

Stack OverFlow Tag Prediction

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>

Youtube : <https://youtu.be/nNDqbUhtIRg>

Research paper : <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>

1.3 Real World / Business Objectives and Constraints

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 Explanation

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-separated 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\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>

```
In [1]: #importing dependencies
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import sqlite3
import csv
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from wordcloud import WordCloud
import re
import os
from sqlalchemy import create_engine # database connection
import datetime as dt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn import metrics
from sklearn.metrics import f1_score, precision_score, recall_score
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn_multilearn.adapt import mlknn
from sklearn_multilearn.problem_transform import ClassifierChain
from sklearn_multilearn.problem_transform import BinaryRelevance
```



```
from skmultilearn.problem_transform import LabelPowerset
from sklearn.naive_bayes import GaussianNB
from datetime import datetime
import nltk
```

3. Exploratory Data Analysis

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to Load the data

```
In [72]: #Creating db file from csv
#Problem was arising when i took Chunksize as 180k and it was still the
re
#until i took Chunksize to be 100 only, now it worked correctly
#Learn SQL: https://www.w3schools.com/sql/default.asp
if not os.path.isfile('train.db'):
    start = datetime.now()
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 100
    j = 0
    index_start = 1
    for df in pd.read_csv('Train.csv', names=['Id', 'Title', 'Body', 'T
ags'],\
                           chunksize=chunksize, iterator=True, encoding=
'utf-8', ):
        df.index += index_start
        j+=1
        if j*chunksize%500000==0:
            print('{} rows'.format(j*chunksize))
        df.to_sql('data', disk_engine, if_exists='append')
        index_start = df.index[-1] + 1
    print("Time taken to run this cell :", datetime.now() - start)
```

3.1.2 Counting the number of rows

```
In [6]: if os.path.isfile('train.db'):
        start = datetime.now()
        con = sqlite3.connect('train.db')
        num_rows = pd.read_sql_query("""SELECT count(*) FROM data""", con)
        #Always remember to close the database
        print("Number of rows in the database :", "\n", num_rows['count(*)'].
values[0])
        con.close()
        print("Time taken to count the number of rows :", datetime.now() -
start)
    else:
        print("Please download the train.db file from drive or run the above cell to generate train.db file")
```

Number of rows in the database :

6034196

Time taken to count the number of rows : 0:04:00.914760

3.1.3 Checking for duplicates

```
In [3]: #Learn SQL: https://www.w3schools.com/sql/default.asp
        if os.path.isfile('train.db'):
            start = datetime.now()
            con = sqlite3.connect('train.db')
            df_no_dup = pd.read_sql_query('SELECT Title, Body, Tags, COUNT(*) \
as cnt_dup FROM data GROUP BY Title, Body, Tags', con)
            con.close()
            print("Time taken to run this cell :", datetime.now() - start)
        else:
            print("Please download the train.db file from drive or run the first cell to generate train.db file")
```

Time taken to run this cell : 0:46:46.061612

```
In [4]: df_no_dup.head()
# we can observe that there are duplicates
```

Out[4]:

	Title	Body	Tags	c
0	Implementing Boundary Value Analysis of S...	<pre> <code>#include<iosstream>\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]: print("number of duplicate questions :", num_rows['count(*)'].values[0]
- df_no_dup.shape[0],\
      "(", (1 - ((df_no_dup.shape[0]) / (num_rows['count(*)'].values[0]))) * 100, "% )")
```

number of duplicate questions : 1827881 (30.292038906260256 %)

```
In [8]: # number of times each question appeared in our database
df_no_dup.cnt_dup.value_counts()
```

Out[8]:

1	2656284
2	1272336

```
3      277575
4          90
5          25
6          5
Name: cnt_dup, dtype: int64
```

```
In [9]: start = datetime.now()
df_no_dup["tag_count"] = df_no_dup["Tags"].apply(lambda text: len(text.
split(" ")) if text is not None else 0)
# adding a new feature number of tags per question
print("Time taken to run this cell :", datetime.now() - start)
df_no_dup.head()
```

Time taken to run this cell : 0:00:03.467926

Out[9]:

	Title	Body	Tags	c
0	Implementing Boundary Value Analysis of S...	<pre> <code>#include<it;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 [10]: # distribution of number of tags per question
df_no_dup.tag_count.value_counts()
```

```
Out[10]: 3    1206157
         2    1111706
         4     814996
         1     568291
         5     505158
         0         7
         Name: tag_count, dtype: int64
```

```
In [11]: #checking datapoints with no tags
df_no_dup[df_no_dup['tag_count']==0]
```

```
Out[11]:
```

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

```
In [12]: #removing datapoints with no tags
df_no_dup= df_no_dup[df_no_dup['tag_count']!=0]
```

```
In [15]: # distribution of number 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 [19]: #Creating a new database with no duplicates
if not os.path.isfile('train_no_dup.db'):
    disk_dup = create_engine("sqlite:///train_no_dup.db")
    no_dup = pd.DataFrame(df_no_dup, columns=['Title', 'Body', 'Tags'])
    no_dup.to_sql('no_dup_train',disk_dup,chunksize=100)
```

```
In [3]: #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')
    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 generate train.db file")
```

Time taken to run this cell : 0:01:49.512123

In [53]: tag_data.head()

Out[53]:

	Tags
1	c# silverlight data-binding
2	c# silverlight data-binding columns
3	jsp jstl
4	java jdbc
5	facebook api facebook-php-sdk

3.2 Analysis of Tags

3.2.1 Total number of unique tags

```
In [21]: # Importing & Initializing the "CountVectorizer" object, which
#is scikit-learn's bag of words tool.

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

```
In [22]: print("Number of data points :", tag_dtm.shape[0])
print("Number of unique tags :", tag_dtm.shape[1])
```

Number of data points : 4206307

Number of unique tags : 42048

```
In [23]: # 'get_feature_name()' gives us the vocabulary.
tags = vectorizer.get_feature_names()
# Lets look at the tags we have.
print("Some of the tags we have :", tags[:10])
```

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

3.2.3 Number of times a tag appeared

```
In [24]: # 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 [25]: # 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()
```

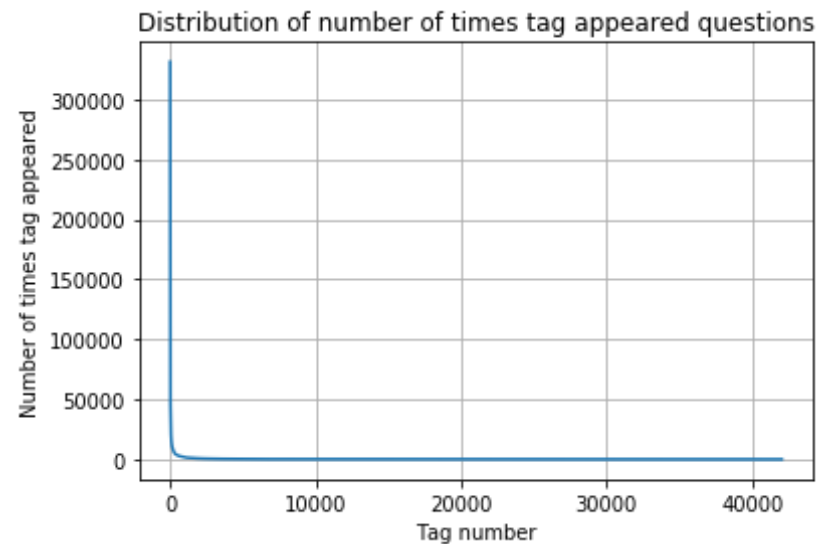
Out[25]:

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

	Tags	Counts
4	.bash-profile	138

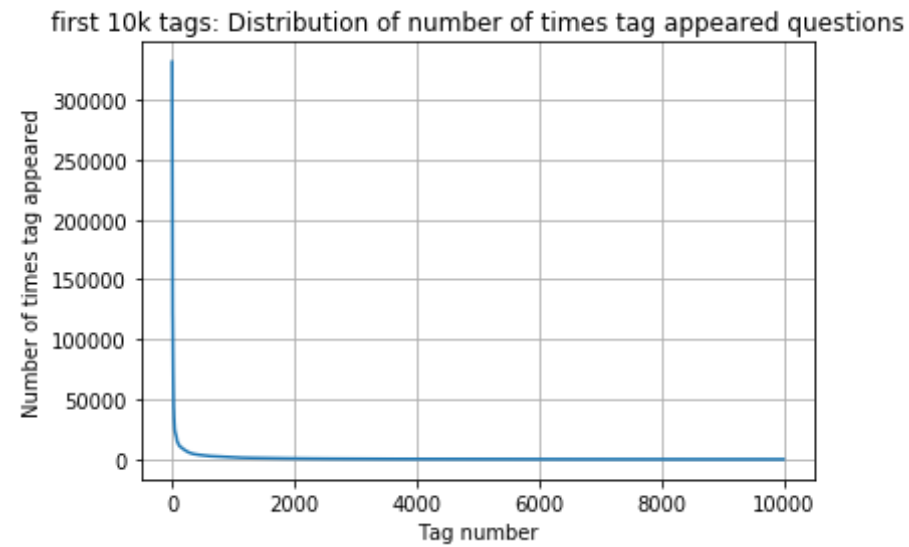
```
In [30]: tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
tag_counts = tag_df_sorted['Counts'].values
```

```
In [31]: 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 [32]: 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")
```

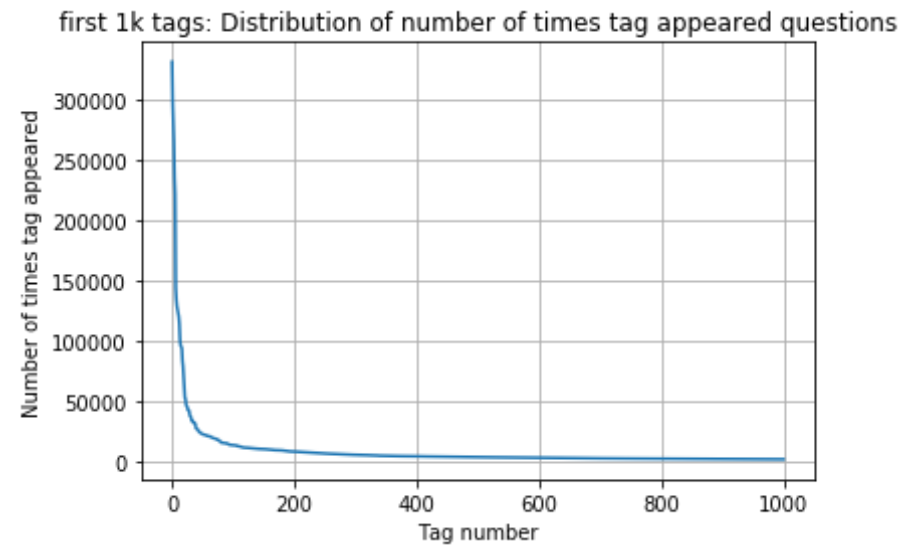
```
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```



```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054    7
151
6466  5865  5370  4983  4526  4281  4144  3929  3750  3593
3453  3299  3123  2986  2891  2738  2647  2527  2431  2331
2259  2186  2097  2020  1959  1900  1828  1770  1723  1673
1631  1574  1532  1479  1448  1406  1365  1328  1300  1266
1245  1222  1197  1181  1158  1139  1121  1101  1076  1056
1038  1023  1006   983   966   952   938   926   911   891
 882   869   856   841   830   816   804   789   779   770
 752   743   733   725   712   702   688   678   671   658
 650   643   634   627   616   607   598   589   583   577
 568   559   552   545   540   533   526   518   512   506
 500   495   490   485   480   477   469   465   457   450
 447   442   437   432   426   422   418   413   408   403
 398   393   388   385   381   378   374   370   367   365
 361   357   354   350   347   344   342   339   336   332
 330   326   323   319   315   312   309   307   304   301
 299   296   293   291   289   286   284   281   278   276
 275   272   270   268   265   262   260   258   256   254
 252   250   249   247   245   243   241   239   238   236
 234   232   230   228   226   224   222   220   218   216
```

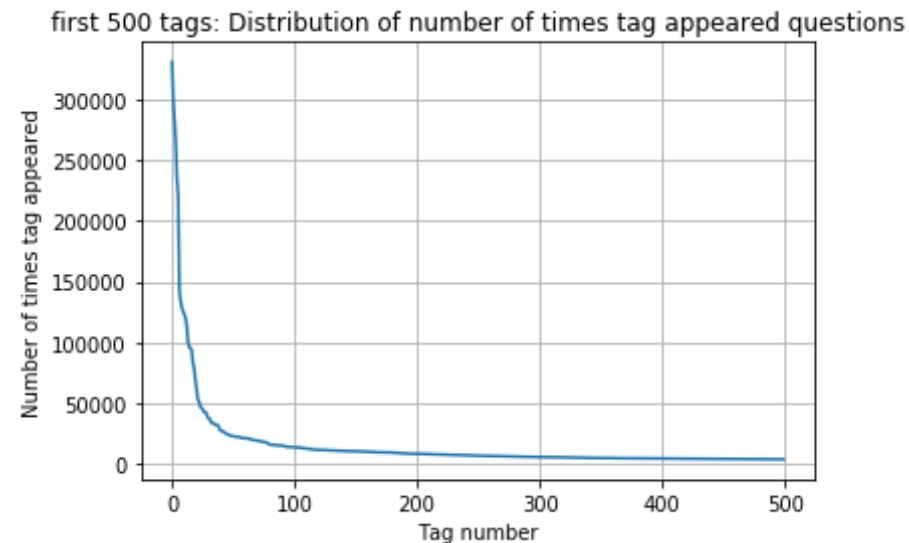
234	233	232	230	228	226	224	222	220	219
217	215	214	212	210	209	207	205	204	203
201	200	199	198	196	194	193	192	191	189
188	186	185	183	182	181	180	179	178	177
175	174	172	171	170	169	168	167	166	165
164	162	161	160	159	158	157	156	156	155
154	153	152	151	150	149	149	148	147	146
145	144	143	142	142	141	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 [33]: 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	24
537										
22429	21820	20957	19758	18905	17728	15533	15097	14884	13703	
13364	13157	12407	11658	11228	11162	10863	10600	10350	10224	
10029	9884	9719	9411	9252	9148	9040	8617	8361	8163	
8054	7867	7702	7564	7274	7151	7052	6847	6656	6553	
6466	6291	6183	6093	5971	5865	5760	5577	5490	5411	
5370	5283	5207	5107	5066	4983	4891	4785	4658	4549	
4526	4487	4429	4335	4310	4281	4239	4228	4195	4159	
4144	4088	4050	4002	3957	3929	3874	3849	3818	3797	
3750	3703	3685	3658	3615	3593	3564	3521	3505	3483	
3453	3427	3396	3363	3326	3299	3272	3232	3196	3168	
3123	3094	3073	3050	3012	2986	2983	2953	2934	2903	
2891	2844	2819	2784	2754	2738	2726	2708	2681	2669	
2647	2621	2604	2594	2556	2527	2510	2482	2460	2444	
2431	2409	2395	2380	2363	2331	2312	2297	2290	2281	
2259	2246	2222	2211	2198	2186	2162	2142	2132	2107	
2097	2078	2057	2045	2036	2020	2011	1994	1971	1965	
1959	1952	1940	1932	1912	1900	1879	1865	1855	1841	
1828	1821	1813	1801	1782	1770	1760	1747	1741	1734	
1723	1707	1697	1688	1683	1673	1665	1656	1646	1639]	

```
In [34]: 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])
```



