In [2]:

```
# Importing necessary libraries

import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import csv
import sqlite3
import os
import re
import datetime as dt
from datetime import datetime
from wordcloud import WordCloud
import pickle
import nltk

from sqlalchemy import create_engine # database connection
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer

from sklearn.metrics import f1_score,precision_score,recall_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn import svm
from skmultilearn.adapt import mlknn
from skmultilearn.problem_transform import ClassifierChain, BinaryRelevance, LabelPowerset
```

# Stack Overflow: Tag Prediction

# 1. Business Problem

## 1.1 Description

### Description

Stack Overflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers.

Stack Overflow is something which every programmer use one way or another. Each month, over 50 million developers come to Stack Overflow to learn, share their knowledge, and build their careers. It features questions and answers on a wide range of topics in computer programming. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg. As of April 2014 Stack Overflow has over 4,000,000 registered users, and it exceeded 10,000,000 questions in late August 2015. Based on the type of tags assigned to questions, the top eight most discussed topics on the site are: Java, JavaScript, C#, PHP, Android, jQuery, Python and HTML.

### Problem Statemtent

Suggest the tags based on the content that was there in the question posted on Stackoverflow.

**Source:** https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/

## 1.2 Source / useful links

Data Source : [https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data](https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data)
Youtube : [https://youtu.be/nNDqbUhtIRg](https://youtu.be/nNDqbUhtIRg)
Research paper : [https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tagging-1.pdf](https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tagging-1.pdf)
Research paper : [https://dl.acm.org/citation.cfm?id=2660970&dl=ACM&coll=DL](https://dl.acm.org/citation.cfm?id=2660970&dl=ACM&coll=DL)

## 1.3 Real World / Business Objectives and Constraints

1. Predict as many tags as possible with high precision and recall.

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

# 2. Machine Learning problem

## 2.1 Data

### 2.1.1 Data Overview

Refer: https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data
All of the data is in 2 files: Train and Test.

> **Train.csv** contains 4 columns: Id,Title,Body,Tags.
>
> **Test.csv** contains the same columns but without the Tags, which you are to predict.
>
> **Size of Train.csv** - 6.75GB
>
> **Size of Test.csv** - 2GB
>
> **Number of rows in Train.csv** = 6034195

The questions are randomized and contains a mix of verbose text sites as well as sites related to math and programming. The number of questions from each site may vary, and no filtering has been performed on the questions (such as closed questions).

__Data Field Explaination__ Dataset contains 6,034,195 rows. The columns in the table are:

> **Id** - Unique identifier for each question
>
> **Title** - The question's title
>
> **Body** - The body of the question
>
> **Tags** - The tags associated with the question in a space-seperated format (all lowercase, should not contain tabs '\t' or ampersands '&')

### 2.1.2 Example Data point

> **Title**: Implementing Boundary Value Analysis of Software Testing in a C++ program?
> **Body** :

```
#include<
iostream>\n
#include<
stdlib.h>\n\n
using namespace std;\n\n
int main()\n
{\n
        int n,a[n],x,c,u[n],m[n],e[n][4];\n
        cout<<"Enter the number of variables";\n        cin>>n;\n\n
        cout<<"Enter the Lower, and Upper Limits of the variables";\n
        for(int y=1; y<n+1; y++)\n
        {\n
          cin>>m[y];\n
          cin>>u[y];\n
        }\n
        for(x=1; x<n+1; x++)\n
        {\n
          a[x] = (m[x] + u[x])/2;\n
        }\n
        c=(n*4)-4;\n
        for(int a1=1; a1<n+1; a1++)\n
        {\n\n
          e[a1][0] = m[a1];\n
          e[a1][1] = m[a1]+1;\n
          e[a1][2] = u[a1]-1;\n
```

```
            e[a1][3] = u[a1];\n
        }\n
        for(int i=1; i<n+1; i++)\n
        {\n
          for(int l=1; l<=i; l++)\n
          {\n
             if(l!=1)\n
             {\n
                cout<<a[l]<<"\\t";\n
             }\n
          }\n
          for(int j=0; j<4; j++)\n
          {\n
             cout<<e[i][j];\n
             for(int k=0; k<n-(i+1); k++)\n
             {\n
                cout<<a[k]<<"\\t";\n
             }\n
             cout<<"\\n";\n
          }\n
        }    \n\n
        system("PAUSE");\n
        return 0;    \n
    }\n
```

\n\n

The answer should come in the form of a table like
\n\n

```
    1        50        50\n
    2        50        50\n
    99       50        50\n
    100      50        50\n
    50       1         50\n
    50       2         50\n
    50       99        50\n
    50       100       50\n
    50       50        1\n
    50       50        2\n
    50       50        99\n
    50       50        100\n
```

\n\n

if the no of inputs is 3 and their ranges are\n
    1,100\n
    1,100\n
    1,100\n
    (could be varied too)
\n\n

The output is not coming,can anyone correct the code or tell me what\'s wrong?
\n'
**Tags** : 'c++ c'

## 2.2 Mapping the real-world problem to a Machine Learning Problem

### 2.2.1 Type of Machine Learning Problem

It is a multi-label classification problem
**Multi-label Classification**: Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A question on Stackoverflow might be about any of C, Pointers, FileIO and/or memory-management at the same time or none of these.
__Credit__: http://scikit-learn.org/stable/modules/multiclass.html

### 2.2.2 Performance metric

**Micro-Averaged F1-Score (Mean F Score)** : The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

*F1 = 2 * (precision * recall) / (precision + recall)*

In the multi-class and multi-label case, this is the weighted average of the F1 score of each class.

**'Micro f1 score':**
Calculate metrics globally by counting the total true positives, false negatives and false positives. This is a better metric when we have class imbalance.

**'Macro f1 score':**
Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

https://www.kaggle.com/wiki/MeanFScore
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

**Hamming loss** : The Hamming loss is the fraction of labels that are incorrectly predicted.
https://www.kaggle.com/wiki/HammingLoss

# 3. Exploratory Data Analysis

## 3.1 Data Loading and Cleaning

### 3.1.1 Using Pandas with SQLite to Load the data

In [2]:

```
"""
# https://stackoverflow.com/a/3451150/10219869

# Train file unzipping
import zipfile
with zipfile.ZipFile('Train.zip', 'r') as zip_ref:
    zip_ref.extractall('lab')

# Test File unzipping
with zipfile.ZipFile('Test.zip', 'r') as zip_ref:
    zip_ref.extractall('lab')

"""
```

Out[2]:

"\n# https://stackoverflow.com/a/3451150/10219869\n\n# Train file unzipping\nimport zipfile\nwith zipfile.ZipFile('Train.zip', 'r') as zip_ref:\n    zip_ref.extractall('lab')\n\n# Test File unzipping\nwith zipfile.ZipFile('Test.zip', 'r') as zip_ref:\n    zip_ref.extractall('lab')\n\n"

In [3]:

```
#Creating db file from csv
#Learn SQL: https://www.w3schools.com/sql/default.asp

from datetime import datetime
if not os.path.isfile('train.db'):
    start = datetime.now()

    # creating new SQLite database engine
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 180000
    j = 0
    index_start = 1

    # importing data into SQLite from CSV
    for df in pd.read_csv('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))

        # 'data' is the name given to the table which consists of above 4 columns
        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

In [3]:

```
if os.path.isfile('train.db'):
    start = datetime.now()

    # connecting to database
    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])

    # disconnecting from database
    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 : 0:03:13.509836
```

### 3.1.3 Checking for duplicates

In [5]:

```
#Learn SQl: https://www.w3schools.com/sql/default.asp
if os.path.isfile('train.db'):
    start = datetime.now()

    # connecting to database
    con = sqlite3.connect('train.db')

    # creating "no duplicates" DataFrame
    df_no_dup = pd.read_sql_query('SELECT Title, Body, Tags, COUNT(*) as cnt_dup FROM data GROUP BY Title, Body, Tags', con)

    # disconnecting from database
    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 : 0:02:45.507073
```

In [6]:

```
df_no_dup.head()
# After grouping all 3 columns except 'Id' (as it is unique obviously), we observe there are duplicates and its count.
```

Out[6]:

| | Title | Body | Tags | cnt_dup |
|---|---|---|---|---|
| 0 | Implementing Boundary Value Analysis of S... | <pre><code>#include<iostream>\n#include&... | c++ c | 1 |
| 1 | Dynamic Datagrid Binding in Silverlight? | <p>I should do binding for datagrid dynamicall... | c# silverlight data-binding | 1 |
| 2 | Dynamic Datagrid Binding in Silverlight? | <p>I should do binding for datagrid dynamicall... | c# silverlight data-binding columns | 1 |
| 3 | java.lang.NoClassDefFoundError: javax/serv... | <p>I followed the guide in <a href="http://sta... | jsp jstl | 1 |
| 4 | java.sql.SQLException:[Microsoft][ODBC Dri... | <p>I use the following code</p>\n\n<pre><code>... | java jdbc | 2 |

In [7]:

```
# Total number of questions which are repeated more than once
df_no_dup[df_no_dup['cnt_dup'] >1 ].count()
```

Out[7]:

```
Title      1550031
```

```
Title      1550031
Body       1550031
Tags       1550030
cnt_dup    1550031
dtype: int64
```

In [8]:

```
# Total number of duplicate questions
print(num_rows['count(*)'].values[0] - df_no_dup.shape[0], 'and its percentage is ', ((num_rows['count(*)'].values[0] - df_no_dup.shape[0]) / num_row
s['count(*)'].values[0]) * 100,'%')
```

```
1827881 and its percentage is  30.292038906260256 %
```

In [9]:

```
# number of times each question appeared in our database 'data'
df_no_dup['cnt_dup'].value_counts()
```

Out[9]:

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

### 3.1.4 Checking for NaN values

In [10]:

```
# Found no duplicate rows in 'Title' feature
df_no_dup[df_no_dup['Title'].isna()]
```

Out[10]:

| Title | Body | Tags | cnt_dup |
|---|---|---|---|

In [11]:

```
# Found no duplicate rows in 'Body' feature
df_no_dup[df_no_dup['Body'].isna()]
```

Out[11]:

| Title | Body | Tags | cnt_dup |
|---|---|---|---|

In [12]:

```
# Found 7 duplicate rows in 'Tags' feature
df_no_dup[df_no_dup['Tags'].isna()]
```

Out[12]:

| | Title | Body | Tags | cnt_dup |
|---|---|---|---|---|
| **777547** | Do we really need NULL? | <blockquote>\n <p><strong>Possible Duplicate:... | None | 1 |
| **962680** | Find all values that are not null and not in a... | <p>I am running into a problem which results i... | None | 1 |
| **1126558** | Handle NullObjects | <p>I have done quite a bit of research on best... | None | 1 |
| **1256102** | How do Germans call null | <p>In german null means 0, so how do they call... | None | 1 |
| **2430668** | Page cannot be null. Please ensure that this o... | <p>I get this error when i remove dynamically ... | None | 1 |
| **3329908** | What is the difference between NULL and "0"? | <p>What is the difference from NULL and "0"?</... | None | 1 |
| **3551595** | a bit of difference between null and space | <p>I was just reading this quote</p>\n\n<block... | None | 2 |

In [13]:

```
# as there are 7 rows of empty Tags, we remove them
df_no_dup.dropna(inplace=True)
```

In [14]:

```
start = datetime.now()

# adding a new feature number of tags per question
a=[]
for i in df_no_dup["Tags"]:
    i = i.split(' ')
    a.append(len(i))

df_no_dup["tag_count"]= a

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

Time taken to run this cell : 0:00:03.892881

Out[14]:

| | Title | Body | Tags | cnt_dup | tag_count |
|---|---|---|---|---|---|
| **0** | Implementing Boundary Value Analysis of S... | <pre><code>#include<iostream>\n#include&... | c++ c | 1 | 2 |
| **1** | Dynamic Datagrid Binding in Silverlight? | <p>I should do binding for datagrid dynamicall... | c# silverlight data-binding | 1 | 3 |
| **2** | Dynamic Datagrid Binding in Silverlight? | <p>I should do binding for datagrid dynamicall... | c# silverlight data-binding columns | 1 | 4 |
| **3** | java.lang.NoClassDefFoundError: javax/serv... | <p>I followed the guide in <a href="http://sta... | jsp jstl | 1 | 2 |
| **4** | java.sql.SQLException:[Microsoft][ODBC Dri... | <p>I use the following code</p>\n\n<pre><code>... | java jdbc | 2 | 2 |

In [15]:

```
#  distribution of no. of tags per question
df_no_dup['tag_count'].value_counts()
```

Out[15]:

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

In [16]:

```
#This method seems more appropriate to work with this much data.
#creating the connection with database file.
if os.path.isfile('train_no_dup.db'):
    start = datetime.now()
    con = sqlite3.connect('train_no_dup.db')

    # making dataframe with no duplicates
    # 'no_dup_train' is table name in database of train_no_dup
    tag_data = pd.read_sql_query("SELECT Tags FROM no_dup_train", con)

    #Always remember to close the database
    con.close()

    # Let's now drop unwanted column.
    tag_data.drop(tag_data.index[0], inplace=True)
    #Printing first 5 columns from our data frame
    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")
```

Time taken to run this cell : 0:00:50.115210

## 3.2 Analysis of Tags

### 3.2.1 Total number of unique tags

In [17]:

```
# Importing & Initializing the "CountVectorizer" object, which is scikit-learn's bag of words (bow) 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'])
```

In [18]:

```
# BOW representation of sparse vectors.
tag_dtm.shape
```

Out[18]:

```
(4206314, 42048)
```

In [19]:

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

```
Number of data points : 4206314
Number of unique tags : 42048
```

In [20]:

```
#'get_feature_names()' used on vectorizer, gives us the vocabulary.
tags = vectorizer.get_feature_names()

#Lets look at the tags we have.
print("Some of the tags we have :\n", 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

In [21]:

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

In [22]:

```
freqs[:10]
```

Out[22]:

```
array([ 18,  37,   1,  21, 138,  53,  14,  47,   1,   8], dtype=int64)
```

In [23]:

