	0.02	0.03	176
396	0.38	0.07	0.12	125
397	0.75	0.30	0.43	224
398	0.88	0.59	0.70	63
399	0.00	0.00	0.00	59
400	0.49	0.32	0.38	63
401	0.44	0.18	0.26	98
402	0.53	0.15	0.24	162
403	0.34	0.13	0.19	83
404	0.81	0.89	0.85	19
405	0.30	0.07	0.11	92
406	0.86	0.15	0.25	41
407	0.59	0.30	0.40	43
408	0.00	0.00	0.00	160
409	0.13	0.08	0.10	50
410	0.00	0.00	0.00	19
411	0.34	0.09	0.14	175
412	0.31	0.07	0.11	72
413	0.40	0.04	0.08	95
414	0.13	0.02	0.04	97
415	0.35	0.17	0.23	48
416	0.44	0.28	0.34	83
417	0.60	0.07	0.13	40
418	0.40	0.09	0.14	91
419	0.49	0.32	0.39	90
420	0.31	0.24	0.27	37
421	0.00	0.00	0.00	66
422	0.60	0.33	0.42	73
423	0.52	0.25	0.34	56
424	0.93	0.82	0.87	33
425	0.00	0.00	0.00	76

426	0.29	0.05	0.08	81
427	0.99	0.67	0.80	150
428	0.95	0.69	0.80	29
429	0.99	0.75	0.85	389
430	0.64	0.35	0.46	167
431	0.48	0.08	0.14	123
432	0.46	0.33	0.39	39
433	0.31	0.18	0.23	82
434	1.00	0.65	0.79	66
435	0.62	0.45	0.52	93
436	0.49	0.22	0.30	87
437	0.15	0.03	0.06	86
438	0.74	0.47	0.58	104
439	0.59	0.13	0.21	100
440	0.20	0.01	0.01	141
441	0.41	0.25	0.31	110
442	0.20	0.07	0.10	123
443	0.42	0.11	0.18	71
444	0.44	0.07	0.13	109
445	0.40	0.21	0.27	48
446	0.43	0.25	0.32	76
447	0.24	0.11	0.15	38
448	0.70	0.52	0.60	81
449	0.35	0.08	0.13	132
450	0.47	0.26	0.33	81
451	0.69	0.36	0.47	76
452	0.00	0.00	0.00	44
453	0.00	0.00	0.00	44
454	0.94	0.43	0.59	70
455	0.35	0.06	0.10	155
456	0.36	0.12	0.18	43
457	0.52	0.21	0.30	72
458	0.33	0.10	0.15	62
459	0.60	0.13	0.21	69
460	0.00	0.00	0.00	119
461	0.80	0.15	0.26	79
462	0.69	0.23	0.35	47
463	0.26	0.09	0.13	104
464	0.63	0.36	0.46	106
465	0.50	0.08	0.14	64
466	0.59	0.31	0.41	173
467	0.78	0.36	0.50	107
468	0.84	0.13	0.22	126
469	0.00	0.00	0.00	114
470	0.94	0.78	0.85	140
471	0.00	0.00	0.00	79
472	0.37	0.27	0.31	143
473	0.70	0.32	0.44	158
474	0.30	0.07	0.11	138

```
         475       0.00      0.00      0.00        59
         476       0.60      0.32      0.41        88
         477       0.87      0.59      0.70       176
         478       0.95      0.75      0.84        24
         479       0.27      0.03      0.06        92
         480       0.83      0.43      0.57       100
         481       0.51      0.17      0.26       103
         482       0.42      0.22      0.29        74
         483       0.80      0.54      0.65       105
         484       0.29      0.02      0.04        83
         485       0.17      0.01      0.02        82
         486       0.38      0.11      0.17        71
         487       0.35      0.16      0.22       120
         488       0.00      0.00      0.00       105
         489       0.65      0.28      0.39        87
         490       1.00      0.81      0.90        32
         491       0.00      0.00      0.00        69
         492       0.25      0.02      0.04        49
         493       0.00      0.00      0.00       117
         494       0.48      0.20      0.28        61
         495       0.99      0.68      0.80       344
         496       0.31      0.15      0.21        52
         497       0.63      0.20      0.30       137
         498       0.28      0.05      0.09        98
         499       0.59      0.16      0.26        79

   micro avg       0.71      0.32      0.44    173812
   macro avg       0.52      0.25      0.33    173812
weighted avg       0.64      0.32      0.41    173812
 samples avg       0.41      0.30      0.33    173812

Time taken to run this cell : 0:18:24.332052
```

Observation:

The model gives a precision of 0.7057, recall of 0.3201 and f1-measure of 0.4404 for micro-averaged F1-score. Though the precision is high but the recall value is less and hence a lower F1-score.

```
import joblib
joblib.dump(classifier, 'lr_with_more_title_weight.pkl')
```

```
['lr_with_more_title_weight.pkl']
```

```
start = datetime.now()
classifier_2 = OneVsRestClassifier(LogisticRegression(penalty='l2'),
n_jobs=-1)
classifier_2.fit(x_train_multilabel, y_train)
predictions_2 = classifier_2.predict(x_test_multilabel)
print("Accuracy :",metrics.accuracy_score(y_test, predictions_2))
```

```
print("Hamming loss ",metrics.hamming_loss(y_test,predictions_2))


precision = precision_score(y_test, predictions_2, average='micro')
recall = recall_score(y_test, predictions_2, average='micro')
f1 = f1_score(y_test, predictions_2, average='micro')

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

precision = precision_score(y_test, predictions_2, average='macro')
recall = recall_score(y_test, predictions_2, average='macro')
f1 = f1_score(y_test, predictions_2, average='macro')

print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

print (metrics.classification_report(y_test, predictions_2))
print("Time taken to run this cell :", datetime.now() - start)

Accuracy : 0.23468
Hamming loss  0.00278914
Micro-average quality numbers
Precision: 0.7300, Recall: 0.3136, F1-measure: 0.4388
Macro-average quality numbers
Precision: 0.5569, Recall: 0.2286, F1-measure: 0.3074
              precision    recall  f1-score   support

           0       0.95      0.69      0.80      5519
           1       0.67      0.31      0.42      8190
           2       0.80      0.38      0.52      6529
           3       0.74      0.37      0.49      3231
           4       0.80      0.42      0.55      6430
           5       0.82      0.37      0.51      2879
           6       0.86      0.51      0.64      5086
           7       0.87      0.57      0.69      4533
           8       0.55      0.12      0.20      3000
           9       0.83      0.56      0.67      2765
          10       0.59      0.18      0.27      3051
          11       0.71      0.34      0.46      3009
          12       0.62      0.25      0.35      2630
          13       0.69      0.19      0.29      1426
          14       0.90      0.53      0.67      2548
          15       0.65      0.19      0.29      2371
          16       0.64      0.23      0.34       873
          17       0.89      0.60      0.72      2151
          18       0.62      0.22      0.32      2204
          19       0.72      0.41      0.52       831
```

20	0.76	0.42	0.54	1860
21	0.28	0.08	0.12	2023
22	0.53	0.22	0.31	1513
23	0.91	0.50	0.65	1207
24	0.56	0.27	0.36	506
25	0.52	0.28	0.36	425
26	0.65	0.39	0.49	793
27	0.59	0.33	0.42	1291
28	0.76	0.35	0.47	1208
29	0.43	0.09	0.15	406
30	0.38	0.04	0.07	504
31	0.30	0.08	0.12	732
32	0.60	0.27	0.38	441
33	0.59	0.20	0.29	1645
34	0.72	0.22	0.34	1058
35	0.83	0.53	0.65	946
36	0.67	0.20	0.31	644
37	0.98	0.67	0.79	136
38	0.64	0.34	0.45	570
39	0.83	0.25	0.39	766
40	0.62	0.29	0.39	1132
41	0.51	0.19	0.28	174
42	0.77	0.48	0.59	210
43	0.82	0.40	0.54	433
44	0.67	0.50	0.57	626
45	0.40	0.09	0.15	852
46	0.78	0.44	0.56	534
47	0.43	0.15	0.22	350
48	0.73	0.49	0.59	496
49	0.79	0.61	0.69	785
50	0.23	0.05	0.08	475
51	0.38	0.10	0.16	305
52	0.50	0.02	0.04	251
53	0.68	0.37	0.48	914
54	0.41	0.14	0.21	728
55	0.25	0.00	0.01	258
56	0.48	0.22	0.30	821
57	0.52	0.08	0.14	541
58	0.70	0.13	0.22	748
59	0.95	0.63	0.76	724
60	0.32	0.06	0.10	660
61	0.67	0.12	0.21	235
62	0.92	0.70	0.80	718
63	0.85	0.62	0.71	468
64	0.54	0.30	0.39	191
65	0.34	0.10	0.15	429
66	0.29	0.05	0.09	415
67	0.72	0.47	0.57	274
68	0.83	0.47	0.60	510

69	0.69	0.39	0.50	466
70	0.28	0.06	0.10	305
71	0.53	0.16	0.25	247
72	0.78	0.47	0.59	401
73	0.99	0.77	0.86	86
74	0.73	0.37	0.49	120
75	0.91	0.60	0.73	129
76	0.40	0.00	0.01	473
77	0.46	0.28	0.35	143
78	0.81	0.42	0.55	347
79	0.70	0.20	0.31	479
80	0.58	0.34	0.43	279
81	0.80	0.14	0.24	461
82	0.37	0.02	0.04	298
83	0.77	0.44	0.56	396
84	0.56	0.28	0.37	184
85	0.35	0.06	0.10	573
86	0.50	0.04	0.07	325
87	0.41	0.19	0.26	273
88	0.40	0.19	0.26	135
89	0.35	0.10	0.15	232
90	0.58	0.34	0.43	409
91	0.54	0.17	0.26	420
92	0.78	0.50	0.61	408
93	0.67	0.44	0.53	241
94	0.33	0.04	0.07	211
95	0.39	0.08	0.14	277
96	0.29	0.03	0.06	410
97	0.89	0.25	0.39	501
98	0.79	0.62	0.69	136
99	0.57	0.31	0.40	239
100	0.63	0.12	0.21	324
101	0.95	0.59	0.73	277
102	0.93	0.69	0.79	613
103	0.48	0.15	0.22	157
104	0.25	0.06	0.10	295
105	0.82	0.35	0.49	334
106	0.91	0.13	0.23	335
107	0.77	0.44	0.56	389
108	0.60	0.22	0.32	251
109	0.58	0.38	0.46	317
110	0.67	0.03	0.06	187
111	0.00	0.00	0.00	140
112	0.72	0.35	0.47	154
113	0.57	0.14	0.22	332
114	0.47	0.24	0.32	323
115	0.52	0.21	0.30	344
116	0.75	0.48	0.58	370
117	0.63	0.20	0.31	313

118	0.79	0.59	0.68	874
119	0.50	0.17	0.26	293
120	0.25	0.01	0.01	200
121	0.79	0.45	0.58	463
122	0.43	0.10	0.16	119
123	0.00	0.00	0.00	256
124	0.89	0.64	0.75	195
125	0.41	0.11	0.17	138
126	0.82	0.48	0.61	376
127	0.27	0.03	0.06	122
128	0.17	0.02	0.04	252
129	0.51	0.15	0.23	144
130	0.42	0.07	0.11	150
131	0.36	0.02	0.04	210
132	0.29	0.02	0.05	361
133	0.94	0.49	0.64	453
134	0.90	0.72	0.80	124
135	0.17	0.01	0.02	91
136	0.70	0.24	0.36	128
137	0.59	0.32	0.42	218
138	0.71	0.08	0.15	243
139	0.45	0.19	0.27	149
140	0.69	0.31	0.42	318
141	0.33	0.09	0.14	159
142	0.63	0.35	0.45	274
143	0.86	0.69	0.76	362
144	0.63	0.16	0.26	118
145	0.66	0.35	0.45	164
146	0.58	0.25	0.35	461
147	0.67	0.33	0.44	159
148	0.35	0.11	0.17	166
149	0.99	0.41	0.58	346
150	0.73	0.05	0.10	350
151	0.97	0.67	0.80	55
152	0.82	0.47	0.60	387
153	0.54	0.09	0.16	150
154	0.59	0.07	0.13	281
155	0.25	0.04	0.07	202
156	0.81	0.62	0.70	130
157	0.35	0.04	0.08	245
158	0.92	0.62	0.74	177
159	0.52	0.25	0.34	130
160	0.51	0.13	0.20	336
161	0.90	0.55	0.68	220
162	0.24	0.03	0.06	229
163	0.91	0.39	0.55	316
164	0.79	0.38	0.51	283
165	0.62	0.25	0.36	197
166	0.65	0.30	0.41	101