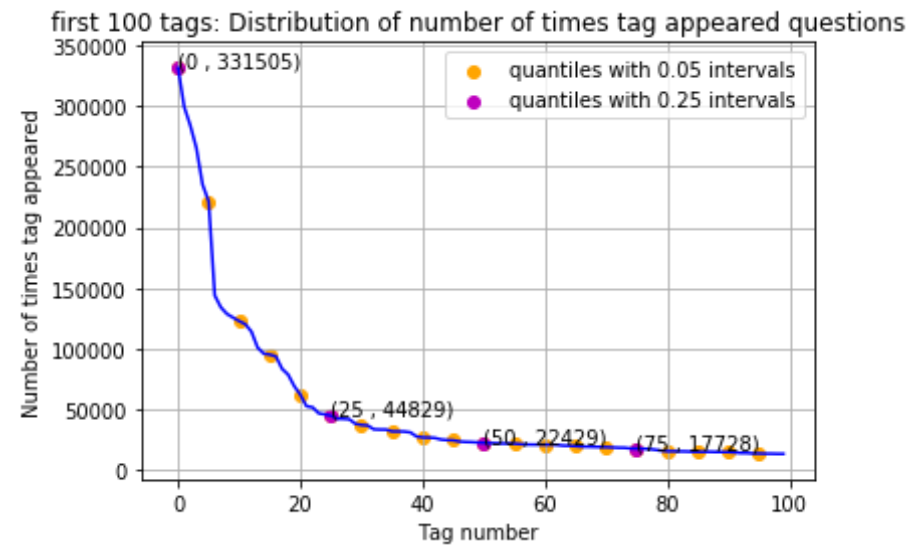
```
100 [331505 221533 122769 95160 62023 44829 37170 31897 26925 24
537
22429 21820 20957 19758 18905 17728 15533 15097 14884 13703
13364 13157 12407 11658 11228 11162 10863 10600 10350 10224
10029 9884 9719 9411 9252 9148 9040 8617 8361 8163
8054 7867 7702 7564 7274 7151 7052 6847 6656 6553
6466 6291 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 [35]: 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', lab
el = "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 245
37
22429 21820 20957 19758 18905 17728 15533 15097 14884 13703]
```

```
In [36]: # Store tags greater than 10K in one list
lst_tags_gt_10k = tag_df[tag_df.Counts>10000].Tags
#Print the length of the list
print ('{} Tags are used more than 10000 times'.format(len(lst_tags_gt_10k)))
# Store tags greater than 100K in one list
lst_tags_gt_100k = tag_df[tag_df.Counts>100000].Tags
#Print the length of the list.
print ('{} Tags are used more than 100000 times'.format(len(lst_tags_gt_100k)))

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

Observations:

1. There are total 153 tags which are used more than 10000 times.
2. 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 frequently than others, Micro-averaged F1-score is the appropriate metric for this problem.

3.2.4 Tags Per Question

```
In [37]: #Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting each value in the 'tag_quest_count' to integer.
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])

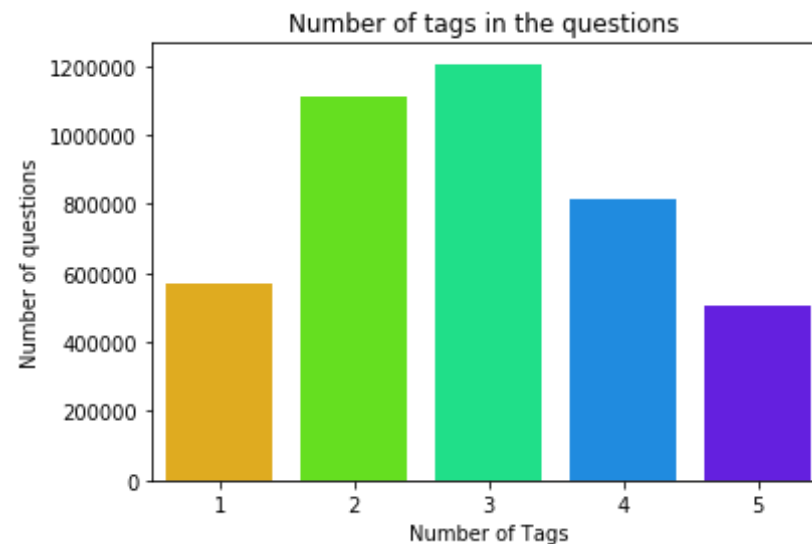
We have total 4206307 datapoints.
[3, 4, 2, 2, 3]
```

```
In [38]: print( "Maximum number of tags per question: %d"%max(tag_quest_count))
```

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

Maximum number of tags per question: 5
Minimum number of tags per question: 1
Avg. number of tags per question: 2.899443

```
In [39]: sns.countplot(tag_quest_count, palette='gist_rainbow')
plt.title("Number of tags in the questions ")
plt.xlabel("Number of Tags")
plt.ylabel("Number of questions")
plt.show()
```



Observations:

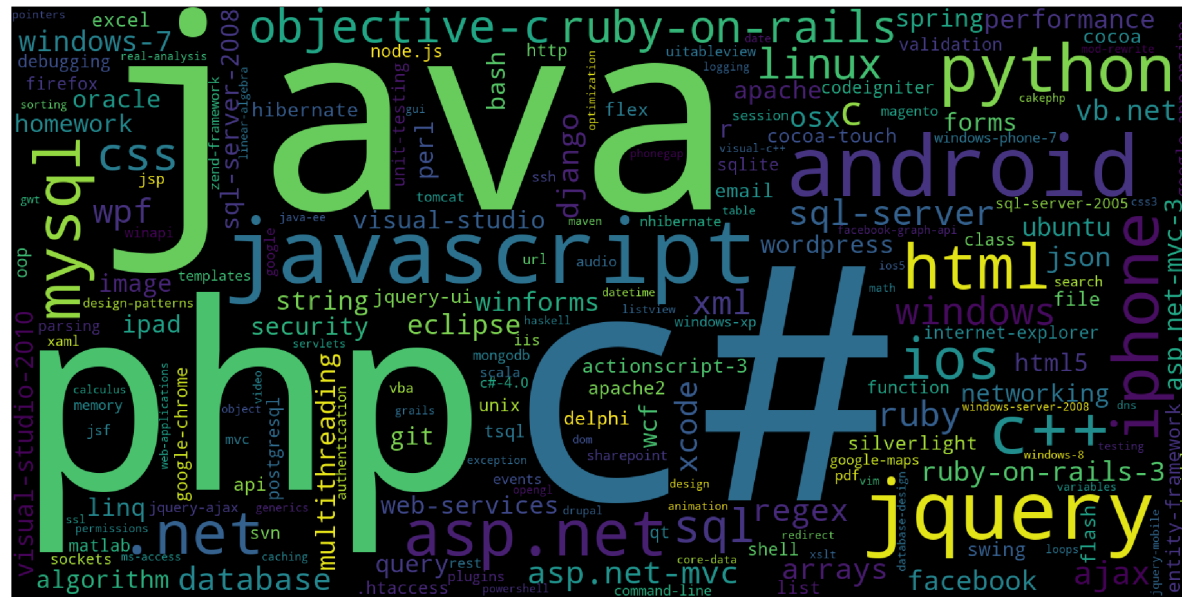
1. Maximum number of tags per question: 5
2. Minimum number of tags per question: 1
3. Avg. number of tags per question: 2.899
4. Most of the questions are having 2 or 3 tags

3.2.5 Most Frequent Tags

```
In [40]: # Plotting 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,
                           ).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)
```



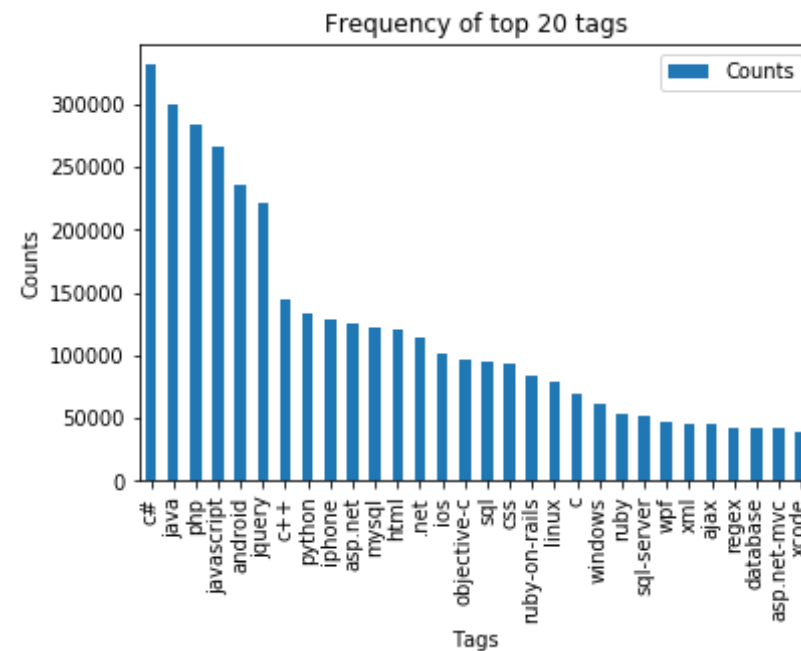
Time taken to run this cell : 0:00:05.454010

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 [41]: i=np.arange(30)
tag_df_sorted.head(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.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.

```
In [5]: def striphtml(data):
        cleanr = re.compile('<.*?>')
        cleantext = re.sub(cleanr, ' ', str(data))
        return cleantext
stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer("english")
```

```
In [6]: #http://www.sqlitetutorial.net/sqlite-python/create-tables/
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

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

```

        print(e)

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

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

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

```

Tables in the database:
QuestionsProcessed

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 (qu
estion 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 [ ]: # 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")
```

4.5.1 Preprocessing of questions

1. Separate Code from Body
2. Remove Special 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 [7]: #http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sql
ite-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.DOT
ALL))

    question=re.sub('<code>(.*?)</code>', '', question, flags=re.MULTIL
INE|re.DOTALL)
    question=stripthtml(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)+" "+questio
n+" "+str(tags)
#         else:
#         question=str(title)+" "+str(title)+" "+str(title)+" "+questio
n

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 if j not in s
top_words and (len(j)!=1 or j=='c'))

len_post+=len(question)
tup = (question,code,tags,x,len(question),is_code)
questions_proccesed += 1
writer.execute("insert into QuestionsProcessed(question,code,tags,w
ords_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_d
up_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: 424
Percent of questions containing code: 57
Time taken to run this cell : 0:19:42.738593

```
In [8]: # never forget to close the connections or else we will end up with data
base locks
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()
```

Sample quesitons after preprocessing of data

```
In [9]: 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 1
0")
            print("Questions after preprocessed")
            print('='*100)
            reader.fetchone()
            for row in reader:
                print(row)
                print('-'*100)
        conn_r.commit()
        conn_r.close()
```

Questions after preprocessed

```
=====
=====
('dynam datagrid bind silverlight dynam datagrid bind silverlight dynam
datagrid bind silverlight bind datagrid dynam code wrote code debug cod
e block seem bind correct grid come column form come grid column althou
gh necessari bind nthank repli advance..',)
-----
-----
('java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryval
id iava lang noclassdeffounderror iavax servlet isp tagext taglibrarvva
```



```
id java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryv  
lid java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryv  
alid follow guid link instal jstl got follow error tri launch jsp page  
java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid  
taglib declar instal jstl 1.1 tomcat webapp tri project work also tri v  
ersion 1.2 jstl still messag caus solv',)
```

```
-----  
-----  
( 'java.sql.sqlexcept microsoft odbc driver manag invalid descriptor ind  
ex java.sql.sqlexcept microsoft odbc driver manag invalid descriptor in  
dex java.sql.sqlexcept microsoft odbc driver manag invalid descriptor i  
ndex use follow code display caus solv',)
```

```
-----  
-----  
( 'better way updat feed fb php sdk better way updat feed fb php sdk bet  
ter way updat feed fb php sdk novic facebook api read mani tutori still  
confused.i find post feed api method like correct second way use curl s  
ometh like way better',)
```

```
-----  
-----  
( 'btnadd click event open two window record ad btnadd click event open  
two window record ad btnadd click event open two window record ad open  
window search.aspx use code hav add button search.aspx nwhen insert rec  
ord btnadd click event open anoth window nafter insert record close win  
dow',)
```

```
-----  
-----  
( 'sql inject issu prevent correct form submit php sql inject issu prev  
ent correct form submit php sql inject issu prevent correct form submi  
ss php check everyth think make sure input field safe type sql inject g  
ood news safe bad news one tag mess form submit place even touch life  
figur exact html use templat file forgiv okay entir php script get exec  
ut see data post none forum field post problem use someth titl field no  
ne data get post current use print post see submit noth work flawless s  
tatement though also mention script work flawless local machin use host  
come across problem state list input test mess',)
```

```
-----  
-----  
( 'countabl subaddit lebesgu measur countabl subaddit lebesgu measur cou  
ntabl subaddit lebesgu measur let lbrace rbrace sequenc set sigma -alge
```

```

      next subaddit lebesgu measur let trace trace sequenc set sigma alge
bra mathcal want show left bigcup right leq sum left right countabl add
it measur defin set sigma algebra mathcal think use monoton properti so
mewher proof start appreci littl help nthank ad han answer make follow
addit construct given han answer clear bigcup bigcup cap emptyset neq l
eft bigcup right left bigcup right sum left right also construct subset
monoton left right leq left right final would sum leq sum result follo
w',)

```

```

-----
('hql equival sql queri hql equival sql queri hql equival sql queri hql
queri replac name class properti name error occur hql error',)
-----

```

```

-----
('undefin symbol architectur i386 objc class skpsmtpmessag referenc err
or undefin symbol architectur i386 objc class skpsmtpmessag referenc er
ror undefin symbol architectur i386 objc class skpsmtpmessag referenc e
rror import framework send email applic background import framework i.e
skpsmtpmessag somebodi suggest get error collect2 ld return exit status
import framework correct sorc taken framework follow mfmcomposeviewc
ontrol question lock field updat answer drag drop folder project click
copi nthat',)
-----

```

Saving Preprocessed data to a Database