```
#Saving this dictionary to csv files.
if not os.path.isfile('tag_counts_dict_dtm.csv'):
    with open('tag_counts_dict_dtm.csv', 'w') as csv_file:
        writer = csv.writer(csv_file)
        for key, value in result.items():
            writer.writerow([key, value])

tag_df = pd.read_csv("tag_counts_dict_dtm.csv", names=['Tags', 'Counts'])
tag_df.head()
```

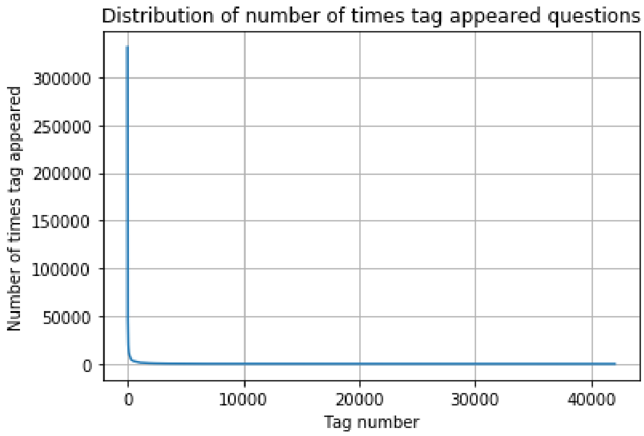
Out[23]:

| | Tags | Counts |
|---|---|---|
| **0** | urlbinding | 5 |
| **1** | mongovue | 11 |
| **2** | createitem | 3 |
| **3** | ipython-notebook | 63 |
| **4** | rubyosa | 2 |

In [24]:

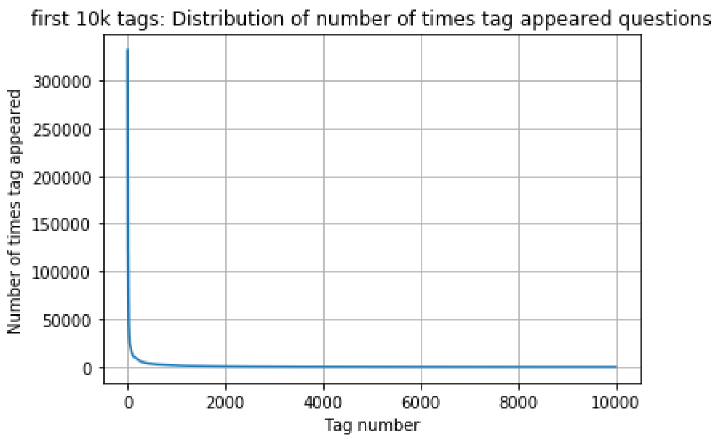
```
tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
tag_counts = tag_df_sorted['Counts'].values

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()
```




In [25]:

```
plt.plot(tag_counts[0:10000])
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])
```

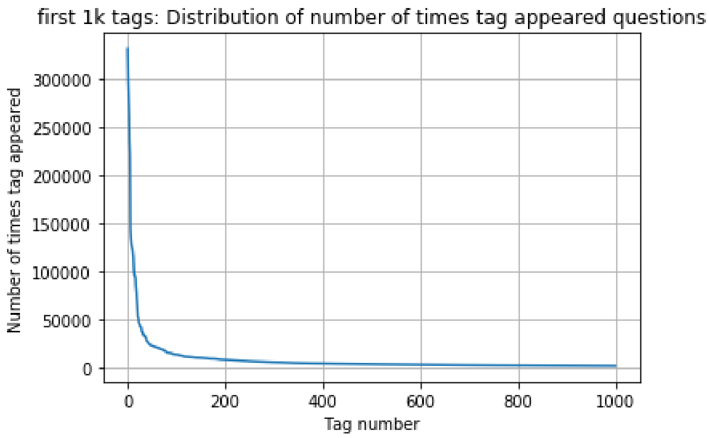



```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054   7151
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2989   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   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]
```

In [26]:

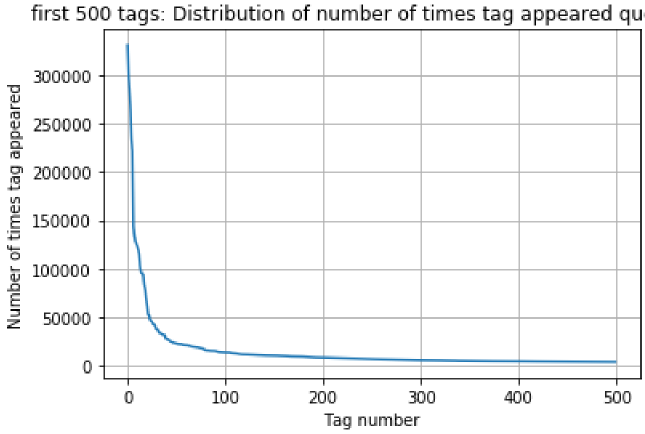
```
plt.plot(tag_counts[0:1000])
plt.title('first 1k tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:1000:5]), tag_counts[0:1000:5])
```




```
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   2989   2984   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]
```

In [27]:

```
plt.plot(tag_counts[0:500])
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])
```



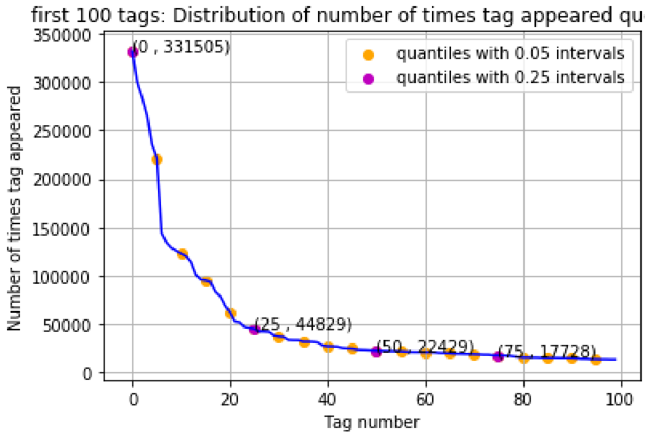

```
100 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   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]
```

In [28]:

```
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(s="({} , {})".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]
```

In [29]:

```
tag_df.columns
```

Out[29]:

Index(['Tags', 'Counts'], dtype='object')

In [30]:

```
#Print the length of the list
print ('{} Tags are used more than 10000 times'.format(len(tag_df[tag_df['Counts'] > 10000])))

#Print the length of the list.
print ('{} Tags are used more than 100000 times'.format(len(tag_df[tag_df['Counts'] > 100000])))
```

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. 14 tags are used more than 100000 times.
3. Most frequent tag (i.e. c#) is used 331505 times.
4. Since some tags occur much more frequenctly than others, Micro-averaged F1-score is the appropriate metric for this probelem.

### 3.2.4 Tags Per Question

In [31]:

```
#Storing the count of tag in each question in list 'tag_count'
# if not .tolist() then the o/p will be in the form of matrix bcoz tag_dtm is a sparse matrix(a result of count vectorizer)
tag_quest_count = tag_dtm.sum(axis=1).tolist()

tag_quest_count[:10]
```

Out[31]:

[[3], [4], [2], [2], [3], [3], [2], [2], [2], [2]]

In [32]:

```
#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_counts=[]
for i in tag_quest_count:
    for j in i:
        tag_quest_counts.append(j)
print ('We have total {} datapoints.'.format(len(tag_quest_counts)))

print(tag_quest_counts[:5])
```

We have total 4206314 datapoints.
[3, 4, 2, 2, 3]

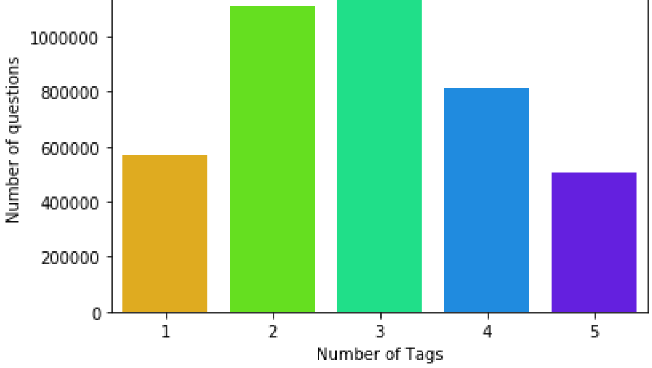
In [33]:

```
print( "Maximum number of tags per question: ", max(tag_quest_counts))
print( "Minimum number of tags per question: ", min(tag_quest_counts))
print( "Avg. number of tags per question: " ,((sum(tag_quest_counts)*1.0)/len(tag_quest_counts)))
```

Maximum number of tags per question:  5
Minimum number of tags per question:  1
Avg. number of tags per question:  2.8994395092710623

In [34]:

```
sns.countplot(tag_quest_counts, palette='gist_rainbow')
plt.title("Number of tags in the questions ")
plt.xlabel("Number of Tags")
plt.ylabel("Number of questions")
plt.show()
```




**Observations:**

1. Maximum number of tags per question: 5
2. Minimum number of tags per question: 1
3. Avg. number of tags per question: 2.899
4. Most of the questions are having 2 or 3 tags

### 3.2.5 Most Frequent Tags

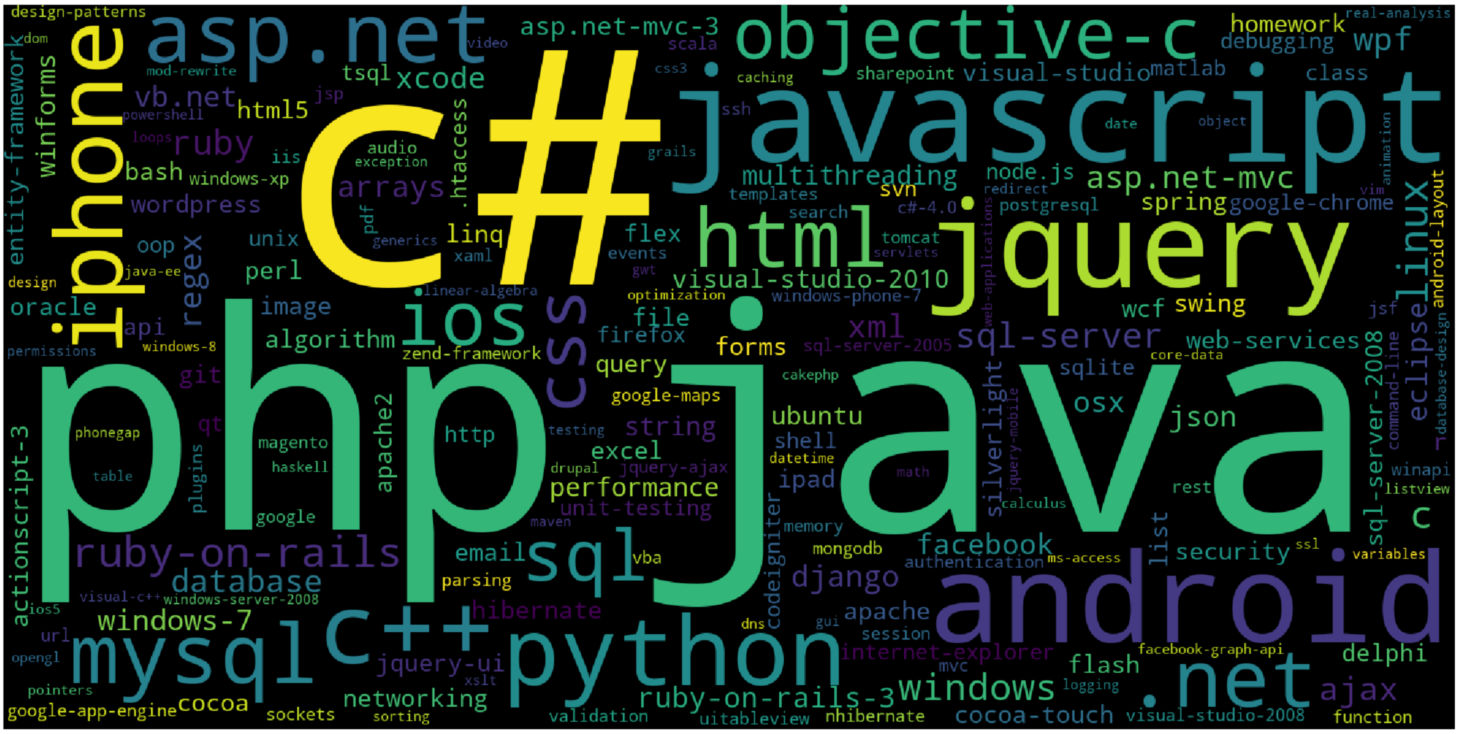
In [35]:

```python
# Ploting word cloud
start = datetime.now()

# Lets 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)
wordcloud.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)
```



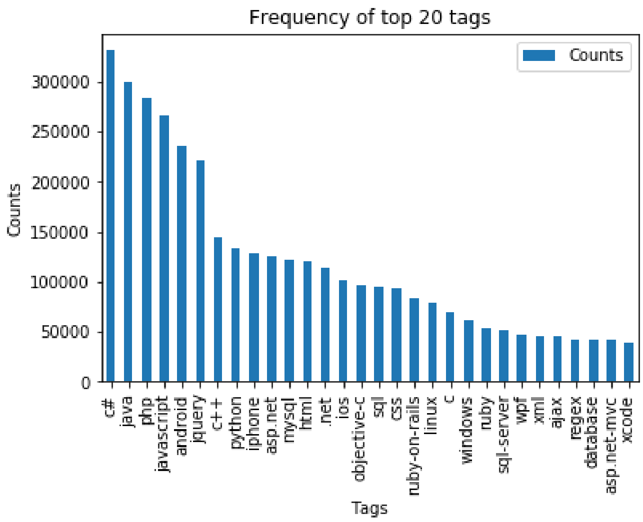

Time taken to run this cell : 0:00:05.045855

**Observations:**
A look at the word cloud shows that "c#", "java", "php", "asp.net", "javascript", "c++" are some of the most frequent tags.

### 3.2.6 The top 20 tags

In [36]:

```
i=np.arange(30)
tag_df_sorted['Counts'][:30].plot(kind='bar')
plt.title('Frequency of top 20 tags')
plt.xticks(i, tag_df_sorted['Tags'])
plt.xlabel('Tags')
plt.ylabel('Counts')
plt.legend()
plt.show()
```




**Observations:**

1. Majority of the most frequent tags are programming language.
2. C# is the top most frequent programming language.
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
2. Separate out code-snippets from Body
3. Remove Spcial characters from Question title and description (not in code)
4. Remove HTML Tags
5. Convert all the characters into small letters
6. Use SnowballStemmer to stem the words

In [4]:

```
def striphtml(data):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', str(data))
    return cleantext
stemmer = SnowballStemmer("english")
```

In [5]:

```
#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:
        connect = sqlite3.connect(db_file)
        return connect
    except Error as e:
        print(e)

    return None
```

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

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

def create_database_table(database, query):
    connect = create_connection(database)
    if connect is not None:
        create_table(connect, query)
        checkTableExists(connect)
    else:
        print("Error! cannot create the database connection.")
    connect.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 database:
QuestionsProcessed
```

In [39]:

```
# 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'
if os.path.isfile(read_db):
    connect_read = create_connection(read_db)
    if connect_read is not None:
        reader =connect_read.cursor()
        # took 1 million random datapoints
        reader.execute("SELECT Title, Body, Tags From no_dup_train ORDER BY RANDOM() LIMIT 1000000;")

write_db = 'Processed.db'
if os.path.isfile(write_db):
    connect_write = create_connection(write_db)
    if connect_write is not None:
        tables = checkTableExists(connect_write)
        writer =connect_write.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 database:
QuestionsProcessed
Cleared All the rows
Time taken to run this cell : 0:02:18.028313
```

**we create a new data base to store the sampled and preprocessed questions**

In [40]:

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

```
[nltk_data] Downloading package punkt to
[nltk_data]     /home/passionateguy_bharat/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

[nltk_data] Package punkt is already up-to-date!

Out[40]:

True

In [41]:

```
#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 from question
    question = ' '.join(str(stemmer.stem(j)) for j in words)
    question = ' '.join([i for i in question.split() if (len(i)>1 or i == '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
number of questions completed =  600000
number of questions completed =  700000
number of questions completed =  800000
number of questions completed =  900000
Avg. length of questions(Title+Body) before processing: 1171
Avg. length of questions(Title+Body) after processing: 490
Percent of questions containing code: 57
Time taken to run this cell : 0:32:06.224758
```

In [42]:

```
# dont forget to close the connections, or else you will end up with locks
connect_read.commit()
connect_write.commit()
connect_read.close()
```

```
connect_write.close()
```

In [43]:

```
if os.path.isfile(write_db):
    connect_read = create_connection(write_db)
    if connect_read is not None:
        reader =connect_read.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)
connect_read.commit()
connect_read.close()
```

```
Questions after preprocessed
====================================================================================================
('javascript div slider no jqueri am look to write javascript div slider tri jqueri plugin for div slide howev sinc our applic is dynam generat somehow the current jqueri plugin don seem to render the slider correct henc thought to see if could write simpl slider that will toggl two div in slider way basic would have two div and one left button and one right bottom so when click the right button the first content should slide to the second horizont same way could flip it back with the left button have an abil to have second delay between transit would be help ani pointer on how to write this will be realli help thx',)
----------------------------------------------------------------------------------------------------
('howto disabl format dection on date how do disabl format detect on date on iphon mobil webapp ni ve tri which doesn work after all it date not phone numb',)
----------------------------------------------------------------------------------------------------
('whi is javascript math floor the slowest way to calcul floor in javascript general not fan of microbenchmark but this one has veri interest result http ernestdelgado com archiv benchmark on the floor it suggest that is the slowest way to calcul floor in javascript all be faster nthis seem pretti shock as would expect that peopl implement javascript in today modern browser would be some pretti smart peopl doe floor do someth import that the other method fail to do is there ani reason to use it',)
----------------------------------------------------------------------------------------------------
('no modul name fcntl am tri to execut this method with ironpython on net use ironpython am use window c code can someon tell me what am do wrong keep gettin that do not have fcntl modul',)
----------------------------------------------------------------------------------------------------
('run regex replac on regex in javascript have the regex which valid csv file but want to be abl to modifi the delimit with ani delimit is it possibl to run regex replac on regex exampl use backtick as the delimit',)
----------------------------------------------------------------------------------------------------
('how do generat breadcrumb for my melt app have use the melt framework to build web applic with the follow structur now want to generat breadcrumb to show in the layout php abov the content of each view what would be the best way to implement breadcrumb in melt',)
----------------------------------------------------------------------------------------------------
('document properti demot from listitem field is not work have content type in librari configur with word document as templat it is the default content type nwhen creat via the ui from new item enter inform into the dip set valu in the document via quick part document properti nthese entri are reflect on the under sharepoint list item and in the bodi of the document if re open the document the chang in the dip do not get reflect in the document bodi if create document via code use the templat file set the list item field the list item properti are not reflect in the document at all they can be found when inspect the xml but they never get demot into the bodi of the document ani bodi got ani pointer as to how we can tri and ensur that the document and the list item properti are kept in sync',)
----------------------------------------------------------------------------------------------------
('how to appli devexpress skin to devexpress textbox control use devexpress theme builder in order to creat theme success appli the theme the problem aris when had differ width aspx textbox decid to creat skin in theme builder and onli chang the width use skinid properti to set the skin on textbox but don see the effect regist my theme with the follow code should do similar registr for my skin',)
----------------------------------------------------------------------------------------------------
('reset video progress bar on click have video progress bar that move to new div when you click on said div and show the play progress of current play video would like to know how can reset the bar when click on the new div instead of when the new video start to play basic click first div video play click anoth div progress bar reset move to new place new video play and bar show have the follow script but as ve not had to work with this befor my grasp of it is littl tenuous ani help would be great appreci',)
----------------------------------------------------------------------------------------------------
```

In [44]:

```
#Taking 2 Million entries to a dataframe.
write_db = 'Processed.db'
if os.path.isfile(write_db):
    connect_read = create_connection(write_db)
    if connect_read is not None:
        preprocessed_data = pd.read_sql_query("SELECT question, Tags FROM QuestionsProcessed", connect_read)
connect_read.commit()
connect_read.close()
```

In [45]:

```
preprocessed_data.head()
```

Out[45]:

| | question | tags |
|---|---|---|

| | question | tags |
|---|---|---|
| 0 | unix termin pager use on os and prefer the uni... | r less readline |
| 1 | javascript div slider no jqueri am look to wri... | javascript jquery slider |
| 2 | howto disabl format dection on date how do dis... | iphone mobile-web |
| 3 | whi is javascript math floor the slowest way t... | javascript optimization |
| 4 | no modul name fcntl am tri to execut this meth... | c# module compiler-errors ironpython fcntl |

In [46]:

```
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

| X | y1 | y2 | y3 | y4 |
|---|---|---|---|---|
| x1 | 0 | 1 | 1 | 0 |
| x1 | 1 | 0 | 0 | 0 |
| x1 | 0 | 1 | 0 | 0 |

In [47]:

```
# binary='true' will give a binary vectorizer (binary BOW)
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 considering all of them (due to limitation of computing power)**

In [6]:

```
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))
```

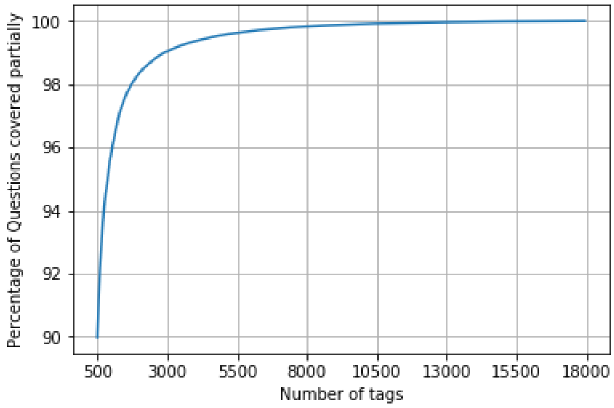
In [49]:

```
questions_explained = []
total_qs= preprocessed_data.shape[0]
total_tags= multilabel_y.shape[1]

for i in range(500, total_tags, 100):
    questions_explained.append(np.round(((total_qs-questions_explained_fn(i))/total_qs)*100,3))
```

In [50]:

```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50) # [-2000, 500, 3000, 5500, 8000, 10500, 13000, 15500, 18000, 20500]
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  99.035 % of questions

In [51]:

```
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 : 9650 out of  999999

In [52]:

```
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 : 35490
number of tags taken : 5500 ( 15.497323189630881 %)

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

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

In [53]:

```
total_size=preprocessed_data.shape[0]
train_size=int(0.80*total_size)

x_train=preprocessed_data.head(train_size) # considering top (head) 80% (train_size)
x_test=preprocessed_data.tail(total_size - train_size)

# sparse matrices of top 5500 features
y_train = multilabel_yx[0:train_size,:]
y_test = multilabel_yx[train_size:total_size,:]
```

In [54]:

```
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

In [55]:

```
start = datetime.now()

vectorizer = TfidfVectorizer(min_df=0.00009, tokenizer = lambda x: x.split(), 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 : 0:11:56.544073

In [56]:

```
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: (799999, 181427) Y : (799999, 5500)
Dimensions of test data X: (200000, 181427) Y: (200000, 5500)
```

In [57]:

```
# 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)
```

Out[57]:

```
"\nfrom skmultilearn.adapt import MLkNN\nclassifier = MLkNN(k=21)\n\n# train\nclassifier.fit(x_train_multilabel, y_train)\n\n# predict\npredictions = classifier.predict(x_test_multilabel)\nprint(accuracy_score(y_test,predictions))\nprint(metrics.f1_score(y_test, predictions, average = 'macro'))\nprint(metrics.f1_score(y_test, predictions, average = 'micro'))\nprint(metrics.hamming_loss(y_test,predictions))\n\n"
```

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

In [7]:

```
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 database:
QuestionsProcessed
```

In [8]:

```
# 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 500001;")

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 database:
QuestionsProcessed
Cleared All the rows

### 4.5.1 Preprocessing of questions

1. Separate Code from Body
2. Remove Spcial characters from Question title and description (not in code)
3. **Give more weightage to title : Add title three times to the question**
4. Remove stop words (Except 'C')
5. Remove HTML Tags
6. Convert all the characters into small letters
7. Use SnowballStemmer to stem the words

In [9]:

```
#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], str(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')

    # adding title three time to the data to increase its weight
    # add tags string to the training data

    question=str(title)+" "+str(title)+" "+str(title)+" "+question

#    if questions_proccesed<=train_datasize:
#        question=str(title)+" "+str(title)+" "+str(title)+" "+question+" "+str(tags)
#    else:
#        question=str(title)+" "+str(title)+" "+str(title)+" "+question

    question=re.sub(r'[^A-Za-z0-9#+.\-]+',' ',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)
    question = ' '.join([i for i in question.split() if (len(i)>1 or i == '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: 607
Percent of questions containing code: 57
Time taken to run this cell : 0:22:43.404308
```

In [10]:

```
# 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 quesitons after preprocessing of data**

In [11]:

```
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 in silverlight dynam datagrid bind in silverlight dynam datagrid bind in silverlight should do bind for datagrid dynam at code wrote the code as below when debug this code block it seem that it doe bind correct but grid come with no column on form whi doesn come grid with column although did necessari bind nthank for the repli in advance..',)
----------------------------------------------------------------------------------------------------
('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 the guid in this link to instal jstl but got the follow error when tri to launch my jsp page java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid the taglib declar is instal jstl 1.1 under in tomcat webapp and tri to do the same in my project but it didn work also tri version 1.2 of jstl and still the same messag how is this caus and how can solv it',)
----------------------------------------------------------------------------------------------------
('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 the follow code it display how is this caus and how can solv it',)
----------------------------------------------------------------------------------------------------
('better way to updat feed on fb with php sdk better way to updat feed on fb with php sdk better way to updat feed on fb with php sdk am novic with the facebook api have read so mani tutori and am still confused.i find that can post to the feed with api method like this and that is correct second way is use curl someth like this which way is better',)
----------------------------------------------------------------------------------------------------
('btnadd click event open two window after record ad btnadd click event open two window after record ad btnadd click event open two window after record ad open window search.aspx use below code hav add button in search.aspx nwhen insert record from btnadd click event then it open anoth window nafter insert record have to close that window how can do that',)
----------------------------------------------------------------------------------------------------
('sql inject issu prevent correct form submiss php sql inject issu prevent correct form submiss php sql inject issu prevent correct form submiss php sove been check everyth can think of to make sure my input field are all safe from ani type of sql inject the good news is they are safe the bad news is there is one tag that mess up the form submiss in place that it shouldn even touch and can for the life of me figur out whi this is the exact html use it in templat file so forgiv the okay this is the entir php script that get execut until it see that there is no data post none of the forum field are be post the problem is when use someth such as or in the titl field none of the data get post am current use print post to see what is be submit and noth is it work flawless for ani other statement though and should also mention that this script work flawless on my local machin but when use my host come across the problem state abov this is list of all the input test that mess it up',)
----------------------------------------------------------------------------------------------------
('countabl subaddit of the lebesgu measur countabl subaddit of the lebesgu measur countabl subaddit of the lebesgu measur let lbrace rbrace be sequenc of set in sigma -algebra mathcal want to show that left bigcup right leq sum left right where is countabl addit measur defin for all set in sigma algebra mathcal think have to use the monoton properti somewher in the proof but don how to start it appreci littl help nthank ad from han answer make the follow addit from the construct given in han answer it is clear the bigcup bigcup and cap emptyset for all neq so left bigcup right left bigcup right sum left right also from the construct we have subset for all and so by monoton we have left right leq left right final we would have sum leq sum and the result follow',)
----------------------------------------------------------------------------------------------------
('hql equival to this sql queri hql equival to this sql queri hql equival to this sql queri what is hql queri for the abov replac all my name with class and p
```

('hql equival to this sql queri hql equival to this sql queri what is hql queri for the abov replac all my name with class and p
roperti name but error occur hql error',)
----------------------------------------------------------------------------------------------------
('undefin symbol for architectur i386 objc class skpsmtpmessag referenc from error undefin symbol for architectur i386 objc class skpsmtpmessag ref
erenc from error undefin symbol for architectur i386 objc class skpsmtpmessag referenc from error have import framework for send email from applic
in background have import framework i.e skpsmtpmessag can somebodi suggest me whi am get this error collect2 ld return exit status have import fr
amework correct sorc from which have taken framework and follow is mfmailcomposeviewcontrol question lock the field updat answer is you just dra
g and drop folder over the project and click copi nthat it',)
----------------------------------------------------------------------------------------------------

### Saving Preprocessed data to a Database

In [12]:

```
#Taking 0.5 Million 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()
```