167	0.49	0.15	0.23	231
168	0.41	0.11	0.17	370
169	0.43	0.16	0.24	258
170	0.38	0.08	0.13	101
171	0.41	0.22	0.29	89
172	0.55	0.32	0.41	193
173	0.42	0.18	0.26	309
174	0.52	0.10	0.17	172
175	0.96	0.67	0.79	95
176	0.95	0.53	0.68	346
177	0.93	0.47	0.62	322
178	0.63	0.42	0.51	232
179	0.36	0.04	0.07	125
180	0.68	0.26	0.38	145
181	0.50	0.06	0.11	77
182	0.11	0.02	0.03	182
183	0.57	0.28	0.38	257
184	0.25	0.02	0.04	216
185	0.32	0.05	0.08	242
186	0.40	0.10	0.16	165
187	0.77	0.52	0.62	263
188	0.45	0.10	0.17	174
189	0.76	0.33	0.46	136
190	0.96	0.45	0.61	202
191	0.46	0.10	0.16	134
192	0.71	0.33	0.45	230
193	0.41	0.14	0.21	90
194	0.59	0.40	0.48	185
195	0.33	0.04	0.07	156
196	0.38	0.04	0.07	160
197	0.24	0.02	0.03	266
198	0.49	0.06	0.11	284
199	0.33	0.03	0.05	145
200	0.96	0.63	0.76	212
201	0.23	0.02	0.03	317
202	0.80	0.50	0.62	427
203	0.37	0.08	0.13	232
204	0.56	0.25	0.35	217
205	0.51	0.32	0.39	527
206	0.22	0.02	0.03	124
207	0.33	0.07	0.11	103
208	0.88	0.44	0.59	287
209	0.36	0.08	0.14	193
210	0.75	0.27	0.40	220
211	0.80	0.03	0.06	140
212	0.11	0.01	0.02	161
213	0.64	0.29	0.40	72
214	0.64	0.39	0.48	396
215	0.91	0.23	0.37	134

216	0.60	0.07	0.13	400
217	0.50	0.20	0.29	75
218	0.97	0.66	0.78	219
219	0.82	0.32	0.46	210
220	0.95	0.53	0.68	298
221	0.97	0.55	0.70	266
222	0.78	0.34	0.47	290
223	0.14	0.01	0.01	128
224	0.73	0.28	0.40	159
225	0.30	0.09	0.13	164
226	0.59	0.28	0.38	144
227	0.58	0.28	0.38	276
228	0.13	0.01	0.02	235
229	0.40	0.01	0.02	216
230	0.35	0.11	0.16	228
231	0.69	0.42	0.52	64
232	0.60	0.06	0.11	103
233	0.72	0.27	0.40	216
234	0.86	0.05	0.10	116
235	0.58	0.32	0.42	77
236	0.95	0.57	0.71	67
237	0.59	0.07	0.13	218
238	0.36	0.07	0.12	139
239	0.50	0.01	0.02	94
240	0.55	0.22	0.31	77
241	0.00	0.00	0.00	167
242	0.85	0.34	0.48	86
243	0.25	0.03	0.06	58
244	0.57	0.17	0.27	269
245	0.30	0.05	0.09	112
246	0.96	0.68	0.80	255
247	0.20	0.02	0.03	58
248	0.00	0.00	0.00	81
249	0.25	0.02	0.03	131
250	0.44	0.16	0.24	93
251	0.75	0.21	0.33	154
252	0.43	0.02	0.04	129
253	0.68	0.34	0.45	83
254	0.39	0.06	0.10	191
255	0.17	0.03	0.05	219
256	0.43	0.05	0.08	130
257	0.52	0.24	0.33	93
258	0.64	0.40	0.49	217
259	0.34	0.09	0.14	141
260	0.80	0.03	0.05	143
261	0.58	0.10	0.16	219
262	0.60	0.25	0.36	107
263	0.45	0.21	0.28	236
264	0.31	0.15	0.20	119

265	0.50	0.12	0.20	72
266	0.00	0.00	0.00	70
267	0.35	0.08	0.14	107
268	0.75	0.41	0.53	169
269	0.27	0.07	0.11	129
270	0.77	0.44	0.56	159
271	0.87	0.41	0.55	190
272	0.32	0.04	0.07	248
273	0.90	0.58	0.70	264
274	0.71	0.23	0.35	105
275	0.75	0.06	0.11	104
276	0.08	0.01	0.02	115
277	0.88	0.55	0.68	170
278	0.69	0.23	0.34	145
279	0.93	0.51	0.66	230
280	0.60	0.36	0.45	80
281	0.67	0.47	0.55	217
282	0.70	0.37	0.48	175
283	0.34	0.06	0.10	269
284	0.72	0.28	0.41	74
285	0.85	0.40	0.54	206
286	0.90	0.53	0.66	227
287	0.91	0.24	0.38	130
288	0.53	0.06	0.11	129
289	0.50	0.01	0.02	80
290	0.38	0.06	0.10	99
291	0.81	0.26	0.40	208
292	0.38	0.04	0.08	67
293	0.90	0.43	0.58	109
294	0.46	0.19	0.26	140
295	0.21	0.08	0.12	241
296	0.32	0.10	0.15	72
297	0.44	0.07	0.11	107
298	0.93	0.43	0.58	61
299	0.97	0.42	0.58	77
300	0.33	0.06	0.11	111
301	0.00	0.00	0.00	126
302	0.00	0.00	0.00	73
303	0.62	0.32	0.43	176
304	0.97	0.65	0.78	230
305	0.96	0.57	0.71	156
306	0.54	0.31	0.39	146
307	0.35	0.08	0.13	98
308	0.00	0.00	0.00	78
309	0.80	0.09	0.15	94
310	0.81	0.28	0.42	162
311	0.83	0.41	0.55	116
312	0.48	0.18	0.26	57
313	0.75	0.05	0.09	65

314	0.52	0.29	0.37	138
315	0.56	0.18	0.27	195
316	0.51	0.26	0.35	69
317	0.48	0.10	0.16	134
318	0.56	0.34	0.42	148
319	0.81	0.35	0.49	161
320	0.22	0.12	0.15	104
321	0.84	0.51	0.63	156
322	0.62	0.34	0.44	134
323	0.58	0.36	0.44	232
324	0.45	0.14	0.21	92
325	0.46	0.18	0.26	197
326	0.22	0.02	0.03	126
327	0.00	0.00	0.00	115
328	0.98	0.49	0.66	198
329	0.61	0.22	0.33	125
330	0.85	0.14	0.23	81
331	0.53	0.09	0.15	94
332	0.67	0.07	0.13	56
333	0.27	0.02	0.04	260
334	0.33	0.03	0.06	60
335	0.44	0.06	0.11	110
336	0.72	0.44	0.54	71
337	0.18	0.03	0.05	66
338	0.47	0.31	0.37	150
339	0.00	0.00	0.00	54
340	0.82	0.45	0.58	195
341	0.85	0.28	0.42	79
342	0.50	0.18	0.27	38
343	0.61	0.33	0.42	43
344	0.52	0.16	0.25	68
345	0.65	0.30	0.41	73
346	0.09	0.01	0.02	116
347	0.89	0.29	0.44	111
348	0.39	0.11	0.17	63
349	0.85	0.51	0.64	104
350	0.57	0.36	0.44	44
351	0.64	0.17	0.27	40
352	0.98	0.33	0.49	136
353	0.41	0.13	0.20	54
354	0.62	0.04	0.07	134
355	0.64	0.30	0.41	120
356	0.47	0.18	0.26	228
357	0.71	0.27	0.39	269
358	0.81	0.31	0.45	80
359	0.84	0.38	0.52	140
360	0.42	0.14	0.21	125
361	0.93	0.53	0.67	169
362	0.12	0.04	0.05	56

363	0.95	0.56	0.70	154
364	0.71	0.09	0.15	58
365	0.21	0.07	0.11	71
366	1.00	0.52	0.68	54
367	0.33	0.04	0.08	116
368	0.00	0.00	0.00	54
369	0.00	0.00	0.00	71
370	0.00	0.00	0.00	61
371	0.43	0.04	0.08	71
372	0.68	0.33	0.44	52
373	0.79	0.30	0.43	150
374	0.38	0.10	0.15	93
375	0.22	0.03	0.05	67
376	0.00	0.00	0.00	76
377	0.67	0.23	0.34	106
378	0.00	0.00	0.00	86
379	0.33	0.07	0.12	14
380	1.00	0.35	0.52	122
381	0.27	0.03	0.05	104
382	0.38	0.09	0.15	66
383	0.53	0.25	0.34	110
384	0.33	0.01	0.01	155
385	0.70	0.14	0.23	50
386	0.24	0.08	0.12	64
387	0.55	0.06	0.12	93
388	0.57	0.23	0.32	102
389	0.00	0.00	0.00	108
390	0.96	0.52	0.68	178
391	0.67	0.14	0.23	115
392	0.93	0.31	0.46	42
393	0.00	0.00	0.00	134
394	0.50	0.03	0.05	112
395	0.00	0.00	0.00	176
396	0.50	0.06	0.10	125
397	0.82	0.28	0.42	224
398	0.93	0.43	0.59	63
399	1.00	0.02	0.03	59
400	0.53	0.27	0.36	63
401	0.48	0.14	0.22	98
402	0.59	0.12	0.20	162
403	0.38	0.12	0.18	83
404	0.88	0.79	0.83	19
405	0.44	0.08	0.13	92
406	0.89	0.20	0.32	41
407	0.65	0.30	0.41	43
408	0.00	0.00	0.00	160
409	0.17	0.08	0.11	50
410	0.00	0.00	0.00	19
411	0.31	0.07	0.11	175

412	0.25	0.03	0.05	72
413	0.57	0.04	0.08	95
414	0.25	0.04	0.07	97
415	0.36	0.08	0.14	48
416	0.48	0.27	0.34	83
417	0.50	0.05	0.09	40
418	0.55	0.13	0.21	91
419	0.49	0.21	0.29	90
420	0.47	0.24	0.32	37
421	0.15	0.03	0.05	66
422	0.64	0.38	0.48	73
423	0.55	0.21	0.31	56
424	0.96	0.79	0.87	33
425	0.00	0.00	0.00	76
426	0.30	0.04	0.07	81
427	0.99	0.51	0.68	150
428	0.95	0.62	0.75	29
429	0.99	0.40	0.57	389
430	0.67	0.26	0.37	167
431	0.56	0.07	0.13	123
432	0.52	0.31	0.39	39
433	0.32	0.13	0.19	82
434	1.00	0.52	0.68	66
435	0.60	0.33	0.43	93
436	0.59	0.22	0.32	87
437	0.10	0.01	0.02	86
438	0.78	0.38	0.51	104
439	0.67	0.10	0.17	100
440	0.00	0.00	0.00	141
441	0.44	0.25	0.32	110
442	0.18	0.04	0.07	123
443	0.50	0.08	0.14	71
444	0.67	0.06	0.10	109
445	0.47	0.19	0.27	48
446	0.40	0.18	0.25	76
447	0.36	0.11	0.16	38
448	0.69	0.47	0.56	81
449	0.43	0.10	0.16	132
450	0.49	0.23	0.32	81
451	0.86	0.25	0.39	76
452	0.00	0.00	0.00	44
453	0.00	0.00	0.00	44
454	0.90	0.40	0.55	70
455	0.54	0.08	0.15	155
456	0.60	0.14	0.23	43
457	0.59	0.18	0.28	72
458	0.55	0.10	0.16	62
459	0.76	0.19	0.30	69
460	0.00	0.00	0.00	119