```

In [8]: #Taking 0.5 Million entries to a dataframe.
write_db = 'Titlmoreweight.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 [9]: preprocessed_data.head()

```

Out[9]:

	question	tags
0	dynam datagrid bind silverlight dynam datagrid...	c# silverlight data-binding
1	dynam datagrid bind silverlight dynam datagrid...	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 updat feed fb php sdk better way up...	facebook api facebook-php-sdk

```
In [10]: 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 [11]: vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

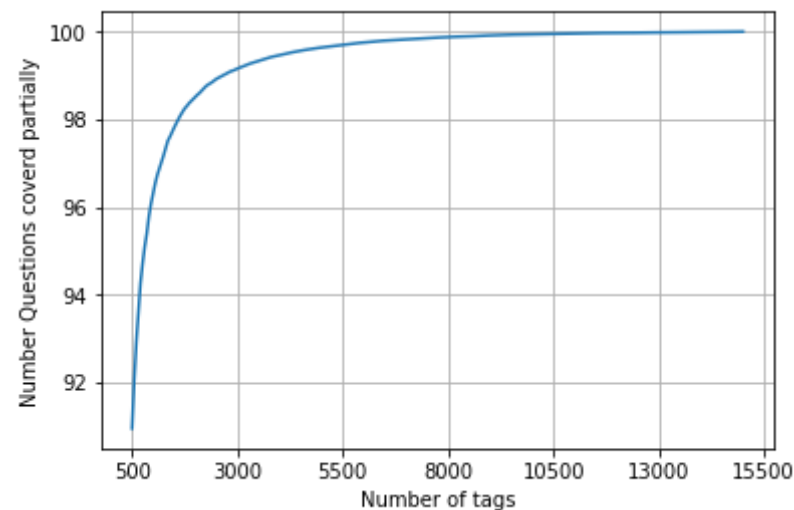
Selecting 500 Tags

```
In [12]: 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 [13]: 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 [81]: 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 covered partially")
plt.grid()
plt.show()
# you can choose any number of tags based on your computing power, mini
mun is 500(it covers 90% of the tags)
print("with ",5500,"tags we are covering ",questions_explained[50],"% o
f questions")
print("with ",500,"tags we are covering ",questions_explained[0],"% of
questions")
```



with 5500 tags we are covering 99.157 % of questions

with 5000 tags we are covering 99.157 % of questions
with 500 tags we are covering 90.956 % of questions

```
In [14]: # 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 [15]: train_datasize= 400000
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 [16]: 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)

Applying different models

1. Try OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-log)
2. Use bag of words upto 4 grams and compute the micro f1 score with Logistic regression(OvR)
3. Perform hyperparam tuning on alpha (or lambda) for Logistic regression to improve the performance using GridSearch
4. Try OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-hinge)

OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-log)

```

In [21]: # job= -1 gave error of unable to pipeline sparse data so taking default value of job
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=10**-5, penalty='l1'))
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 : 0.10695
Hamming loss 0.00600426
Micro-average quality numbers
Precision: 0.2841, Recall: 0.4786, F1-measure: 0.3566
Macro-average quality numbers
Precision: 0.2078, Recall: 0.4110, F1-measure: 0.2678
precision recall f1-score support

```

0	0.72	0.79	0.75	5519
1	0.42	0.46	0.44	8190
2	0.50	0.51	0.50	6529
3	0.48	0.62	0.54	3231
4	0.54	0.54	0.54	6430
5	0.40	0.51	0.45	2879
6	0.59	0.63	0.61	5086
7	0.57	0.67	0.61	4533
8	0.21	0.23	0.22	3000
9	0.53	0.66	0.59	2765
10	0.32	0.31	0.31	3051
11	0.43	0.54	0.48	3009
12	0.38	0.41	0.40	2630
13	0.33	0.46	0.39	1426
14	0.57	0.68	0.62	2548
15	0.34	0.38	0.36	2371
16	0.25	0.35	0.30	873
17	0.57	0.71	0.63	2151
18	0.30	0.35	0.32	2204
19	0.26	0.50	0.34	831
20	0.50	0.57	0.53	1860
21	0.17	0.24	0.20	2023
22	0.29	0.39	0.33	1513
23	0.46	0.67	0.55	1207
24	0.24	0.46	0.31	506
25	0.20	0.45	0.28	425
26	0.34	0.50	0.40	793
27	0.39	0.48	0.43	1291
28	0.40	0.50	0.45	1208
29	0.13	0.22	0.16	406
30	0.19	0.39	0.25	504
31	0.13	0.21	0.16	732
32	0.22	0.45	0.30	441
33	0.33	0.47	0.39	1645
34	0.29	0.33	0.31	1058
35	0.44	0.62	0.51	946
36	0.19	0.42	0.27	644
37	0.28	0.72	0.40	136

38	0.27	0.53	0.36	570
39	0.24	0.42	0.31	766
40	0.33	0.45	0.38	1132
41	0.09	0.33	0.14	174
42	0.31	0.62	0.41	210
43	0.34	0.54	0.42	433
44	0.36	0.53	0.43	626
45	0.28	0.46	0.35	852
46	0.31	0.59	0.41	534
47	0.16	0.36	0.22	350
48	0.34	0.54	0.42	496
49	0.54	0.71	0.61	785
50	0.12	0.26	0.17	475
51	0.09	0.28	0.14	305
52	0.07	0.18	0.10	251
53	0.30	0.45	0.36	914
54	0.22	0.30	0.26	728
55	0.07	0.15	0.10	258
56	0.21	0.38	0.27	821
57	0.14	0.22	0.17	541
58	0.28	0.43	0.34	748
59	0.65	0.73	0.68	724
60	0.19	0.32	0.24	660
61	0.15	0.36	0.21	235
62	0.52	0.80	0.63	718
63	0.49	0.76	0.59	468
64	0.17	0.47	0.25	191
65	0.11	0.25	0.15	429
66	0.13	0.22	0.17	415
67	0.28	0.61	0.39	274
68	0.36	0.64	0.46	510
69	0.29	0.54	0.37	466
70	0.09	0.20	0.12	305
71	0.13	0.30	0.18	247
72	0.36	0.57	0.44	401
73	0.35	0.83	0.49	86
74	0.18	0.49	0.26	120
75	0.32	0.77	0.46	129
76	0.09	0.13	0.10	473

77	0.10	0.36	0.16	143
78	0.33	0.60	0.43	347
79	0.20	0.37	0.26	479
80	0.20	0.52	0.29	279
81	0.19	0.32	0.24	461
82	0.07	0.13	0.09	298
83	0.36	0.57	0.44	396
84	0.15	0.45	0.23	184
85	0.25	0.38	0.30	573
86	0.11	0.18	0.13	325
87	0.18	0.47	0.26	273
88	0.09	0.36	0.15	135
89	0.13	0.31	0.18	232
90	0.27	0.45	0.33	409
91	0.25	0.45	0.32	420
92	0.41	0.58	0.48	408
93	0.24	0.57	0.34	241
94	0.08	0.18	0.11	211
95	0.14	0.30	0.19	277
96	0.12	0.22	0.15	410
97	0.43	0.61	0.50	501
98	0.20	0.70	0.30	136
99	0.21	0.45	0.29	239
100	0.13	0.26	0.17	324
101	0.47	0.77	0.58	277
102	0.69	0.79	0.74	613
103	0.12	0.32	0.17	157
104	0.09	0.18	0.12	295
105	0.31	0.52	0.39	334
106	0.26	0.40	0.31	335
107	0.31	0.57	0.41	389
108	0.18	0.41	0.25	251
109	0.28	0.46	0.35	317
110	0.07	0.19	0.10	187
111	0.09	0.34	0.14	140
112	0.25	0.63	0.36	154
113	0.22	0.36	0.27	332
114	0.20	0.38	0.26	323
115	0.19	0.35	0.25	344

116	0.41	0.57	0.47	370
117	0.21	0.41	0.27	313
118	0.61	0.81	0.69	874
119	0.14	0.33	0.20	293
120	0.04	0.17	0.07	200
121	0.38	0.57	0.45	463
122	0.11	0.31	0.16	119
123	0.04	0.05	0.05	256
124	0.48	0.74	0.58	195
125	0.10	0.32	0.15	138
126	0.36	0.62	0.46	376
127	0.02	0.09	0.04	122
128	0.06	0.12	0.08	252
129	0.19	0.38	0.25	144
130	0.08	0.23	0.12	150
131	0.05	0.13	0.08	210
132	0.22	0.40	0.28	361
133	0.53	0.70	0.61	453
134	0.43	0.82	0.57	124
135	0.05	0.20	0.07	91
136	0.10	0.41	0.16	128
137	0.20	0.47	0.28	218
138	0.12	0.33	0.18	243
139	0.12	0.36	0.18	149
140	0.39	0.58	0.47	318
141	0.08	0.19	0.11	159
142	0.28	0.47	0.35	274
143	0.57	0.83	0.68	362
144	0.08	0.33	0.13	118
145	0.15	0.41	0.21	164
146	0.24	0.45	0.32	461
147	0.24	0.60	0.34	159
148	0.07	0.26	0.11	166
149	0.44	0.65	0.52	346
150	0.16	0.24	0.19	350
151	0.19	0.73	0.30	55
152	0.37	0.63	0.47	387
153	0.20	0.31	0.25	150
154	0.10	0.18	0.13	281

155	0.09	0.22	0.13	202
156	0.29	0.69	0.41	130
157	0.13	0.26	0.17	245
158	0.48	0.73	0.58	177
159	0.13	0.45	0.20	130
160	0.18	0.34	0.24	336
161	0.40	0.73	0.52	220
162	0.09	0.26	0.14	229
163	0.32	0.55	0.40	316
164	0.27	0.53	0.36	283
165	0.18	0.43	0.25	197
166	0.17	0.57	0.27	101
167	0.13	0.29	0.18	231
168	0.19	0.39	0.25	370
169	0.19	0.40	0.26	258
170	0.06	0.25	0.10	101
171	0.09	0.28	0.14	89
172	0.19	0.46	0.27	193
173	0.25	0.44	0.32	309
174	0.08	0.24	0.12	172
175	0.32	0.85	0.46	95
176	0.57	0.73	0.64	346
177	0.42	0.67	0.52	322
178	0.28	0.59	0.38	232
179	0.06	0.16	0.09	125
180	0.15	0.47	0.23	145
181	0.04	0.25	0.07	77
182	0.08	0.21	0.11	182
183	0.23	0.52	0.32	257
184	0.08	0.21	0.11	216
185	0.13	0.29	0.18	242
186	0.10	0.24	0.14	165
187	0.32	0.60	0.42	263
188	0.08	0.20	0.11	174
189	0.41	0.55	0.47	136
190	0.35	0.68	0.46	202
191	0.08	0.28	0.13	134
192	0.23	0.51	0.32	230
193	0.06	0.21	0.09	90

194	0.26	0.57	0.36	185
195	0.04	0.14	0.07	156
196	0.06	0.23	0.10	160
197	0.12	0.28	0.17	266
198	0.16	0.31	0.21	284
199	0.08	0.18	0.11	145
200	0.48	0.83	0.61	212
201	0.21	0.43	0.28	317
202	0.46	0.67	0.54	427
203	0.11	0.25	0.16	232
204	0.16	0.41	0.23	217
205	0.35	0.56	0.43	527
206	0.04	0.15	0.07	124
207	0.17	0.36	0.23	103
208	0.33	0.59	0.42	287
209	0.08	0.19	0.12	193
210	0.22	0.47	0.30	220
211	0.10	0.36	0.15	140
212	0.06	0.24	0.10	161
213	0.24	0.64	0.35	72
214	0.39	0.49	0.44	396
215	0.19	0.50	0.27	134
216	0.24	0.35	0.29	400
217	0.10	0.44	0.16	75
218	0.57	0.81	0.67	219
219	0.25	0.49	0.33	210
220	0.47	0.70	0.56	298
221	0.55	0.76	0.64	266
222	0.39	0.53	0.45	290
223	0.02	0.11	0.04	128
224	0.22	0.52	0.31	159
225	0.15	0.39	0.22	164
226	0.20	0.42	0.27	144
227	0.27	0.49	0.35	276
228	0.03	0.09	0.05	235
229	0.05	0.16	0.08	216
230	0.09	0.24	0.13	228
231	0.15	0.62	0.24	64
232	0.07	0.24	0.11	103

233	0.24	0.44	0.31	216
234	0.14	0.28	0.19	116
235	0.15	0.44	0.22	77
236	0.32	0.81	0.45	67
237	0.11	0.23	0.15	218
238	0.09	0.27	0.14	139
239	0.03	0.07	0.04	94
240	0.09	0.40	0.15	77
241	0.06	0.16	0.09	167
242	0.21	0.47	0.29	86
243	0.05	0.26	0.08	58
244	0.30	0.47	0.37	269
245	0.09	0.37	0.15	112
246	0.62	0.83	0.71	255
247	0.06	0.29	0.10	58
248	0.02	0.16	0.04	81
249	0.03	0.13	0.05	131
250	0.11	0.40	0.18	93
251	0.14	0.49	0.22	154
252	0.05	0.15	0.07	129
253	0.13	0.41	0.20	83
254	0.09	0.19	0.13	191
255	0.09	0.19	0.12	219
256	0.05	0.18	0.07	130
257	0.12	0.41	0.19	93
258	0.34	0.59	0.43	217
259	0.11	0.33	0.17	141
260	0.25	0.40	0.31	143
261	0.13	0.29	0.18	219
262	0.14	0.45	0.21	107
263	0.21	0.33	0.26	236
264	0.08	0.34	0.13	119
265	0.11	0.38	0.17	72
266	0.06	0.23	0.10	70
267	0.11	0.28	0.16	107
268	0.27	0.54	0.36	169
269	0.17	0.33	0.22	129
270	0.32	0.55	0.40	159
271	0.27	0.58	0.37	190

272	0.19	0.35	0.24	248
273	0.61	0.82	0.70	264
274	0.48	0.75	0.59	105
275	0.06	0.21	0.09	104
276	0.02	0.09	0.04	115
277	0.41	0.69	0.51	170
278	0.30	0.57	0.39	145
279	0.55	0.79	0.65	230
280	0.13	0.46	0.20	80
281	0.40	0.67	0.50	217
282	0.35	0.65	0.45	175
283	0.15	0.26	0.19	269
284	0.19	0.49	0.27	74
285	0.30	0.58	0.39	206
286	0.45	0.74	0.56	227
287	0.21	0.59	0.31	130
288	0.06	0.14	0.08	129
289	0.07	0.29	0.11	80
290	0.06	0.21	0.09	99
291	0.28	0.50	0.36	208
292	0.05	0.27	0.09	67
293	0.27	0.61	0.38	109
294	0.14	0.45	0.21	140
295	0.11	0.27	0.16	241
296	0.09	0.31	0.14	72
297	0.06	0.24	0.10	107
298	0.27	0.61	0.37	61
299	0.31	0.55	0.40	77
300	0.06	0.21	0.09	111
301	0.01	0.03	0.02	126
302	0.06	0.15	0.08	73
303	0.27	0.58	0.36	176
304	0.65	0.80	0.71	230
305	0.49	0.79	0.61	156
306	0.24	0.51	0.33	146
307	0.08	0.28	0.12	98
308	0.02	0.10	0.03	78
309	0.09	0.28	0.13	94
310	0.19	0.47	0.27	162

311	0.37	0.58	0.45	116
312	0.08	0.37	0.14	57
313	0.04	0.12	0.06	65
314	0.18	0.44	0.26	138
315	0.23	0.38	0.29	195
316	0.15	0.39	0.22	69
317	0.10	0.30	0.15	134
318	0.25	0.45	0.32	148
319	0.41	0.59	0.48	161
320	0.07	0.32	0.12	104
321	0.38	0.62	0.47	156
322	0.22	0.51	0.31	134
323	0.34	0.52	0.41	232
324	0.11	0.32	0.16	92
325	0.16	0.42	0.23	197
326	0.05	0.13	0.07	126
327	0.03	0.10	0.05	115
328	0.48	0.74	0.58	198
329	0.14	0.46	0.22	125
330	0.13	0.37	0.19	81
331	0.08	0.20	0.12	94
332	0.06	0.21	0.09	56
333	0.08	0.16	0.11	260
334	0.07	0.25	0.11	60
335	0.09	0.17	0.12	110
336	0.23	0.63	0.34	71
337	0.04	0.17	0.06	66
338	0.20	0.45	0.28	150
339	0.01	0.04	0.01	54
340	0.47	0.67	0.55	195
341	0.29	0.61	0.39	79
342	0.09	0.42	0.14	38
343	0.11	0.47	0.18	43
344	0.18	0.32	0.24	68
345	0.20	0.49	0.28	73
346	0.05	0.15	0.07	116
347	0.18	0.58	0.27	111
348	0.05	0.21	0.08	63
349	0.43	0.69	0.53	104

350	0.18	0.55	0.27	44
351	0.14	0.35	0.20	40
352	0.36	0.59	0.45	136
353	0.10	0.41	0.16	54
354	0.07	0.29	0.12	134
355	0.22	0.49	0.30	120
356	0.28	0.46	0.35	228
357	0.34	0.52	0.41	269
358	0.16	0.46	0.24	80
359	0.33	0.70	0.45	140
360	0.09	0.27	0.13	125
361	0.66	0.74	0.70	169
362	0.03	0.12	0.05	56
363	0.54	0.80	0.65	154
364	0.10	0.28	0.15	58
365	0.11	0.30	0.16	71
366	0.35	0.74	0.48	54
367	0.04	0.13	0.07	116
368	0.03	0.11	0.05	54
369	0.02	0.11	0.04	71
370	0.01	0.03	0.01	61
371	0.04	0.15	0.06	71
372	0.18	0.50	0.26	52
373	0.33	0.54	0.41	150
374	0.12	0.40	0.19	93
375	0.02	0.07	0.03	67
376	0.01	0.04	0.02	76
377	0.15	0.42	0.22	106
378	0.03	0.10	0.05	86
379	0.03	0.36	0.05	14
380	0.23	0.58	0.33	122
381	0.03	0.13	0.05	104
382	0.04	0.20	0.06	66
383	0.15	0.43	0.22	110
384	0.04	0.10	0.06	155
385	0.07	0.48	0.13	50
386	0.09	0.36	0.14	64
387	0.06	0.19	0.09	93
388	0.17	0.45	0.25	102

389	0.02	0.07	0.04	108
390	0.61	0.75	0.67	178
391	0.13	0.32	0.18	115
392	0.14	0.55	0.23	42
393	0.02	0.04	0.02	134
394	0.05	0.19	0.09	112
395	0.15	0.40	0.21	176
396	0.10	0.26	0.14	125
397	0.38	0.60	0.47	224
398	0.32	0.76	0.45	63
399	0.02	0.14	0.04	59
400	0.11	0.46	0.18	63
401	0.12	0.40	0.19	98
402	0.14	0.35	0.20	162
403	0.11	0.39	0.17	83
404	0.18	0.79	0.29	19
405	0.08	0.21	0.12	92
406	0.09	0.49	0.15	41
407	0.20	0.51	0.29	43
408	0.28	0.53	0.37	160
409	0.08	0.28	0.13	50
410	0.01	0.11	0.02	19
411	0.15	0.31	0.21	175
412	0.06	0.19	0.10	72
413	0.05	0.14	0.07	95
414	0.12	0.31	0.17	97
415	0.06	0.21	0.10	48
416	0.17	0.41	0.24	83
417	0.04	0.20	0.07	40
418	0.10	0.25	0.14	91
419	0.19	0.46	0.26	90
420	0.08	0.38	0.13	37
421	0.07	0.24	0.11	66
422	0.14	0.45	0.21	73
423	0.09	0.36	0.15	56
424	0.57	0.94	0.71	33
425	0.07	0.22	0.11	76
426	0.04	0.11	0.06	81
427	0.58	0.79	0.67	150

428	0.34	0.72	0.46	29
429	0.87	0.89	0.88	389
430	0.31	0.54	0.40	167
431	0.06	0.20	0.09	123
432	0.13	0.51	0.21	39
433	0.14	0.45	0.21	82
434	0.48	0.74	0.58	66
435	0.21	0.53	0.30	93
436	0.18	0.52	0.26	87
437	0.05	0.15	0.08	86
438	0.36	0.64	0.46	104
439	0.08	0.26	0.13	100
440	0.07	0.14	0.09	141
441	0.14	0.46	0.21	110
442	0.09	0.29	0.13	123
443	0.12	0.31	0.17	71
444	0.09	0.21	0.13	109
445	0.08	0.42	0.14	48
446	0.13	0.43	0.20	76
447	0.05	0.32	0.08	38
448	0.25	0.75	0.37	81
449	0.22	0.36	0.27	132
450	0.15	0.38	0.22	81
451	0.13	0.50	0.21	76
452	0.08	0.23	0.12	44
453	0.02	0.07	0.03	44
454	0.26	0.64	0.37	70
455	0.13	0.37	0.19	155
456	0.07	0.26	0.11	43
457	0.15	0.53	0.23	72
458	0.03	0.18	0.05	62
459	0.07	0.33	0.11	69
460	0.03	0.09	0.05	119
461	0.31	0.43	0.36	79
462	0.05	0.17	0.08	47
463	0.11	0.38	0.17	104
464	0.28	0.47	0.35	106
465	0.09	0.31	0.14	64
466	0.26	0.44	0.32	173

467	0.21	0.48	0.29	107
468	0.15	0.36	0.21	126
469	0.05	0.11	0.07	114
470	0.64	0.84	0.73	140
471	0.21	0.47	0.29	79
472	0.22	0.44	0.30	143
473	0.33	0.53	0.41	158
474	0.09	0.22	0.13	138
475	0.04	0.20	0.07	59
476	0.24	0.49	0.32	88
477	0.46	0.73	0.57	176
478	0.38	0.88	0.53	24
479	0.07	0.24	0.10	92
480	0.32	0.56	0.41	100
481	0.23	0.43	0.30	103
482	0.10	0.35	0.16	74
483	0.40	0.73	0.52	105
484	0.06	0.16	0.09	83
485	0.03	0.11	0.05	82
486	0.07	0.30	0.11	71
487	0.18	0.36	0.24	120
488	0.05	0.13	0.07	105
489	0.26	0.53	0.35	87
490	0.47	0.84	0.61	32
491	0.01	0.07	0.02	69
492	0.03	0.10	0.04	49
493	0.02	0.06	0.03	117
494	0.12	0.38	0.19	61
495	0.78	0.84	0.81	344
496	0.09	0.17	0.12	52
497	0.20	0.31	0.25	137
498	0.13	0.28	0.17	98
499	0.06	0.27	0.10	79

avg / total	0.34	0.48	0.39	173812
-------------	------	------	------	--------

Time taken to run this cell : 0:35:22.530511

1 Applying Logistic Regression

Featurizing data with BOW vectorizer