In [13]:

```
preprocessed_data.head()
```

Out[13]:

| | question | tags |
|---|---|---|
| 0 | dynam datagrid bind in silverlight dynam datag... | c# silverlight data-binding |
| 1 | dynam datagrid bind in silverlight dynam datag... | c# silverlight data-binding columns |
| 2 | java.lang.noclassdeffounderror javax servlet j... | jsp jstl |
| 3 | java.sql.sqlexcept microsoft odbc driver manag... | java jdbc |
| 4 | better way to updat feed on fb with php sdk be... | facebook api facebook-php-sdk |

In [14]:

```
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 string Tags to multilable output variables

In [15]:

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

### Selecting 500 Tags

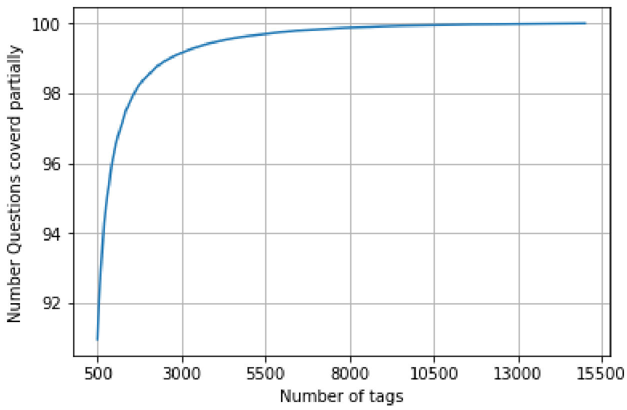
In [16]:

```
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))
```

In [17]:

```
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.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

In [18]:

```
# 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

In [19]:

```
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],:]
```

In [20]:

```
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

In [21]:

```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, smooth_idf=True, norm="l2", tokenizer = lambda x: x.split(), 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 : 0:05:48.254292

In [22]:

```
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, 192729) Y : (400000, 500)
Dimensions of test data X: (100000, 192729) Y: (100000, 500)

### 4.5.3 Applying SGD Classifier with loss='log' with OneVsRest Classifier

In [23]:

```
metrics.SCORERS.keys()
```

Out[23]:

```
dict_keys(['max_error', 'average_precision', 'jaccard_samples', 'jaccard', 'f1_macro', 'precision_macro', 'f1_micro', 'r2', 'neg_mean_absolute_error', 're
call_micro', 'f1_weighted', 'explained_variance', 'jaccard_micro', 'precision_micro', 'fowlkes_mallows_score', 'adjusted_rand_score', 'balanced_accura
cy', 'neg_log_loss', 'f1', 'completeness_score', 'roc_auc', 'precision_weighted', 'recall_samples', 'precision', 'recall_macro', 'jaccard_weighted', 'recall', '
precision_samples', 'normalized_mutual_info_score', 'neg_median_absolute_error', 'recall_weighted', 'neg_mean_squared_error', 'neg_mean_square
d_log_error', 'adjusted_mutual_info_score', 'brier_score_loss', 'accuracy', 'v_measure_score', 'jaccard_macro', 'homogeneity_score', 'mutual_info_sco
re', 'f1_samples'])
```

In [24]:

```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
```

In [32]:

```
start = datetime.now()
parameters= {'estimator__alpha':[10**-3, 10**-2, 10**-1, 1]}

rscv = RandomizedSearchCV(estimator = OneVsRestClassifier(SGDClassifier(loss='log', penalty='l1', n_jobs= -1)), param_distributions = parameter
s,
                        n_jobs= -1, return_train_score=True, scoring ='f1_micro')
rscv.fit(x_train_multilabel, y_train)
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 5:20:54.521809
```

In [34]:

```
print('Best parameters: \n', rscv.best_estimator_)
print()
print('ROC AUC Score: ', rscv.score(x_test_multilabel, y_test))
```

```
Best parameters:
 OneVsRestClassifier(estimator=SGDClassifier(alpha=0.001, average=False,
                                            class_weight=None,
                                            early_stopping=False, epsilon=0.1,
                                            eta0=0.0, fit_intercept=True,
                                            l1_ratio=0.15,
                                            learning_rate='optimal', loss='log',
                                            max_iter=1000, n_iter_no_change=5,
                                            n_jobs=-1, penalty='l1',
                                            power_t=0.5, random_state=None,
                                            shuffle=True, tol=0.001,
                                            validation_fraction=0.1, verbose=0,
                                            warm_start=False),
                    n_jobs=None)

ROC AUC Score:  0.03726603243900106
```

In [27]:

```
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 : 0.22845
Hamming loss  0.00281536
Micro-average quality numbers
Precision: 0.7291, Recall: 0.3025, F1-measure: 0.4276
Macro-average quality numbers
Precision: 0.5457, Recall: 0.2320, F1-measure: 0.3085
             precision    recall  f1-score   support

          0       0.95      0.61      0.74      5519
          1       0.69      0.24      0.36      8190
          2       0.81      0.36      0.49      6529
          3       0.82      0.40      0.54      3231
          4       0.82      0.39      0.53      6430
          5       0.81      0.32      0.46      2879
          6       0.88      0.48      0.62      5086
          7       0.88      0.53      0.66      4533
          8       0.61      0.12      0.20      3000
          9       0.81      0.51      0.63      2765
         10       0.60      0.15      0.24      3051
         11       0.72      0.31      0.44      3009
         12       0.66      0.22      0.33      2630
         13       0.74      0.21      0.33      1426
         14       0.91      0.51      0.65      2548
         15       0.70      0.16      0.26      2371
         16       0.64      0.20      0.30       873
         17       0.89      0.59      0.71      2151
         18       0.63      0.21      0.32      2204
         19       0.72      0.38      0.49       831
         20       0.78      0.40      0.53      1860
         21       0.26      0.07      0.11      2023
         22       0.48      0.18      0.26      1513
         23       0.91      0.46      0.61      1207
         24       0.57      0.28      0.37       506
         25       0.70      0.29      0.41       425
         26       0.64      0.37      0.47       793
         27       0.60      0.28      0.38      1291
         28       0.75      0.35      0.48      1208
         29       0.43      0.08      0.14       406
         30       0.79      0.17      0.28       504
         31       0.29      0.09      0.14       732
         32       0.58      0.19      0.29       441
         33       0.56      0.15      0.23      1645
         34       0.73      0.25      0.37      1058
         35       0.82      0.53      0.64       946
         36       0.70      0.18      0.29       644
         37       0.97      0.65      0.78       136
         38       0.63      0.33      0.44       570
         39       0.84      0.28      0.42       766
         40       0.63      0.26      0.36      1132
         41       0.39      0.13      0.19       174
         42       0.80      0.47      0.59       210
         43       0.82      0.39      0.53       433
         44       0.66      0.47      0.55       626
         45       0.75      0.29      0.42       852
         46       0.75      0.38      0.50       534
         47       0.35      0.12      0.18       350
         48       0.75      0.48      0.58       496
         49       0.80      0.58      0.67       785
         50       0.18      0.04      0.07       475
         51       0.42      0.08      0.14       305
         52       0.43      0.02      0.05       251
         53       0.69      0.38      0.49       914
         54       0.45      0.14      0.22       728
         55       0.00      0.00      0.00       258
         56       0.48      0.18      0.26       821
         57       0.45      0.08      0.13       541
         58       0.77      0.26      0.39       748
         59       0.94      0.61      0.74       724
         60       0.28      0.05      0.08       660
         61       0.85      0.17      0.29       235
         62       0.91      0.68      0.78       718
         63       0.85      0.60      0.70       468
         64       0.51      0.26      0.35       191
         65       0.35      0.10      0.16       429
         66       0.32      0.06      0.09       415
         67       0.76      0.45      0.56       274
```

| | | | | |
|---|---|---|---|---|
| 67 | [illegible] | [illegible] | [illegible] | [illegible] |
| 68 | 0.83 | 0.50 | 0.62 | 510 |
| 69 | 0.68 | 0.43 | 0.53 | 466 |
| 70 | 0.28 | 0.06 | 0.10 | 305 |
| 71 | 0.51 | 0.17 | 0.26 | 247 |
| 72 | 0.77 | 0.45 | 0.57 | 401 |
| 73 | 0.98 | 0.73 | 0.84 | 86 |
| 74 | 0.77 | 0.33 | 0.47 | 120 |
| 75 | 0.91 | 0.68 | 0.78 | 129 |
| 76 | 0.40 | 0.00 | 0.01 | 473 |
| 77 | 0.32 | 0.18 | 0.23 | 143 |
| 78 | 0.79 | 0.41 | 0.54 | 347 |
| 79 | 0.73 | 0.24 | 0.36 | 479 |
| 80 | 0.51 | 0.28 | 0.36 | 279 |
| 81 | 0.79 | 0.16 | 0.27 | 461 |
| 82 | 0.20 | 0.01 | 0.02 | 298 |
| 83 | 0.78 | 0.43 | 0.56 | 396 |
| 84 | 0.59 | 0.28 | 0.38 | 184 |
| 85 | 0.65 | 0.19 | 0.29 | 573 |
| 86 | 0.46 | 0.04 | 0.07 | 325 |
| 87 | 0.47 | 0.20 | 0.28 | 273 |
| 88 | 0.38 | 0.17 | 0.23 | 135 |
| 89 | 0.31 | 0.09 | 0.14 | 232 |
| 90 | 0.54 | 0.27 | 0.36 | 409 |
| 91 | 0.64 | 0.23 | 0.34 | 420 |
| 92 | 0.76 | 0.50 | 0.60 | 408 |
| 93 | 0.68 | 0.46 | 0.55 | 241 |
| 94 | 0.25 | 0.02 | 0.04 | 211 |
| 95 | 0.27 | 0.05 | 0.08 | 277 |
| 96 | 0.28 | 0.03 | 0.06 | 410 |
| 97 | 0.89 | 0.27 | 0.41 | 501 |
| 98 | 0.77 | 0.58 | 0.66 | 136 |
| 99 | 0.52 | 0.25 | 0.34 | 239 |
| 100 | 0.59 | 0.11 | 0.19 | 324 |
| 101 | 0.94 | 0.54 | 0.68 | 277 |
| 102 | 0.91 | 0.68 | 0.78 | 613 |
| 103 | 0.53 | 0.13 | 0.21 | 157 |
| 104 | 0.21 | 0.05 | 0.08 | 295 |
| 105 | 0.85 | 0.28 | 0.43 | 334 |
| 106 | 0.78 | 0.09 | 0.17 | 335 |
| 107 | 0.75 | 0.47 | 0.58 | 389 |
| 108 | 0.52 | 0.18 | 0.26 | 251 |
| 109 | 0.54 | 0.38 | 0.44 | 317 |
| 110 | 0.75 | 0.05 | 0.09 | 187 |
| 111 | 0.71 | 0.07 | 0.13 | 140 |
| 112 | 0.68 | 0.30 | 0.41 | 154 |
| 113 | 0.65 | 0.16 | 0.26 | 332 |
| 114 | 0.43 | 0.23 | 0.30 | 323 |
| 115 | 0.48 | 0.17 | 0.26 | 344 |
| 116 | 0.76 | 0.47 | 0.58 | 370 |
| 117 | 0.56 | 0.19 | 0.29 | 313 |
| 118 | 0.78 | 0.62 | 0.69 | 874 |
| 119 | 0.45 | 0.17 | 0.25 | 293 |
| 120 | 0.00 | 0.00 | 0.00 | 200 |
| 121 | 0.77 | 0.45 | 0.56 | 463 |
| 122 | 0.36 | 0.07 | 0.11 | 119 |
| 123 | 0.50 | 0.01 | 0.02 | 256 |
| 124 | 0.91 | 0.70 | 0.79 | 195 |
| 125 | 0.39 | 0.09 | 0.14 | 138 |
| 126 | 0.81 | 0.43 | 0.56 | 376 |
| 127 | 0.17 | 0.03 | 0.05 | 122 |
| 128 | 0.17 | 0.02 | 0.04 | 252 |
| 129 | 0.56 | 0.06 | 0.11 | 144 |
| 130 | 0.46 | 0.08 | 0.14 | 150 |
| 131 | 0.25 | 0.01 | 0.02 | 210 |
| 132 | 0.67 | 0.23 | 0.34 | 361 |
| 133 | 0.94 | 0.52 | 0.67 | 453 |
| 134 | 0.87 | 0.72 | 0.79 | 124 |
| 135 | 0.12 | 0.01 | 0.02 | 91 |
| 136 | 0.70 | 0.16 | 0.27 | 128 |
| 137 | 0.59 | 0.33 | 0.42 | 218 |
| 138 | 0.88 | 0.12 | 0.22 | 243 |
| 139 | 0.39 | 0.15 | 0.22 | 149 |
| 140 | 0.76 | 0.40 | 0.53 | 318 |
| 141 | 0.29 | 0.09 | 0.13 | 159 |
| 142 | 0.66 | 0.33 | 0.44 | 274 |
| 143 | 0.86 | 0.70 | 0.78 | 362 |
| 144 | 0.55 | 0.18 | 0.27 | 118 |
| 145 | 0.67 | 0.38 | 0.48 | 164 |
| 146 | 0.59 | 0.25 | 0.35 | 461 |
| 147 | 0.67 | 0.38 | 0.48 | 159 |
| 148 | 0.35 | 0.13 | 0.19 | 166 |
| 149 | 0.99 | 0.41 | 0.58 | 346 |