461	0.75	0.15	0.25	79
462	0.62	0.17	0.27	47
463	0.38	0.05	0.09	104
464	0.67	0.29	0.41	106
465	0.92	0.17	0.29	64
466	0.57	0.21	0.31	173
467	0.81	0.27	0.41	107
468	0.78	0.11	0.19	126
469	0.00	0.00	0.00	114
470	0.95	0.74	0.83	140
471	0.00	0.00	0.00	79
472	0.41	0.24	0.30	143
473	0.76	0.26	0.39	158
474	0.38	0.04	0.07	138
475	0.00	0.00	0.00	59
476	0.55	0.18	0.27	88
477	0.88	0.49	0.63	176
478	1.00	0.75	0.86	24
479	0.14	0.01	0.02	92
480	0.86	0.38	0.53	100
481	0.57	0.20	0.30	103
482	0.41	0.16	0.23	74
483	0.83	0.42	0.56	105
484	0.00	0.00	0.00	83
485	0.00	0.00	0.00	82
486	0.47	0.10	0.16	71
487	0.36	0.11	0.17	120
488	0.00	0.00	0.00	105
489	0.76	0.25	0.38	87
490	1.00	0.69	0.81	32
491	0.00	0.00	0.00	69
492	0.50	0.02	0.04	49
493	0.00	0.00	0.00	117
494	0.44	0.11	0.18	61
495	0.99	0.44	0.60	344
496	0.30	0.12	0.17	52
497	0.62	0.18	0.27	137
498	0.25	0.03	0.05	98
499	0.85	0.14	0.24	79
micro avg	0.73	0.31	0.44	173812
macro avg	0.56	0.23	0.31	173812
weighted avg	0.66	0.31	0.41	173812
samples avg	0.41	0.30	0.32	173812

```
Time taken to run this cell : 1:14:16.939599
```

5.1 Using Bag of Words upto 4 grams

```
# we will be taking 500 tags
multilabel_yx = tags_to_choose(500)
print("number of questions that are not covered :",
questions_explained_fn(500),"out of ", total_qs)

x_train=preprocessed_data.head(train_datasize)
x_test=preprocessed_data.tail(preprocessed_data.shape[0] - 400000)

y_train = multilabel_yx[0:train_datasize,:]
y_test = multilabel_yx[train_datasize:preprocessed_data.shape[0],:]

print("Number of data points in train data :", y_train.shape)
print("Number of data points in test data :", y_test.shape)
```

```
number of questions that are not covered : 98536 out of  999999
Number of data points in train data : (400000, 500)
Number of data points in test data : (599999, 500)
```

```
start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, max_features=100000,
analyzer='word', tokenizer = lambda x: x.split(), ngram_range=(1,4))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel= vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:06:04.141498
```

```
print("Dimensions of train data X:",x_train_multilabel.shape,
"Y :",y_train.shape)
print("Dimensions of test data
X:",x_test_multilabel.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (400000, 90222) Y : (400000, 500)
Dimensions of test data X: (599999, 90222) Y: (599999, 500)
```

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

start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log',
alpha=0.00001, penalty='l1'), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


print("Accuracy :",metrics.accuracy_score(y_test, predictions))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions))


precision = precision_score(y_test, predictions, average='micro')
```

```
recall = recall_score(y_test, predictions, average='micro')
f1 = f1_score(y_test, predictions, average='micro')

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')

print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

print (metrics.classification_report(y_test, predictions))
print("Time taken to run this cell :", datetime.now() - start)
```

```
Accuracy : 0.12515187525312543
Hamming loss  0.00441820736367894
Micro-average quality numbers
Precision: 0.4000, Recall: 0.4472, F1-measure: 0.4223
Macro-average quality numbers
Precision: 0.2958, Recall: 0.3773, F1-measure: 0.3294
              precision    recall  f1-score   support

           0       0.44      0.42      0.43     47557
           1       0.56      0.54      0.55     42709
           2       0.64      0.62      0.63     40596
           3       0.54      0.50      0.52     37872
           4       0.84      0.83      0.84     33658
           5       0.71      0.66      0.68     31574
           6       0.49      0.46      0.47     20635
           7       0.72      0.69      0.70     19120
           8       0.52      0.50      0.51     18338
           9       0.53      0.51      0.52     17947
          10       0.67      0.64      0.66     17320
          11       0.31      0.28      0.30     17137
          12       0.24      0.21      0.22     16416
          13       0.41      0.39      0.40     14502
          14       0.39      0.35      0.37     13595
          15       0.38      0.37      0.38     13517
          16       0.61      0.57      0.59     13434
          17       0.61      0.60      0.61     11833
          18       0.38      0.34      0.36     11121
          19       0.36      0.36      0.36      9672
          20       0.21      0.20      0.20      8837
          21       0.48      0.48      0.48      7501
          22       0.41      0.37      0.39      7320
          23       0.69      0.73      0.71      6737
```

24	0.46	0.45	0.46	6484
25	0.43	0.44	0.43	6430
26	0.18	0.17	0.17	6134
27	0.39	0.39	0.39	6068
28	0.66	0.73	0.70	6032
29	0.29	0.28	0.29	5477
30	0.34	0.36	0.35	5352
31	0.76	0.82	0.79	5277
32	0.39	0.39	0.39	4902
33	0.41	0.44	0.43	4885
34	0.32	0.35	0.33	4722
35	0.61	0.65	0.63	4580
36	0.51	0.59	0.55	4512
37	0.54	0.53	0.54	4547
38	0.26	0.26	0.26	4376
39	0.23	0.23	0.23	3918
40	0.22	0.23	0.22	3852
41	0.40	0.51	0.45	3801
42	0.37	0.41	0.39	3793
43	0.38	0.41	0.39	3604
44	0.26	0.27	0.26	3508
45	0.18	0.21	0.19	3432
46	0.18	0.17	0.18	3463
47	0.26	0.28	0.27	3293
48	0.24	0.26	0.25	3273
49	0.42	0.50	0.45	3230
50	0.15	0.14	0.15	3299
51	0.50	0.53	0.52	3173
52	0.29	0.33	0.31	3175
53	0.24	0.27	0.26	3137
54	0.65	0.77	0.71	3159
55	0.20	0.18	0.19	3107
56	0.37	0.41	0.39	3028
57	0.52	0.60	0.56	3022
58	0.17	0.20	0.18	2993
59	0.50	0.56	0.53	2976
60	0.12	0.12	0.12	2947
61	0.70	0.84	0.76	2937
62	0.20	0.22	0.21	2897
63	0.65	0.74	0.69	2937
64	0.49	0.62	0.55	2822
65	0.39	0.47	0.43	2858
66	0.34	0.39	0.36	2771
67	0.31	0.35	0.33	2721
68	0.41	0.49	0.44	2722
69	0.24	0.27	0.26	2700
70	0.55	0.62	0.59	2746
71	0.50	0.57	0.53	2671
72	0.10	0.08	0.09	2667

73	0.37	0.38	0.37	2651
74	0.40	0.44	0.42	2561
75	0.34	0.40	0.37	2545
76	0.65	0.77	0.70	2510
77	0.44	0.54	0.48	2421
78	0.09	0.10	0.09	2350
79	0.17	0.20	0.18	2260
80	0.39	0.45	0.42	2268
81	0.33	0.40	0.36	2234
82	0.22	0.22	0.22	2245
83	0.29	0.32	0.30	2191
84	0.51	0.65	0.58	2130
85	0.44	0.52	0.48	2148
86	0.52	0.61	0.56	2130
87	0.29	0.35	0.32	2186
88	0.61	0.70	0.65	2094
89	0.64	0.76	0.69	2121
90	0.18	0.27	0.22	2010
91	0.58	0.69	0.63	2072
92	0.42	0.52	0.46	1997
93	0.41	0.50	0.45	2020
94	0.40	0.47	0.43	1976
95	0.20	0.19	0.19	1953
96	0.21	0.26	0.23	1973
97	0.68	0.79	0.73	1952
98	0.13	0.14	0.14	1969
99	0.66	0.78	0.71	1912
100	0.24	0.29	0.27	1891
101	0.24	0.25	0.24	1884
102	0.64	0.78	0.70	1899
103	0.72	0.83	0.77	1873
104	0.52	0.69	0.59	1888
105	0.12	0.13	0.12	1827
106	0.38	0.53	0.44	1789
107	0.22	0.27	0.24	1806
108	0.34	0.47	0.39	1753
109	0.11	0.13	0.11	1713
110	0.15	0.22	0.18	1785
111	0.42	0.54	0.47	1740
112	0.32	0.32	0.32	1784
113	0.26	0.31	0.28	1704
114	0.65	0.79	0.71	1618
115	0.39	0.49	0.43	1673
116	0.23	0.29	0.26	1622
117	0.43	0.50	0.46	1662
118	0.18	0.25	0.21	1596
119	0.17	0.17	0.17	1630
120	0.26	0.30	0.28	1644
121	0.55	0.67	0.60	1598

122	0.19	0.28	0.23	1605
123	0.14	0.18	0.16	1633
124	0.41	0.52	0.46	1607
125	0.34	0.41	0.37	1627
126	0.78	0.90	0.84	1538
127	0.09	0.13	0.11	1508
128	0.66	0.78	0.71	1555
129	0.24	0.32	0.27	1583
130	0.19	0.21	0.20	1566
131	0.15	0.20	0.17	1513
132	0.33	0.45	0.38	1502
133	0.30	0.37	0.33	1522
134	0.38	0.52	0.44	1514
135	0.39	0.51	0.44	1484
136	0.44	0.59	0.50	1555
137	0.11	0.13	0.12	1492
138	0.10	0.12	0.11	1477
139	0.33	0.42	0.37	1507
140	0.01	0.01	0.01	1519
141	0.15	0.21	0.17	1451
142	0.68	0.77	0.72	1481
143	0.14	0.17	0.15	1482
144	0.34	0.40	0.36	1496
145	0.22	0.32	0.26	1428
146	0.68	0.81	0.74	1438
147	0.17	0.23	0.20	1467
148	0.32	0.42	0.36	1419
149	0.11	0.14	0.12	1464
150	0.19	0.18	0.18	1436
151	0.45	0.53	0.49	1479
152	0.16	0.20	0.17	1447
153	0.09	0.09	0.09	1409
154	0.44	0.60	0.51	1379
155	0.13	0.20	0.16	1375
156	0.21	0.26	0.23	1417
157	0.15	0.22	0.18	1373
158	0.15	0.21	0.17	1367
159	0.20	0.23	0.22	1364
160	0.30	0.38	0.33	1381
161	0.12	0.15	0.13	1330
162	0.33	0.39	0.36	1356
163	0.43	0.58	0.49	1346
164	0.75	0.85	0.80	1359
165	0.21	0.35	0.26	1343
166	0.13	0.22	0.16	1323
167	0.22	0.31	0.26	1327
168	0.24	0.32	0.28	1317
169	0.63	0.80	0.70	1328
170	0.29	0.38	0.32	1337

171	0.39	0.56	0.46	1338
172	0.43	0.68	0.53	1318
173	0.55	0.69	0.61	1312
174	0.72	0.83	0.77	1302
175	0.10	0.16	0.13	1230
176	0.31	0.40	0.35	1296
177	0.14	0.23	0.17	1281
178	0.64	0.82	0.72	1271
179	0.45	0.56	0.50	1298
180	0.36	0.51	0.42	1289
181	0.07	0.10	0.08	1276
182	0.19	0.24	0.21	1215
183	0.09	0.08	0.08	1244
184	0.38	0.51	0.44	1261
185	0.51	0.66	0.57	1217
186	0.17	0.25	0.20	1255
187	0.70	0.80	0.75	1224
188	0.25	0.32	0.28	1227
189	0.15	0.17	0.16	1216
190	0.28	0.35	0.31	1188
191	0.09	0.08	0.08	1172
192	0.22	0.28	0.25	1237
193	0.14	0.18	0.16	1171
194	0.21	0.30	0.25	1173
195	0.54	0.70	0.61	1222
196	0.21	0.29	0.24	1167
197	0.36	0.47	0.41	1192
198	0.61	0.74	0.67	1193
199	0.18	0.26	0.22	1180
200	0.21	0.21	0.21	1154
201	0.11	0.16	0.13	1179
202	0.58	0.76	0.66	1141
203	0.66	0.85	0.74	1151
204	0.42	0.56	0.48	1144
205	0.42	0.58	0.49	1144
206	0.35	0.47	0.40	1165
207	0.07	0.10	0.08	1140
208	0.35	0.45	0.39	1119
209	0.14	0.17	0.15	1068
210	0.14	0.14	0.14	1124
211	0.31	0.43	0.36	1107
212	0.69	0.83	0.75	1104
213	0.04	0.06	0.05	1076
214	0.09	0.14	0.11	1099
215	0.39	0.57	0.47	1051
216	0.35	0.46	0.39	1086
217	0.38	0.47	0.42	1080
218	0.13	0.12	0.12	1087
219	0.23	0.34	0.27	1035

220	0.31	0.48	0.38	1037
221	0.44	0.60	0.51	1030
222	0.70	0.83	0.76	1044
223	0.17	0.21	0.19	1046
224	0.34	0.48	0.40	1028
225	0.14	0.20	0.17	1011
226	0.64	0.80	0.71	1049
227	0.42	0.54	0.47	1016
228	0.17	0.28	0.21	1020
229	0.26	0.37	0.31	988
230	0.22	0.34	0.27	985
231	0.09	0.11	0.10	998
232	0.17	0.23	0.20	974
233	0.06	0.05	0.06	982
234	0.06	0.09	0.07	980
235	0.38	0.51	0.44	985
236	0.17	0.26	0.20	957
237	0.12	0.17	0.14	962
238	0.27	0.39	0.32	945
239	0.33	0.47	0.39	947
240	0.12	0.10	0.11	970
241	0.12	0.17	0.14	936
242	0.17	0.22	0.19	953
243	0.48	0.63	0.54	953
244	0.51	0.68	0.58	905
245	0.32	0.45	0.37	965
246	0.42	0.65	0.51	938
247	0.22	0.29	0.25	917
248	0.27	0.40	0.32	938
249	0.35	0.50	0.41	920
250	0.47	0.66	0.55	910
251	0.34	0.53	0.41	902
252	0.22	0.34	0.27	884
253	0.07	0.09	0.08	910
254	0.34	0.50	0.41	916
255	0.28	0.41	0.33	902
256	0.24	0.37	0.29	892
257	0.22	0.26	0.24	899
258	0.11	0.16	0.13	875
259	0.34	0.43	0.38	869
260	0.18	0.25	0.21	869
261	0.06	0.07	0.06	873
262	0.15	0.21	0.18	914
263	0.35	0.46	0.40	869
264	0.46	0.58	0.52	878
265	0.65	0.75	0.70	892
266	0.02	0.02	0.02	850
267	0.31	0.48	0.38	862
268	0.10	0.18	0.12	851

269	0.51	0.66	0.58	823
270	0.31	0.43	0.36	852
271	0.08	0.08	0.08	835
272	0.11	0.21	0.14	824
273	0.04	0.06	0.05	833
274	0.08	0.09	0.08	831
275	0.22	0.35	0.27	831
276	0.30	0.45	0.36	829
277	0.52	0.70	0.60	788
278	0.66	0.81	0.73	822
279	0.26	0.35	0.30	827
280	0.39	0.60	0.47	796
281	0.44	0.54	0.49	818
282	0.32	0.49	0.38	771
283	0.73	0.85	0.79	801
284	0.07	0.06	0.07	843
285	0.20	0.27	0.23	823
286	0.10	0.14	0.12	803
287	0.19	0.16	0.17	796
288	0.18	0.21	0.19	768
289	0.55	0.69	0.61	794
290	0.15	0.24	0.18	761
291	0.06	0.06	0.06	784
292	0.06	0.09	0.07	761
293	0.26	0.41	0.32	772
294	0.14	0.22	0.17	770
295	0.07	0.08	0.08	787
296	0.13	0.14	0.13	776
297	0.02	0.02	0.02	767
298	0.25	0.38	0.30	800
299	0.23	0.36	0.28	772
300	0.14	0.21	0.17	761
301	0.12	0.18	0.15	749
302	0.26	0.40	0.31	761
303	0.15	0.26	0.19	763
304	0.08	0.10	0.09	776
305	0.10	0.11	0.11	738
306	0.24	0.36	0.29	733
307	0.50	0.71	0.59	738
308	0.12	0.19	0.15	738
309	0.68	0.85	0.76	751
310	0.54	0.72	0.62	716
311	0.20	0.32	0.25	744
312	0.21	0.34	0.26	749
313	0.24	0.35	0.28	733
314	0.13	0.18	0.15	761
315	0.29	0.45	0.35	715
316	0.16	0.27	0.20	740
317	0.40	0.55	0.46	725

318	0.20	0.26	0.23	722
319	0.35	0.53	0.42	733
320	0.43	0.61	0.51	732
321	0.08	0.08	0.08	744
322	0.08	0.10	0.09	712
323	0.18	0.11	0.14	713
324	0.57	0.66	0.61	737
325	0.03	0.03	0.03	706
326	0.13	0.23	0.16	724
327	0.03	0.04	0.04	702
328	0.28	0.45	0.34	693
329	0.14	0.21	0.17	687
330	0.20	0.29	0.24	662
331	0.22	0.30	0.25	708
332	0.29	0.40	0.33	698
333	0.28	0.36	0.31	725
334	0.27	0.38	0.32	686
335	0.33	0.46	0.39	687
336	0.13	0.18	0.15	700
337	0.12	0.21	0.15	640
338	0.09	0.11	0.10	710
339	0.11	0.15	0.12	701
340	0.11	0.19	0.14	646
341	0.07	0.07	0.07	667
342	0.23	0.39	0.29	665
343	0.29	0.50	0.36	655
344	0.23	0.31	0.26	676
345	0.32	0.51	0.39	643
346	0.27	0.41	0.33	661
347	0.07	0.12	0.09	679
348	0.14	0.23	0.17	620
349	0.28	0.45	0.34	627
350	0.12	0.13	0.13	690
351	0.06	0.08	0.06	653
352	0.21	0.44	0.28	595
353	0.17	0.28	0.21	640
354	0.49	0.60	0.54	651
355	0.31	0.50	0.39	650
356	0.20	0.33	0.25	616
357	0.18	0.23	0.21	663
358	0.17	0.31	0.22	639
359	0.15	0.25	0.19	644
360	0.18	0.30	0.22	660
361	0.47	0.60	0.53	637
362	0.13	0.22	0.16	616
363	0.17	0.28	0.21	630
364	0.09	0.15	0.11	614
365	0.60	0.79	0.68	627
366	0.10	0.14	0.12	645

367	0.30	0.41	0.34	588
368	0.11	0.16	0.13	594
369	0.12	0.21	0.15	579
370	0.17	0.28	0.21	621
371	0.18	0.22	0.20	653
372	0.26	0.41	0.32	615
373	0.09	0.15	0.11	595
374	0.04	0.09	0.06	602
375	0.55	0.69	0.61	600
376	0.45	0.60	0.51	620
377	0.71	0.87	0.78	621
378	0.08	0.06	0.06	600
379	0.14	0.26	0.18	591
380	0.19	0.26	0.22	613
381	0.06	0.10	0.08	628
382	0.26	0.40	0.31	596
383	0.08	0.13	0.10	589
384	0.50	0.66	0.57	612
385	0.34	0.59	0.43	602
386	0.08	0.19	0.12	589
387	0.59	0.68	0.63	617
388	0.41	0.64	0.50	585
389	0.30	0.44	0.36	609
390	0.48	0.59	0.53	606
391	0.10	0.15	0.12	599
392	0.06	0.08	0.07	562
393	0.68	0.80	0.73	618
394	0.37	0.59	0.45	592
395	0.12	0.20	0.15	604
396	0.71	0.84	0.77	598
397	0.42	0.54	0.47	605
398	0.30	0.20	0.24	610
399	0.21	0.39	0.28	583
400	0.29	0.44	0.35	592
401	0.19	0.31	0.23	589
402	0.49	0.65	0.56	595
403	0.10	0.14	0.11	573
404	0.37	0.53	0.43	616
405	0.09	0.12	0.10	573
406	0.11	0.13	0.12	573
407	0.28	0.42	0.33	573
408	0.19	0.24	0.21	600
409	0.07	0.15	0.09	583
410	0.21	0.25	0.23	578
411	0.17	0.22	0.19	570
412	0.44	0.53	0.48	609
413	0.03	0.03	0.03	597
414	0.40	0.66	0.50	571
415	0.06	0.12	0.08	551

416	0.12	0.18	0.15	594
417	0.12	0.20	0.15	558
418	0.07	0.10	0.08	567
419	0.18	0.25	0.21	587
420	0.13	0.20	0.16	545
421	0.07	0.14	0.10	561
422	0.32	0.51	0.40	558
423	0.42	0.61	0.50	562
424	0.28	0.36	0.31	539
425	0.63	0.74	0.68	570
426	0.42	0.65	0.51	562
427	0.23	0.43	0.30	559
428	0.18	0.32	0.23	549
429	0.29	0.47	0.36	565
430	0.18	0.27	0.22	563
431	0.06	0.10	0.07	527
432	0.20	0.29	0.24	546
433	0.21	0.29	0.25	565
434	0.59	0.69	0.64	545
435	0.15	0.22	0.18	573
436	0.09	0.16	0.11	543
437	0.11	0.19	0.14	549
438	0.18	0.29	0.22	534
439	0.13	0.23	0.17	538
440	0.05	0.07	0.06	536
441	0.52	0.68	0.59	532
442	0.02	0.03	0.03	553
443	0.50	0.61	0.55	579
444	0.15	0.20	0.17	523
445	0.25	0.36	0.29	567
446	0.22	0.32	0.26	531
447	0.26	0.42	0.32	526
448	0.23	0.39	0.29	516
449	0.08	0.13	0.10	528
450	0.18	0.27	0.21	528
451	0.62	0.72	0.67	512
452	0.62	0.81	0.70	550
453	0.61	0.80	0.69	527
454	0.20	0.31	0.24	521
455	0.24	0.31	0.27	529
456	0.12	0.21	0.15	510
457	0.18	0.30	0.22	513
458	0.18	0.26	0.22	564
459	0.04	0.05	0.05	526
460	0.07	0.07	0.07	526
461	0.02	0.03	0.02	506
462	0.37	0.56	0.45	523
463	0.15	0.27	0.20	496
464	0.49	0.59	0.53	546

```
         465       0.09      0.16      0.12       541
         466       0.56      0.75      0.64       536
         467       0.32      0.46      0.38       479
         468       0.19      0.31      0.24       534
         469       0.29      0.34      0.31       546
         470       0.12      0.17      0.14       509
         471       0.22      0.41      0.29       520
         472       0.14      0.18      0.16       509
         473       0.28      0.41      0.33       523
         474       0.15      0.23      0.18       538
         475       0.25      0.23      0.24       504
         476       0.19      0.34      0.24       526
         477       0.44      0.64      0.52       509
         478       0.48      0.67      0.56       520
         479       0.39      0.54      0.45       524
         480       0.17      0.27      0.21       498
         481       0.26      0.35      0.30       534
         482       0.50      0.53      0.51       508
         483       0.17      0.25      0.20       519
         484       0.58      0.71      0.64       523
         485       0.10      0.16      0.12       508
         486       0.21      0.35      0.26       505
         487       0.10      0.11      0.10       512
         488       0.06      0.12      0.08       497
         489       0.16      0.32      0.21       521
         490       0.35      0.41      0.37       512
         491       0.14      0.24      0.17       509
         492       0.05      0.09      0.07       502
         493       0.78      0.85      0.82       489
         494       0.25      0.37      0.29       496
         495       0.07      0.13      0.09       496
         496       0.27      0.50      0.35       503
         497       0.26      0.35      0.30       507
         498       0.41      0.64      0.50       486
         499       0.11      0.20      0.14       497

   micro avg       0.40      0.45      0.42   1083243
   macro avg       0.30      0.38      0.33   1083243
weighted avg       0.42      0.45      0.43   1083243
 samples avg       0.40      0.43      0.38   1083243

Time taken to run this cell : 0:48:38.197112
```

```
import joblib
joblib.dump(classifier, 'lr_with_more_title_weight_4gram.pkl')
```

```
['lr_with_more_title_weight_4gram.pkl']
```

5.2 Hyperparameter tuning using GridSearch

```
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")

start = datetime.now()
param_grid = {
    'estimator__loss': ['log'],
    'estimator__alpha': [0.0001, 0.001, 0.01, 1],
    'estimator__penalty': ['l1']}

classifier = OneVsRestClassifier(SGDClassifier(random_state=21))
grid = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=3,
scoring='f1_micro', verbose=3)
grid.fit(x_train_multilabel, y_train)
predictions = grid.predict (x_test_multilabel)


print("Accuracy :",metrics.accuracy_score(y_test, predictions))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions))


precision = precision_score(y_test, predictions, average='micro')
recall = recall_score(y_test, predictions, average='micro')
f1 = f1_score(y_test, predictions, average='micro')

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')

print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

print (metrics.classification_report(y_test, predictions))
print("Time taken to run this cell :", datetime.now() - start)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV 1/3] END estimator__alpha=0.0001, estimator__loss=log,
estimator__penalty=l1;, score=0.458 total time=34.7min
[CV 2/3] END estimator__alpha=0.0001, estimator__loss=log,
estimator__penalty=l1;, score=0.455 total time=30.0min
[CV 3/3] END estimator__alpha=0.0001, estimator__loss=log,
estimator__penalty=l1;, score=0.455 total time=31.5min
[CV 1/3] END estimator__alpha=0.001, estimator__loss=log,
estimator__penalty=l1;, score=0.376 total time=15.1min
```

```
[CV 2/3] END estimator__alpha=0.001, estimator__loss=log,
estimator__penalty=l1;, score=0.377 total time=16.1min
[CV 3/3] END estimator__alpha=0.001, estimator__loss=log,
estimator__penalty=l1;, score=0.376 total time=16.4min
[CV 1/3] END estimator__alpha=0.01, estimator__loss=log,
estimator__penalty=l1;, score=0.160 total time=14.8min
[CV 2/3] END estimator__alpha=0.01, estimator__loss=log,
estimator__penalty=l1;, score=0.163 total time=13.9min
[CV 3/3] END estimator__alpha=0.01, estimator__loss=log,
estimator__penalty=l1;, score=0.165 total time=14.5min
[CV 1/3] END estimator__alpha=1, estimator__loss=log,
estimator__penalty=l1;, score=0.000 total time=13.9min
[CV 2/3] END estimator__alpha=1, estimator__loss=log,
estimator__penalty=l1;, score=0.000 total time=14.1min
[CV 3/3] END estimator__alpha=1, estimator__loss=log,
estimator__penalty=l1;, score=0.000 total time=14.0min
Accuracy : 0.19106365177275295
Hamming loss  0.0032512420854034756
Micro-average quality numbers
Precision: 0.5733, Recall: 0.3821, F1-measure: 0.4585
Macro-average quality numbers
Precision: 0.4173, Recall: 0.3142, F1-measure: 0.3443
              precision    recall  f1-score   support

           0       0.55      0.24      0.33     47519
           1       0.72      0.46      0.56     42622
           2       0.75      0.54      0.63     40419
           3       0.69      0.42      0.52     38060
           4       0.87      0.81      0.84     33345
           5       0.80      0.66      0.72     31858
           6       0.58      0.36      0.45     20316
           7       0.80      0.67      0.73     19101
           8       0.66      0.40      0.50     18173
           9       0.67      0.47      0.55     18161
          10       0.72      0.64      0.68     17620
          11       0.41      0.16      0.23     17096
          12       0.37      0.11      0.17     16461
          13       0.55      0.29      0.38     14389
          14       0.52      0.26      0.34     13676
          15       0.48      0.30      0.37     13625
          16       0.71      0.50      0.59     13316
          17       0.73      0.59      0.65     11794
          18       0.52      0.27      0.36     11205
          19       0.42      0.21      0.28      9992
          20       0.23      0.07      0.11      8838
          21       0.72      0.40      0.51      7526
          22       0.51      0.34      0.41      7297
          23       0.76      0.73      0.74      6614
          24       0.56      0.46      0.50      6485
```

25	0.61	0.41	0.49	6390
26	0.77	0.74	0.75	6028
27	0.27	0.06	0.10	6108
28	0.48	0.37	0.42	5966
29	0.50	0.30	0.37	5484
30	0.38	0.31	0.34	5279
31	0.81	0.86	0.84	5248
32	0.44	0.36	0.40	4854
33	0.74	0.38	0.51	4795
34	0.42	0.18	0.25	4743
35	0.65	0.50	0.56	4578
36	0.57	0.61	0.59	4627
37	0.74	0.61	0.67	4526
38	0.33	0.12	0.18	4401
39	0.35	0.10	0.16	3911
40	0.37	0.10	0.16	3832
41	0.39	0.45	0.42	3714
42	0.59	0.35	0.44	3756
43	0.45	0.44	0.44	3574
44	0.33	0.11	0.17	3578
45	0.38	0.12	0.18	3516
46	0.25	0.09	0.14	3320
47	0.51	0.13	0.20	3278
48	0.37	0.17	0.23	3255
49	0.34	0.05	0.09	3196
50	0.25	0.23	0.24	3169
51	0.53	0.50	0.51	3116
52	0.61	0.52	0.56	3132
53	0.33	0.25	0.29	3119
54	0.73	0.80	0.76	3085
55	0.31	0.10	0.15	3058
56	0.38	0.14	0.20	3015
57	0.70	0.58	0.64	3012
58	0.43	0.41	0.42	3014
59	0.13	0.09	0.10	2993
60	0.79	0.81	0.80	2964
61	0.71	0.50	0.59	2965
62	0.25	0.09	0.14	2905
63	0.65	0.61	0.63	2876
64	0.62	0.32	0.43	2876
65	0.83	0.74	0.78	2885
66	0.35	0.18	0.24	2778
67	0.62	0.49	0.55	2711
68	0.74	0.58	0.65	2765
69	0.63	0.30	0.41	2661
70	0.50	0.28	0.36	2715
71	0.21	0.03	0.06	2592
72	0.66	0.32	0.43	2552
73	0.49	0.45	0.47	2592

74	0.40	0.33	0.36	2571
75	0.77	0.78	0.78	2510
76	0.46	0.47	0.46	2481
77	0.09	0.02	0.03	2435
78	0.55	0.55	0.55	2455
79	0.26	0.14	0.18	2227
80	0.63	0.44	0.51	2243
81	0.38	0.27	0.31	2206
82	0.26	0.10	0.15	2177
83	0.86	0.71	0.78	2155
84	0.73	0.64	0.68	2183
85	0.47	0.26	0.33	2182
86	0.43	0.36	0.39	2166
87	0.80	0.55	0.65	2130
88	0.83	0.68	0.75	2192
89	0.49	0.60	0.54	2155
90	0.68	0.75	0.72	2121
91	0.23	0.13	0.17	2045
92	0.56	0.53	0.54	2045
93	0.53	0.50	0.51	1969
94	0.24	0.11	0.15	1904
95	0.37	0.49	0.42	1931
96	0.14	0.03	0.06	1932
97	0.42	0.18	0.25	1921
98	0.82	0.83	0.82	1886
99	0.61	0.68	0.64	1945
100	0.44	0.20	0.28	1895
101	0.76	0.80	0.78	1871
102	0.13	0.05	0.07	1895
103	0.40	0.22	0.28	1902
104	0.82	0.85	0.83	1832
105	0.71	0.73	0.72	1854
106	0.23	0.15	0.19	1753
107	0.32	0.19	0.24	1853
108	0.20	0.03	0.05	1752
109	0.58	0.54	0.56	1719
110	0.68	0.43	0.53	1763
111	0.52	0.43	0.47	1758
112	0.36	0.31	0.34	1741
113	0.49	0.50	0.50	1642
114	0.83	0.82	0.83	1671
115	0.43	0.20	0.27	1682
116	0.35	0.27	0.31	1682
117	0.52	0.14	0.22	1624
118	0.34	0.19	0.24	1626
119	0.30	0.07	0.11	1632
120	0.34	0.24	0.28	1609
121	0.20	0.13	0.16	1603
122	0.51	0.29	0.37	1566

123	0.88	0.76	0.81	1594
124	0.72	0.71	0.72	1579
125	0.64	0.48	0.55	1619
126	0.28	0.08	0.13	1563
127	0.39	0.30	0.34	1552
128	0.42	0.32	0.36	1540
129	0.72	0.90	0.80	1537
130	0.26	0.15	0.19	1525
131	0.17	0.10	0.13	1521
132	0.62	0.38	0.47	1547
133	0.13	0.07	0.09	1543
134	0.37	0.09	0.15	1509
135	0.00	0.00	0.00	1537
136	0.85	0.85	0.85	1506
137	0.49	0.47	0.48	1498
138	0.61	0.57	0.59	1514
139	0.66	0.41	0.51	1508
140	0.46	0.45	0.46	1476
141	0.50	0.37	0.43	1502
142	0.21	0.04	0.07	1522
143	0.76	0.82	0.79	1485
144	0.70	0.51	0.59	1477
145	0.24	0.12	0.16	1469
146	0.46	0.25	0.33	1467
147	0.39	0.24	0.30	1465
148	0.21	0.20	0.20	1434
149	0.45	0.11	0.18	1460
150	0.07	0.07	0.07	1391
151	0.47	0.36	0.41	1444
152	0.31	0.12	0.17	1453
153	0.14	0.11	0.12	1439
154	0.40	0.32	0.36	1432
155	0.37	0.35	0.36	1434
156	0.78	0.85	0.81	1399
157	0.19	0.04	0.07	1376
158	0.24	0.15	0.18	1382
159	0.62	0.46	0.53	1389
160	0.35	0.15	0.21	1380
161	0.47	0.48	0.47	1384
162	0.37	0.23	0.29	1355
163	0.33	0.14	0.19	1365
164	0.87	0.87	0.87	1390
165	0.15	0.18	0.17	1367
166	0.42	0.13	0.20	1317
167	0.38	0.15	0.22	1281
168	0.17	0.04	0.06	1303
169	0.36	0.27	0.31	1329
170	0.81	0.82	0.82	1293
171	0.65	0.41	0.50	1328

172	0.49	0.28	0.36	1282
173	0.58	0.55	0.57	1322
174	0.78	0.83	0.80	1289
175	0.39	0.23	0.29	1284
176	0.65	0.76	0.70	1304
177	0.25	0.19	0.22	1335
178	0.51	0.53	0.52	1281
179	0.56	0.50	0.53	1217
180	0.66	0.34	0.45	1270
181	0.37	0.25	0.30	1245
182	0.31	0.18	0.23	1261
183	0.30	0.24	0.27	1246
184	0.57	0.55	0.56	1209
185	0.29	0.08	0.12	1235
186	0.23	0.02	0.03	1206
187	0.82	0.84	0.83	1225
188	0.23	0.03	0.05	1214
189	0.09	0.02	0.03	1210
190	0.14	0.19	0.16	1212
191	0.43	0.09	0.15	1185
192	0.92	0.88	0.90	1195
193	0.54	0.51	0.53	1209
194	0.35	0.14	0.20	1184
195	0.42	0.21	0.28	1158
196	0.42	0.25	0.31	1147
197	0.48	0.48	0.48	1138
198	0.28	0.25	0.26	1156
199	0.77	0.70	0.73	1136
200	0.75	0.75	0.75	1152
201	0.45	0.36	0.40	1107
202	0.27	0.27	0.27	1142
203	0.65	0.55	0.60	1123
204	0.48	0.43	0.46	1108
205	0.10	0.04	0.06	1108
206	0.17	0.07	0.10	1138
207	0.26	0.03	0.06	1161
208	0.86	0.70	0.77	1138
209	0.47	0.51	0.49	1117
210	0.23	0.05	0.08	1096
211	0.21	0.04	0.07	1067
212	0.16	0.09	0.12	1064
213	0.51	0.54	0.53	1123
214	0.67	0.85	0.75	1072
215	0.20	0.09	0.13	1098
216	0.07	0.03	0.04	1069
217	0.23	0.15	0.18	1092
218	0.51	0.51	0.51	1043
219	0.54	0.30	0.39	1043
220	0.58	0.60	0.59	1030

221	0.46	0.57	0.51	1043
222	0.57	0.52	0.54	1049
223	0.41	0.20	0.27	1047
224	0.28	0.16	0.20	1027
225	0.39	0.17	0.23	995
226	0.46	0.37	0.41	990
227	0.61	0.76	0.68	999
228	0.14	0.02	0.04	995
229	0.75	0.85	0.80	995
230	0.21	0.12	0.16	999
231	0.30	0.14	0.19	997
232	0.64	0.43	0.52	974
233	0.54	0.54	0.54	995
234	0.13	0.04	0.05	998
235	0.16	0.03	0.06	951
236	0.30	0.13	0.18	1000
237	0.46	0.43	0.44	976
238	0.35	0.33	0.34	953
239	0.52	0.46	0.49	960
240	0.74	0.67	0.71	925
241	0.10	0.01	0.02	961
242	0.58	0.62	0.60	974
243	0.34	0.21	0.26	943
244	0.16	0.04	0.07	928
245	0.20	0.20	0.20	936
246	0.20	0.09	0.12	928
247	0.48	0.39	0.43	917
248	0.59	0.32	0.41	938
249	0.60	0.64	0.62	910
250	0.36	0.11	0.17	935
251	0.52	0.30	0.38	916
252	0.39	0.19	0.25	909
253	0.44	0.49	0.46	878
254	0.15	0.13	0.14	901
255	0.36	0.37	0.37	904
256	0.14	0.16	0.15	873
257	0.37	0.25	0.30	891
258	0.56	0.59	0.58	880
259	0.20	0.20	0.20	886
260	0.62	0.39	0.48	900
261	0.32	0.34	0.33	871
262	0.10	0.05	0.07	897
263	0.78	0.80	0.79	873
264	0.32	0.19	0.24	877
265	0.24	0.03	0.06	870
266	0.10	0.01	0.01	846
267	0.66	0.61	0.63	865
268	0.48	0.28	0.35	849
269	0.19	0.11	0.14	861

270	0.03	0.01	0.02	842
271	0.49	0.42	0.46	844
272	0.55	0.72	0.62	844
273	0.82	0.83	0.82	822
274	0.83	0.88	0.85	845
275	0.28	0.09	0.14	828
276	0.67	0.64	0.65	819
277	0.33	0.16	0.21	843
278	0.54	0.30	0.38	814
279	0.09	0.05	0.07	798
280	0.55	0.37	0.44	783
281	0.56	0.49	0.52	789
282	0.20	0.04	0.06	813
283	0.52	0.45	0.48	769
284	0.43	0.25	0.32	837
285	0.72	0.69	0.71	776
286	0.12	0.06	0.08	764
287	0.25	0.12	0.16	756
288	0.08	0.00	0.00	777
289	0.10	0.16	0.12	772
290	0.00	0.00	0.00	795
291	0.70	0.49	0.58	779
292	0.83	0.88	0.86	741
293	0.19	0.16	0.18	785
294	0.00	0.00	0.00	802
295	0.31	0.04	0.07	773
296	0.07	0.03	0.04	799
297	0.29	0.31	0.30	750
298	0.30	0.14	0.19	769
299	0.81	0.66	0.73	767
300	0.63	0.09	0.15	769
301	0.48	0.12	0.19	742
302	0.13	0.15	0.14	738
303	0.28	0.13	0.18	744
304	0.38	0.22	0.28	730
305	0.19	0.04	0.06	705
306	0.50	0.16	0.25	772
307	0.31	0.26	0.29	730
308	0.68	0.73	0.70	781
309	0.71	0.80	0.75	733
310	0.31	0.28	0.30	761
311	0.30	0.33	0.31	727
312	0.29	0.29	0.29	718
313	0.41	0.22	0.28	735
314	0.57	0.37	0.45	734
315	0.41	0.33	0.37	735
316	0.73	0.51	0.60	714
317	0.14	0.09	0.11	690
318	0.18	0.04	0.06	749

319	0.26	0.11	0.16	722
320	0.05	0.01	0.01	723
321	0.12	0.02	0.03	700
322	0.45	0.30	0.36	688
323	0.23	0.32	0.27	706
324	0.11	0.04	0.06	713
325	0.18	0.16	0.17	713
326	0.39	0.34	0.37	646
327	0.51	0.47	0.49	710
328	0.05	0.01	0.02	689
329	0.19	0.21	0.20	673
330	0.27	0.12	0.16	657
331	0.29	0.18	0.23	706
332	0.51	0.36	0.42	675
333	0.46	0.34	0.39	694
334	0.14	0.16	0.15	661
335	0.16	0.06	0.08	659
336	0.40	0.46	0.43	659
337	0.15	0.04	0.06	669
338	0.44	0.54	0.49	676
339	0.45	0.25	0.32	673
340	0.58	0.17	0.27	676
341	0.44	0.14	0.21	683
342	0.16	0.09	0.12	664
343	0.37	0.16	0.22	662
344	0.20	0.13	0.16	676
345	0.17	0.02	0.04	650
346	0.50	0.43	0.46	647
347	0.13	0.17	0.15	636
348	0.35	0.29	0.32	648
349	0.34	0.09	0.14	643
350	0.11	0.02	0.04	652
351	0.11	0.02	0.04	666
352	0.69	0.55	0.61	635
353	0.24	0.25	0.24	632
354	0.36	0.34	0.35	639
355	0.37	0.24	0.29	614
356	0.34	0.38	0.36	637
357	0.29	0.19	0.23	637
358	0.63	0.33	0.43	614
359	0.41	0.30	0.35	592
360	0.27	0.40	0.32	651
361	0.37	0.38	0.38	634
362	0.41	0.37	0.39	590
363	0.48	0.24	0.32	650
364	0.26	0.06	0.10	621
365	0.76	0.60	0.67	629
366	0.90	0.90	0.90	619
367	0.81	0.74	0.77	631

368	0.25	0.12	0.16	629
369	0.32	0.21	0.26	630
370	0.27	0.27	0.27	614
371	0.09	0.02	0.03	635
372	0.69	0.39	0.50	637
373	0.37	0.12	0.18	645
374	0.10	0.04	0.05	606
375	0.18	0.05	0.08	588
376	0.69	0.78	0.73	620
377	0.65	0.70	0.67	613
378	0.22	0.08	0.12	590
379	0.53	0.57	0.55	607
380	0.13	0.03	0.05	623
381	0.30	0.12	0.17	587
382	0.44	0.59	0.50	613
383	0.61	0.84	0.71	584
384	0.25	0.24	0.25	612
385	0.45	0.27	0.34	635
386	0.11	0.11	0.11	608
387	0.23	0.10	0.14	585
388	0.04	0.00	0.01	618
389	0.16	0.05	0.07	610
390	0.08	0.09	0.08	582
391	0.20	0.08	0.12	612
392	0.36	0.47	0.41	569
393	0.23	0.01	0.02	600
394	0.88	0.65	0.75	605
395	0.29	0.32	0.30	586
396	0.07	0.07	0.07	588
397	0.14	0.19	0.16	595
398	0.28	0.02	0.04	575
399	0.65	0.44	0.53	609
400	0.19	0.03	0.05	584
401	0.60	0.69	0.64	595
402	0.66	0.50	0.57	576
403	0.09	0.05	0.06	604
404	0.33	0.33	0.33	584
405	0.50	0.02	0.03	587
406	0.21	0.11	0.15	560
407	0.20	0.15	0.17	578
408	0.53	0.42	0.47	571
409	0.77	0.73	0.75	564
410	0.21	0.13	0.16	575
411	0.09	0.03	0.04	583
412	0.22	0.20	0.21	580
413	0.28	0.18	0.22	551
414	0.47	0.51	0.49	553
415	0.63	0.51	0.56	576
416	0.40	0.22	0.28	564

417	0.18	0.08	0.11	602
418	0.15	0.08	0.11	566
419	0.18	0.06	0.09	575
420	0.39	0.32	0.35	569
421	0.42	0.07	0.12	559
422	0.13	0.15	0.14	562
423	0.75	0.59	0.66	524
424	0.54	0.60	0.57	569
425	0.25	0.14	0.18	563
426	0.81	0.62	0.70	553
427	0.16	0.07	0.10	523
428	0.03	0.00	0.00	555
429	0.24	0.18	0.21	580
430	0.38	0.15	0.21	553
431	0.51	0.43	0.46	552
432	0.41	0.18	0.25	566
433	0.83	0.67	0.74	564
434	0.72	0.52	0.60	556
435	0.25	0.06	0.10	536
436	0.64	0.58	0.61	552
437	0.65	0.57	0.61	569
438	0.56	0.66	0.60	527
439	0.50	0.30	0.37	542
440	0.28	0.14	0.19	551
441	0.43	0.25	0.31	550
442	0.11	0.02	0.03	548
443	0.32	0.32	0.32	531
444	0.46	0.44	0.45	538
445	0.12	0.09	0.10	550
446	0.55	0.07	0.12	523
447	0.07	0.07	0.07	539
448	0.38	0.22	0.28	511
449	0.16	0.05	0.07	538
450	0.13	0.20	0.16	522
451	0.37	0.07	0.11	533
452	0.47	0.65	0.55	535
453	0.89	0.78	0.83	522
454	0.03	0.00	0.00	525
455	0.00	0.00	0.00	528
456	0.10	0.02	0.03	510
457	0.74	0.55	0.63	575
458	0.87	0.74	0.80	505
459	0.59	0.30	0.40	528
460	0.70	0.40	0.51	522
461	0.40	0.39	0.40	527
462	0.09	0.03	0.05	520
463	0.18	0.03	0.06	520
464	0.68	0.59	0.63	529
465	0.40	0.20	0.27	507
466	0.28	0.18	0.22	517

```
         467       0.36      0.19      0.25       503
         468       0.34      0.16      0.22       542
         469       0.51      0.08      0.14       503
         470       0.27      0.22      0.24       504
         471       0.37      0.16      0.23       528
         472       0.08      0.03      0.05       523
         473       0.40      0.23      0.29       502
         474       0.46      0.28      0.34       501
         475       0.81      0.61      0.70       508
         476       0.27      0.12      0.17       517
         477       0.11      0.02      0.04       526
         478       0.50      0.23      0.31       503
         479       0.06      0.01      0.02       496
         480       0.70      0.77      0.73       524
         481       0.26      0.14      0.18       535
         482       0.34      0.26      0.30       502
         483       0.36      0.23      0.28       487
         484       0.13      0.16      0.14       502
         485       0.79      0.83      0.81       520
         486       0.41      0.10      0.16       513
         487       0.00      0.00      0.00       512
         488       0.49      0.27      0.34       489
         489       0.28      0.26      0.27       480
         490       0.43      0.52      0.47       502
         491       0.28      0.29      0.28       495
         492       0.39      0.48      0.43       500
         493       0.15      0.10      0.12       484
         494       0.46      0.27      0.34       499
         495       0.21      0.11      0.15       495
         496       0.17      0.06      0.09       475
         497       0.05      0.01      0.02       485
         498       0.80      0.72      0.75       495
         499       0.19      0.16      0.18       509

   micro avg       0.57      0.38      0.46   1080935
   macro avg       0.42      0.31      0.34   1080935
weighted avg       0.54      0.38      0.43   1080935
 samples avg       0.44      0.37      0.38   1080935

Time taken to run this cell : 4:35:14.374099
```

The best score is obtained at alpha = 0.0001 that is 0.458.

5.3 OneVsRest with Linear-SVM

```
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")
```

```
start = datetime.now()
param_grid = {
    'estimator__loss': ['hinge'],
    'estimator__alpha': [0.0001, 0.001, 0.01, 1],
    'estimator__penalty': ['l1']}

classifier = OneVsRestClassifier(SGDClassifier(random_state=21))
grid = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=3,
scoring='f1_micro', verbose=3)
grid.fit(x_train_multilabel, y_train)

best_alpha = grid.best_estimator_.get_params()['estimator__alpha']
print('Best alpha ', best_alpha)
print("Time taken to run this cell :", datetime.now() - start)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV 1/3] END estimator__alpha=0.0001, estimator__loss=hinge,
estimator__penalty=l1;, score=0.450 total time=26.5min
[CV 2/3] END estimator__alpha=0.0001, estimator__loss=hinge,
estimator__penalty=l1;, score=0.452 total time=28.4min
[CV 3/3] END estimator__alpha=0.0001, estimator__loss=hinge,
estimator__penalty=l1;, score=0.451 total time=27.1min
[CV 1/3] END estimator__alpha=0.001, estimator__loss=hinge,
estimator__penalty=l1;, score=0.389 total time=11.0min
[CV 2/3] END estimator__alpha=0.001, estimator__loss=hinge,
estimator__penalty=l1;, score=0.374 total time=12.0min
[CV 3/3] END estimator__alpha=0.001, estimator__loss=hinge,
estimator__penalty=l1;, score=0.385 total time=13.1min
[CV 1/3] END estimator__alpha=0.01, estimator__loss=hinge,
estimator__penalty=l1;, score=0.199 total time= 8.2min
[CV 2/3] END estimator__alpha=0.01, estimator__loss=hinge,
estimator__penalty=l1;, score=0.162 total time= 9.0min
[CV 3/3] END estimator__alpha=0.01, estimator__loss=hinge,
estimator__penalty=l1;, score=0.190 total time=10.4min
[CV 1/3] END estimator__alpha=1, estimator__loss=hinge,
estimator__penalty=l1;, score=0.000 total time=22.7min
[CV 2/3] END estimator__alpha=1, estimator__loss=hinge,
estimator__penalty=l1;, score=0.000 total time=22.7min
[CV 3/3] END estimator__alpha=1, estimator__loss=hinge,
estimator__penalty=l1;, score=0.000 total time=23.2min
Best alpha  0.0001
Time taken to run this cell : 4:14:10.638587
```

```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge',
alpha=best_alpha, penalty='l1', random_state=21), n_jobs=-1)

classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)
```

```
print("Accuracy :",metrics.accuracy_score(y_test, predictions))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions))


precision = precision_score(y_test, predictions, average='micro')
recall = recall_score(y_test, predictions, average='micro')
f1 = f1_score(y_test, predictions, average='micro')

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')

print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure:
{:.4f}".format(precision, recall, f1))

print (metrics.classification_report(y_test, predictions))
print("Time taken to run this cell :", datetime.now() - start)
```

```
Accuracy : 0.18382697304495507
Hamming loss  0.0033420955701592837
Micro-average quality numbers
Precision: 0.5538, Recall: 0.3812, F1-measure: 0.4516
Macro-average quality numbers
Precision: 0.3756, Recall: 0.3192, F1-measure: 0.3295
              precision    recall  f1-score   support