```
In [18]: start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, max_features=200000, \
                             tokenizer = lambda x: x.split(), ngram_range=(1,4))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

Time taken to run this cell : 0:07:59.145589

```
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, 95585) Y : (400000, 500)
Dimensions of test data X: (100000, 95585) Y: (100000, 500)

Applying Logistic Regression with OneVsRest Classifier

```
In [34]: # job= -1 gave error of unable to pipeline sparse data so taking default value of job
start = datetime.now()
classifier_2 = OneVsRestClassifier(LogisticRegression(penalty='l1', tol=0.001))
classifier_2.fit(x_train_multilabel, y_train)
predictions_2 = classifier_2.predict(x_test_multilabel)
print("Accuracy :",metrics.accuracy_score(y_test, predictions_2))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions_2))

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

```

f1 = f1_score(y_test, predictions_2, average='micro')

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

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

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

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

```

```

Accuracy : 0.21238
Hamming loss 0.00312846
Micro-average quality numbers
Precision: 0.5695, Recall: 0.4098, F1-measure: 0.4767
Macro-average quality numbers
Precision: 0.4520, Recall: 0.3340, F1-measure: 0.3805

```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.90	0.74	0.81	5519
1	0.52	0.41	0.46	8190
2	0.64	0.47	0.54	6529
3	0.69	0.53	0.60	3231
4	0.66	0.49	0.56	6430
5	0.62	0.42	0.50	2879
6	0.74	0.57	0.64	5086
7	0.75	0.62	0.68	4533
8	0.34	0.18	0.24	3000
9	0.69	0.59	0.64	2765
10	0.42	0.30	0.35	3051
11	0.59	0.45	0.51	3009
12	0.48	0.36	0.41	2630
13	0.54	0.38	0.44	1426
14	0.80	0.61	0.69	2548

15	0.48	0.30	0.37	2371
16	0.54	0.29	0.38	873
17	0.79	0.65	0.71	2151
18	0.44	0.30	0.36	2204
19	0.56	0.45	0.49	831
20	0.70	0.49	0.57	1860
21	0.26	0.18	0.21	2023
22	0.40	0.30	0.34	1513
23	0.77	0.58	0.66	1207
24	0.45	0.34	0.39	506
25	0.53	0.37	0.43	425
26	0.57	0.44	0.50	793
27	0.54	0.41	0.47	1291
28	0.60	0.41	0.49	1208
29	0.28	0.17	0.21	406
30	0.46	0.24	0.32	504
31	0.20	0.14	0.16	732
32	0.47	0.32	0.38	441
33	0.52	0.36	0.43	1645
34	0.48	0.29	0.36	1058
35	0.73	0.57	0.64	946
36	0.47	0.29	0.36	644
37	0.90	0.70	0.79	136
38	0.51	0.38	0.43	570
39	0.63	0.34	0.44	766
40	0.53	0.42	0.47	1132
41	0.35	0.31	0.33	174
42	0.66	0.55	0.60	210
43	0.64	0.45	0.53	433
44	0.60	0.48	0.53	626
45	0.57	0.38	0.46	852
46	0.64	0.48	0.55	534
47	0.30	0.24	0.26	350
48	0.64	0.54	0.58	496
49	0.76	0.64	0.69	785
50	0.20	0.12	0.15	475
51	0.26	0.21	0.23	305
52	0.24	0.11	0.15	251
53	0.54	0.41	0.47	914

54	0.39	0.25	0.30	728
55	0.16	0.08	0.11	258
56	0.35	0.29	0.32	821
57	0.35	0.19	0.25	541
58	0.59	0.35	0.44	748
59	0.90	0.70	0.79	724
60	0.35	0.19	0.24	660
61	0.44	0.24	0.31	235
62	0.88	0.72	0.79	718
63	0.78	0.69	0.73	468
64	0.44	0.31	0.36	191
65	0.29	0.18	0.22	429
66	0.22	0.12	0.16	415
67	0.68	0.54	0.60	274
68	0.72	0.54	0.62	510
69	0.61	0.50	0.55	466
70	0.26	0.16	0.20	305
71	0.34	0.22	0.27	247
72	0.71	0.52	0.60	401
73	0.85	0.80	0.83	86
74	0.56	0.42	0.48	120
75	0.83	0.71	0.76	129
76	0.13	0.05	0.07	473
77	0.37	0.29	0.32	143
78	0.68	0.47	0.56	347
79	0.50	0.27	0.35	479
80	0.44	0.37	0.40	279
81	0.51	0.28	0.36	461
82	0.13	0.06	0.09	298
83	0.72	0.52	0.60	396
84	0.40	0.36	0.38	184
85	0.43	0.30	0.35	573
86	0.27	0.12	0.17	325
87	0.51	0.42	0.46	273
88	0.46	0.32	0.38	135
89	0.25	0.17	0.20	232
90	0.50	0.41	0.45	409
91	0.51	0.33	0.40	420
92	0.69	0.56	0.62	408

93	0.55	0.50	0.52	241
94	0.20	0.09	0.13	211
95	0.30	0.17	0.21	277
96	0.22	0.12	0.15	410
97	0.77	0.47	0.58	501
98	0.67	0.65	0.66	136
99	0.46	0.37	0.41	239
100	0.34	0.20	0.25	324
101	0.85	0.74	0.79	277
102	0.89	0.75	0.82	613
103	0.37	0.23	0.28	157
104	0.20	0.11	0.14	295
105	0.67	0.46	0.54	334
106	0.69	0.36	0.47	335
107	0.70	0.58	0.63	389
108	0.52	0.33	0.41	251
109	0.56	0.47	0.51	317
110	0.30	0.11	0.16	187
111	0.44	0.18	0.25	140
112	0.58	0.47	0.52	154
113	0.49	0.28	0.36	332
114	0.43	0.28	0.34	323
115	0.43	0.32	0.37	344
116	0.67	0.54	0.60	370
117	0.44	0.30	0.36	313
118	0.76	0.75	0.75	874
119	0.36	0.26	0.30	293
120	0.13	0.09	0.10	200
121	0.66	0.50	0.57	463
122	0.23	0.12	0.16	119
123	0.14	0.04	0.06	256
124	0.86	0.71	0.78	195
125	0.29	0.17	0.22	138
126	0.71	0.53	0.60	376
127	0.15	0.07	0.09	122
128	0.13	0.06	0.08	252
129	0.42	0.39	0.41	144
130	0.31	0.18	0.23	150
131	0.19	0.09	0.12	210