| | | | | |
|---|---|---|---|---|
| 150 | 0.65 | 0.06 | 0.10 | 350 |
| 151 | 0.94 | 0.58 | 0.72 | 55 |
| 152 | 0.81 | 0.44 | 0.57 | 387 |
| 153 | 0.43 | 0.07 | 0.12 | 150 |
| 154 | 0.59 | 0.11 | 0.18 | 281 |
| 155 | 0.31 | 0.05 | 0.09 | 202 |
| 156 | 0.74 | 0.62 | 0.67 | 130 |
| 157 | 0.26 | 0.06 | 0.09 | 245 |
| 158 | 0.88 | 0.56 | 0.69 | 177 |
| 159 | 0.51 | 0.21 | 0.30 | 130 |
| 160 | 0.53 | 0.12 | 0.20 | 336 |
| 161 | 0.93 | 0.54 | 0.68 | 220 |
| 162 | 0.12 | 0.02 | 0.03 | 229 |
| 163 | 0.90 | 0.39 | 0.54 | 316 |
| 164 | 0.74 | 0.33 | 0.46 | 283 |
| 165 | 0.55 | 0.26 | 0.35 | 197 |
| 166 | 0.49 | 0.25 | 0.33 | 101 |
| 167 | 0.46 | 0.14 | 0.22 | 231 |
| 168 | 0.55 | 0.19 | 0.28 | 370 |
| 169 | 0.44 | 0.17 | 0.25 | 258 |
| 170 | 0.29 | 0.05 | 0.08 | 101 |
| 171 | 0.40 | 0.19 | 0.26 | 89 |
| 172 | 0.45 | 0.25 | 0.32 | 193 |
| 173 | 0.38 | 0.20 | 0.27 | 309 |
| 174 | 0.55 | 0.13 | 0.21 | 172 |
| 175 | 0.93 | 0.67 | 0.78 | 95 |
| 176 | 0.95 | 0.55 | 0.69 | 346 |
| 177 | 0.94 | 0.41 | 0.57 | 322 |
| 178 | 0.63 | 0.40 | 0.49 | 232 |
| 179 | 0.36 | 0.04 | 0.07 | 125 |
| 180 | 0.49 | 0.23 | 0.31 | 145 |
| 181 | 0.37 | 0.09 | 0.15 | 77 |
| 182 | 0.13 | 0.02 | 0.03 | 182 |
| 183 | 0.61 | 0.28 | 0.38 | 257 |
| 184 | 0.15 | 0.01 | 0.03 | 216 |
| 185 | 0.35 | 0.08 | 0.13 | 242 |
| 186 | 0.42 | 0.15 | 0.22 | 165 |
| 187 | 0.76 | 0.53 | 0.62 | 263 |
| 188 | 0.30 | 0.08 | 0.13 | 174 |
| 189 | 0.71 | 0.25 | 0.37 | 136 |
| 190 | 0.92 | 0.46 | 0.61 | 202 |
| 191 | 0.33 | 0.08 | 0.13 | 134 |
| 192 | 0.75 | 0.40 | 0.52 | 230 |
| 193 | 0.31 | 0.10 | 0.15 | 90 |
| 194 | 0.57 | 0.42 | 0.48 | 185 |
| 195 | 0.18 | 0.03 | 0.05 | 156 |
| 196 | 0.35 | 0.04 | 0.07 | 160 |
| 197 | 0.52 | 0.04 | 0.08 | 266 |
| 198 | 0.42 | 0.03 | 0.05 | 284 |
| 199 | 0.28 | 0.05 | 0.08 | 145 |
| 200 | 0.93 | 0.67 | 0.78 | 212 |
| 201 | 0.66 | 0.21 | 0.32 | 317 |
| 202 | 0.78 | 0.50 | 0.61 | 427 |
| 203 | 0.29 | 0.07 | 0.11 | 232 |
| 204 | 0.54 | 0.20 | 0.29 | 217 |
| 205 | 0.49 | 0.38 | 0.42 | 527 |
| 206 | 0.08 | 0.01 | 0.01 | 124 |
| 207 | 0.43 | 0.06 | 0.10 | 103 |
| 208 | 0.87 | 0.46 | 0.60 | 287 |
| 209 | 0.39 | 0.07 | 0.12 | 193 |
| 210 | 0.73 | 0.30 | 0.43 | 220 |
| 211 | 0.86 | 0.14 | 0.23 | 140 |
| 212 | 0.11 | 0.02 | 0.03 | 161 |
| 213 | 0.61 | 0.24 | 0.34 | 72 |
| 214 | 0.63 | 0.46 | 0.54 | 396 |
| 215 | 0.89 | 0.30 | 0.45 | 134 |
| 216 | 0.50 | 0.04 | 0.07 | 400 |
| 217 | 0.55 | 0.24 | 0.33 | 75 |
| 218 | 0.97 | 0.70 | 0.81 | 219 |
| 219 | 0.78 | 0.35 | 0.48 | 210 |
| 220 | 0.93 | 0.55 | 0.69 | 298 |
| 221 | 0.97 | 0.58 | 0.72 | 266 |
| 222 | 0.75 | 0.38 | 0.50 | 290 |
| 223 | 0.08 | 0.01 | 0.01 | 128 |
| 224 | 0.85 | 0.38 | 0.52 | 159 |
| 225 | 0.59 | 0.24 | 0.34 | 164 |
| 226 | 0.62 | 0.35 | 0.44 | 144 |
| 227 | 0.53 | 0.29 | 0.37 | 276 |
| 228 | 0.21 | 0.01 | 0.02 | 235 |
| 229 | 0.20 | 0.00 | 0.01 | 216 |
| 230 | 0.33 | 0.16 | 0.22 | 228 |
| 231 | 0.69 | 0.42 | 0.52 | 64 |
| 232 | 0.38 | 0.05 | 0.09 | 103 |

| | | | | |
|---|---|---|---|---|
| 233 | 0.68 | 0.29 | 0.41 | 216 |
| 234 | 0.62 | 0.04 | 0.08 | 116 |
| 235 | 0.54 | 0.35 | 0.43 | 77 |
| 236 | 0.98 | 0.63 | 0.76 | 67 |
| 237 | 0.68 | 0.07 | 0.12 | 218 |
| 238 | 0.21 | 0.04 | 0.06 | 139 |
| 239 | 0.12 | 0.01 | 0.02 | 94 |
| 240 | 0.51 | 0.23 | 0.32 | 77 |
| 241 | 0.54 | 0.08 | 0.15 | 167 |
| 242 | 0.85 | 0.27 | 0.41 | 86 |
| 243 | 0.39 | 0.12 | 0.18 | 58 |
| 244 | 0.57 | 0.14 | 0.22 | 269 |
| 245 | 0.15 | 0.03 | 0.05 | 112 |
| 246 | 0.95 | 0.71 | 0.81 | 255 |
| 247 | 0.34 | 0.17 | 0.23 | 58 |
| 248 | 0.50 | 0.04 | 0.07 | 81 |
| 249 | 0.06 | 0.01 | 0.01 | 131 |
| 250 | 0.38 | 0.16 | 0.23 | 93 |
| 251 | 0.64 | 0.23 | 0.34 | 154 |
| 252 | 0.36 | 0.03 | 0.06 | 129 |
| 253 | 0.58 | 0.27 | 0.36 | 83 |
| 254 | 0.28 | 0.04 | 0.07 | 191 |
| 255 | 0.11 | 0.02 | 0.03 | 219 |
| 256 | 0.24 | 0.03 | 0.05 | 130 |
| 257 | 0.49 | 0.29 | 0.36 | 93 |
| 258 | 0.68 | 0.40 | 0.50 | 217 |
| 259 | 0.37 | 0.08 | 0.13 | 141 |
| 260 | 0.95 | 0.13 | 0.23 | 143 |
| 261 | 0.51 | 0.11 | 0.17 | 219 |
| 262 | 0.54 | 0.25 | 0.34 | 107 |
| 263 | 0.40 | 0.23 | 0.29 | 236 |
| 264 | 0.32 | 0.14 | 0.20 | 119 |
| 265 | 0.26 | 0.08 | 0.13 | 72 |
| 266 | 0.00 | 0.00 | 0.00 | 70 |
| 267 | 0.29 | 0.12 | 0.17 | 107 |
| 268 | 0.66 | 0.38 | 0.49 | 169 |
| 269 | 0.37 | 0.12 | 0.18 | 129 |
| 270 | 0.73 | 0.52 | 0.61 | 159 |
| 271 | 0.86 | 0.28 | 0.43 | 190 |
| 272 | 0.62 | 0.20 | 0.30 | 248 |
| 273 | 0.91 | 0.67 | 0.77 | 264 |
| 274 | 0.91 | 0.59 | 0.72 | 105 |
| 275 | 0.60 | 0.06 | 0.11 | 104 |
| 276 | 0.06 | 0.01 | 0.02 | 115 |
| 277 | 0.85 | 0.61 | 0.71 | 170 |
| 278 | 0.57 | 0.16 | 0.25 | 145 |
| 279 | 0.92 | 0.51 | 0.66 | 230 |
| 280 | 0.51 | 0.35 | 0.41 | 80 |
| 281 | 0.68 | 0.54 | 0.61 | 217 |
| 282 | 0.74 | 0.46 | 0.57 | 175 |
| 283 | 0.34 | 0.04 | 0.08 | 269 |
| 284 | 0.56 | 0.19 | 0.28 | 74 |
| 285 | 0.85 | 0.46 | 0.59 | 206 |
| 286 | 0.90 | 0.58 | 0.70 | 227 |
| 287 | 0.84 | 0.25 | 0.38 | 130 |
| 288 | 0.36 | 0.04 | 0.07 | 129 |
| 289 | 0.33 | 0.01 | 0.02 | 80 |
| 290 | 0.11 | 0.04 | 0.06 | 99 |
| 291 | 0.79 | 0.27 | 0.41 | 208 |
| 292 | 0.43 | 0.04 | 0.08 | 67 |
| 293 | 0.89 | 0.39 | 0.54 | 109 |
| 294 | 0.35 | 0.20 | 0.26 | 140 |
| 295 | 0.20 | 0.07 | 0.10 | 241 |
| 296 | 0.31 | 0.15 | 0.20 | 72 |
| 297 | 0.19 | 0.03 | 0.05 | 107 |
| 298 | 0.87 | 0.33 | 0.48 | 61 |
| 299 | 0.92 | 0.30 | 0.45 | 77 |
| 300 | 0.20 | 0.04 | 0.06 | 111 |
| 301 | 0.00 | 0.00 | 0.00 | 126 |
| 302 | 0.00 | 0.00 | 0.00 | 73 |
| 303 | 0.60 | 0.37 | 0.46 | 176 |
| 304 | 0.97 | 0.68 | 0.80 | 230 |
| 305 | 0.97 | 0.56 | 0.71 | 156 |
| 306 | 0.53 | 0.29 | 0.38 | 146 |
| 307 | 0.20 | 0.04 | 0.07 | 98 |
| 308 | 0.00 | 0.00 | 0.00 | 78 |
| 309 | 0.57 | 0.04 | 0.08 | 94 |
| 310 | 0.79 | 0.30 | 0.44 | 162 |
| 311 | 0.77 | 0.47 | 0.59 | 116 |
| 312 | 0.44 | 0.21 | 0.29 | 57 |
| 313 | 1.00 | 0.02 | 0.03 | 65 |
| 314 | 0.44 | 0.33 | 0.38 | 138 |

| | | | | |
|---|---|---|---|---|
| 315 | 0.60 | 0.19 | 0.29 | 195 |
| 316 | 0.43 | 0.22 | 0.29 | 69 |
| 317 | 0.37 | 0.08 | 0.13 | 134 |
| 318 | 0.47 | 0.28 | 0.35 | 148 |
| 319 | 0.83 | 0.39 | 0.53 | 161 |
| 320 | 0.23 | 0.12 | 0.16 | 104 |
| 321 | 0.86 | 0.51 | 0.64 | 156 |
| 322 | 0.54 | 0.25 | 0.35 | 134 |
| 323 | 0.57 | 0.38 | 0.45 | 232 |
| 324 | 0.47 | 0.15 | 0.23 | 92 |
| 325 | 0.42 | 0.16 | 0.23 | 197 |
| 326 | 0.05 | 0.01 | 0.01 | 126 |
| 327 | 0.33 | 0.03 | 0.05 | 115 |
| 328 | 0.98 | 0.60 | 0.75 | 198 |
| 329 | 0.64 | 0.30 | 0.40 | 125 |
| 330 | 0.75 | 0.15 | 0.25 | 81 |
| 331 | 0.57 | 0.09 | 0.15 | 94 |
| 332 | 1.00 | 0.02 | 0.04 | 56 |
| 333 | 0.11 | 0.01 | 0.02 | 260 |
| 334 | 0.00 | 0.00 | 0.00 | 60 |
| 335 | 0.30 | 0.06 | 0.11 | 110 |
| 336 | 0.64 | 0.35 | 0.45 | 71 |
| 337 | 0.20 | 0.05 | 0.07 | 66 |
| 338 | 0.42 | 0.23 | 0.30 | 150 |
| 339 | 0.00 | 0.00 | 0.00 | 54 |
| 340 | 0.86 | 0.50 | 0.63 | 195 |
| 341 | 0.90 | 0.11 | 0.20 | 79 |
| 342 | 0.44 | 0.21 | 0.29 | 38 |
| 343 | 0.74 | 0.40 | 0.52 | 43 |
| 344 | 0.50 | 0.16 | 0.24 | 68 |
| 345 | 0.64 | 0.40 | 0.49 | 73 |
| 346 | 0.12 | 0.02 | 0.03 | 116 |
| 347 | 0.92 | 0.32 | 0.47 | 111 |
| 348 | 0.35 | 0.10 | 0.15 | 63 |
| 349 | 0.87 | 0.58 | 0.69 | 104 |
| 350 | 0.59 | 0.39 | 0.47 | 44 |
| 351 | 0.62 | 0.12 | 0.21 | 40 |
| 352 | 1.00 | 0.35 | 0.51 | 136 |
| 353 | 0.36 | 0.19 | 0.24 | 54 |
| 354 | 0.60 | 0.02 | 0.04 | 134 |
| 355 | 0.55 | 0.24 | 0.34 | 120 |
| 356 | 0.56 | 0.18 | 0.27 | 228 |
| 357 | 0.68 | 0.23 | 0.34 | 269 |
| 358 | 0.74 | 0.29 | 0.41 | 80 |
| 359 | 0.83 | 0.39 | 0.53 | 140 |
| 360 | 0.38 | 0.12 | 0.18 | 125 |
| 361 | 0.92 | 0.59 | 0.71 | 169 |
| 362 | 0.11 | 0.04 | 0.05 | 56 |
| 363 | 0.95 | 0.61 | 0.74 | 154 |
| 364 | 0.27 | 0.05 | 0.09 | 58 |
| 365 | 0.27 | 0.14 | 0.19 | 71 |
| 366 | 1.00 | 0.61 | 0.76 | 54 |
| 367 | 0.30 | 0.03 | 0.05 | 116 |
| 368 | 0.00 | 0.00 | 0.00 | 54 |
| 369 | 0.00 | 0.00 | 0.00 | 71 |
| 370 | 0.33 | 0.05 | 0.09 | 61 |
| 371 | 0.38 | 0.04 | 0.08 | 71 |
| 372 | 0.66 | 0.44 | 0.53 | 52 |
| 373 | 0.78 | 0.30 | 0.43 | 150 |
| 374 | 0.44 | 0.16 | 0.24 | 93 |
| 375 | 0.15 | 0.03 | 0.05 | 67 |
| 376 | 0.00 | 0.00 | 0.00 | 76 |
| 377 | 0.58 | 0.07 | 0.12 | 106 |
| 378 | 0.00 | 0.00 | 0.00 | 86 |
| 379 | 0.50 | 0.07 | 0.12 | 14 |
| 380 | 1.00 | 0.32 | 0.48 | 122 |
| 381 | 0.17 | 0.02 | 0.03 | 104 |
| 382 | 0.44 | 0.11 | 0.17 | 66 |
| 383 | 0.54 | 0.26 | 0.35 | 110 |
| 384 | 0.00 | 0.00 | 0.00 | 155 |
| 385 | 0.33 | 0.06 | 0.10 | 50 |
| 386 | 0.22 | 0.08 | 0.11 | 64 |
| 387 | 0.43 | 0.06 | 0.11 | 93 |
| 388 | 0.64 | 0.29 | 0.40 | 102 |
| 389 | 0.17 | 0.01 | 0.02 | 108 |
| 390 | 0.96 | 0.60 | 0.74 | 178 |
| 391 | 0.65 | 0.17 | 0.27 | 115 |
| 392 | 0.78 | 0.33 | 0.47 | 42 |
| 393 | 0.00 | 0.00 | 0.00 | 134 |
| 394 | 0.25 | 0.01 | 0.02 | 112 |
| 395 | 0.48 | 0.12 | 0.20 | 176 |
| 396 | 0.40 | 0.06 | 0.11 | 125 |
| 397 | 0.73 | 0.20 | 0.31 | 224 |