           0       0.61      0.16      0.26     47225
           1       0.69      0.49      0.57     42820
           2       0.77      0.57      0.66     40062
           3       0.71      0.46      0.56     38059
           4       0.87      0.81      0.84     33755
           5       0.79      0.66      0.72     31548
           6       0.54      0.38      0.45     20369
           7       0.78      0.66      0.72     18948
           8       0.58      0.44      0.50     18350
           9       0.68      0.47      0.56     17813
          10       0.76      0.64      0.69     17374
          11       0.47      0.05      0.10     17218
          12       0.43      0.06      0.11     16136
          13       0.50      0.29      0.37     14400
          14       0.60      0.23      0.34     13682
          15       0.49      0.24      0.33     13625
          16       0.67      0.54      0.60     13343
          17       0.73      0.61      0.66     12000
```

18	0.51	0.25	0.33	11217
19	0.40	0.17	0.24	9901
20	0.24	0.05	0.09	8786
21	0.69	0.43	0.53	7597
22	0.52	0.33	0.40	7457
23	0.80	0.72	0.76	6760
24	0.54	0.45	0.49	6670
25	0.59	0.40	0.48	6400
26	0.26	0.05	0.08	6038
27	0.74	0.75	0.75	6061
28	0.50	0.37	0.43	5953
29	0.48	0.29	0.36	5434
30	0.80	0.86	0.83	5301
31	0.44	0.20	0.28	5115
32	0.35	0.22	0.27	4843
33	0.64	0.42	0.51	4723
34	0.50	0.31	0.38	4769
35	0.57	0.54	0.56	4604
36	0.61	0.58	0.60	4657
37	0.65	0.69	0.67	4600
38	0.31	0.17	0.22	4456
39	0.22	0.12	0.16	3900
40	0.31	0.11	0.16	3811
41	0.51	0.47	0.49	3834
42	0.46	0.38	0.42	3704
43	0.45	0.44	0.44	3567
44	0.36	0.15	0.21	3513
45	0.24	0.10	0.14	3443
46	0.22	0.04	0.07	3298
47	0.30	0.16	0.21	3281
48	0.58	0.06	0.11	3277
49	0.25	0.07	0.11	3244
50	0.34	0.10	0.16	3231
51	0.68	0.81	0.74	3205
52	0.28	0.11	0.16	3123
53	0.52	0.48	0.50	3149
54	0.62	0.54	0.57	3062
55	0.43	0.18	0.25	3027
56	0.25	0.16	0.20	3057
57	0.80	0.85	0.82	3017
58	0.13	0.01	0.01	2986
59	0.42	0.42	0.42	3043
60	0.66	0.57	0.61	3014
61	0.64	0.57	0.60	2942
62	0.19	0.06	0.09	2935
63	0.57	0.31	0.40	2882
64	0.68	0.75	0.71	2938
65	0.64	0.61	0.63	2873
66	0.49	0.33	0.39	2817

67	0.51	0.28	0.37	2734
68	0.72	0.63	0.67	2740
69	0.36	0.07	0.12	2750
70	0.52	0.49	0.50	2684
71	0.72	0.61	0.66	2675
72	0.36	0.03	0.06	2575
73	0.65	0.35	0.46	2558
74	0.75	0.76	0.76	2611
75	0.44	0.47	0.46	2507
76	0.38	0.35	0.37	2460
77	0.60	0.57	0.59	2427
78	0.07	0.00	0.00	2371
79	0.20	0.08	0.12	2227
80	0.36	0.35	0.36	2215
81	0.64	0.47	0.54	2187
82	0.44	0.35	0.39	2146
83	0.67	0.68	0.68	2215
84	0.27	0.03	0.05	2217
85	0.77	0.69	0.72	2182
86	0.69	0.59	0.64	2172
87	0.64	0.53	0.58	2186
88	0.29	0.26	0.27	2170
89	0.77	0.60	0.68	2151
90	0.73	0.71	0.72	2168
91	0.54	0.10	0.17	2125
92	0.25	0.10	0.14	1969
93	0.46	0.51	0.49	1994
94	0.50	0.58	0.54	1986
95	0.69	0.77	0.73	1954
96	0.34	0.19	0.25	1917
97	0.78	0.80	0.79	1899
98	0.27	0.25	0.26	1945
99	0.43	0.20	0.28	1922
100	0.14	0.09	0.11	1954
101	0.52	0.42	0.46	1901
102	0.09	0.01	0.01	1926
103	0.77	0.76	0.77	1896
104	0.72	0.74	0.73	1905
105	0.17	0.00	0.01	1897
106	0.70	0.74	0.72	1901
107	0.33	0.17	0.23	1825
108	0.41	0.47	0.44	1843
109	0.22	0.15	0.18	1783
110	0.57	0.54	0.55	1765
111	0.56	0.49	0.52	1769
112	0.30	0.34	0.32	1757
113	0.37	0.27	0.31	1708
114	0.31	0.31	0.31	1673
115	0.35	0.05	0.08	1660

116	0.44	0.17	0.24	1652
117	0.77	0.84	0.80	1624
118	0.13	0.06	0.08	1607
119	0.45	0.45	0.45	1641
120	0.35	0.31	0.33	1608
121	0.32	0.16	0.21	1578
122	0.23	0.11	0.15	1605
123	0.83	0.76	0.80	1588
124	0.28	0.32	0.30	1583
125	0.50	0.56	0.53	1605
126	0.49	0.55	0.52	1566
127	0.26	0.00	0.01	1557
128	0.68	0.87	0.76	1563
129	0.26	0.14	0.18	1577
130	0.54	0.62	0.58	1520
131	0.37	0.01	0.01	1518
132	0.74	0.68	0.71	1554
133	0.36	0.29	0.32	1481
134	0.75	0.81	0.78	1510
135	0.35	0.34	0.34	1494
136	0.40	0.50	0.45	1465
137	0.29	0.08	0.12	1486
138	0.00	0.00	0.00	1515
139	0.24	0.14	0.18	1524
140	0.23	0.08	0.12	1499
141	0.33	0.39	0.36	1507
142	0.62	0.53	0.57	1503
143	0.31	0.11	0.16	1484
144	0.59	0.62	0.61	1460
145	0.10	0.01	0.02	1466
146	0.29	0.07	0.11	1461
147	0.40	0.39	0.39	1495
148	0.77	0.78	0.77	1478
149	0.16	0.05	0.08	1502
150	0.10	0.04	0.06	1418
151	0.34	0.37	0.36	1419
152	0.38	0.01	0.02	1424
153	0.45	0.60	0.51	1387
154	0.25	0.32	0.28	1384
155	0.85	0.84	0.85	1446
156	0.14	0.10	0.11	1376
157	0.44	0.39	0.41	1424
158	0.18	0.09	0.12	1387
159	0.29	0.38	0.33	1387
160	0.39	0.07	0.12	1383
161	0.36	0.10	0.15	1356
162	0.42	0.49	0.45	1387
163	0.23	0.07	0.11	1339
164	0.78	0.84	0.81	1365

165	0.38	0.31	0.34	1354
166	0.59	0.56	0.58	1344
167	0.24	0.15	0.19	1379
168	0.28	0.18	0.22	1337
169	0.31	0.26	0.29	1325
170	0.35	0.21	0.26	1322
171	0.50	0.49	0.49	1337
172	0.21	0.27	0.24	1330
173	0.40	0.14	0.21	1299
174	0.62	0.78	0.69	1276
175	0.16	0.07	0.10	1335
176	0.24	0.20	0.22	1350
177	0.55	0.55	0.55	1247
178	0.39	0.12	0.18	1295
179	0.61	0.74	0.67	1296
180	0.28	0.27	0.27	1265
181	0.08	0.03	0.04	1244
182	0.67	0.75	0.71	1236
183	0.57	0.35	0.43	1264
184	0.66	0.77	0.71	1252
185	0.10	0.05	0.07	1213
186	0.46	0.53	0.49	1187
187	0.42	0.21	0.28	1202
188	0.09	0.00	0.00	1213
189	0.15	0.03	0.05	1217
190	0.65	0.61	0.63	1212
191	0.24	0.22	0.23	1203
192	0.41	0.26	0.32	1154
193	0.52	0.43	0.47	1155
194	0.61	0.61	0.61	1185
195	0.86	0.82	0.84	1125
196	0.36	0.11	0.17	1173
197	0.67	0.57	0.62	1154
198	0.88	0.74	0.80	1160
199	0.46	0.53	0.49	1148
200	0.16	0.02	0.03	1128
201	0.62	0.72	0.66	1205
202	0.36	0.14	0.20	1158
203	0.32	0.12	0.17	1190
204	0.26	0.08	0.12	1154
205	0.45	0.40	0.42	1138
206	0.24	0.02	0.04	1132
207	0.34	0.15	0.21	1137
208	0.55	0.36	0.44	1128
209	0.42	0.45	0.44	1153
210	0.00	0.00	0.00	1130
211	0.03	0.00	0.00	1075
212	0.50	0.00	0.00	1090
213	0.42	0.39	0.41	1091

214	0.61	0.62	0.62	1034
215	0.15	0.10	0.12	1070
216	0.52	0.55	0.53	1064
217	0.66	0.77	0.71	1043
218	0.18	0.10	0.13	1069
219	0.18	0.16	0.17	1006
220	0.53	0.55	0.54	1038
221	0.34	0.28	0.31	1028
222	0.54	0.54	0.54	1039
223	0.47	0.50	0.49	1017
224	0.18	0.32	0.23	1013
225	0.81	0.87	0.84	1043
226	0.80	0.83	0.81	1019
227	0.35	0.35	0.35	1011
228	0.33	0.07	0.12	1006
229	0.30	0.20	0.24	1018
230	0.40	0.43	0.42	1019
231	0.07	0.02	0.04	1010
232	0.26	0.12	0.16	1006
233	0.17	0.01	0.02	962
234	0.36	0.03	0.05	981
235	0.52	0.55	0.54	1003
236	0.13	0.05	0.07	955
237	0.34	0.14	0.20	984
238	0.47	0.58	0.52	1003
239	0.12	0.01	0.01	984
240	0.24	0.10	0.14	950
241	0.67	0.64	0.65	975
242	0.42	0.43	0.42	910
243	0.07	0.08	0.08	880
244	0.42	0.40	0.41	942
245	0.20	0.29	0.23	932
246	0.20	0.34	0.25	929
247	0.02	0.00	0.00	909
248	0.63	0.59	0.61	921
249	0.45	0.33	0.38	893
250	0.16	0.18	0.17	875
251	0.64	0.68	0.66	932
252	0.34	0.37	0.36	906
253	0.37	0.21	0.26	916
254	0.44	0.51	0.47	904
255	0.26	0.38	0.31	871
256	0.56	0.64	0.60	907
257	0.81	0.78	0.80	870
258	0.19	0.10	0.13	885
259	0.09	0.11	0.10	878
260	0.30	0.35	0.33	876
261	0.60	0.62	0.61	910
262	0.28	0.20	0.23	887

263	0.43	0.41	0.42	874
264	0.46	0.61	0.52	853
265	0.19	0.16	0.17	894
266	0.62	0.66	0.64	893
267	0.00	0.00	0.00	831
268	0.62	0.46	0.53	902
269	0.26	0.22	0.24	866
270	0.00	0.00	0.00	871
271	0.06	0.01	0.02	875
272	0.08	0.01	0.02	873
273	0.43	0.45	0.44	824
274	0.80	0.83	0.82	834
275	0.39	0.45	0.42	807
276	0.23	0.24	0.23	814
277	0.25	0.16	0.19	847
278	0.37	0.28	0.32	814
279	0.34	0.38	0.36	856
280	0.01	0.00	0.00	827
281	0.87	0.81	0.84	751
282	0.36	0.31	0.33	787
283	0.08	0.03	0.04	786
284	0.57	0.42	0.49	800
285	0.42	0.37	0.40	799
286	0.57	0.61	0.59	796
287	0.50	0.54	0.52	795
288	0.08	0.07	0.08	777
289	0.68	0.71	0.69	766
290	0.32	0.11	0.16	824
291	0.71	0.76	0.73	777
292	0.25	0.26	0.25	800
293	0.00	0.00	0.00	794
294	0.30	0.29	0.30	779
295	0.12	0.03	0.04	745
296	0.33	0.43	0.37	732
297	0.26	0.08	0.12	797
298	0.04	0.01	0.01	778
299	0.62	0.69	0.65	763
300	0.20	0.00	0.00	784
301	0.74	0.86	0.80	761
302	0.09	0.00	0.00	773
303	0.27	0.40	0.32	758
304	0.08	0.15	0.10	764
305	0.01	0.00	0.01	740
306	0.12	0.01	0.03	764
307	0.09	0.08	0.08	741
308	0.00	0.00	0.00	772
309	0.32	0.26	0.29	746
310	0.35	0.36	0.35	767
311	0.23	0.20	0.21	786

312	0.29	0.32	0.31	732
313	0.19	0.02	0.04	749
314	0.09	0.04	0.06	746
315	0.46	0.44	0.45	755
316	0.28	0.06	0.10	720
317	0.35	0.39	0.37	758
318	0.26	0.25	0.25	732
319	0.33	0.38	0.35	744
320	0.66	0.75	0.70	716
321	0.30	0.31	0.31	730
322	0.39	0.40	0.40	728
323	0.19	0.18	0.19	715
324	0.02	0.00	0.00	723
325	0.35	0.29	0.32	719
326	0.16	0.23	0.19	695
327	0.52	0.61	0.56	711
328	0.16	0.21	0.18	672
329	0.13	0.11	0.12	703
330	0.31	0.17	0.22	693
331	0.32	0.35	0.34	684
332	0.07	0.01	0.02	699
333	0.44	0.46	0.45	675
334	0.13	0.07	0.09	690
335	0.67	0.00	0.01	683
336	0.35	0.15	0.21	680
337	0.27	0.36	0.31	670
338	0.11	0.02	0.04	693
339	0.26	0.24	0.25	677
340	0.48	0.51	0.49	660
341	0.00	0.00	0.00	651
342	0.35	0.41	0.38	662
343	0.05	0.02	0.03	649
344	0.70	0.58	0.63	614
345	0.12	0.17	0.14	673
346	0.80	0.67	0.73	659
347	0.21	0.08	0.11	642
348	0.35	0.42	0.39	639
349	0.15	0.13	0.14	660
350	0.32	0.20	0.24	653
351	0.27	0.21	0.23	616
352	0.00	0.00	0.00	669
353	0.34	0.34	0.34	635
354	0.10	0.18	0.13	630
355	0.66	0.64	0.65	644
356	0.30	0.43	0.35	662
357	0.08	0.14	0.11	632
358	0.78	0.68	0.72	634
359	0.33	0.37	0.35	635
360	0.25	0.27	0.26	609

361	0.27	0.02	0.04	649
362	0.12	0.01	0.02	639
363	0.40	0.45	0.42	623
364	0.24	0.27	0.25	651
365	0.43	0.52	0.47	618
366	0.26	0.29	0.28	625
367	0.00	0.00	0.00	634
368	0.74	0.82	0.78	617
369	0.42	0.35	0.38	656
370	0.41	0.48	0.44	613
371	0.09	0.04	0.06	629
372	0.05	0.02	0.03	615
373	0.36	0.26	0.30	580
374	0.76	0.67	0.71	598
375	0.16	0.04	0.07	612
376	0.47	0.58	0.52	606
377	0.88	0.83	0.86	594
378	0.26	0.01	0.02	611
379	0.76	0.84	0.80	610
380	0.32	0.30	0.31	605
381	0.15	0.11	0.13	606
382	0.11	0.11	0.11	602
383	0.15	0.16	0.16	609
384	0.53	0.56	0.54	597
385	0.10	0.02	0.03	618
386	0.25	0.34	0.29	617
387	0.25	0.23	0.24	628
388	0.35	0.29	0.32	638
389	0.10	0.03	0.04	591
390	0.46	0.40	0.43	608
391	0.09	0.13	0.11	614
392	0.16	0.05	0.08	612
393	0.08	0.09	0.08	610
394	0.03	0.01	0.01	578
395	0.15	0.14	0.14	612
396	0.00	0.00	0.00	583
397	0.21	0.02	0.03	592
398	0.13	0.12	0.13	587
399	0.19	0.22	0.20	563
400	0.14	0.19	0.16	588
401	0.63	0.71	0.67	608
402	0.13	0.05	0.08	585
403	0.38	0.44	0.41	582
404	0.08	0.01	0.02	592
405	0.07	0.00	0.00	611
406	0.29	0.05	0.09	588
407	0.10	0.16	0.13	598
408	0.21	0.01	0.02	604
409	0.18	0.17	0.17	578

410	0.25	0.16	0.20	581
411	0.35	0.40	0.37	588
412	0.70	0.57	0.63	566
413	0.30	0.23	0.26	568
414	0.11	0.03	0.04	556
415	0.05	0.04	0.04	559
416	0.16	0.14	0.15	559
417	0.76	0.06	0.12	541
418	0.57	0.60	0.59	564
419	0.21	0.04	0.06	572
420	0.30	0.27	0.28	577
421	0.04	0.01	0.01	580
422	0.74	0.65	0.70	567
423	0.43	0.52	0.47	547
424	0.60	0.58	0.59	565
425	0.74	0.77	0.76	549
426	0.16	0.18	0.17	555
427	0.34	0.30	0.32	548
428	0.57	0.60	0.58	542
429	0.43	0.39	0.41	551
430	0.38	0.50	0.43	555
431	0.52	0.59	0.55	548
432	0.60	0.70	0.65	544
433	0.02	0.01	0.01	546
434	0.35	0.41	0.38	548
435	0.51	0.57	0.54	572
436	0.24	0.26	0.25	547
437	0.22	0.19	0.20	543
438	0.00	0.00	0.00	533
439	0.51	0.43	0.47	550
440	0.25	0.22	0.23	525
441	0.80	0.60	0.69	565
442	0.16	0.09	0.12	555
443	0.64	0.60	0.62	548
444	0.14	0.12	0.13	518
445	0.60	0.65	0.62	535
446	0.16	0.14	0.15	547
447	0.13	0.14	0.13	548
448	0.25	0.27	0.26	565
449	0.00	0.00	0.00	561
450	0.00	0.00	0.00	549
451	0.29	0.30	0.29	542
452	0.23	0.30	0.26	523
453	0.27	0.29	0.28	530
454	0.08	0.01	0.01	531
455	0.14	0.11	0.12	540
456	0.50	0.29	0.36	523
457	0.37	0.16	0.22	523
458	0.12	0.12	0.12	528

459	0.40	0.42	0.41	503
460	0.17	0.08	0.11	519
461	0.17	0.18	0.18	521
462	0.25	0.27	0.26	512
463	0.16	0.03	0.04	520
464	0.08	0.08	0.08	501
465	0.79	0.70	0.74	530
466	0.04	0.00	0.01	528
467	0.55	0.61	0.58	499
468	0.66	0.69	0.68	526
469	0.74	0.50	0.60	511
470	0.12	0.09	0.10	517
471	0.07	0.01	0.02	499
472	0.27	0.37	0.31	504
473	0.43	0.51	0.46	518
474	0.29	0.31	0.29	521
475	0.43	0.49	0.45	511
476	0.25	0.28	0.26	509
477	0.78	0.68	0.73	510
478	0.19	0.25	0.21	498
479	0.13	0.22	0.17	478
480	0.05	0.03	0.04	521
481	0.91	0.74	0.82	520
482	0.38	0.43	0.40	516
483	0.19	0.18	0.19	522
484	0.85	0.75	0.79	496
485	0.05	0.02	0.03	504
486	0.00	0.00	0.00	519
487	0.43	0.59	0.50	538
488	0.42	0.54	0.48	512
489	0.36	0.39	0.37	480
490	0.32	0.36	0.34	483
491	0.18	0.07	0.10	499
492	0.06	0.03	0.04	474
493	0.12	0.08	0.10	502
494	0.79	0.77	0.78	496
495	0.11	0.06	0.07	495
496	0.27	0.30	0.28	497
497	0.13	0.04	0.06	486
498	0.26	0.33	0.29	474
499	0.10	0.00	0.00	472
micro avg	0.55	0.38	0.45	1082861
macro avg	0.38	0.32	0.33	1082861
weighted avg	0.52	0.38	0.42	1082861
samples avg	0.44	0.37	0.37	1082861

Time taken to run this cell : 0:21:41.271880

Conclusion

```
from prettytable import PrettyTable
table = PrettyTable()
table.field_names = ["Classifier", "Vectorizer", "Hyperparameter",
"Regularization", "Micro F1 score"]
table.add_row(["Logistic Regression", "Tfidf", 0.00001, 'L1', 0.4410])
table.add_row(["Logistic Regression", "Tfidf", 1.0, 'L2', 0.4388])
table.add_row(["Logistic Regression", "Count vectorizer (4-gram)",
0.00001, 'L1', 0.4223])
table.add_row(["Logistic Regression", "Count vectorizer (4-gram)",
0.0001, 'L1', 0.4585])
table.add_row(["Linear SVM", "Count vectorizer (4-gram)", 0.0001,
'L1', 0.4516])

print(table)
```

```
+---------------------+---------------------------+----------------
+----------------+----------------+
|      Classifier     |         Vectorizer        | Hyperparameter |
Regularization | Micro F1 score |
+---------------------+---------------------------+----------------
+----------------+----------------+
| Logistic Regression |           Tfidf           |     1e-05      |
L1        |     0.441      |
| Logistic Regression |           Tfidf           |      1.0       |
L2        |     0.4388     |
| Logistic Regression | Count vectorizer (4-gram) |     1e-05      |
L1        |     0.4223     |
| Logistic Regression | Count vectorizer (4-gram) |     0.0001     |
L1        |     0.4585     |
|      Linear SVM     | Count vectorizer (4-gram) |     0.0001     |
L1        |     0.4516     |
+---------------------+---------------------------+----------------
+----------------+----------------+
```

- High Dimensionality - Here we limited ourselves to simple linear models likes Logistic Regression and Linear SVM. Because we used Count vectorizers and Tf-idf vectorizers and our data is high dimensional, Decision Tree, Random Forest, GBDT would fail to work.
- Time Complexity – As we have considered 500 labels, so we have to train 500 model and hence Linear model makes more sense.