132	0.51	0.33	0.40	361
133	0.84	0.64	0.73	453
134	0.80	0.77	0.79	124
135	0.15	0.12	0.13	91
136	0.51	0.38	0.44	128
137	0.45	0.39	0.42	218
138	0.35	0.22	0.27	243
139	0.29	0.19	0.23	149
140	0.70	0.53	0.60	318
141	0.19	0.12	0.15	159
142	0.57	0.43	0.49	274
143	0.81	0.83	0.82	362
144	0.43	0.28	0.34	118
145	0.50	0.43	0.46	164
146	0.51	0.39	0.44	461
147	0.69	0.43	0.53	159
148	0.33	0.20	0.25	166
149	0.91	0.59	0.72	346
150	0.49	0.22	0.30	350
151	0.90	0.67	0.77	55
152	0.70	0.52	0.60	387
153	0.37	0.33	0.35	150
154	0.32	0.15	0.20	281
155	0.25	0.19	0.22	202
156	0.75	0.65	0.70	130
157	0.21	0.10	0.14	245
158	0.90	0.69	0.78	177
159	0.46	0.40	0.43	130
160	0.38	0.24	0.29	336
161	0.79	0.65	0.71	220
162	0.19	0.10	0.13	229
163	0.78	0.46	0.58	316
164	0.63	0.42	0.50	283
165	0.53	0.37	0.44	197
166	0.54	0.53	0.54	101
167	0.37	0.23	0.28	231
168	0.43	0.35	0.39	370
169	0.39	0.23	0.29	258
170	0.27	0.16	0.20	101

171	0.34	0.28	0.31	89
172	0.48	0.37	0.42	193
173	0.46	0.33	0.39	309
174	0.31	0.14	0.19	172
175	0.78	0.75	0.76	95
176	0.85	0.63	0.72	346
177	0.81	0.60	0.69	322
178	0.54	0.47	0.50	232
179	0.21	0.10	0.14	125
180	0.46	0.41	0.43	145
181	0.30	0.21	0.25	77
182	0.18	0.12	0.14	182
183	0.51	0.37	0.43	257
184	0.22	0.13	0.16	216
185	0.31	0.20	0.24	242
186	0.33	0.22	0.26	165
187	0.68	0.56	0.61	263
188	0.19	0.10	0.13	174
189	0.64	0.47	0.54	136
190	0.82	0.59	0.68	202
191	0.29	0.21	0.24	134
192	0.59	0.46	0.52	230
193	0.25	0.19	0.22	90
194	0.56	0.56	0.56	185
195	0.16	0.08	0.10	156
196	0.14	0.09	0.11	160
197	0.28	0.17	0.21	266
198	0.28	0.15	0.20	284
199	0.19	0.08	0.11	145
200	0.86	0.77	0.81	212
201	0.49	0.26	0.34	317
202	0.70	0.63	0.66	427
203	0.20	0.14	0.17	232
204	0.38	0.29	0.33	217
205	0.49	0.48	0.49	527
206	0.15	0.06	0.09	124
207	0.41	0.35	0.38	103
208	0.77	0.55	0.64	287
209	0.20	0.11	0.15	193

210	0.55	0.39	0.45	220
211	0.45	0.20	0.28	140
212	0.15	0.09	0.11	161
213	0.46	0.53	0.49	72
214	0.60	0.43	0.50	396
215	0.67	0.42	0.51	134
216	0.47	0.26	0.34	400
217	0.32	0.25	0.28	75
218	0.93	0.77	0.84	219
219	0.59	0.42	0.49	210
220	0.84	0.67	0.74	298
221	0.89	0.71	0.79	266
222	0.66	0.45	0.53	290
223	0.12	0.05	0.07	128
224	0.70	0.48	0.57	159
225	0.40	0.35	0.37	164
226	0.47	0.34	0.39	144
227	0.54	0.39	0.46	276
228	0.09	0.04	0.06	235
229	0.14	0.06	0.09	216
230	0.32	0.20	0.25	228
231	0.63	0.53	0.58	64
232	0.23	0.16	0.18	103
233	0.62	0.39	0.48	216
234	0.51	0.23	0.32	116
235	0.45	0.32	0.38	77
236	0.88	0.69	0.77	67
237	0.29	0.18	0.22	218
238	0.26	0.18	0.21	139
239	0.22	0.06	0.10	94
240	0.41	0.31	0.35	77
241	0.31	0.13	0.19	167
242	0.62	0.41	0.49	86
243	0.31	0.24	0.27	58
244	0.53	0.40	0.46	269
245	0.12	0.08	0.10	112
246	0.92	0.82	0.87	255
247	0.21	0.21	0.21	58
248	0.14	0.07	0.10	81

249	0.05	0.02	0.03	131
250	0.39	0.26	0.31	93
251	0.56	0.34	0.43	154
252	0.10	0.05	0.06	129
253	0.45	0.34	0.39	83
254	0.24	0.13	0.17	191
255	0.11	0.06	0.08	219
256	0.13	0.08	0.10	130
257	0.39	0.31	0.35	93
258	0.63	0.52	0.57	217
259	0.27	0.18	0.22	141
260	0.65	0.24	0.35	143
261	0.40	0.18	0.25	219
262	0.47	0.36	0.41	107
263	0.36	0.23	0.28	236
264	0.26	0.20	0.23	119
265	0.43	0.28	0.34	72
266	0.11	0.06	0.07	70
267	0.34	0.23	0.28	107
268	0.53	0.47	0.50	169
269	0.29	0.17	0.21	129
270	0.69	0.53	0.60	159
271	0.77	0.53	0.63	190
272	0.45	0.33	0.39	248
273	0.85	0.75	0.80	264
274	0.83	0.67	0.74	105
275	0.23	0.13	0.17	104
276	0.05	0.03	0.03	115
277	0.77	0.60	0.67	170
278	0.72	0.48	0.57	145
279	0.88	0.75	0.81	230
280	0.59	0.40	0.48	80
281	0.65	0.55	0.59	217
282	0.69	0.52	0.59	175
283	0.26	0.17	0.21	269
284	0.54	0.38	0.44	74
285	0.70	0.51	0.59	206
286	0.84	0.70	0.76	227
287	0.66	0.42	0.52	130

288	0.16	0.08	0.10	129
289	0.16	0.11	0.13	80
290	0.19	0.14	0.16	99
291	0.59	0.39	0.47	208
292	0.24	0.12	0.16	67
293	0.77	0.54	0.63	109
294	0.35	0.27	0.31	140
295	0.24	0.17	0.20	241
296	0.22	0.14	0.17	72
297	0.21	0.15	0.18	107
298	0.64	0.57	0.60	61
299	0.72	0.55	0.62	77
300	0.15	0.12	0.13	111
301	0.03	0.01	0.01	126
302	0.17	0.11	0.13	73
303	0.55	0.44	0.49	176
304	0.91	0.81	0.86	230
305	0.82	0.71	0.76	156
306	0.43	0.37	0.40	146
307	0.21	0.11	0.15	98
308	0.04	0.01	0.02	78
309	0.47	0.17	0.25	94
310	0.58	0.38	0.46	162
311	0.72	0.50	0.59	116
312	0.45	0.35	0.40	57
313	0.37	0.11	0.17	65
314	0.41	0.36	0.39	138
315	0.49	0.28	0.36	195
316	0.40	0.32	0.35	69
317	0.26	0.21	0.23	134
318	0.54	0.41	0.47	148
319	0.80	0.56	0.66	161
320	0.18	0.18	0.18	104
321	0.69	0.62	0.65	156
322	0.54	0.46	0.49	134
323	0.53	0.44	0.48	232
324	0.21	0.16	0.19	92
325	0.37	0.23	0.29	197
326	0.10	0.07	0.08	126

327	0.14	0.06	0.08	115
328	0.95	0.71	0.81	198
329	0.43	0.30	0.35	125
330	0.57	0.26	0.36	81
331	0.34	0.15	0.21	94
332	0.29	0.20	0.23	56
333	0.15	0.08	0.10	260
334	0.16	0.12	0.13	60
335	0.22	0.10	0.14	110
336	0.58	0.54	0.56	71
337	0.13	0.09	0.11	66
338	0.41	0.45	0.43	150
339	0.05	0.04	0.04	54
340	0.79	0.57	0.66	195
341	0.68	0.51	0.58	79
342	0.38	0.50	0.43	38
343	0.55	0.37	0.44	43
344	0.33	0.28	0.30	68
345	0.64	0.34	0.45	73
346	0.11	0.07	0.08	116
347	0.61	0.49	0.54	111
348	0.24	0.19	0.21	63
349	0.84	0.71	0.77	104
350	0.58	0.57	0.57	44
351	0.25	0.28	0.26	40
352	0.77	0.62	0.69	136
353	0.40	0.22	0.29	54
354	0.25	0.12	0.16	134
355	0.52	0.42	0.46	120
356	0.43	0.30	0.35	228
357	0.55	0.42	0.48	269
358	0.56	0.36	0.44	80
359	0.75	0.63	0.68	140
360	0.30	0.19	0.23	125
361	0.87	0.74	0.80	169
362	0.17	0.12	0.15	56
363	0.83	0.76	0.79	154
364	0.19	0.19	0.19	58
365	0.23	0.15	0.18	71

366	0.90	0.67	0.77	54
367	0.14	0.10	0.12	116
368	0.26	0.19	0.22	54
369	0.09	0.06	0.07	71
370	0.23	0.11	0.15	61
371	0.29	0.10	0.15	71
372	0.51	0.44	0.47	52
373	0.59	0.47	0.52	150
374	0.26	0.22	0.23	93
375	0.12	0.09	0.10	67
376	0.08	0.03	0.04	76
377	0.46	0.37	0.41	106
378	0.07	0.02	0.04	86
379	0.23	0.21	0.22	14
380	0.76	0.56	0.64	122
381	0.10	0.06	0.07	104
382	0.26	0.15	0.19	66
383	0.49	0.40	0.44	110
384	0.17	0.07	0.10	155
385	0.41	0.36	0.38	50
386	0.21	0.12	0.16	64
387	0.26	0.12	0.16	93
388	0.48	0.33	0.39	102
389	0.12	0.06	0.08	108
390	0.91	0.68	0.78	178
391	0.36	0.21	0.26	115
392	0.66	0.45	0.54	42
393	0.04	0.01	0.01	134
394	0.28	0.15	0.20	112
395	0.39	0.32	0.36	176
396	0.29	0.16	0.21	125
397	0.64	0.45	0.53	224
398	0.75	0.67	0.71	63
399	0.11	0.07	0.09	59
400	0.44	0.41	0.43	63
401	0.45	0.33	0.38	98
402	0.45	0.24	0.31	162
403	0.24	0.18	0.21	83
404	0.64	0.84	0.73	19

405	0.21	0.15	0.17	92
406	0.47	0.39	0.43	41
407	0.50	0.33	0.39	43
408	0.68	0.49	0.57	160
409	0.12	0.08	0.10	50
410	0.00	0.00	0.00	19
411	0.30	0.22	0.25	175
412	0.22	0.15	0.18	72
413	0.22	0.09	0.13	95
414	0.24	0.14	0.18	97
415	0.16	0.10	0.12	48
416	0.43	0.33	0.37	83
417	0.23	0.15	0.18	40
418	0.22	0.12	0.16	91
419	0.52	0.42	0.47	90
420	0.28	0.24	0.26	37
421	0.07	0.05	0.06	66
422	0.47	0.41	0.44	73
423	0.38	0.29	0.33	56
424	0.88	0.88	0.88	33
425	0.11	0.04	0.06	76
426	0.06	0.02	0.04	81
427	0.92	0.73	0.81	150
428	1.00	0.76	0.86	29
429	0.98	0.95	0.97	389
430	0.56	0.44	0.49	167
431	0.46	0.15	0.22	123
432	0.26	0.18	0.21	39
433	0.32	0.28	0.30	82
434	0.90	0.71	0.80	66
435	0.55	0.46	0.50	93
436	0.49	0.37	0.42	87
437	0.17	0.09	0.12	86
438	0.63	0.49	0.55	104
439	0.46	0.21	0.29	100
440	0.17	0.07	0.10	141
441	0.41	0.42	0.41	110
442	0.24	0.20	0.22	123
443	0.28	0.20	0.23	71

444	0.22	0.12	0.15	109
445	0.42	0.35	0.39	48
446	0.40	0.28	0.33	76
447	0.23	0.26	0.25	38
448	0.59	0.56	0.57	81
449	0.43	0.27	0.33	132
450	0.42	0.33	0.37	81
451	0.67	0.38	0.49	76
452	0.11	0.07	0.08	44
453	0.00	0.00	0.00	44
454	0.73	0.54	0.62	70
455	0.29	0.25	0.27	155
456	0.31	0.26	0.28	43
457	0.37	0.31	0.33	72
458	0.19	0.16	0.17	62
459	0.42	0.32	0.36	69
460	0.13	0.08	0.10	119
461	0.61	0.35	0.45	79
462	0.31	0.26	0.28	47
463	0.34	0.26	0.29	104
464	0.56	0.42	0.48	106
465	0.34	0.27	0.30	64
466	0.45	0.33	0.38	173
467	0.60	0.44	0.51	107
468	0.41	0.29	0.34	126
469	0.15	0.05	0.08	114
470	0.91	0.81	0.86	140
471	0.58	0.38	0.46	79
472	0.40	0.40	0.40	143
473	0.64	0.39	0.49	158
474	0.28	0.12	0.16	138
475	0.20	0.15	0.17	59
476	0.63	0.45	0.53	88
477	0.74	0.67	0.70	176
478	0.90	0.79	0.84	24
479	0.29	0.17	0.22	92
480	0.68	0.58	0.63	100
481	0.38	0.36	0.37	103
482	0.26	0.15	0.19	74

483	0.72	0.60	0.65	105
484	0.19	0.07	0.10	83
485	0.05	0.04	0.04	82
486	0.30	0.18	0.23	71
487	0.37	0.23	0.28	120
488	0.23	0.10	0.13	105
489	0.54	0.39	0.45	87
490	0.90	0.84	0.87	32
491	0.05	0.03	0.04	69
492	0.14	0.06	0.09	49
493	0.06	0.04	0.05	117
494	0.49	0.38	0.43	61
495	0.95	0.80	0.87	344
496	0.19	0.12	0.14	52
497	0.49	0.34	0.41	137
498	0.34	0.15	0.21	98
499	0.32	0.23	0.27	79
avg / total	0.55	0.41	0.47	173812

Time taken to run this cell : 3:27:49.280084

- This model performed better than SGDClassifier with log loss with $\alpha = 10^{-5}$

2 Hyperparameter tuning using gridsearch

```
In [14]: # taking 4 parameters on which gridsearch is processed
from sklearn.model_selection import GridSearchCV
params= {"estimator__C": [10,1,0.1,0.01]}
logis= LogisticRegression(penalty='l1', tol=0.001)
gridcv= GridSearchCV(OneVsRestClassifier(logis), param_grid=params, scoring='f1_micro', verbose=5 ,n_jobs=-1)
gridcv.fit(x_train_multilabel, y_train)
```

Fitting 3 folds for each of 4 candidates, totalling 12 fits

```
[Parallel(n_jobs=-1)]: Done    8 out of   12 | elapsed: 1127.7min remaining: 563.8min
```