| | | | | |
|---|---|---|---|---|
| 398 | 0.88 | 0.48 | 0.62 | 63 |
| 399 | 0.00 | 0.00 | 0.00 | 59 |
| 400 | 0.42 | 0.25 | 0.32 | 63 |
| 401 | 0.39 | 0.12 | 0.19 | 98 |
| 402 | 0.53 | 0.12 | 0.20 | 162 |
| 403 | 0.50 | 0.14 | 0.22 | 83 |
| 404 | 0.73 | 0.84 | 0.78 | 19 |
| 405 | 0.21 | 0.03 | 0.06 | 92 |
| 406 | 0.86 | 0.15 | 0.25 | 41 |
| 407 | 0.71 | 0.28 | 0.40 | 43 |
| 408 | 0.79 | 0.26 | 0.39 | 160 |
| 409 | 0.22 | 0.12 | 0.16 | 50 |
| 410 | 0.00 | 0.00 | 0.00 | 19 |
| 411 | 0.30 | 0.06 | 0.10 | 175 |
| 412 | 0.38 | 0.07 | 0.12 | 72 |
| 413 | 0.50 | 0.02 | 0.04 | 95 |
| 414 | 0.17 | 0.04 | 0.07 | 97 |
| 415 | 0.27 | 0.08 | 0.13 | 48 |
| 416 | 0.45 | 0.27 | 0.33 | 83 |
| 417 | 0.40 | 0.05 | 0.09 | 40 |
| 418 | 0.43 | 0.11 | 0.18 | 91 |
| 419 | 0.51 | 0.24 | 0.33 | 90 |
| 420 | 0.30 | 0.24 | 0.27 | 37 |
| 421 | 0.00 | 0.00 | 0.00 | 66 |
| 422 | 0.60 | 0.29 | 0.39 | 73 |
| 423 | 0.46 | 0.21 | 0.29 | 56 |
| 424 | 0.89 | 0.76 | 0.82 | 33 |
| 425 | 0.00 | 0.00 | 0.00 | 76 |
| 426 | 0.23 | 0.04 | 0.06 | 81 |
| 427 | 1.00 | 0.62 | 0.77 | 150 |
| 428 | 0.95 | 0.66 | 0.78 | 29 |
| 429 | 0.99 | 0.46 | 0.62 | 389 |
| 430 | 0.64 | 0.30 | 0.41 | 167 |
| 431 | 0.50 | 0.07 | 0.12 | 123 |
| 432 | 0.41 | 0.28 | 0.33 | 39 |
| 433 | 0.32 | 0.15 | 0.20 | 82 |
| 434 | 1.00 | 0.59 | 0.74 | 66 |
| 435 | 0.51 | 0.30 | 0.38 | 93 |
| 436 | 0.50 | 0.20 | 0.28 | 87 |
| 437 | 0.20 | 0.03 | 0.06 | 86 |
| 438 | 0.71 | 0.40 | 0.52 | 104 |
| 439 | 0.60 | 0.12 | 0.20 | 100 |
| 440 | 0.43 | 0.02 | 0.04 | 141 |
| 441 | 0.51 | 0.35 | 0.41 | 110 |
| 442 | 0.35 | 0.11 | 0.17 | 123 |
| 443 | 0.44 | 0.10 | 0.16 | 71 |
| 444 | 0.45 | 0.05 | 0.08 | 109 |
| 445 | 0.45 | 0.19 | 0.26 | 48 |
| 446 | 0.44 | 0.24 | 0.31 | 76 |
| 447 | 0.07 | 0.03 | 0.04 | 38 |
| 448 | 0.70 | 0.48 | 0.57 | 81 |
| 449 | 0.58 | 0.11 | 0.18 | 132 |
| 450 | 0.43 | 0.22 | 0.29 | 81 |
| 451 | 0.88 | 0.28 | 0.42 | 76 |
| 452 | 0.00 | 0.00 | 0.00 | 44 |
| 453 | 0.00 | 0.00 | 0.00 | 44 |
| 454 | 0.88 | 0.41 | 0.56 | 70 |
| 455 | 0.29 | 0.03 | 0.06 | 155 |
| 456 | 0.38 | 0.14 | 0.20 | 43 |
| 457 | 0.48 | 0.17 | 0.25 | 72 |
| 458 | 0.29 | 0.08 | 0.13 | 62 |
| 459 | 0.77 | 0.14 | 0.24 | 69 |
| 460 | 0.00 | 0.00 | 0.00 | 119 |
| 461 | 0.89 | 0.10 | 0.18 | 79 |
| 462 | 0.45 | 0.11 | 0.17 | 47 |
| 463 | 0.30 | 0.03 | 0.05 | 104 |
| 464 | 0.69 | 0.29 | 0.41 | 106 |
| 465 | 0.60 | 0.05 | 0.09 | 64 |
| 466 | 0.59 | 0.26 | 0.36 | 173 |
| 467 | 0.79 | 0.32 | 0.45 | 107 |
| 468 | 0.85 | 0.09 | 0.16 | 126 |
| 469 | 0.25 | 0.01 | 0.02 | 114 |
| 470 | 0.94 | 0.73 | 0.82 | 140 |
| 471 | 0.94 | 0.19 | 0.32 | 79 |
| 472 | 0.36 | 0.25 | 0.30 | 143 |
| 473 | 0.66 | 0.27 | 0.39 | 158 |
| 474 | 0.45 | 0.07 | 0.11 | 138 |
| 475 | 0.00 | 0.00 | 0.00 | 59 |
| 476 | 0.58 | 0.25 | 0.35 | 88 |
| 477 | 0.88 | 0.55 | 0.67 | 176 |
| 478 | 0.89 | 0.71 | 0.79 | 24 |
| 479 | 0.14 | 0.01 | 0.02 | 92 |

```
480     0.85      0.45      0.59       100
481     0.39      0.12      0.18       103
482     0.36      0.14      0.20        74
483     0.84      0.55      0.67       105
484     0.00      0.00      0.00        83
485     0.00      0.00      0.00        82
486     0.24      0.08      0.12        71
487     0.40      0.17      0.24       120
488     0.00      0.00      0.00       105
489     0.79      0.26      0.40        87
490     1.00      0.69      0.81        32
491     1.00      0.01      0.03        69
492     0.00      0.00      0.00        49
493     0.17      0.01      0.02       117
494     0.57      0.20      0.29        61
495     1.00      0.15      0.26       344
496     0.30      0.13      0.19        52
497     0.61      0.14      0.23       137
498     0.36      0.04      0.07        98
499     0.75      0.15      0.25        79

   micro avg       0.73      0.30      0.43    173812
   macro avg       0.55      0.23      0.31    173812
weighted avg       0.67      0.30      0.40    173812
 samples avg       0.39      0.29      0.31    173812

Time taken to run this cell : 0:19:03.567914
```

### 4.5.4 Applying SGD Classifier with loss='hinge' with OneVsRest Classifier

In [25]:

```
start = datetime.now()
parameters= {'estimator__alpha':[10**-3, 10**-2, 10**-1, 1]}

rscv = RandomizedSearchCV(estimator = OneVsRestClassifier(SGDClassifier(loss='hinge', penalty='l1', n_jobs= -1)), param_distributions = paramet
ers,
                    n_jobs= -1, return_train_score=True, scoring ='f1_micro')
rscv.fit(x_train_multilabel, y_train)
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 2:42:20.782226
```

In [26]:

```
print('Best parameters: \n', rscv.best_estimator_)
print()
print('ROC AUC Score: ', rscv.score(x_test_multilabel, y_test))
```

```
Best parameters:
 OneVsRestClassifier(estimator=SGDClassifier(alpha=0.001, average=False,
                                            class_weight=None,
                                            early_stopping=False, epsilon=0.1,
                                            eta0=0.0, fit_intercept=True,
                                            l1_ratio=0.15,
                                            learning_rate='optimal',
                                            loss='hinge', max_iter=1000,
                                            n_iter_no_change=5, n_jobs=-1,
                                            penalty='l1', power_t=0.5,
                                            random_state=None, shuffle=True,
                                            tol=0.001, validation_fraction=0.1,
                                            verbose=0, warm_start=False),
                    n_jobs=None)

ROC AUC Score:  0.1138582559024317
```

In [27]:

```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', 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 : 0.24237
Hamming loss  0.0027244
Micro-average quality numbers
Precision: 0.8137, Recall: 0.2805, F1-measure: 0.4172
Macro-average quality numbers
Precision: 0.3977, Recall: 0.2025, F1-measure: 0.2472
             precision    recall  f1-score   support

          0       0.95      0.66      0.77      5519
          1       0.70      0.20      0.31      8190
          2       0.85      0.34      0.48      6529
          3       0.81      0.40      0.53      3231
          4       0.86      0.35      0.50      6430
          5       0.82      0.34      0.48      2879
          6       0.87      0.51      0.64      5086
          7       0.90      0.53      0.67      4533
          8       0.62      0.14      0.23      3000
          9       0.83      0.49      0.62      2765
         10       0.71      0.01      0.01      3051
         11       0.79      0.30      0.43      3009
         12       0.74      0.21      0.33      2630
         13       0.73      0.16      0.26      1426
         14       0.91      0.55      0.69      2548
         15       0.82      0.14      0.25      2371
         16       0.67      0.22      0.33       873
         17       0.89      0.59      0.71      2151
         18       0.75      0.18      0.28      2204
         19       0.69      0.48      0.56       831
         20       0.77      0.44      0.56      1860
         21       0.00      0.00      0.00      2023
         22       0.63      0.02      0.04      1513
         23       0.88      0.53      0.66      1207
         24       1.00      0.00      0.00       506
         25       0.73      0.35      0.47       425
         26       0.63      0.40      0.49       793
         27       0.67      0.23      0.34      1291
         28       0.81      0.35      0.49      1208
         29       0.50      0.00      0.01       406
         30       0.79      0.16      0.27       504
         31       0.00      0.00      0.00       732
         32       0.64      0.20      0.31       441
         33       0.00      0.00      0.00      1645
         34       0.72      0.27      0.39      1058
         35       0.83      0.59      0.69       946
         36       0.76      0.15      0.25       644
         37       0.94      0.78      0.85       136
         38       0.64      0.29      0.40       570
         39       0.87      0.26      0.40       766
         40       0.72      0.13      0.22      1132
         41       0.37      0.06      0.11       174
         42       0.75      0.57      0.65       210
         43       0.78      0.42      0.55       433
         44       0.68      0.55      0.61       626
         45       0.82      0.19      0.31       852
         46       0.81      0.22      0.35       534
         47       1.00      0.01      0.01       350
         48       0.74      0.56      0.63       496
         49       0.78      0.69      0.73       785
         50       0.00      0.00      0.00       475
         51       0.00      0.00      0.00       305
         52       0.00      0.00      0.00       251
         53       0.68      0.32      0.43       914
         54       0.00      0.00      0.00       728
         55       0.00      0.00      0.00       258
```