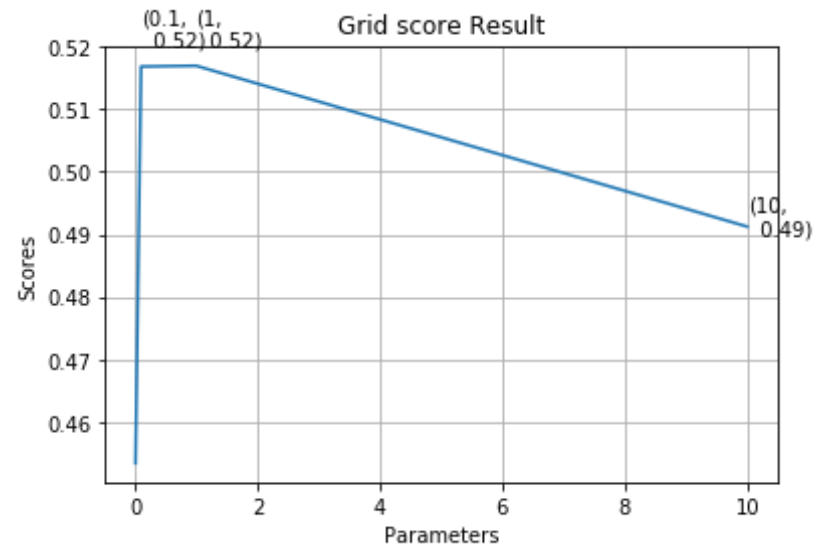
```
[Parallel(n_jobs=-1)]: Done 12 out of 12 | elapsed: 1187.4min finished
```

```
Out[14]: GridSearchCV(cv=None, error_score='raise',
                    estimator=OneVsRestClassifier(estimator=LogisticRegression(C=1.
0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=
1,
                    penalty='l1', random_state=None, solver='liblinear', tol=0.00
1,
                    verbose=0, warm_start=False),
                    n_jobs=1),
                    fit_params=None, iid=True, n_jobs=-1,
                    param_grid={'estimator__C': [10, 1, 0.1, 0.01]},
                    pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                    scoring='f1_micro', verbose=5)
```

```
In [37]: print(list(gridcv.cv_results_['mean_test_score']))
print(list(gridcv.cv_results_['param_estimator__C']))
```

```
[0.4912677727967754, 0.5169441162382484, 0.5168229185664773, 0.45350096
112743066]
[10, 1, 0.1, 0.01]
```

```
In [63]: scores = list(gridcv.cv_results_['mean_test_score'])
parm=list(gridcv.cv_results_['param_estimator__C'])
dff= pd.DataFrame(scores,parm)
plt.xlabel('Parameters')
plt.ylabel('Scores')
plt.title('Grid score Result')
for xy in zip(parm,np.round(scores,2)):
    plt.annotate('(%s,\n %s)' %xy,xy=xy,textcoords='data')
plt.grid()
plt.plot(dff)
plt.show()
```



- After doing hyperparameter tuning we got C=1 as optimal one which is the by default value for logistic regression function
- Now we put C=1 in logistic regression and check the performance

```
In [19]: # Putting C=1 in Model
start = datetime.now()
logis= LogisticRegression(penalty='l1',C=1, tol=0.001)
classifier = OneVsRestClassifier(logis)
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 : 0.21244
Hamming loss 0.00312824
Micro-average quality numbers
Precision: 0.5696, Recall: 0.4098, F1-measure: 0.4766
Macro-average quality numbers
Precision: 0.4523, Recall: 0.3341, F1-measure: 0.3806

```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.90	0.74	0.81	5519
1	0.52	0.41	0.46	8190
2	0.64	0.47	0.54	6529
3	0.68	0.53	0.59	3231
4	0.66	0.49	0.56	6430
5	0.62	0.42	0.50	2879
6	0.74	0.57	0.64	5086
7	0.75	0.62	0.68	4533
8	0.34	0.18	0.24	3000
9	0.69	0.59	0.64	2765
10	0.42	0.29	0.35	3051
11	0.59	0.45	0.51	3009
12	0.48	0.36	0.41	2630
13	0.54	0.38	0.44	1426
14	0.80	0.61	0.69	2548
15	0.48	0.31	0.37	2371

16	0.54	0.29	0.38	873
17	0.79	0.65	0.71	2151
18	0.44	0.30	0.36	2204
19	0.55	0.44	0.49	831
20	0.70	0.49	0.57	1860
21	0.26	0.18	0.21	2023
22	0.40	0.29	0.34	1513
23	0.76	0.58	0.66	1207
24	0.45	0.34	0.39	506
25	0.53	0.37	0.43	425
26	0.57	0.44	0.50	793
27	0.54	0.41	0.47	1291
28	0.60	0.41	0.49	1208
29	0.27	0.17	0.21	406
30	0.47	0.24	0.32	504
31	0.20	0.14	0.16	732
32	0.47	0.32	0.38	441
33	0.52	0.36	0.43	1645
34	0.48	0.29	0.36	1058
35	0.73	0.57	0.64	946
36	0.47	0.29	0.36	644
37	0.87	0.69	0.77	136
38	0.51	0.38	0.43	570
39	0.63	0.34	0.44	766
40	0.53	0.42	0.47	1132
41	0.35	0.31	0.33	174
42	0.66	0.55	0.60	210
43	0.64	0.45	0.53	433
44	0.60	0.49	0.54	626
45	0.56	0.38	0.46	852
46	0.64	0.48	0.55	534
47	0.30	0.24	0.26	350
48	0.64	0.54	0.58	496
49	0.76	0.64	0.70	785
50	0.19	0.11	0.14	475
51	0.26	0.21	0.23	305
52	0.24	0.11	0.15	251
53	0.53	0.41	0.47	914
54	0.39	0.25	0.30	728
55	0.16	0.08	0.11	258

55	0.10	0.08	0.11	258
56	0.35	0.29	0.31	821
57	0.35	0.19	0.25	541
58	0.59	0.35	0.44	748
59	0.90	0.70	0.79	724
60	0.35	0.19	0.24	660
61	0.44	0.24	0.31	235
62	0.88	0.72	0.79	718
63	0.78	0.70	0.74	468
64	0.44	0.31	0.36	191
65	0.29	0.18	0.22	429
66	0.22	0.12	0.16	415
67	0.68	0.55	0.61	274
68	0.72	0.54	0.62	510
69	0.61	0.50	0.55	466
70	0.26	0.16	0.20	305
71	0.34	0.22	0.27	247
72	0.71	0.52	0.60	401
73	0.84	0.79	0.81	86
74	0.56	0.42	0.48	120
75	0.83	0.71	0.76	129
76	0.13	0.05	0.07	473
77	0.37	0.29	0.32	143
78	0.69	0.47	0.56	347
79	0.50	0.27	0.35	479
80	0.44	0.37	0.40	279
81	0.51	0.28	0.36	461
82	0.13	0.06	0.09	298
83	0.72	0.52	0.60	396
84	0.39	0.36	0.38	184
85	0.43	0.30	0.35	573
86	0.27	0.12	0.16	325
87	0.51	0.42	0.46	273
88	0.46	0.32	0.38	135
89	0.25	0.17	0.20	232
90	0.50	0.41	0.45	409
91	0.51	0.33	0.40	420
92	0.68	0.56	0.62	408
93	0.55	0.49	0.52	241
94	0.20	0.00	0.10	211

94	0.20	0.09	0.13	211
95	0.31	0.17	0.22	277
96	0.22	0.12	0.15	410
97	0.77	0.47	0.58	501
98	0.67	0.64	0.65	136
99	0.46	0.37	0.41	239
100	0.34	0.20	0.25	324
101	0.86	0.74	0.79	277
102	0.89	0.76	0.82	613
103	0.37	0.23	0.28	157
104	0.21	0.11	0.15	295
105	0.67	0.46	0.54	334
106	0.68	0.36	0.47	335
107	0.70	0.58	0.63	389
108	0.52	0.33	0.41	251
109	0.56	0.47	0.51	317
110	0.30	0.11	0.16	187
111	0.46	0.19	0.26	140
112	0.57	0.47	0.52	154
113	0.48	0.27	0.35	332
114	0.43	0.28	0.34	323
115	0.43	0.32	0.37	344
116	0.68	0.54	0.60	370
117	0.44	0.30	0.36	313
118	0.76	0.75	0.75	874
119	0.36	0.26	0.30	293
120	0.13	0.09	0.10	200
121	0.66	0.51	0.57	463
122	0.23	0.12	0.16	119
123	0.15	0.04	0.06	256
124	0.86	0.72	0.78	195
125	0.29	0.17	0.22	138
126	0.71	0.52	0.60	376
127	0.16	0.07	0.09	122
128	0.13	0.06	0.08	252
129	0.43	0.38	0.40	144
130	0.31	0.18	0.23	150
131	0.19	0.09	0.12	210
132	0.51	0.33	0.40	361
133	0.34	0.24	0.27	452

133	0.84	0.64	0.73	453
134	0.80	0.77	0.79	124
135	0.15	0.12	0.14	91
136	0.51	0.38	0.43	128
137	0.45	0.39	0.42	218
138	0.34	0.21	0.26	243
139	0.29	0.19	0.23	149
140	0.69	0.54	0.61	318
141	0.19	0.12	0.15	159
142	0.57	0.43	0.49	274
143	0.81	0.82	0.82	362
144	0.42	0.27	0.33	118
145	0.50	0.43	0.46	164
146	0.51	0.39	0.44	461
147	0.69	0.43	0.53	159
148	0.33	0.20	0.25	166
149	0.90	0.59	0.72	346
150	0.50	0.22	0.30	350
151	0.90	0.67	0.77	55
152	0.71	0.51	0.60	387
153	0.36	0.31	0.33	150
154	0.32	0.15	0.20	281
155	0.25	0.19	0.22	202
156	0.75	0.65	0.70	130
157	0.22	0.10	0.14	245
158	0.90	0.69	0.78	177
159	0.46	0.40	0.43	130
160	0.38	0.24	0.29	336
161	0.79	0.65	0.71	220
162	0.19	0.10	0.14	229
163	0.78	0.46	0.58	316
164	0.63	0.42	0.50	283
165	0.53	0.38	0.44	197
166	0.54	0.53	0.54	101
167	0.37	0.23	0.28	231
168	0.43	0.35	0.39	370
169	0.39	0.24	0.29	258
170	0.26	0.16	0.20	101
171	0.34	0.28	0.31	89
172	0.48	0.27	0.42	102

172	0.48	0.37	0.42	193
173	0.46	0.33	0.39	309
174	0.31	0.14	0.19	172
175	0.79	0.74	0.76	95
176	0.85	0.63	0.73	346
177	0.81	0.60	0.69	322
178	0.54	0.47	0.50	232
179	0.21	0.10	0.14	125
180	0.46	0.39	0.42	145
181	0.30	0.21	0.25	77
182	0.18	0.12	0.14	182
183	0.51	0.37	0.43	257
184	0.23	0.13	0.17	216
185	0.31	0.20	0.24	242
186	0.33	0.22	0.26	165
187	0.68	0.56	0.61	263
188	0.19	0.10	0.13	174
189	0.64	0.47	0.54	136
190	0.81	0.59	0.68	202
191	0.29	0.21	0.24	134
192	0.58	0.46	0.51	230
193	0.26	0.20	0.23	90
194	0.56	0.55	0.55	185
195	0.16	0.08	0.10	156
196	0.14	0.09	0.11	160
197	0.29	0.17	0.21	266
198	0.28	0.15	0.20	284
199	0.18	0.08	0.11	145
200	0.86	0.77	0.81	212
201	0.49	0.26	0.34	317
202	0.70	0.63	0.66	427
203	0.20	0.14	0.17	232
204	0.39	0.29	0.33	217
205	0.49	0.48	0.48	527
206	0.15	0.06	0.09	124
207	0.41	0.35	0.38	103
208	0.77	0.55	0.64	287
209	0.20	0.11	0.15	193
210	0.55	0.39	0.45	220
211	0.45	0.30	0.30	140

211	0.45	0.20	0.28	140
212	0.14	0.09	0.11	161
213	0.46	0.53	0.49	72
214	0.60	0.43	0.50	396
215	0.67	0.42	0.51	134
216	0.47	0.27	0.34	400
217	0.32	0.25	0.28	75
218	0.93	0.76	0.84	219
219	0.59	0.42	0.49	210
220	0.84	0.67	0.74	298
221	0.89	0.71	0.79	266
222	0.66	0.45	0.53	290
223	0.12	0.05	0.07	128
224	0.70	0.48	0.57	159
225	0.41	0.35	0.38	164
226	0.47	0.34	0.39	144
227	0.54	0.39	0.45	276
228	0.09	0.04	0.06	235
229	0.14	0.06	0.09	216
230	0.33	0.21	0.25	228
231	0.64	0.55	0.59	64
232	0.23	0.16	0.18	103
233	0.62	0.39	0.48	216
234	0.50	0.22	0.31	116
235	0.45	0.32	0.38	77
236	0.88	0.69	0.77	67
237	0.29	0.18	0.22	218
238	0.26	0.18	0.21	139
239	0.22	0.06	0.10	94
240	0.40	0.31	0.35	77
241	0.31	0.13	0.19	167
242	0.64	0.41	0.50	86
243	0.31	0.24	0.27	58
244	0.53	0.40	0.46	269
245	0.12	0.08	0.10	112
246	0.92	0.82	0.87	255
247	0.21	0.21	0.21	58
248	0.14	0.07	0.10	81
249	0.05	0.02	0.03	131
250	0.40	0.26	0.31	62

250	0.40	0.20	0.31	93
251	0.56	0.34	0.43	154
252	0.10	0.05	0.06	129
253	0.47	0.34	0.39	83
254	0.24	0.13	0.17	191
255	0.11	0.06	0.08	219
256	0.13	0.08	0.10	130
257	0.38	0.31	0.34	93
258	0.63	0.52	0.57	217
259	0.27	0.18	0.22	141
260	0.68	0.24	0.35	143
261	0.40	0.18	0.25	219
262	0.48	0.37	0.42	107
263	0.36	0.23	0.28	236
264	0.25	0.20	0.22	119
265	0.43	0.28	0.34	72
266	0.11	0.06	0.07	70
267	0.35	0.24	0.29	107
268	0.53	0.47	0.50	169
269	0.29	0.17	0.21	129
270	0.69	0.53	0.60	159
271	0.77	0.53	0.63	190
272	0.46	0.33	0.39	248
273	0.85	0.75	0.80	264
274	0.84	0.67	0.74	105
275	0.23	0.13	0.17	104
276	0.05	0.03	0.03	115
277	0.77	0.60	0.67	170
278	0.71	0.48	0.57	145
279	0.88	0.75	0.81	230
280	0.59	0.40	0.48	80
281	0.65	0.55	0.59	217
282	0.69	0.52	0.59	175
283	0.26	0.17	0.21	269
284	0.54	0.38	0.44	74
285	0.70	0.51	0.59	206
286	0.83	0.70	0.76	227
287	0.66	0.42	0.52	130
288	0.16	0.08	0.10	129
289	0.16	0.11	0.12	80

289	0.10	0.11	0.13	80
290	0.19	0.14	0.16	99
291	0.59	0.39	0.47	208
292	0.26	0.13	0.18	67
293	0.77	0.54	0.63	109
294	0.35	0.27	0.31	140
295	0.23	0.16	0.19	241
296	0.22	0.14	0.17	72
297	0.21	0.15	0.18	107
298	0.64	0.57	0.60	61
299	0.73	0.56	0.63	77
300	0.15	0.12	0.13	111
301	0.03	0.01	0.01	126
302	0.17	0.11	0.13	73
303	0.55	0.44	0.49	176
304	0.90	0.81	0.86	230
305	0.82	0.71	0.76	156
306	0.43	0.37	0.40	146
307	0.21	0.11	0.15	98
308	0.04	0.01	0.02	78
309	0.48	0.17	0.25	94
310	0.58	0.38	0.46	162
311	0.71	0.51	0.59	116
312	0.45	0.35	0.40	57
313	0.37	0.11	0.17	65
314	0.41	0.36	0.39	138
315	0.50	0.28	0.36	195
316	0.40	0.32	0.35	69
317	0.26	0.21	0.23	134
318	0.54	0.41	0.47	148
319	0.80	0.56	0.66	161
320	0.18	0.18	0.18	104
321	0.69	0.62	0.65	156
322	0.54	0.46	0.49	134
323	0.53	0.44	0.48	232
324	0.21	0.16	0.18	92
325	0.37	0.23	0.29	197
326	0.10	0.07	0.08	126
327	0.14	0.06	0.08	115
328	0.05	0.01	0.01	100

328	0.95	0.71	0.81	198
329	0.42	0.30	0.35	125
330	0.54	0.26	0.35	81
331	0.34	0.15	0.21	94
332	0.29	0.20	0.23	56
333	0.15	0.08	0.10	260
334	0.16	0.12	0.13	60
335	0.22	0.10	0.14	110
336	0.58	0.54	0.56	71
337	0.13	0.09	0.11	66
338	0.40	0.45	0.43	150
339	0.05	0.04	0.04	54
340	0.79	0.57	0.66	195
341	0.68	0.51	0.58	79
342	0.38	0.50	0.43	38
343	0.55	0.37	0.44	43
344	0.33	0.28	0.30	68
345	0.64	0.34	0.45	73
346	0.11	0.07	0.08	116
347	0.60	0.49	0.54	111
348	0.24	0.19	0.21	63
349	0.84	0.71	0.77	104
350	0.58	0.57	0.57	44
351	0.25	0.28	0.26	40
352	0.77	0.62	0.69	136
353	0.39	0.22	0.28	54
354	0.25	0.12	0.16	134
355	0.52	0.42	0.46	120
356	0.44	0.31	0.37	228
357	0.56	0.43	0.49	269
358	0.56	0.36	0.44	80
359	0.75	0.63	0.68	140
360	0.30	0.19	0.23	125
361	0.87	0.74	0.80	169
362	0.17	0.12	0.14	56
363	0.83	0.75	0.79	154
364	0.19	0.19	0.19	58
365	0.23	0.15	0.18	71
366	0.90	0.67	0.77	54
367	0.14	0.10	0.12	116

367	0.14	0.10	0.12	110
368	0.26	0.19	0.22	54
369	0.09	0.06	0.07	71
370	0.23	0.11	0.15	61
371	0.29	0.10	0.15	71
372	0.51	0.44	0.47	52
373	0.59	0.46	0.52	150
374	0.26	0.22	0.23	93
375	0.12	0.09	0.10	67
376	0.12	0.04	0.06	76
377	0.45	0.37	0.41	106
378	0.07	0.02	0.04	86
379	0.23	0.21	0.22	14
380	0.76	0.56	0.64	122
381	0.10	0.06	0.07	104
382	0.26	0.15	0.19	66
383	0.49	0.40	0.44	110
384	0.17	0.07	0.10	155
385	0.41	0.36	0.38	50
386	0.22	0.12	0.16	64
387	0.26	0.12	0.16	93
388	0.48	0.33	0.39	102
389	0.12	0.06	0.08	108
390	0.91	0.68	0.78	178
391	0.35	0.21	0.26	115
392	0.66	0.45	0.54	42
393	0.04	0.01	0.01	134
394	0.28	0.15	0.20	112
395	0.39	0.32	0.35	176
396	0.29	0.16	0.21	125
397	0.64	0.45	0.52	224
398	0.75	0.67	0.71	63
399	0.11	0.07	0.09	59
400	0.44	0.41	0.43	63
401	0.44	0.33	0.38	98
402	0.45	0.24	0.31	162
403	0.24	0.18	0.21	83
404	0.64	0.84	0.73	19
405	0.21	0.15	0.17	92
406	0.47	0.30	0.42	41

406	0.47	0.39	0.43	41
407	0.50	0.33	0.39	43
408	0.69	0.49	0.57	160
409	0.12	0.08	0.10	50
410	0.00	0.00	0.00	19
411	0.30	0.22	0.25	175
412	0.22	0.15	0.18	72
413	0.23	0.09	0.13	95
414	0.24	0.14	0.18	97
415	0.15	0.10	0.12	48
416	0.43	0.33	0.37	83
417	0.23	0.15	0.18	40
418	0.22	0.12	0.16	91
419	0.52	0.42	0.47	90
420	0.28	0.24	0.26	37
421	0.10	0.06	0.07	66
422	0.47	0.41	0.44	73
423	0.37	0.29	0.32	56
424	0.88	0.88	0.88	33
425	0.10	0.04	0.06	76
426	0.06	0.02	0.04	81
427	0.92	0.73	0.81	150
428	1.00	0.76	0.86	29
429	0.98	0.95	0.97	389
430	0.56	0.44	0.50	167
431	0.45	0.15	0.22	123
432	0.26	0.18	0.21	39
433	0.32	0.28	0.30	82
434	0.89	0.71	0.79	66
435	0.55	0.45	0.49	93
436	0.49	0.37	0.42	87
437	0.17	0.09	0.12	86
438	0.65	0.49	0.56	104
439	0.48	0.21	0.29	100
440	0.17	0.07	0.10	141
441	0.41	0.43	0.42	110
442	0.24	0.20	0.22	123
443	0.28	0.20	0.23	71
444	0.22	0.12	0.16	109
445	0.42	0.25	0.20	40

445	0.42	0.35	0.39	48
446	0.40	0.28	0.33	76
447	0.23	0.26	0.25	38
448	0.59	0.56	0.57	81
449	0.44	0.28	0.34	132
450	0.42	0.33	0.37	81
451	0.67	0.38	0.49	76
452	0.11	0.07	0.08	44
453	0.00	0.00	0.00	44
454	0.73	0.54	0.62	70
455	0.29	0.25	0.27	155
456	0.31	0.26	0.28	43
457	0.36	0.31	0.33	72
458	0.19	0.16	0.18	62
459	0.42	0.32	0.36	69
460	0.13	0.08	0.10	119
461	0.61	0.35	0.45	79
462	0.31	0.26	0.28	47
463	0.34	0.27	0.30	104
464	0.55	0.42	0.47	106
465	0.34	0.27	0.30	64
466	0.45	0.33	0.38	173
467	0.60	0.44	0.51	107
468	0.42	0.29	0.35	126
469	0.15	0.05	0.08	114
470	0.92	0.81	0.86	140
471	0.59	0.38	0.46	79
472	0.40	0.40	0.40	143
473	0.64	0.39	0.49	158
474	0.28	0.12	0.16	138
475	0.20	0.15	0.17	59
476	0.63	0.45	0.53	88
477	0.74	0.68	0.71	176
478	0.90	0.79	0.84	24
479	0.29	0.17	0.22	92
480	0.68	0.58	0.63	100
481	0.39	0.37	0.38	103
482	0.27	0.15	0.19	74
483	0.72	0.60	0.65	105
484	0.10	0.07	0.10	82

484	0.19	0.07	0.10	83
485	0.05	0.04	0.04	82
486	0.30	0.18	0.23	71
487	0.38	0.23	0.28	120
488	0.23	0.10	0.13	105
489	0.54	0.39	0.45	87
490	0.90	0.84	0.87	32
491	0.05	0.03	0.04	69
492	0.14	0.06	0.09	49
493	0.06	0.04	0.05	117
494	0.49	0.38	0.43	61
495	0.95	0.80	0.87	344
496	0.19	0.12	0.14	52
497	0.50	0.34	0.41	137
498	0.34	0.15	0.21	98
499	0.32	0.23	0.27	79
avg / total	0.55	0.41	0.47	173812

Time taken to run this cell : 3:39:20.613677

- Best alpha after hyperparameter tuning is 1 which gives the micro-F1-Score of 47.66

3 OneVsRest with SGDClassifier with loss-hinge

```
In [20]: # job= -1 gave error of unable to pipeline sparse data so taking default value of job
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=10**-5, penalty='l1'))
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 : 0.10998
Hamming loss 0.0059325
Micro-average quality numbers
Precision: 0.2889, Recall: 0.4835, F1-measure: 0.3617
Macro-average quality numbers
Precision: 0.2093, Recall: 0.4119, F1-measure: 0.2699

```

	precision	recall	f1-score	support
0	0.72	0.81	0.76	5519
1	0.45	0.44	0.45	8190
2	0.51	0.54	0.52	6529
3	0.49	0.60	0.54	3231
4	0.52	0.56	0.54	6430
5	0.46	0.49	0.47	2879
6	0.56	0.63	0.59	5086
7	0.60	0.66	0.63	4533
8	0.23	0.21	0.22	3000
9	0.55	0.66	0.60	2765
10	0.31	0.37	0.34	3051
11	0.44	0.55	0.49	3009

12	0.35	0.47	0.40	2630
13	0.35	0.47	0.40	1426
14	0.57	0.67	0.62	2548
15	0.38	0.37	0.37	2371
16	0.23	0.42	0.30	873
17	0.59	0.70	0.64	2151
18	0.30	0.40	0.34	2204
19	0.26	0.52	0.35	831
20	0.50	0.59	0.54	1860
21	0.18	0.22	0.20	2023
22	0.27	0.39	0.32	1513
23	0.50	0.69	0.58	1207
24	0.21	0.39	0.28	506
25	0.25	0.47	0.32	425
26	0.38	0.47	0.42	793
27	0.38	0.49	0.43	1291
28	0.42	0.52	0.47	1208
29	0.12	0.23	0.16	406
30	0.18	0.36	0.24	504
31	0.13	0.25	0.17	732
32	0.21	0.45	0.29	441
33	0.36	0.46	0.40	1645
34	0.28	0.37	0.32	1058
35	0.44	0.62	0.52	946
36	0.24	0.44	0.31	644
37	0.29	0.76	0.42	136
38	0.29	0.52	0.37	570
39	0.29	0.45	0.35	766
40	0.31	0.48	0.38	1132
41	0.09	0.29	0.14	174
42	0.38	0.57	0.45	210
43	0.30	0.56	0.39	433
44	0.31	0.57	0.40	626
45	0.27	0.46	0.34	852
46	0.29	0.56	0.38	534
47	0.15	0.29	0.20	350
48	0.32	0.57	0.41	496
49	0.56	0.69	0.62	785
50	0.13	0.23	0.16	475

51	0.10	0.28	0.14	305
52	0.06	0.12	0.08	251
53	0.32	0.46	0.38	914
54	0.20	0.29	0.24	728
55	0.07	0.15	0.10	258
56	0.22	0.38	0.28	821
57	0.13	0.25	0.17	541
58	0.30	0.45	0.36	748
59	0.60	0.74	0.67	724
60	0.18	0.28	0.22	660
61	0.17	0.34	0.23	235
62	0.54	0.77	0.63	718
63	0.44	0.75	0.56	468
64	0.15	0.43	0.22	191
65	0.11	0.26	0.16	429
66	0.11	0.20	0.14	415
67	0.27	0.60	0.38	274
68	0.39	0.62	0.48	510
69	0.30	0.55	0.38	466
70	0.12	0.22	0.15	305
71	0.12	0.28	0.17	247
72	0.35	0.59	0.44	401
73	0.27	0.83	0.41	86
74	0.21	0.56	0.31	120
75	0.38	0.74	0.50	129
76	0.07	0.11	0.09	473
77	0.10	0.35	0.15	143
78	0.38	0.63	0.47	347
79	0.21	0.32	0.26	479
80	0.16	0.49	0.24	279
81	0.21	0.37	0.27	461
82	0.10	0.21	0.13	298
83	0.32	0.61	0.42	396
84	0.16	0.49	0.24	184
85	0.20	0.40	0.27	573
86	0.11	0.18	0.13	325
87	0.19	0.51	0.28	273
88	0.10	0.28	0.14	135
89	0.12	0.28	0.17	232

90	0.30	0.52	0.38	409
91	0.23	0.45	0.31	420
92	0.39	0.62	0.48	408
93	0.21	0.54	0.30	241
94	0.07	0.17	0.10	211
95	0.13	0.29	0.18	277
96	0.12	0.27	0.17	410
97	0.46	0.59	0.52	501
98	0.20	0.68	0.32	136
99	0.21	0.49	0.29	239
100	0.12	0.22	0.16	324
101	0.53	0.77	0.63	277
102	0.68	0.78	0.73	613
103	0.11	0.32	0.16	157
104	0.08	0.17	0.11	295
105	0.33	0.57	0.42	334
106	0.24	0.44	0.31	335
107	0.32	0.60	0.42	389
108	0.17	0.39	0.24	251
109	0.26	0.49	0.34	317
110	0.07	0.19	0.10	187
111	0.07	0.24	0.11	140
112	0.20	0.58	0.30	154
113	0.18	0.43	0.25	332
114	0.17	0.39	0.24	323
115	0.19	0.38	0.25	344
116	0.36	0.59	0.45	370
117	0.18	0.40	0.24	313
118	0.59	0.80	0.68	874
119	0.14	0.37	0.20	293
120	0.05	0.17	0.08	200
121	0.38	0.60	0.46	463
122	0.11	0.32	0.16	119
123	0.03	0.05	0.03	256
124	0.44	0.74	0.55	195
125	0.10	0.35	0.16	138
126	0.41	0.63	0.49	376
127	0.04	0.11	0.06	122
128	0.08	0.15	0.10	252