| | | | | |
|---|---|---|---|---|
| 56 | 0.00 | 0.00 | 0.00 | 821 |
| 57 | 0.00 | 0.00 | 0.00 | 541 |
| 58 | 0.80 | 0.26 | 0.40 | 748 |
| 59 | 0.91 | 0.67 | 0.77 | 724 |
| 60 | 0.75 | 0.01 | 0.02 | 660 |
| 61 | 0.89 | 0.20 | 0.33 | 235 |
| 62 | 0.91 | 0.74 | 0.81 | 718 |
| 63 | 0.82 | 0.66 | 0.73 | 468 |
| 64 | 0.50 | 0.33 | 0.40 | 191 |
| 65 | 0.00 | 0.00 | 0.00 | 429 |
| 66 | 0.00 | 0.00 | 0.00 | 415 |
| 67 | 0.76 | 0.54 | 0.63 | 274 |
| 68 | 0.83 | 0.57 | 0.68 | 510 |
| 69 | 0.66 | 0.51 | 0.57 | 466 |
| 70 | 0.00 | 0.00 | 0.00 | 305 |
| 71 | 0.00 | 0.00 | 0.00 | 247 |
| 72 | 0.83 | 0.50 | 0.62 | 401 |
| 73 | 0.94 | 0.78 | 0.85 | 86 |
| 74 | 0.81 | 0.43 | 0.57 | 120 |
| 75 | 0.88 | 0.74 | 0.80 | 129 |
| 76 | 0.00 | 0.00 | 0.00 | 473 |
| 77 | 0.36 | 0.18 | 0.24 | 143 |
| 78 | 0.76 | 0.54 | 0.63 | 347 |
| 79 | 0.76 | 0.22 | 0.35 | 479 |
| 80 | 0.74 | 0.18 | 0.29 | 279 |
| 81 | 0.83 | 0.22 | 0.35 | 461 |
| 82 | 0.00 | 0.00 | 0.00 | 298 |
| 83 | 0.76 | 0.51 | 0.61 | 396 |
| 84 | 0.56 | 0.03 | 0.05 | 184 |
| 85 | 1.00 | 0.02 | 0.04 | 573 |
| 86 | 0.67 | 0.02 | 0.05 | 325 |
| 87 | 0.00 | 0.00 | 0.00 | 273 |
| 88 | 0.00 | 0.00 | 0.00 | 135 |
| 89 | 0.00 | 0.00 | 0.00 | 232 |
| 90 | 0.00 | 0.00 | 0.00 | 409 |
| 91 | 0.00 | 0.00 | 0.00 | 420 |
| 92 | 0.75 | 0.57 | 0.65 | 408 |
| 93 | 0.64 | 0.48 | 0.55 | 241 |
| 94 | 0.00 | 0.00 | 0.00 | 211 |
| 95 | 0.00 | 0.00 | 0.00 | 277 |
| 96 | 0.00 | 0.00 | 0.00 | 410 |
| 97 | 0.87 | 0.46 | 0.60 | 501 |
| 98 | 0.73 | 0.72 | 0.72 | 136 |
| 99 | 0.00 | 0.00 | 0.00 | 239 |
| 100 | 0.00 | 0.00 | 0.00 | 324 |
| 101 | 0.92 | 0.71 | 0.80 | 277 |
| 102 | 0.92 | 0.70 | 0.79 | 613 |
| 103 | 0.00 | 0.00 | 0.00 | 157 |
| 104 | 0.00 | 0.00 | 0.00 | 295 |
| 105 | 0.86 | 0.30 | 0.45 | 334 |
| 106 | 0.96 | 0.20 | 0.33 | 335 |
| 107 | 0.78 | 0.49 | 0.60 | 389 |
| 108 | 0.00 | 0.00 | 0.00 | 251 |
| 109 | 0.51 | 0.46 | 0.48 | 317 |
| 110 | 0.00 | 0.00 | 0.00 | 187 |
| 111 | 0.77 | 0.14 | 0.24 | 140 |
| 112 | 0.67 | 0.38 | 0.48 | 154 |
| 113 | 0.76 | 0.13 | 0.22 | 332 |
| 114 | 0.00 | 0.00 | 0.00 | 323 |
| 115 | 0.00 | 0.00 | 0.00 | 344 |
| 116 | 0.76 | 0.53 | 0.63 | 370 |
| 117 | 0.61 | 0.12 | 0.21 | 313 |
| 118 | 0.78 | 0.66 | 0.72 | 874 |
| 119 | 0.00 | 0.00 | 0.00 | 293 |
| 120 | 0.00 | 0.00 | 0.00 | 200 |
| 121 | 0.75 | 0.53 | 0.62 | 463 |
| 122 | 0.00 | 0.00 | 0.00 | 119 |
| 123 | 0.00 | 0.00 | 0.00 | 256 |
| 124 | 0.87 | 0.79 | 0.83 | 195 |
| 125 | 0.00 | 0.00 | 0.00 | 138 |
| 126 | 0.81 | 0.38 | 0.52 | 376 |
| 127 | 0.00 | 0.00 | 0.00 | 122 |
| 128 | 0.00 | 0.00 | 0.00 | 252 |
| 129 | 0.00 | 0.00 | 0.00 | 144 |
| 130 | 0.00 | 0.00 | 0.00 | 150 |
| 131 | 0.00 | 0.00 | 0.00 | 210 |
| 132 | 0.00 | 0.00 | 0.00 | 361 |
| 133 | 0.93 | 0.56 | 0.70 | 453 |
| 134 | 0.86 | 0.82 | 0.84 | 124 |
| 135 | 0.00 | 0.00 | 0.00 | 91 |
| 136 | 0.80 | 0.12 | 0.22 | 128 |
| 137 | 0.61 | 0.29 | 0.39 | 218 |
| 138 | 0.00 | 0.00 | 0.00 | 243 |

| | | | | |
|---|---|---|---|---|
| 139 | 0.00 | 0.00 | 0.00 | 149 |
| 140 | 0.74 | 0.54 | 0.63 | 318 |
| 141 | 0.00 | 0.00 | 0.00 | 159 |
| 142 | 0.65 | 0.45 | 0.53 | 274 |
| 143 | 0.86 | 0.78 | 0.82 | 362 |
| 144 | 0.00 | 0.00 | 0.00 | 118 |
| 145 | 0.62 | 0.42 | 0.50 | 164 |
| 146 | 0.00 | 0.00 | 0.00 | 461 |
| 147 | 0.65 | 0.53 | 0.58 | 159 |
| 148 | 0.00 | 0.00 | 0.00 | 166 |
| 149 | 0.94 | 0.56 | 0.70 | 346 |
| 150 | 0.91 | 0.03 | 0.06 | 350 |
| 151 | 0.82 | 0.67 | 0.74 | 55 |
| 152 | 0.81 | 0.44 | 0.57 | 387 |
| 153 | 0.46 | 0.04 | 0.07 | 150 |
| 154 | 1.00 | 0.00 | 0.01 | 281 |
| 155 | 0.00 | 0.00 | 0.00 | 202 |
| 156 | 0.72 | 0.67 | 0.69 | 130 |
| 157 | 0.00 | 0.00 | 0.00 | 245 |
| 158 | 0.85 | 0.67 | 0.75 | 177 |
| 159 | 0.68 | 0.32 | 0.44 | 130 |
| 160 | 0.00 | 0.00 | 0.00 | 336 |
| 161 | 0.93 | 0.62 | 0.74 | 220 |
| 162 | 0.00 | 0.00 | 0.00 | 229 |
| 163 | 0.87 | 0.45 | 0.59 | 316 |
| 164 | 0.76 | 0.37 | 0.50 | 283 |
| 165 | 0.60 | 0.28 | 0.39 | 197 |
| 166 | 0.76 | 0.31 | 0.44 | 101 |
| 167 | 0.00 | 0.00 | 0.00 | 231 |
| 168 | 0.00 | 0.00 | 0.00 | 370 |
| 169 | 0.00 | 0.00 | 0.00 | 258 |
| 170 | 0.00 | 0.00 | 0.00 | 101 |
| 171 | 0.58 | 0.12 | 0.20 | 89 |
| 172 | 0.00 | 0.00 | 0.00 | 193 |
| 173 | 0.00 | 0.00 | 0.00 | 309 |
| 174 | 0.00 | 0.00 | 0.00 | 172 |
| 175 | 0.91 | 0.84 | 0.87 | 95 |
| 176 | 0.91 | 0.58 | 0.71 | 346 |
| 177 | 0.90 | 0.48 | 0.62 | 322 |
| 178 | 0.65 | 0.48 | 0.56 | 232 |
| 179 | 0.00 | 0.00 | 0.00 | 125 |
| 180 | 0.42 | 0.41 | 0.42 | 145 |
| 181 | 0.00 | 0.00 | 0.00 | 77 |
| 182 | 0.00 | 0.00 | 0.00 | 182 |
| 183 | 0.00 | 0.00 | 0.00 | 257 |
| 184 | 0.00 | 0.00 | 0.00 | 216 |
| 185 | 0.00 | 0.00 | 0.00 | 242 |
| 186 | 0.00 | 0.00 | 0.00 | 165 |
| 187 | 0.75 | 0.62 | 0.68 | 263 |
| 188 | 0.00 | 0.00 | 0.00 | 174 |
| 189 | 0.92 | 0.08 | 0.15 | 136 |
| 190 | 0.94 | 0.58 | 0.72 | 202 |
| 191 | 0.00 | 0.00 | 0.00 | 134 |
| 192 | 0.72 | 0.47 | 0.57 | 230 |
| 193 | 0.00 | 0.00 | 0.00 | 90 |
| 194 | 0.58 | 0.46 | 0.51 | 185 |
| 195 | 0.00 | 0.00 | 0.00 | 156 |
| 196 | 0.00 | 0.00 | 0.00 | 160 |
| 197 | 0.66 | 0.10 | 0.18 | 266 |
| 198 | 0.00 | 0.00 | 0.00 | 284 |
| 199 | 0.00 | 0.00 | 0.00 | 145 |
| 200 | 0.92 | 0.74 | 0.82 | 212 |
| 201 | 0.69 | 0.16 | 0.26 | 317 |
| 202 | 0.76 | 0.58 | 0.66 | 427 |
| 203 | 0.00 | 0.00 | 0.00 | 232 |
| 204 | 0.00 | 0.00 | 0.00 | 217 |
| 205 | 0.00 | 0.00 | 0.00 | 527 |
| 206 | 0.00 | 0.00 | 0.00 | 124 |
| 207 | 1.00 | 0.02 | 0.04 | 103 |
| 208 | 0.89 | 0.52 | 0.65 | 287 |
| 209 | 0.00 | 0.00 | 0.00 | 193 |
| 210 | 0.83 | 0.14 | 0.23 | 220 |
| 211 | 0.75 | 0.21 | 0.33 | 140 |
| 212 | 0.00 | 0.00 | 0.00 | 161 |
| 213 | 0.60 | 0.25 | 0.35 | 72 |
| 214 | 0.00 | 0.00 | 0.00 | 396 |
| 215 | 0.84 | 0.40 | 0.54 | 134 |
| 216 | 0.71 | 0.01 | 0.02 | 400 |
| 217 | 1.00 | 0.01 | 0.03 | 75 |
| 218 | 0.96 | 0.79 | 0.86 | 219 |
| 219 | 0.86 | 0.24 | 0.37 | 210 |
| 220 | 0.88 | 0.66 | 0.75 | 298 |

| | | | | |
|---|---|---|---|---|
| 221 | 0.93 | 0.69 | 0.79 | 266 |
| 222 | 0.78 | 0.40 | 0.53 | 290 |
| 223 | 0.00 | 0.00 | 0.00 | 128 |
| 224 | 0.82 | 0.43 | 0.56 | 159 |
| 225 | 0.68 | 0.27 | 0.39 | 164 |
| 226 | 0.62 | 0.40 | 0.49 | 144 |
| 227 | 0.00 | 0.00 | 0.00 | 276 |
| 228 | 0.00 | 0.00 | 0.00 | 235 |
| 229 | 0.00 | 0.00 | 0.00 | 216 |
| 230 | 0.00 | 0.00 | 0.00 | 228 |
| 231 | 0.74 | 0.62 | 0.68 | 64 |
| 232 | 0.00 | 0.00 | 0.00 | 103 |
| 233 | 0.73 | 0.35 | 0.47 | 216 |
| 234 | 0.00 | 0.00 | 0.00 | 116 |
| 235 | 0.57 | 0.45 | 0.51 | 77 |
| 236 | 0.87 | 0.72 | 0.79 | 67 |
| 237 | 0.70 | 0.07 | 0.13 | 218 |
| 238 | 0.00 | 0.00 | 0.00 | 139 |
| 239 | 0.00 | 0.00 | 0.00 | 94 |
| 240 | 0.55 | 0.21 | 0.30 | 77 |
| 241 | 0.00 | 0.00 | 0.00 | 167 |
| 242 | 0.80 | 0.33 | 0.46 | 86 |
| 243 | 0.00 | 0.00 | 0.00 | 58 |
| 244 | 0.74 | 0.13 | 0.22 | 269 |
| 245 | 0.00 | 0.00 | 0.00 | 112 |
| 246 | 0.94 | 0.79 | 0.86 | 255 |
| 247 | 0.53 | 0.14 | 0.22 | 58 |
| 248 | 0.00 | 0.00 | 0.00 | 81 |
| 249 | 0.00 | 0.00 | 0.00 | 131 |
| 250 | 0.00 | 0.00 | 0.00 | 93 |
| 251 | 0.00 | 0.00 | 0.00 | 154 |
| 252 | 0.00 | 0.00 | 0.00 | 129 |
| 253 | 0.66 | 0.25 | 0.37 | 83 |
| 254 | 0.00 | 0.00 | 0.00 | 191 |
| 255 | 0.00 | 0.00 | 0.00 | 219 |
| 256 | 0.00 | 0.00 | 0.00 | 130 |
| 257 | 0.00 | 0.00 | 0.00 | 93 |
| 258 | 0.67 | 0.53 | 0.59 | 217 |
| 259 | 0.00 | 0.00 | 0.00 | 141 |
| 260 | 0.81 | 0.15 | 0.26 | 143 |
| 261 | 0.00 | 0.00 | 0.00 | 219 |
| 262 | 0.00 | 0.00 | 0.00 | 107 |
| 263 | 0.00 | 0.00 | 0.00 | 236 |
| 264 | 0.25 | 0.01 | 0.02 | 119 |
| 265 | 0.33 | 0.01 | 0.03 | 72 |
| 266 | 0.00 | 0.00 | 0.00 | 70 |
| 267 | 0.00 | 0.00 | 0.00 | 107 |
| 268 | 0.67 | 0.45 | 0.54 | 169 |
| 269 | 0.00 | 0.00 | 0.00 | 129 |
| 270 | 0.71 | 0.62 | 0.66 | 159 |
| 271 | 0.90 | 0.44 | 0.59 | 190 |
| 272 | 0.00 | 0.00 | 0.00 | 248 |
| 273 | 0.89 | 0.78 | 0.83 | 264 |
| 274 | 0.89 | 0.70 | 0.78 | 105 |
| 275 | 0.00 | 0.00 | 0.00 | 104 |
| 276 | 0.00 | 0.00 | 0.00 | 115 |
| 277 | 0.83 | 0.68 | 0.75 | 170 |
| 278 | 0.67 | 0.23 | 0.35 | 145 |
| 279 | 0.91 | 0.68 | 0.78 | 230 |
| 280 | 0.52 | 0.15 | 0.23 | 80 |
| 281 | 0.67 | 0.72 | 0.69 | 217 |
| 282 | 0.74 | 0.62 | 0.68 | 175 |
| 283 | 0.00 | 0.00 | 0.00 | 269 |
| 284 | 0.52 | 0.19 | 0.28 | 74 |
| 285 | 0.83 | 0.51 | 0.64 | 206 |
| 286 | 0.89 | 0.65 | 0.75 | 227 |
| 287 | 0.90 | 0.34 | 0.49 | 130 |
| 288 | 0.00 | 0.00 | 0.00 | 129 |
| 289 | 0.00 | 0.00 | 0.00 | 80 |
| 290 | 0.00 | 0.00 | 0.00 | 99 |
| 291 | 0.78 | 0.30 | 0.43 | 208 |
| 292 | 0.00 | 0.00 | 0.00 | 67 |
| 293 | 0.95 | 0.36 | 0.52 | 109 |
| 294 | 0.00 | 0.00 | 0.00 | 140 |
| 295 | 0.00 | 0.00 | 0.00 | 241 |
| 296 | 0.00 | 0.00 | 0.00 | 72 |
| 297 | 0.00 | 0.00 | 0.00 | 107 |
| 298 | 0.81 | 0.28 | 0.41 | 61 |
| 299 | 0.86 | 0.31 | 0.46 | 77 |
| 300 | 0.00 | 0.00 | 0.00 | 111 |
| 301 | 0.00 | 0.00 | 0.00 | 126 |
| 302 | 0.00 | 0.00 | 0.00 | 73 |
| 303 | 0.64 | 0.39 | 0.49 | 176 |