129	0.21	0.37	0.26	144
130	0.09	0.33	0.15	150
131	0.05	0.12	0.07	210
132	0.22	0.37	0.27	361
133	0.52	0.68	0.59	453
134	0.36	0.81	0.50	124
135	0.03	0.12	0.05	91
136	0.13	0.44	0.20	128
137	0.21	0.48	0.29	218
138	0.15	0.33	0.21	243
139	0.13	0.34	0.19	149
140	0.37	0.57	0.44	318
141	0.07	0.21	0.11	159
142	0.32	0.53	0.40	274
143	0.58	0.86	0.69	362
144	0.09	0.38	0.14	118
145	0.18	0.45	0.26	164
146	0.24	0.43	0.31	461
147	0.19	0.58	0.28	159
148	0.10	0.29	0.14	166
149	0.42	0.63	0.51	346
150	0.14	0.27	0.19	350
151	0.22	0.75	0.33	55
152	0.42	0.60	0.49	387
153	0.18	0.31	0.23	150
154	0.11	0.21	0.15	281
155	0.08	0.24	0.12	202
156	0.27	0.73	0.39	130
157	0.14	0.20	0.16	245
158	0.48	0.72	0.58	177
159	0.11	0.42	0.17	130
160	0.16	0.33	0.22	336
161	0.41	0.71	0.52	220
162	0.09	0.21	0.12	229
163	0.33	0.53	0.41	316
164	0.28	0.54	0.36	283
165	0.20	0.43	0.27	197
166	0.20	0.55	0.29	101
167	0.14	0.29	0.19	231

168	0.19	0.42	0.26	370
169	0.20	0.38	0.26	258
170	0.05	0.25	0.09	101
171	0.10	0.36	0.16	89
172	0.19	0.43	0.26	193
173	0.25	0.46	0.32	309
174	0.08	0.24	0.12	172
175	0.35	0.81	0.48	95
176	0.57	0.73	0.64	346
177	0.42	0.65	0.51	322
178	0.27	0.55	0.36	232
179	0.06	0.19	0.09	125
180	0.20	0.48	0.28	145
181	0.04	0.23	0.07	77
182	0.09	0.23	0.13	182
183	0.26	0.53	0.35	257
184	0.06	0.22	0.10	216
185	0.14	0.30	0.19	242
186	0.12	0.25	0.16	165
187	0.36	0.66	0.47	263
188	0.08	0.20	0.11	174
189	0.38	0.58	0.46	136
190	0.40	0.69	0.51	202
191	0.09	0.24	0.14	134
192	0.25	0.48	0.33	230
193	0.07	0.24	0.11	90
194	0.24	0.61	0.35	185
195	0.03	0.11	0.05	156
196	0.06	0.17	0.09	160
197	0.12	0.26	0.17	266
198	0.18	0.30	0.23	284
199	0.07	0.20	0.10	145
200	0.55	0.82	0.66	212
201	0.20	0.33	0.25	317
202	0.50	0.63	0.56	427
203	0.13	0.25	0.17	232
204	0.18	0.43	0.25	217
205	0.37	0.60	0.46	527
206	0.03	0.10	0.04	124

207	0.22	0.35	0.27	103
208	0.38	0.58	0.46	287
209	0.07	0.17	0.10	193
210	0.23	0.46	0.30	220
211	0.11	0.41	0.17	140
212	0.06	0.23	0.10	161
213	0.20	0.56	0.30	72
214	0.40	0.56	0.47	396
215	0.21	0.44	0.29	134
216	0.20	0.30	0.24	400
217	0.09	0.36	0.14	75
218	0.59	0.82	0.69	219
219	0.25	0.51	0.34	210
220	0.51	0.74	0.60	298
221	0.49	0.74	0.59	266
222	0.40	0.53	0.45	290
223	0.03	0.11	0.05	128
224	0.17	0.54	0.26	159
225	0.13	0.44	0.20	164
226	0.22	0.43	0.29	144
227	0.29	0.56	0.38	276
228	0.03	0.11	0.05	235
229	0.08	0.18	0.11	216
230	0.09	0.26	0.13	228
231	0.17	0.69	0.28	64
232	0.05	0.24	0.09	103
233	0.18	0.50	0.26	216
234	0.16	0.36	0.22	116
235	0.17	0.58	0.26	77
236	0.35	0.72	0.47	67
237	0.14	0.27	0.18	218
238	0.08	0.22	0.12	139
239	0.04	0.12	0.05	94
240	0.14	0.36	0.21	77
241	0.05	0.16	0.08	167
242	0.22	0.52	0.31	86
243	0.05	0.28	0.09	58
244	0.28	0.42	0.34	269
245	0.08	0.20	0.12	112

246	0.66	0.84	0.74	255
247	0.07	0.41	0.12	58
248	0.02	0.11	0.04	81
249	0.04	0.13	0.06	131
250	0.11	0.35	0.17	93
251	0.17	0.42	0.24	154
252	0.03	0.09	0.04	129
253	0.11	0.41	0.18	83
254	0.11	0.19	0.14	191
255	0.07	0.17	0.10	219
256	0.04	0.12	0.06	130
257	0.16	0.43	0.23	93
258	0.32	0.59	0.41	217
259	0.09	0.28	0.14	141
260	0.20	0.36	0.26	143
261	0.12	0.24	0.16	219
262	0.12	0.39	0.19	107
263	0.24	0.39	0.30	236
264	0.12	0.35	0.18	119
265	0.09	0.36	0.14	72
266	0.05	0.16	0.07	70
267	0.14	0.36	0.20	107
268	0.28	0.55	0.37	169
269	0.18	0.40	0.24	129
270	0.31	0.62	0.41	159
271	0.23	0.59	0.33	190
272	0.19	0.40	0.26	248
273	0.60	0.80	0.68	264
274	0.38	0.78	0.52	105
275	0.09	0.24	0.14	104
276	0.02	0.06	0.03	115
277	0.37	0.67	0.48	170
278	0.29	0.59	0.39	145
279	0.54	0.79	0.64	230
280	0.17	0.51	0.26	80
281	0.42	0.74	0.53	217
282	0.38	0.59	0.47	175
283	0.15	0.29	0.20	269
284	0.19	0.47	0.28	74

285	0.33	0.61	0.43	206
286	0.50	0.71	0.59	227
287	0.20	0.54	0.29	130
288	0.07	0.12	0.09	129
289	0.04	0.23	0.07	80
290	0.07	0.25	0.11	99
291	0.26	0.48	0.33	208
292	0.04	0.21	0.07	67
293	0.22	0.63	0.33	109
294	0.15	0.42	0.22	140
295	0.14	0.31	0.19	241
296	0.08	0.26	0.13	72
297	0.08	0.25	0.12	107
298	0.28	0.54	0.36	61
299	0.34	0.61	0.44	77
300	0.06	0.20	0.09	111
301	0.01	0.02	0.01	126
302	0.07	0.21	0.10	73
303	0.25	0.58	0.35	176
304	0.65	0.80	0.71	230
305	0.48	0.77	0.59	156
306	0.29	0.48	0.36	146
307	0.08	0.28	0.12	98
308	0.01	0.03	0.01	78
309	0.07	0.19	0.10	94
310	0.25	0.52	0.34	162
311	0.32	0.62	0.42	116
312	0.11	0.42	0.18	57
313	0.03	0.14	0.05	65
314	0.17	0.43	0.24	138
315	0.23	0.37	0.28	195
316	0.12	0.41	0.19	69
317	0.11	0.35	0.17	134
318	0.22	0.40	0.28	148
319	0.38	0.55	0.45	161
320	0.09	0.37	0.15	104
321	0.38	0.62	0.47	156
322	0.21	0.44	0.28	134
323	0.30	0.49	0.37	232

324	0.08	0.23	0.11	92
325	0.17	0.38	0.23	197
326	0.06	0.22	0.10	126
327	0.02	0.08	0.04	115
328	0.56	0.74	0.64	198
329	0.17	0.43	0.24	125
330	0.13	0.35	0.18	81
331	0.08	0.27	0.13	94
332	0.10	0.36	0.16	56
333	0.08	0.16	0.11	260
334	0.08	0.23	0.12	60
335	0.09	0.25	0.13	110
336	0.19	0.58	0.29	71
337	0.04	0.17	0.06	66
338	0.16	0.47	0.24	150
339	0.01	0.06	0.02	54
340	0.51	0.67	0.58	195
341	0.31	0.58	0.40	79
342	0.08	0.50	0.14	38
343	0.13	0.51	0.20	43
344	0.17	0.35	0.23	68
345	0.24	0.47	0.31	73
346	0.04	0.10	0.06	116
347	0.19	0.55	0.28	111
348	0.04	0.19	0.06	63
349	0.33	0.75	0.45	104
350	0.20	0.55	0.29	44
351	0.13	0.38	0.19	40
352	0.37	0.63	0.47	136
353	0.08	0.24	0.12	54
354	0.08	0.22	0.12	134
355	0.18	0.43	0.26	120
356	0.30	0.56	0.39	228
357	0.33	0.52	0.40	269
358	0.18	0.49	0.26	80
359	0.34	0.69	0.45	140
360	0.13	0.32	0.19	125
361	0.51	0.77	0.61	169
362	0.05	0.20	0.08	56

363	0.51	0.76	0.61	154
364	0.06	0.21	0.09	58
365	0.07	0.28	0.11	71
366	0.46	0.70	0.56	54
367	0.05	0.20	0.08	116
368	0.03	0.07	0.04	54
369	0.03	0.20	0.06	71
370	0.02	0.11	0.04	61
371	0.04	0.17	0.07	71
372	0.23	0.63	0.34	52
373	0.29	0.56	0.39	150
374	0.12	0.46	0.20	93
375	0.04	0.13	0.06	67
376	0.02	0.04	0.02	76
377	0.18	0.38	0.25	106
378	0.03	0.10	0.04	86
379	0.02	0.36	0.04	14
380	0.24	0.59	0.34	122
381	0.04	0.14	0.06	104
382	0.06	0.24	0.09	66
383	0.18	0.41	0.25	110
384	0.03	0.10	0.05	155
385	0.08	0.44	0.13	50
386	0.08	0.25	0.12	64
387	0.08	0.20	0.11	93
388	0.19	0.46	0.27	102
389	0.04	0.13	0.06	108
390	0.57	0.74	0.64	178
391	0.12	0.27	0.17	115
392	0.20	0.64	0.30	42
393	0.03	0.07	0.04	134
394	0.06	0.21	0.09	112
395	0.15	0.37	0.21	176
396	0.10	0.26	0.15	125
397	0.44	0.61	0.51	224
398	0.35	0.70	0.46	63
399	0.02	0.08	0.03	59
400	0.12	0.41	0.19	63
401	0.11	0.44	0.17	98

402	0.13	0.36	0.20	162
403	0.09	0.28	0.13	83
404	0.36	0.89	0.52	19
405	0.11	0.34	0.17	92
406	0.08	0.51	0.14	41
407	0.20	0.37	0.26	43
408	0.24	0.51	0.33	160
409	0.09	0.26	0.14	50
410	0.01	0.11	0.02	19
411	0.15	0.24	0.19	175
412	0.06	0.22	0.09	72
413	0.06	0.16	0.08	95
414	0.09	0.24	0.13	97
415	0.05	0.23	0.09	48
416	0.19	0.43	0.26	83
417	0.07	0.23	0.11	40
418	0.10	0.23	0.14	91
419	0.18	0.48	0.27	90
420	0.11	0.35	0.17	37
421	0.07	0.29	0.12	66
422	0.13	0.48	0.20	73
423	0.09	0.32	0.14	56
424	0.49	0.94	0.65	33
425	0.04	0.13	0.06	76
426	0.02	0.09	0.04	81
427	0.54	0.81	0.65	150
428	0.38	0.79	0.51	29
429	0.87	0.85	0.86	389
430	0.25	0.50	0.34	167
431	0.08	0.16	0.11	123
432	0.10	0.41	0.16	39
433	0.16	0.43	0.23	82
434	0.38	0.74	0.50	66
435	0.19	0.48	0.28	93
436	0.22	0.49	0.30	87
437	0.06	0.17	0.09	86
438	0.38	0.58	0.46	104
439	0.11	0.30	0.16	100
440	0.06	0.16	0.08	141

441	0.20	0.52	0.28	110
442	0.08	0.21	0.12	123
443	0.12	0.32	0.17	71
444	0.09	0.23	0.13	109
445	0.09	0.35	0.15	48
446	0.13	0.43	0.20	76
447	0.05	0.37	0.09	38
448	0.29	0.72	0.41	81
449	0.23	0.41	0.30	132
450	0.14	0.41	0.21	81
451	0.17	0.42	0.25	76
452	0.04	0.11	0.06	44
453	0.02	0.07	0.02	44
454	0.23	0.64	0.34	70
455	0.15	0.39	0.22	155
456	0.09	0.26	0.13	43
457	0.16	0.44	0.24	72
458	0.03	0.13	0.05	62
459	0.10	0.38	0.15	69
460	0.03	0.06	0.04	119
461	0.33	0.48	0.39	79
462	0.08	0.30	0.13	47
463	0.10	0.37	0.16	104
464	0.25	0.49	0.33	106
465	0.09	0.34	0.14	64
466	0.26	0.46	0.33	173
467	0.28	0.50	0.36	107
468	0.15	0.37	0.22	126
469	0.06	0.13	0.08	114
470	0.56	0.81	0.66	140
471	0.22	0.48	0.30	79
472	0.21	0.38	0.27	143
473	0.37	0.63	0.47	158
474	0.11	0.19	0.14	138
475	0.03	0.15	0.05	59
476	0.23	0.50	0.31	88
477	0.39	0.67	0.49	176
478	0.44	0.92	0.59	24
479	0.07	0.20	0.10	92

480	0.39	0.65	0.49	100
481	0.21	0.48	0.30	103
482	0.09	0.32	0.15	74
483	0.37	0.70	0.48	105
484	0.07	0.20	0.10	83
485	0.01	0.06	0.02	82
486	0.10	0.37	0.15	71
487	0.13	0.29	0.18	120
488	0.04	0.11	0.06	105
489	0.23	0.52	0.31	87
490	0.34	0.84	0.48	32
491	0.03	0.12	0.05	69
492	0.02	0.06	0.02	49
493	0.05	0.11	0.07	117
494	0.09	0.30	0.14	61
495	0.79	0.82	0.81	344
496	0.08	0.15	0.10	52
497	0.21	0.36	0.27	137
498	0.10	0.24	0.15	98
499	0.08	0.32	0.13	79
avg / total	0.35	0.48	0.40	173812

Time taken to run this cell : 0:19:33.461137

- SGDClassifier with hinge loss performed better than SGDClassifier with log loss

Summary

- Micro-F1-Score on SGDClassifier with log loss is 35.66
- Micro-F1-Score on Logistic Regression is 47.67
- Best Hyperparameter alpha after gridsearch is 1 which gave Micro-F1-Score 47.66
- Micro-F1-Score on SGDClassifier with hinge loss is 36.17

Model	Hyperparameter	Micro-F1-Score
-------	----------------	----------------

Model	Hyperparameter	Micro-F1-Score
OneVrest SGDClassifier(log-loss)	$\alpha = 10^{-5}$	35.66
OneVrest Logistic Regression	$C = 1$	47.67
OneVrest SGDClassifier(hinge-loss)	$\alpha = 10^{-5}$	36.17