| | | | | |
|---|---|---|---|---|
| 304 | 0.93 | 0.74 | 0.83 | 230 |
| 305 | 0.91 | 0.74 | 0.82 | 156 |
| 306 | 0.00 | 0.00 | 0.00 | 146 |
| 307 | 0.00 | 0.00 | 0.00 | 98 |
| 308 | 0.00 | 0.00 | 0.00 | 78 |
| 309 | 0.57 | 0.13 | 0.21 | 94 |
| 310 | 1.00 | 0.02 | 0.04 | 162 |
| 311 | 0.76 | 0.59 | 0.66 | 116 |
| 312 | 0.50 | 0.33 | 0.40 | 57 |
| 313 | 0.00 | 0.00 | 0.00 | 65 |
| 314 | 0.43 | 0.22 | 0.29 | 138 |
| 315 | 0.00 | 0.00 | 0.00 | 195 |
| 316 | 1.00 | 0.01 | 0.03 | 69 |
| 317 | 0.62 | 0.06 | 0.11 | 134 |
| 318 | 0.00 | 0.00 | 0.00 | 148 |
| 319 | 0.83 | 0.55 | 0.66 | 161 |
| 320 | 0.00 | 0.00 | 0.00 | 104 |
| 321 | 0.84 | 0.63 | 0.72 | 156 |
| 322 | 0.67 | 0.04 | 0.08 | 134 |
| 323 | 0.00 | 0.00 | 0.00 | 232 |
| 324 | 0.00 | 0.00 | 0.00 | 92 |
| 325 | 0.00 | 0.00 | 0.00 | 197 |
| 326 | 0.00 | 0.00 | 0.00 | 126 |
| 327 | 0.00 | 0.00 | 0.00 | 115 |
| 328 | 0.97 | 0.71 | 0.82 | 198 |
| 329 | 0.55 | 0.25 | 0.34 | 125 |
| 330 | 0.67 | 0.02 | 0.05 | 81 |
| 331 | 0.00 | 0.00 | 0.00 | 94 |
| 332 | 0.00 | 0.00 | 0.00 | 56 |
| 333 | 0.00 | 0.00 | 0.00 | 260 |
| 334 | 0.00 | 0.00 | 0.00 | 60 |
| 335 | 0.00 | 0.00 | 0.00 | 110 |
| 336 | 0.61 | 0.54 | 0.57 | 71 |
| 337 | 0.00 | 0.00 | 0.00 | 66 |
| 338 | 0.00 | 0.00 | 0.00 | 150 |
| 339 | 0.00 | 0.00 | 0.00 | 54 |
| 340 | 0.86 | 0.63 | 0.73 | 195 |
| 341 | 0.67 | 0.10 | 0.18 | 79 |
| 342 | 0.00 | 0.00 | 0.00 | 38 |
| 343 | 0.68 | 0.49 | 0.57 | 43 |
| 344 | 0.00 | 0.00 | 0.00 | 68 |
| 345 | 1.00 | 0.01 | 0.03 | 73 |
| 346 | 0.00 | 0.00 | 0.00 | 116 |
| 347 | 0.83 | 0.40 | 0.54 | 111 |
| 348 | 0.00 | 0.00 | 0.00 | 63 |
| 349 | 0.84 | 0.71 | 0.77 | 104 |
| 350 | 0.62 | 0.57 | 0.60 | 44 |
| 351 | 0.83 | 0.12 | 0.22 | 40 |
| 352 | 0.90 | 0.49 | 0.63 | 136 |
| 353 | 0.00 | 0.00 | 0.00 | 54 |
| 354 | 0.00 | 0.00 | 0.00 | 134 |
| 355 | 0.79 | 0.28 | 0.41 | 120 |
| 356 | 0.00 | 0.00 | 0.00 | 228 |
| 357 | 0.00 | 0.00 | 0.00 | 269 |
| 358 | 0.00 | 0.00 | 0.00 | 80 |
| 359 | 0.84 | 0.59 | 0.69 | 140 |
| 360 | 0.00 | 0.00 | 0.00 | 125 |
| 361 | 0.90 | 0.73 | 0.80 | 169 |
| 362 | 0.00 | 0.00 | 0.00 | 56 |
| 363 | 0.93 | 0.71 | 0.80 | 154 |
| 364 | 0.00 | 0.00 | 0.00 | 58 |
| 365 | 0.00 | 0.00 | 0.00 | 71 |
| 366 | 0.97 | 0.70 | 0.82 | 54 |
| 367 | 0.00 | 0.00 | 0.00 | 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 | 0.00 | 0.00 | 0.00 | 71 |
| 372 | 0.61 | 0.54 | 0.57 | 52 |
| 373 | 0.81 | 0.43 | 0.57 | 150 |
| 374 | 0.00 | 0.00 | 0.00 | 93 |
| 375 | 0.00 | 0.00 | 0.00 | 67 |
| 376 | 0.00 | 0.00 | 0.00 | 76 |
| 377 | 0.00 | 0.00 | 0.00 | 106 |
| 378 | 0.00 | 0.00 | 0.00 | 86 |
| 379 | 0.75 | 0.21 | 0.33 | 14 |
| 380 | 0.86 | 0.57 | 0.68 | 122 |
| 381 | 0.00 | 0.00 | 0.00 | 104 |
| 382 | 0.00 | 0.00 | 0.00 | 66 |
| 383 | 1.00 | 0.03 | 0.05 | 110 |
| 384 | 0.00 | 0.00 | 0.00 | 155 |
| 385 | 0.30 | 0.06 | 0.10 | 50 |

| | | | | |
|---|---|---|---|---|
| 386 | 0.00 | 0.00 | 0.00 | 64 |
| 387 | 0.00 | 0.00 | 0.00 | 93 |
| 388 | 0.00 | 0.00 | 0.00 | 102 |
| 389 | 0.00 | 0.00 | 0.00 | 108 |
| 390 | 0.94 | 0.70 | 0.80 | 178 |
| 391 | 0.00 | 0.00 | 0.00 | 115 |
| 392 | 0.88 | 0.50 | 0.64 | 42 |
| 393 | 0.00 | 0.00 | 0.00 | 134 |
| 394 | 0.00 | 0.00 | 0.00 | 112 |
| 395 | 0.00 | 0.00 | 0.00 | 176 |
| 396 | 0.00 | 0.00 | 0.00 | 125 |
| 397 | 0.73 | 0.23 | 0.35 | 224 |
| 398 | 0.76 | 0.71 | 0.74 | 63 |
| 399 | 0.00 | 0.00 | 0.00 | 59 |
| 400 | 0.50 | 0.02 | 0.03 | 63 |
| 401 | 0.00 | 0.00 | 0.00 | 98 |
| 402 | 0.00 | 0.00 | 0.00 | 162 |
| 403 | 0.00 | 0.00 | 0.00 | 83 |
| 404 | 0.64 | 0.84 | 0.73 | 19 |
| 405 | 0.00 | 0.00 | 0.00 | 92 |
| 406 | 0.86 | 0.15 | 0.25 | 41 |
| 407 | 0.90 | 0.21 | 0.34 | 43 |
| 408 | 0.76 | 0.32 | 0.45 | 160 |
| 409 | 0.00 | 0.00 | 0.00 | 50 |
| 410 | 0.00 | 0.00 | 0.00 | 19 |
| 411 | 0.00 | 0.00 | 0.00 | 175 |
| 412 | 0.00 | 0.00 | 0.00 | 72 |
| 413 | 0.00 | 0.00 | 0.00 | 95 |
| 414 | 0.00 | 0.00 | 0.00 | 97 |
| 415 | 0.00 | 0.00 | 0.00 | 48 |
| 416 | 0.33 | 0.02 | 0.04 | 83 |
| 417 | 0.00 | 0.00 | 0.00 | 40 |
| 418 | 0.00 | 0.00 | 0.00 | 91 |
| 419 | 0.00 | 0.00 | 0.00 | 90 |
| 420 | 0.00 | 0.00 | 0.00 | 37 |
| 421 | 0.00 | 0.00 | 0.00 | 66 |
| 422 | 0.00 | 0.00 | 0.00 | 73 |
| 423 | 0.42 | 0.23 | 0.30 | 56 |
| 424 | 0.91 | 0.88 | 0.89 | 33 |
| 425 | 0.00 | 0.00 | 0.00 | 76 |
| 426 | 0.00 | 0.00 | 0.00 | 81 |
| 427 | 0.97 | 0.75 | 0.85 | 150 |
| 428 | 0.91 | 0.72 | 0.81 | 29 |
| 429 | 0.99 | 0.92 | 0.96 | 389 |
| 430 | 0.69 | 0.31 | 0.43 | 167 |
| 431 | 0.00 | 0.00 | 0.00 | 123 |
| 432 | 0.00 | 0.00 | 0.00 | 39 |
| 433 | 0.00 | 0.00 | 0.00 | 82 |
| 434 | 0.96 | 0.71 | 0.82 | 66 |
| 435 | 0.60 | 0.19 | 0.29 | 93 |
| 436 | 0.69 | 0.23 | 0.34 | 87 |
| 437 | 0.00 | 0.00 | 0.00 | 86 |
| 438 | 0.78 | 0.45 | 0.57 | 104 |
| 439 | 0.00 | 0.00 | 0.00 | 100 |
| 440 | 0.00 | 0.00 | 0.00 | 141 |
| 441 | 0.00 | 0.00 | 0.00 | 110 |
| 442 | 0.00 | 0.00 | 0.00 | 123 |
| 443 | 0.25 | 0.03 | 0.05 | 71 |
| 444 | 0.00 | 0.00 | 0.00 | 109 |
| 445 | 0.00 | 0.00 | 0.00 | 48 |
| 446 | 0.00 | 0.00 | 0.00 | 76 |
| 447 | 0.00 | 0.00 | 0.00 | 38 |
| 448 | 0.65 | 0.65 | 0.65 | 81 |
| 449 | 1.00 | 0.05 | 0.09 | 132 |
| 450 | 0.00 | 0.00 | 0.00 | 81 |
| 451 | 0.87 | 0.34 | 0.49 | 76 |
| 452 | 0.00 | 0.00 | 0.00 | 44 |
| 453 | 0.00 | 0.00 | 0.00 | 44 |
| 454 | 0.82 | 0.47 | 0.60 | 70 |
| 455 | 0.00 | 0.00 | 0.00 | 155 |
| 456 | 1.00 | 0.02 | 0.05 | 43 |
| 457 | 0.00 | 0.00 | 0.00 | 72 |
| 458 | 0.00 | 0.00 | 0.00 | 62 |
| 459 | 0.88 | 0.10 | 0.18 | 69 |
| 460 | 0.00 | 0.00 | 0.00 | 119 |
| 461 | 0.00 | 0.00 | 0.00 | 79 |
| 462 | 0.00 | 0.00 | 0.00 | 47 |
| 463 | 0.00 | 0.00 | 0.00 | 104 |
| 464 | 0.00 | 0.00 | 0.00 | 106 |
| 465 | 0.00 | 0.00 | 0.00 | 64 |
| 466 | 0.00 | 0.00 | 0.00 | 173 |
| 467 | 0.71 | 0.33 | 0.45 | 107 |
| 468 | 0.89 | 0.06 | 0.12 | 126 |

```
        469      0.00      0.00      0.00       114
        470      0.93      0.81      0.87       140
        471      1.00      0.37      0.54        79
        472      0.00      0.00      0.00       143
        473      0.71      0.34      0.46       158
        474      0.00      0.00      0.00       138
        475      0.00      0.00      0.00        59
        476      0.63      0.14      0.22        88
        477      0.83      0.67      0.74       176
        478      0.90      0.79      0.84        24
        479      0.00      0.00      0.00        92
        480      0.79      0.55      0.65       100
        481      0.54      0.07      0.12       103
        482      0.00      0.00      0.00        74
        483      0.78      0.60      0.68       105
        484      0.00      0.00      0.00        83
        485      0.00      0.00      0.00        82
        486      0.00      0.00      0.00        71
        487      0.00      0.00      0.00       120
        488      0.67      0.08      0.14       105
        489      0.71      0.28      0.40        87
        490      1.00      0.81      0.90        32
        491      0.00      0.00      0.00        69
        492      0.00      0.00      0.00        49
        493      0.00      0.00      0.00       117
        494      0.52      0.26      0.35        61
        495      0.97      0.75      0.85       344
        496      0.00      0.00      0.00        52
        497      0.00      0.00      0.00       137
        498      0.00      0.00      0.00        98
        499      0.65      0.19      0.29        79

   micro avg       0.81      0.28      0.42    173812
   macro avg       0.40      0.20      0.25    173812
weighted avg       0.60      0.28      0.36    173812
 samples avg       0.40      0.27      0.31    173812

Time taken to run this cell : 0:11:50.750816
```

# Conclusions

In [29]:

```python
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["SNo", 'Classifier', 'Penalty', 'hyperparameter', "F1_micro average"]
x.add_row(["1", 'SGDC(loss=log)', 'L1', 'C = 0.00001', "0.43" ])
x.add_row(["2", 'SGDC(loss=hinge)', 'L1', 'C = 0.00001', "0.42" ])

print(x)
```

```
+------+------------------+---------+----------------+------------------+
| SNo  |    Classifier    | Penalty | hyperparameter | F1_micro average |
+------+------------------+---------+----------------+------------------+
|  1   |  SGDC(loss=log)  |    L1   |  C = 0.00001   |       0.43       |
|  2   | SGDC(loss=hinge) |    L1   |  C = 0.00001   |       0.42       |
+------+------------------+---------+----------------+------------------+
```

In [ ]: