

▼ Stack Overflow: Tag Prediction

1. Business Problem

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 Statement

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

▼ LOAD REQUIRED LIBRARIES

```
1 import warnings
2 warnings.filterwarnings("ignore")
3 import pandas as pd
4 import sqlite3
5 import csv
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8 import numpy as np
9 from wordcloud import WordCloud
10 import re
11 import os
12 from sqlalchemy import create_engine # database connection
13 import datetime as dt
14 from nltk.corpus import stopwords
15 from nltk.tokenize import word_tokenize
16 from nltk.stem.snowball import SnowballStemmer
17 from sklearn.feature_extraction.text import CountVectorizer
18 from sklearn.feature_extraction.text import TfidfVectorizer
19 from sklearn.multiclass import OneVsRestClassifier
20 from sklearn.linear_model import SGDClassifier
21 from sklearn import metrics
22 from sklearn.metrics import f1_score, precision_score, recall_score
23 from sklearn import svm
24 from sklearn.linear_model import LogisticRegression
25 from sklearn.naive_bayes import GaussianNB
26 from datetime import datetime
```

▼ CONNECT TO GOOGLE COLAB

```
1 from google.colab import drive
2
3 drive.mount('/content/drive')
```

↳ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", for

▼ LOAD THE DATA FROM CSV FILE TO TRAIN.DB

```
1 if not os.path.isfile('train.db'):
2     start = datetime.now()
3     disk_engine = create_engine('sqlite:///train.db')
4     start=dt.datetime.now()
5     chunksize=180000
6     j=0
7     index_start=1
8     for df in pd.read_csv('/content/drive/My Drive/Colab Notebooks/StackOverflow Tag Predictor/Train.csv',names=['Ic
9         df.index +=index_start
10        j+=1
11        df.to_sql('data', disk_engine, if_exists='append')
12        index_start = df.index[-1] + 1
13    print("Time taken to run this cell :", datetime.now() - start)
```

↳ Time taken to run this cell : 0:04:30.307245

▼ COUNT THE TOTAL NO OF ROWS

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

↳ Number of rows in the database :
6034196
Time taken to count the number of rows : 0:00:14.149502

▼ CHECK FOR DUPLICATES

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

↳ Time taken to run this cell : 0:05:11.238108

```
1 df_no_duplicate.head()
```

↳

	Title	Body	Tags	cnt_dup
0	Implementing Boundary Value Analysis of S...	<pre>\n#include<iostream>\n#include<...\n</pre>	c++ c	1

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

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

Dynamic Datagrid Binding in Silverlight?	dynamicall	columns	
--	------------	---------	--

```
1 df_no_duplicate.cnt_dup.value_counts()
```

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

```
1 df_no_duplicate["Tags"].head()
```

```
0 c++ c
1 c# silverlight data-binding
2 c# silverlight data-binding columns
3 jsp jstl
4 java jdbc
Name: Tags, dtype: object
```

```
1 start = datetime.now()
2 df_no_duplicate["tag_count"] = df_no_duplicate["Tags"].apply(lambda text: len(str(text).split()))
3 print("Time taken to run this cell :", datetime.now() - start)
4 df_no_duplicate.head()
```

```
Time taken to run this cell : 0:00:03.016791
```

	Title	Body	Tags	cnt_dup	tag_count
0	Implementing Boundary Value Analysis of S...	<pre>\n#include<iostream>\n#include<...\n</pre>	c++ c	1	2
1	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding	1	3
2	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding columns	1	4

```
1 df_no_duplicate.tag_count.value_counts()
```

```
3 1206157
2 1111706
4 814996
1 568298
5 505158
Name: tag_count, dtype: int64
```

```
1 if not os.path.isfile('train_no_dup.db'):
2     disk_dup = create_engine("sqlite:///train_no_dup.db")
3     no_dup = pd.DataFrame(df_no_duplicate, columns=['Title', 'Body', 'Tags'])
4     no_dup.to_sql('no_dup_train',disk_dup)
```

```
1 if os.path.isfile('train_no_dup.db'):
2     start = datetime.now()
3     con = sqlite3.connect('train_no_dup.db')
4     tag_data = pd.read_sql_query("""SELECT Tags FROM no_dup_train""", con)
```

```

tag_data = pd.read_sql_query('SELECT tags FROM tag_data', con)
5 #Always remember to close the database
6 con.close()
7
8 # Let's now drop unwanted column.
9 tag_data.drop(tag_data.index[0], inplace=True)
10 #Printing first 5 columns from our data frame
11 tag_data.head()
12 print("Time taken to run this cell :", datetime.now() - start)
13 else:
14     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 : 0:01:54.541361

```
1 tag_data.head()
```

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

```
1 tag_data.loc[tag_data['Tags'].isnull(), 'Tags'] = ''
```

▼ ANALYSIS OF TAGS --> COUNTVECTORIZER

```

1 vectorizer=CountVectorizer(tokenizer = lambda x : str(x).split())
2
3 tag_dtm=vectorizer.fit_transform(tag_data['Tags'])

```

```

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

```

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

```

1 feature_tags=vectorizer.get_feature_names()
2
3 print("Some of the tags we have :", feature_tags[:10])

```

Some of the tags we have : ['.a', '.app', '.asp.net-mvc', '.aspxauth', '.bash-profile', '.class-file', '.cs-fil

```

1 freqs = tag_dtm.sum(axis=0).A1
2 result = dict(zip(feature_tags, freqs))

```

```

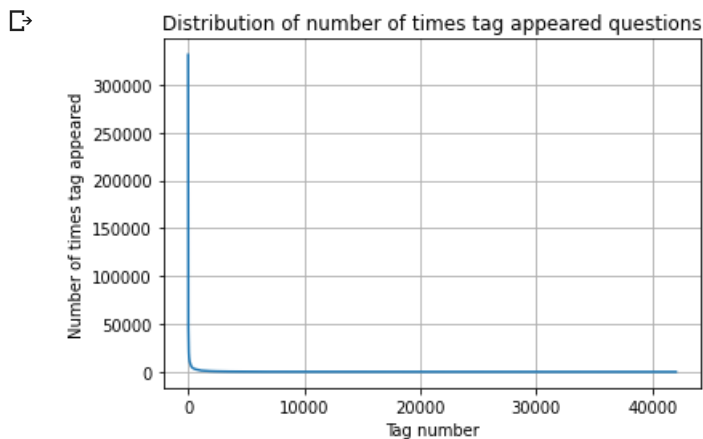
1 if not os.path.isfile('tag_counts_dict_dtm.csv'):
2     with open('tag_counts_dict_dtm.csv', 'w') as csv_file:
3         writer = csv.writer(csv_file)
4         for key, value in result.items():
5             writer.writerow([key, value])
6 tag_df = pd.read_csv("tag_counts_dict_dtm.csv", names=['Tags', 'Counts'])
7 tag_df.head()

```

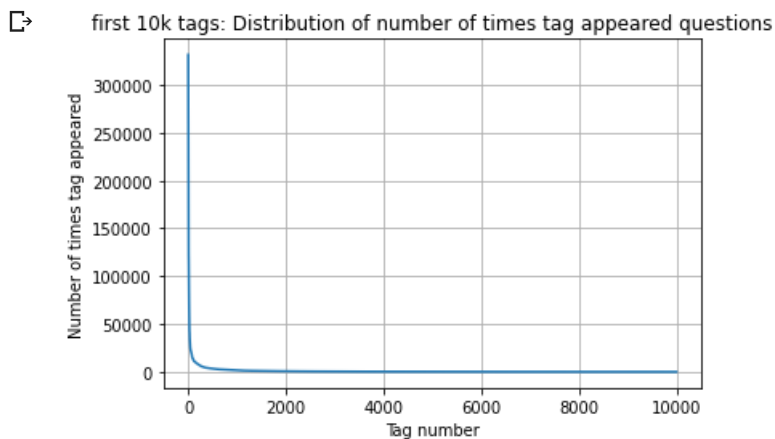
	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21

```
1 tag_df_sorted=tag_df.sort_values(['Counts'],ascending=False)
2
3 tag_counts=tag_df_sorted['Counts'].values
```

```
1 plt.plot(tag_counts)
2 plt.title("Distribution of number of times tag appeared questions")
3 plt.grid()
4 plt.xlabel("Tag number")
5 plt.ylabel("Number of times tag appeared")
6 plt.show()
```



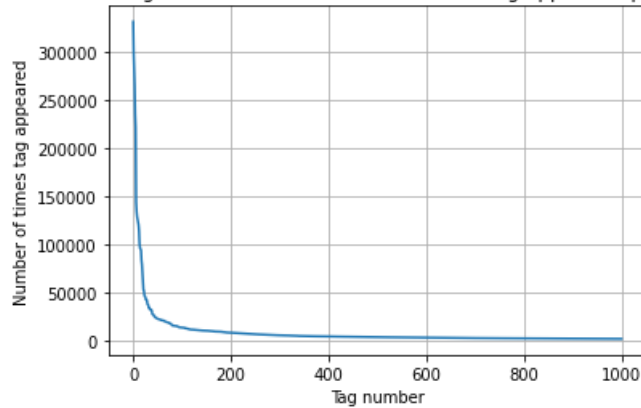
```
1 plt.plot(tag_counts[0:10000])
2 plt.title('first 10k tags: Distribution of number of times tag appeared questions')
3 plt.grid()
4 plt.xlabel("Tag number")
5 plt.ylabel("Number of times tag appeared")
6 plt.show()
```



```
1 plt.plot(tag_counts[0:1000])
2 plt.title('first 10k tags: Distribution of number of times tag appeared questions')
3 plt.grid()
4 plt.xlabel("Tag number")
```

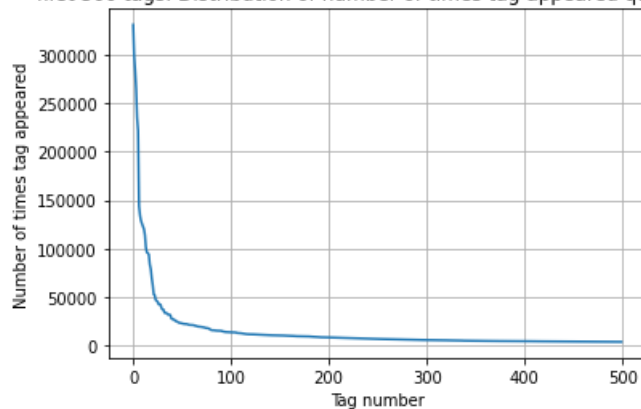
```
5 plt.ylabel("Number of times tag appeared")
6 plt.show()
```

first 10k tags: Distribution of number of times tag appeared questions



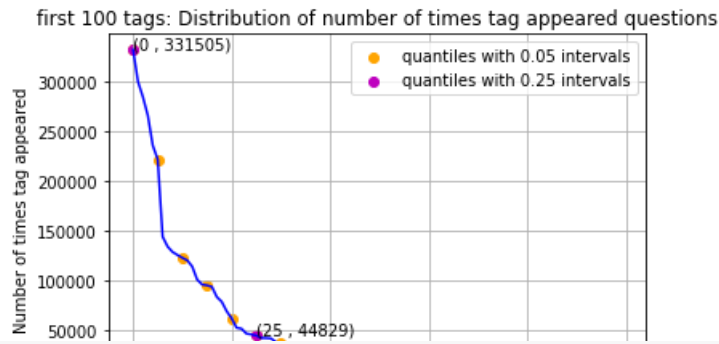
```
1 plt.plot(tag_counts[0:500])
2 plt.title('first 500 tags: Distribution of number of times tag appeared questions')
3 plt.grid()
4 plt.xlabel("Tag number")
5 plt.ylabel("Number of times tag appeared")
6 plt.show()
```

first 500 tags: Distribution of number of times tag appeared questions



```
1 plt.plot(tag_counts[0:100], c='b')
2 plt.scatter(x=list(range(0,100,5)), y=tag_counts[0:100:5], c='orange', label="quantiles with 0.05 intervals")
3 # quantiles with 0.25 difference
4 plt.scatter(x=list(range(0,100,25)), y=tag_counts[0:100:25], c='m', label = "quantiles with 0.25 intervals")
5
6 for x,y in zip(list(range(0,100,25)), tag_counts[0:100:25]):
7     plt.annotate(s="({} , {})".format(x,y), xy=(x,y), xytext=(x-0.05, y+500))
8
9 plt.title('first 100 tags: Distribution of number of times tag appeared questions')
10 plt.grid()
11 plt.xlabel("Tag number")
12 plt.ylabel("Number of times tag appeared")
13 plt.legend()
14 plt.show()
15 print(len(tag_counts[0:100:5]), tag_counts[0:100:5])
```

first 100 tags: Distribution of number of times tag appeared questions



```
1 # Store tags greater than 10K in one list
2 lst_tags_gt_10k = tag_df[tag_df.Counts>10000].Tags
3 #Print the length of the list
4 print ('{} Tags are used more than 10000 times'.format(len(lst_tags_gt_10k)))
5 # Store tags greater than 100K in one list
6 lst_tags_gt_100k = tag_df[tag_df.Counts>100000].Tags
7 #Print the length of the list.
8 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:

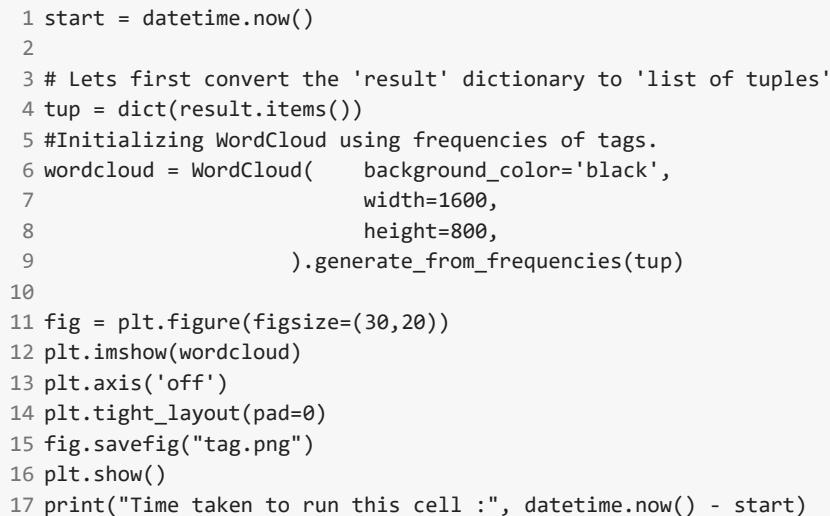
1. There are total 153 tags which are used more than 10000 times.
2. 14 tags are used more than 100000 times.
3. Most frequent tag (i.e. c#) is used 331505 times.
4. Since some tags occur much more frequently than others
5. Micro-averaged. F1-score is the appropriate metric for this problem.

▼ TAGS PER QUESTION

```
1 tag_quest_count = tag_dtm.sum(axis=1).tolist()
2
3 tag_quest_count=[int(j) for i in tag_quest_count for j in i]
4 print ('We have total {} datapoints.'.format(len(tag_quest_count)))
5
6 print(tag_quest_count[:5])
```

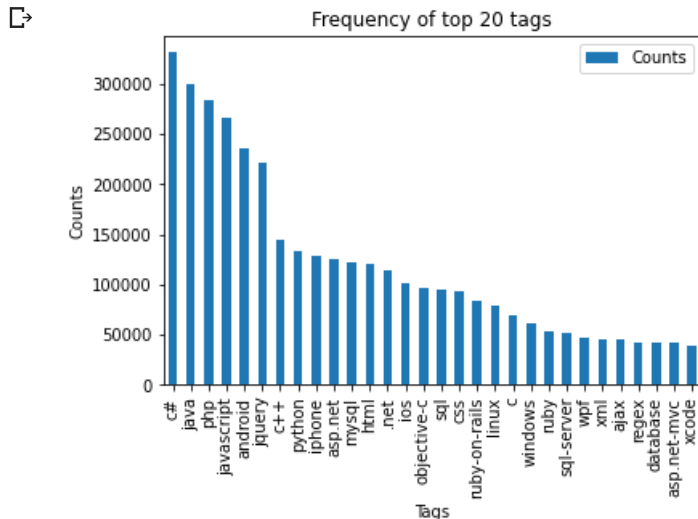
```
We have total 4206314 datapoints.
[3, 4, 2, 2, 3]
```

```
1 sns.countplot(tag_quest_count, palette='gist_rainbow')
2 plt.title("Number of tags in the questions ")
3 plt.xlabel("Number of Tags")
4 plt.ylabel("Number of questions")
5 plt.show()
```



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

```
1 i=np.arange(30)
2 tag_df_sorted.head(30).plot(kind='bar')
3 plt.title('Frequency of top 20 tags')
4 plt.xticks(i, tag_df_sorted['Tags'])
5 plt.xlabel('Tags')
6 plt.ylabel('Counts')
7 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.

▼ CLEANING & PREPROCESSING OF QUESTIONS

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

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

```
1 def striphtml(data):
2     cleanr = re.compile('<.*?>')
3     cleantext = re.sub(cleanr, ' ', str(data))
4     return cleantext
5 stop_words = set(stopwords.words('english'))
6 stemmer = SnowballStemmer("english")
```

```
1 def create_connection(db_file):
2     """ create a database connection to the SQLite database
3         specified by db_file
4     :param db_file: database file
5     :return: Connection object or None
6     """
7     try:
8         conn = sqlite3.connect(db_file)
```

```

9         return conn
10    except Error as e:
11        print(e)
12
13    return None
14
15 def create_table(conn, create_table_sql):
16     """ create a table from the create_table_sql statement
17     :param conn: Connection object
18     :param create_table_sql: a CREATE TABLE statement
19     :return:
20     """
21     try:
22         c = conn.cursor()
23         c.execute(create_table_sql)
24     except Error as e:
25         print(e)
26
27 def checkTableExists(dbcon):
28     cursr = dbcon.cursor()
29     str = "select name from sqlite_master where type='table'"
30     table_names = cursr.execute(str)
31     print("Tables in the databse:")
32     tables =table_names.fetchall()
33     print(tables[0][0])
34     return(len(tables))
35
36 def create_database_table(database, query):
37     conn = create_connection(database)
38     if conn is not None:
39         create_table(conn, query)
40         checkTableExists(conn)
41     else:
42         print("Error! cannot create the database connection.")
43     conn.close()
44
45 sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (question text NOT NULL, code text, tags text,
46 create_database_table("Processed.db", sql_create_table)

```

➤ Tables in the databse:
QuestionsProcessed

```

1 start = datetime.now()
2 read_db = 'train_no_dup.db'
3 write_db = 'Processed.db'
4 if os.path.isfile(read_db):
5     conn_r = create_connection(read_db)
6     if conn_r is not None:
7         reader =conn_r.cursor()
8         reader.execute("SELECT Title, Body, Tags From no_dup_train ORDER BY RANDOM() LIMIT 1000000;")
9
10 if os.path.isfile(write_db):
11     conn_w = create_connection(write_db)
12     if conn_w is not None:
13         tables = checkTableExists(conn_w)
14         writer =conn_w.cursor()
15         if tables != 0:
16             writer.execute("DELETE FROM QuestionsProcessed WHERE 1")
17             print("Cleared All the rows")
18 print("Time taken to run this cell :", datetime.now() - start)

```

➤

Tables in the database:

QuestionsProcessed

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
1 start = datetime.now()
2 preprocessed_data_list=[]
3 reader.fetchone()
4 questions_with_code=0
5 len_pre=0
6 len_post=0
7 questions_proccesed = 0
8 for row in reader:
9
10     is_code = 0
11
12     title, question, tags = row[0], row[1], row[2]
13
14     if '<code>' in question:
15         questions_with_code+=1
16         is_code = 1
17     x = len(question)+len(title)
18     len_pre+=x
19
20     code = str(re.findall(r'<code>(.*?)</code>', question, flags=re.DOTALL))
21
22     question=re.sub('<code>(.*?)</code>', '', question, flags=re.MULTILINE|re.DOTALL)
23     question=stripthtml(question.encode('utf-8'))
24
25     title=title.encode('utf-8')
26
27     question=str(title)+" "+str(question)
28     question=re.sub(r'^[A-Za-z]+'+', ' ',question)
29     words=word_tokenize(str(question.lower()))
30
31     #Removing all single letter and and stopwords from question exceptt for the letter 'c'
32     question=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words and (len(j)!=1 or j=='c'))
33
34     len_post+=len(question)
35     tup = (question,code,tags,x,len(question),is_code)
36     questions_proccesed += 1
37     writer.execute("insert into QuestionsProcessed(question,code,tags,words_pre,words_post,is_code) values (?,?=?,?)")
38     if (questions_proccesed%100000==0):
39         print("number of questions completed=",questions_proccesed)
40
41 no_dup_avg_len_pre=(len_pre*1.0)/questions_proccesed
42 no_dup_avg_len_post=(len_post*1.0)/questions_proccesed
43
44 print( "Avg. length of questions(Title+Body) before processing: %d"%no_dup_avg_len_pre)
45 print( "Avg. length of questions(Title+Body) after processing: %d"%no_dup_avg_len_post)
46 print( "Percent of questions containing code: %d"%((questions_with_code*100.0)/questions_proccesed))
47
48 print("Time taken to run this cell :", datetime.now() - start)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```

number of questions completed= 100000
number of questions completed= 200000
number of questions completed= 300000
number of questions completed= 400000
number of questions completed= 500000
number of questions completed= 600000
number of questions completed= 700000

```

```

1 conn_r.commit()
2 conn_w.commit()
3 conn_r.close()
4 conn_w.close()

```

Time taken to run this cell : 0.25.27.309922

```

1 if os.path.isfile(write_db):
2     conn_r = create_connection(write_db)
3     if conn_r is not None:
4         reader = conn_r.cursor()
5         reader.execute("SELECT question From QuestionsProcessed LIMIT 10")
6         print("Questions after preprocessed")
7         print('='*100)
8         reader.fetchone()
9         for row in reader:
10             print(row)
11             print('-'*100)
12 conn_r.commit()
13 conn_r.close()

```

↳ Questions after preprocessed

```

=====
('right wrong practic use statement foo foo true throughout short career far program mere student work internsh
-----
('java anim import imag netbean hi want ask move import jpg imag left right bottom beginn java pleas help thank
-----
('help sql inner join tri inner join temp tabl ni know done done complet forgot npleas advis queri tri execut f
-----
('possibl pitfal use extens method base shorthand regular want access properti possibl null object use often sn
-----
('chrome extens run background file function everi sec extens need synch data server everi second also backgrou
-----
('iphon xcode mutablecopi still immut look place regard problem tri creat nsuser default object add mutabl arra
-----
('could due servic endpoint bind use http protocol wcf servic run fine local machin put server receiv follow er
-----
('gotoandstop bug reason one instanc get gotoandstop go stop second frame movi clip two frame load ad movieclip
-----
('traffic shape tp ipsec vpn via account connect need abl control amount bandwidth specif user account use vpn
=====

```

```

1 start = datetime.now()
2 con=sqlite3.connect('/content/drive/My Drive/Colab Notebooks/StackOverflow Tag Predictor/Processed.db')
3 preprocessed_data = pd.read_sql_query("""SELECT question, Tags FROM QuestionsProcessed""",con)
4 con.close()
5 print("Time taken to count the number of rows :", datetime.now() - start)

```

↳ Time taken to count the number of rows : 0:00:21.968940

```
1 preprocessed_data.head()
```

↳

	question	tags
0	macport gcc select error tri exec appl darwin ...	macports selection gcc
1	right wrong practic use statement foo foo true	validation if-statement logic

▼ MACHINE LEARNING MODELS

▼ Converting Tags to MultiLabel Problems

```
1 preprocessed_data[preprocessed_data.isnull().any(axis=1)]
```

```
↳
```

	question	tags
642062	handl nullobject done quit bit research best w...	None

```
1 preprocessed_data.loc[preprocessed_data['tags'].isnull(), 'tags'] = ''
```

```
1 # binary = true , means it will return binary vector
2 count_vectorizer=CountVectorizer(tokenizer = lambda x : str(x).split() , binary='true')
3
4 multilabel_y=count_vectorizer.fit_transform(preprocessed_data['tags'])
```

```
1 def tags_to_choose(n):
2     t = multilabel_y.sum(axis=0).tolist()[0]
3     sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)
4     multilabel_yn=multilabel_y[:,sorted_tags_i[:n]]
5     return multilabel_yn
6
7 def questions_explained_fn(n):
8     multilabel_yn = tags_to_choose(n)
9     x= multilabel_yn.sum(axis=1)
10    return (np.count_nonzero(x==0))
```

```
1 multilabel_y.shape[1]
```

```
↳ 35388
```

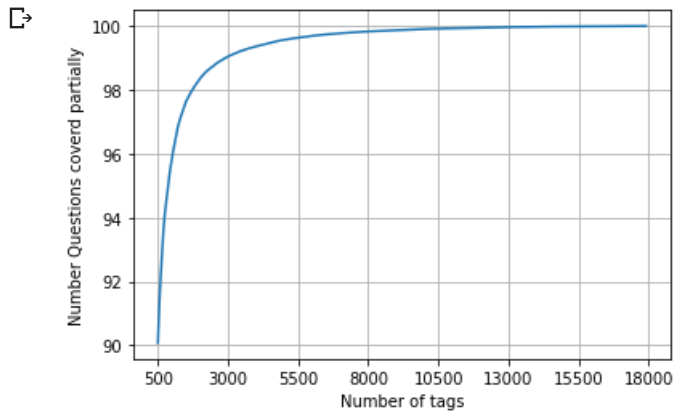
```
1 preprocessed_data.shape[0]
```

```
↳ 999997
```

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

```
1 fig, ax = plt.subplots()
2 ax.plot(questions_explained)
3 xlabel = list(500+np.array(range(-50,450,50))*50)
4 ax.set_xticklabels(xlabel)
5 plt.xlabel("Number of tags")
6 plt.ylabel("Number Questions covered partially")
7 plt.grid()
8 plt.show()
9 # you can choose any number of tags based on your computing power, minimun is 50(it covers 90% of the tags)
```

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



with 5500 tags we are covering 99.048 % of questions

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

number of questions that are not covered : 9517 out of 999997

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

Number of tags in sample : 35388
number of tags taken : 5500 (15.541991635582683 %)

▼ Split the Data into Test & Train

```
1 total_size=preprocessed_data.shape[0]
2 train_size=int(0.80*total_size)
3
4 x_train=preprocessed_data.head(train_size)
5 x_test=preprocessed_data.tail(total_size - train_size)
6
7 y_train = multilabel_yx[0:train_size,:]
8 y_test = multilabel_yx[train_size:total_size,:]
```

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

Number of data points in train data : (799997, 5500)
Number of data points in test data : (200000, 5500)

▼ Featurizing of data

▼ TF-IDF Featurization Vector

```
1 start = datetime.now()
2 vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True, norm="l2", \
3                               tokenizer = lambda x: x.split(), sublinear_tf=False, ngram_range=(1,1))
```

```
1 x_train_multilabel = vectorizer.fit_transform(x_train['question'])
```

```
1 x_test_multilabel = vectorizer.transform(x_test['question'])
2 print("Time taken to run this cell :", datetime.now() - start)
```

Time taken to run this cell : 0:00:37.624874

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

Dimensions of train data X: (799997, 9366) Y : (799997, 5500)
Dimensions of test data X: (200000, 9366) Y: (200000, 5500)

```
1 classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty='l1', n_jobs=-1)
2 classifier.fit(x_train_multilabel, y_train)
3 predictions = classifier.predict(x_test_multilabel)
4
5 print("accuracy :",metrics.accuracy_score(y_test,predictions))
6 print("macro f1 score :",metrics.f1_score(y_test, predictions, average = 'macro'))
7 print("micro f1 scoore :",metrics.f1_score(y_test, predictions, average = 'micro'))
8 print("hamming loss :",metrics.hamming_loss(y_test,predictions))
9 print("Precision recall report :\n",metrics.classification_report(y_test, predictions))
```

```
1 start = datetime.now()
2 con=sqlite3.connect('/content/drive/My Drive/Colab Notebooks/StackOverflow Tag Predictor/Titlemoreweight.db')
3 preprocessed_data = pd.read_sql_query("""SELECT question, Tags FROM QuestionsProcessed LIMIT 200000""",con)
4 con.close()
5 print("Time taken to count the number of rows :", datetime.now() - start)
```

Time taken to count the number of rows : 0:04:14.738980

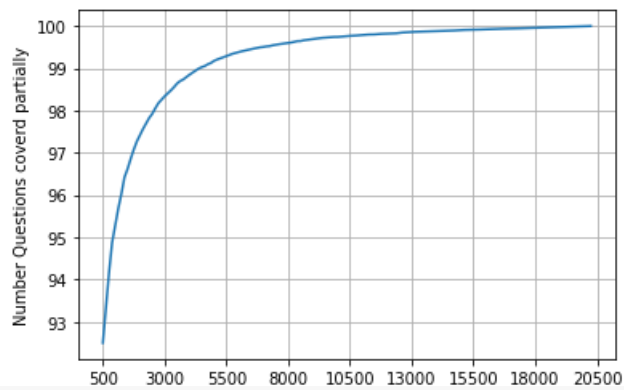
```
1 print("number of data points in sample :", preprocessed_data.shape[0])
2 print("number of dimensions :", preprocessed_data.shape[1])
```

number of data points in sample : 200000
number of dimensions : 2

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

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

```
1 fig, ax = plt.subplots()
2 ax.plot(questions_explained)
3 xlabel = list(500+np.array(range(-50,450,50))*50)
4 ax.set_xticklabels(xlabel)
5 plt.xlabel("Number of tags")
6 plt.ylabel("Number Questions covered partially")
7 plt.grid()
8 plt.show()
9 # you can choose any number of tags based on your computing power, minimum is 500(it covers 90% of the tags)
10 print("with ",5500,"tags we are covering ",questions_explained[50],"% of questions")
11 print("with ",500,"tags we are covering ",questions_explained[0],"% of questions")
```



```
1 multilabel_yx = tags_to_choose(500)
2 print("number of questions that are not covered :", questions_explained_fn(500),"out of ", total_qs)
```

↳ number of questions that are not covered : 15044 out of 200000

```
1 train_datasize = 130000
2 x_train=preprocessed_data.head(train_datasize)
3 x_test=preprocessed_data.tail(preprocessed_data.shape[0] - 130000)
4
5 y_train = multilabel_yx[0:train_datasize,:]
6 y_test = multilabel_yx[train_datasize:preprocessed_data.shape[0],:]
```

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

↳ Number of data points in train data : (130000, 500)
Number of data points in test data : (70000, 500)

```
1 start = datetime.now()
2 vectorizer = CountVectorizer(min_df=0.00009, max_features=200000,
3                               tokenizer = lambda x: str(x).split(),ngram_range=(1,4))
4 x_train_multilabel_bow = vectorizer.fit_transform(x_train['question'])
5 x_test_multilabel_bow = vectorizer.transform(x_test['question'])
6 print("Time taken to run this cell :", datetime.now() - start)
```

↳ Time taken to run this cell : 0:02:30.165520

```
1 print("Dimensions of train data X:",x_train_multilabel_bow.shape, "Y :",y_train.shape)
2 print("Dimensions of test data X:",x_test_multilabel_bow.shape,"Y:",y_test.shape)
```

↳ Dimensions of train data X: (130000, 100181) Y : (130000, 500)
Dimensions of test data X: (70000, 100181) Y: (70000, 500)

```
1 start = datetime.now()
2 classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty='l1'), n_jobs=1)
3 classifier.fit(x_train_multilabel_bow, y_train)
4 predictions = classifier.predict (x_test_multilabel_bow)
5
6
7 print("Accuracy :",metrics.accuracy_score(y_test, predictions))
8 print("Hamming loss ",metrics.hamming_loss(y_test,predictions))
9
10
11 precision = precision_score(y_test, predictions, average='micro')
12 recall = recall_score(y_test, predictions, average='micro')
13 f1 = f1_score(y_test, predictions, average='micro')
14
15 print("Micro-average quality numbers")
```



```
16 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
17
18 precision = precision_score(y_test, predictions, average='macro')
19 recall = recall_score(y_test, predictions, average='macro')
20 f1 = f1_score(y_test, predictions, average='macro')
21
22 print("Macro-average quality numbers")
23 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
24
25 print(metrics.classification_report(y_test, predictions))
26 print("Time taken to run this cell :", datetime.now() - start)
```



Accuracy : 0.11258571428571429
 Hamming loss 0.006103285714285714
 Micro-average quality numbers
 Precision: 0.3424, Recall: 0.5517, F1-measure: 0.4225
 Macro-average quality numbers
 Precision: 0.1623, Recall: 0.3566, F1-measure: 0.2083

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

0	0.97	0.96	0.96	42802
1	0.25	0.32	0.28	1764
2	0.27	0.41	0.33	942
3	0.24	0.28	0.26	6539
4	0.35	0.45	0.40	2540
5	0.50	0.58	0.54	2156
6	0.47	0.54	0.50	1990
7	0.13	0.21	0.16	611
8	0.20	0.33	0.25	324
9	0.69	0.74	0.71	2335
10	0.27	0.43	0.33	736
11	0.23	0.38	0.29	1199
12	0.31	0.48	0.38	842
13	0.30	0.44	0.36	770
14	0.40	0.63	0.49	975
15	0.22	0.33	0.26	523
16	0.09	0.36	0.14	143
17	0.34	0.57	0.43	428
18	0.39	0.60	0.47	552
19	0.44	0.59	0.50	708
20	0.26	0.63	0.37	89
21	0.17	0.39	0.24	219
22	0.29	0.29	0.29	662
23	0.18	0.20	0.19	1965
24	0.23	0.38	0.29	1046
25	0.23	0.34	0.27	488
26	0.26	0.38	0.31	458
27	0.25	0.39	0.31	1193
28	0.51	0.44	0.47	1492
29	0.09	0.22	0.13	354
30	0.10	0.47	0.17	159
31	0.09	0.34	0.15	157
32	0.17	0.26	0.21	740
33	0.40	0.44	0.42	1337
34	0.21	0.38	0.27	226
35	0.25	0.50	0.33	535
36	0.22	0.37	0.28	315
37	0.12	0.28	0.17	167
38	0.39	0.62	0.48	1074
39	0.23	0.33	0.27	336
40	0.13	0.45	0.21	312
41	0.13	0.19	0.15	682
42	0.36	0.65	0.46	265
43	0.46	0.65	0.54	478
44	0.22	0.42	0.29	598
45	0.19	0.33	0.24	1054
46	0.26	0.42	0.32	253
47	0.14	0.21	0.17	936
48	0.10	0.28	0.15	133
49	0.09	0.23	0.13	152
50	0.35	0.42	0.38	943
51	0.10	0.18	0.13	243
52	0.13	0.36	0.19	135
53	0.92	0.84	0.88	969
54	0.13	0.19	0.16	801
55	0.13	0.72	0.22	50
56	0.12	0.40	0.18	142
57	0.13	0.36	0.19	125
58	0.02	0.15	0.03	39
59	0.10	0.50	0.16	36
60	0.22	0.42	0.29	205
61	0.11	0.21	0.14	354
62	0.10	0.20	0.13	266
63	0.17	0.51	0.26	178

63	0.17	0.57	0.20	170
64	0.36	0.76	0.49	163
65	0.13	0.48	0.21	46
66	0.13	0.25	0.17	269
67	0.29	0.79	0.43	133
68	0.18	0.33	0.23	577
69	0.12	0.23	0.16	212
70	0.31	0.70	0.43	120
71	0.09	0.20	0.13	258
72	0.13	0.23	0.17	354
73	0.42	0.58	0.49	159
74	0.07	0.31	0.11	89
75	0.22	0.37	0.28	125
76	0.08	0.24	0.12	94
77	0.37	0.37	0.37	456
78	0.20	0.31	0.24	712
79	0.15	0.22	0.18	299
80	0.06	0.23	0.10	53
81	0.12	0.40	0.18	275
82	0.21	0.40	0.28	623
83	0.18	0.43	0.25	75
84	0.01	0.22	0.02	27
85	0.24	0.39	0.29	230
86	0.26	0.42	0.32	177
87	0.03	0.27	0.06	30
88	0.32	0.62	0.42	655
89	0.06	0.21	0.10	126
90	0.31	0.30	0.31	422
91	0.05	0.12	0.07	130
92	0.53	0.87	0.66	451
93	0.05	0.17	0.08	77
94	0.04	0.12	0.06	461
95	0.26	0.50	0.34	104
96	0.07	0.15	0.10	454
97	0.20	0.49	0.28	345
98	0.17	0.49	0.26	125
99	0.25	0.47	0.33	144
100	0.09	0.22	0.12	279
101	0.08	0.25	0.12	99
102	0.34	0.53	0.42	553
103	0.12	0.34	0.18	123
104	0.05	0.12	0.07	542
105	0.11	0.18	0.14	542
106	0.07	0.18	0.10	118
107	0.18	0.58	0.27	73
108	0.07	0.24	0.11	191
109	0.06	0.13	0.08	180
110	0.10	0.30	0.15	121
111	0.05	0.24	0.08	41
112	0.07	0.16	0.10	254
113	0.06	0.18	0.10	146
114	0.38	0.41	0.39	279
115	0.13	0.22	0.17	245
116	0.14	0.29	0.19	102
117	0.37	0.35	0.36	469
118	0.08	0.10	0.09	248
119	0.15	0.28	0.19	98
120	0.20	0.40	0.27	105
121	0.07	0.24	0.10	164
122	0.12	0.42	0.19	95
123	0.19	0.42	0.26	208
124	0.23	0.41	0.29	85
125	0.18	0.48	0.26	98
126	0.03	0.27	0.05	41
127	0.32	0.41	0.36	431
128	0.23	0.37	0.28	111
129	0.10	0.35	0.16	74
130	0.10	0.27	0.15	116
131	0.14	0.49	0.21	126
132	0.23	0.46	0.31	270
133	0.05	0.43	0.10	35
134	0.07	0.30	0.11	64
135	0.15	0.59	0.22	212

136	0.75	0.81	0.78	345
137	0.12	0.25	0.16	174
138	0.11	0.34	0.17	183
139	0.50	0.43	0.46	454
140	0.52	0.74	0.61	302
141	0.16	0.40	0.23	82
142	0.34	0.82	0.48	82
143	0.18	0.37	0.24	98
144	0.41	0.72	0.52	137
145	0.22	0.32	0.26	412
146	0.02	0.10	0.03	224
147	0.11	0.25	0.15	153
148	0.25	0.70	0.37	64
149	0.17	0.51	0.26	68
150	0.08	0.25	0.12	126
151	0.07	0.10	0.09	202
152	0.02	0.31	0.04	39
153	0.09	0.39	0.15	36
154	0.20	0.49	0.28	136
155	0.07	0.13	0.09	212
156	0.23	0.45	0.31	51
157	0.17	0.45	0.25	94
158	0.04	0.13	0.06	286
159	0.28	0.48	0.35	350
160	0.07	0.36	0.12	22
161	0.05	0.20	0.07	120
162	0.06	0.17	0.09	144
163	0.24	0.50	0.33	119
164	0.07	0.29	0.11	42
165	0.66	0.83	0.73	361
166	0.11	0.24	0.15	206
167	0.16	0.40	0.23	87
168	0.06	0.35	0.10	112
169	0.15	0.27	0.20	298
170	0.19	0.24	0.21	191
171	0.17	0.44	0.24	91
172	0.22	0.52	0.31	100
173	0.04	0.14	0.07	167
174	0.32	0.31	0.32	344
175	0.03	0.07	0.04	76
176	0.06	0.17	0.08	198
177	0.09	0.27	0.13	127
178	0.08	0.26	0.12	102
179	0.10	0.52	0.16	31
180	0.23	0.42	0.30	139
181	0.17	0.51	0.25	63
182	0.17	0.28	0.21	367
183	0.19	0.55	0.28	67
184	0.01	0.04	0.01	46
185	0.46	0.16	0.24	381
186	0.04	0.17	0.06	29
187	0.11	0.26	0.16	111
188	0.14	0.38	0.21	121
189	0.01	0.02	0.02	82
190	0.15	0.36	0.21	118
191	0.13	0.43	0.20	77
192	0.19	0.53	0.28	118
193	0.05	0.15	0.07	159
194	0.26	0.30	0.28	269
195	0.25	0.67	0.36	81
196	0.22	0.43	0.29	299
197	0.04	0.11	0.06	47
198	0.30	0.77	0.43	47
199	0.08	0.27	0.12	62
200	0.01	0.08	0.02	24
201	0.18	0.42	0.25	86
202	0.29	0.27	0.28	308
203	0.40	0.55	0.46	321
204	0.03	0.06	0.04	67
205	0.06	0.25	0.09	28
206	0.17	0.61	0.26	59
207	0.14	0.21	0.19	245

207	0.17	0.51	0.15	27
208	0.25	0.68	0.37	53
209	0.03	0.11	0.05	274
210	0.02	0.38	0.04	8
211	0.20	0.46	0.28	95
212	0.18	0.37	0.24	129
213	0.13	0.50	0.20	34
214	0.17	0.30	0.22	89
215	0.14	0.43	0.21	67
216	0.07	0.36	0.12	25
217	0.49	0.80	0.61	109
218	0.09	0.16	0.11	134
219	0.20	0.46	0.28	70
220	0.14	0.42	0.21	67
221	0.14	0.32	0.20	79
222	0.22	0.44	0.29	50
223	0.34	0.75	0.47	93
224	0.13	0.35	0.19	94
225	0.04	0.11	0.06	180
226	0.03	0.14	0.05	79
227	0.13	0.45	0.20	64
228	0.03	0.10	0.05	50
229	0.21	0.64	0.31	53
230	0.08	0.34	0.13	44
231	0.11	0.28	0.16	61
232	0.11	0.59	0.19	49
233	0.40	0.75	0.52	72
234	0.12	0.18	0.14	233
235	0.11	0.16	0.13	166
236	0.21	0.60	0.31	58
237	0.06	0.20	0.10	152
238	0.23	0.30	0.26	302
239	0.19	0.52	0.28	42
240	0.25	0.52	0.34	269
241	0.05	0.13	0.07	54
242	0.08	0.25	0.12	162
243	0.06	0.30	0.10	23
244	0.24	0.44	0.31	66
245	0.05	0.33	0.09	40
246	0.22	0.42	0.29	73
247	0.49	0.71	0.58	78
248	0.08	0.18	0.11	131
249	0.10	0.26	0.14	82
250	0.30	0.65	0.41	57
251	0.15	0.18	0.16	296
252	0.13	0.24	0.17	87
253	0.13	0.27	0.18	96
254	0.38	0.30	0.34	280
255	0.03	0.21	0.05	24
256	0.10	0.24	0.14	88
257	0.04	0.57	0.07	7
258	0.08	0.21	0.11	136
259	0.28	0.51	0.36	73
260	0.42	0.19	0.26	268
261	0.30	0.91	0.45	11
262	0.39	0.55	0.46	82
263	0.04	0.60	0.07	5
264	0.13	0.32	0.18	108
265	0.26	0.63	0.36	78
266	0.06	0.20	0.09	69
267	0.05	0.19	0.08	80
268	0.13	0.43	0.20	28
269	0.03	0.14	0.05	44
270	0.23	0.60	0.33	42
271	0.14	0.34	0.20	114
272	0.08	0.17	0.11	59
273	0.09	0.23	0.13	130
274	0.10	0.33	0.15	48
275	0.04	0.20	0.07	227
276	0.30	0.73	0.43	75
277	0.13	0.22	0.16	68
278	0.40	0.56	0.46	143
279	0.01	0.22	0.07	72

279	0.07	0.22	0.07	78
280	0.30	0.63	0.41	78
281	0.30	0.52	0.39	61
282	0.04	0.11	0.06	61
283	0.08	0.27	0.12	52
284	0.02	0.12	0.03	24
285	0.03	0.10	0.05	125
286	0.11	0.25	0.15	138
287	0.11	0.25	0.15	171
288	0.38	0.50	0.43	157
289	0.09	0.43	0.14	30
290	0.02	0.17	0.04	30
291	0.21	0.47	0.29	64
292	0.02	0.33	0.04	9
293	0.13	0.30	0.18	123
294	0.08	0.29	0.13	35
295	0.01	0.05	0.01	22
296	0.24	0.47	0.32	184
297	0.18	0.30	0.23	140
298	0.19	0.29	0.23	224
299	0.19	0.46	0.27	97
300	0.05	0.12	0.07	65
301	0.04	0.16	0.06	44
302	0.09	0.39	0.14	38
303	0.22	0.30	0.26	98
304	0.08	0.42	0.13	31
305	0.30	0.40	0.34	235
306	0.39	0.73	0.51	249
307	0.13	0.10	0.11	247
308	0.12	0.45	0.19	122
309	0.13	0.20	0.16	230
310	0.09	0.23	0.13	166
311	0.05	0.28	0.09	40
312	0.05	0.24	0.08	17
313	0.06	0.31	0.10	36
314	0.27	0.47	0.34	109
315	0.00	0.01	0.01	67
316	0.35	0.67	0.46	79
317	0.10	0.15	0.12	197
318	0.10	0.49	0.17	47
319	0.47	0.39	0.43	222
320	0.11	0.59	0.19	27
321	0.50	0.63	0.56	207
322	0.42	0.39	0.40	240
323	0.20	0.15	0.18	215
324	0.20	0.43	0.27	120
325	0.12	0.37	0.18	130
326	0.23	0.57	0.33	28
327	0.07	0.16	0.10	166
328	0.32	0.60	0.42	45
329	0.33	0.54	0.41	180
330	0.04	0.24	0.06	62
331	0.10	0.24	0.14	105
332	0.23	0.69	0.34	39
333	0.06	1.00	0.12	4
334	0.13	0.41	0.19	113
335	0.13	0.47	0.20	78
336	0.08	0.22	0.11	51
337	0.20	0.18	0.19	147
338	0.01	0.03	0.02	135
339	0.04	0.15	0.07	27
340	0.06	0.14	0.08	79
341	0.39	0.73	0.51	30
342	0.10	0.30	0.15	54
343	0.26	0.31	0.28	195
344	0.13	0.36	0.19	39
345	0.11	0.89	0.19	9
346	0.47	0.69	0.56	86
347	0.05	0.16	0.08	44
348	0.25	0.42	0.32	185
349	0.21	0.62	0.31	66
350	0.03	0.67	0.06	3
351	0.10	0.10	0.15	25

352	0.37	0.38	0.38	216
353	0.22	0.40	0.28	42
354	0.04	0.50	0.08	6
355	0.02	1.00	0.04	3
356	0.03	0.36	0.05	14
357	0.09	0.55	0.15	31
358	0.11	0.33	0.16	204
359	0.05	0.05	0.05	211
360	0.30	0.26	0.28	184
361	0.18	0.25	0.21	108
362	0.02	0.06	0.03	54
363	0.05	0.27	0.08	56
364	0.09	0.21	0.12	97
365	0.12	0.40	0.19	72
366	0.02	0.17	0.03	12
367	0.29	0.22	0.25	185
368	0.12	0.16	0.13	193
369	0.03	0.12	0.05	34
370	0.18	0.34	0.23	164
371	0.14	0.83	0.24	18
372	0.08	0.29	0.13	65
373	0.07	0.35	0.11	20
374	0.01	0.03	0.01	29
375	0.27	0.42	0.33	71
376	0.04	0.05	0.04	164
377	0.17	0.53	0.25	185
378	0.05	0.21	0.08	24
379	0.09	0.33	0.14	52
380	0.02	0.14	0.04	57
381	0.09	0.17	0.12	59
382	0.11	0.29	0.16	117
383	0.17	0.62	0.27	39
384	0.28	0.45	0.34	125
385	0.07	0.25	0.11	130
386	0.26	0.61	0.37	74
387	0.19	0.66	0.29	35
388	0.10	0.67	0.17	21
389	0.20	0.25	0.22	175
390	0.07	0.11	0.09	54
391	0.00	0.00	0.00	29
392	0.07	0.14	0.09	63
393	0.10	0.47	0.17	34
394	0.20	0.66	0.31	38
395	0.05	0.27	0.09	15
396	0.01	0.10	0.01	10
397	0.08	0.27	0.13	49
398	0.24	0.31	0.27	169
399	0.08	0.30	0.13	33
400	0.35	0.56	0.43	84
401	0.27	0.61	0.37	31
402	0.10	0.54	0.17	24
403	0.57	0.26	0.36	187
404	0.03	0.50	0.06	6
405	0.12	0.33	0.18	33
406	0.02	0.18	0.04	17
407	0.03	0.19	0.05	21
408	0.07	0.21	0.10	62
409	0.07	0.14	0.09	78
410	0.46	0.47	0.46	147
411	0.06	0.29	0.10	31
412	0.00	0.00	0.00	14
413	0.19	0.31	0.23	103
414	0.42	0.75	0.53	36
415	0.28	0.57	0.38	68
416	0.02	0.19	0.04	43
417	0.18	0.45	0.26	73
418	0.51	0.56	0.54	62
419	0.06	0.23	0.10	97
420	0.06	0.21	0.09	42
421	0.17	0.41	0.24	41
422	0.01	0.05	0.02	38
423	0.00	0.00	0.00	26

423	0.00	0.00	0.00	0
424	0.04	0.23	0.07	13
425	0.06	0.33	0.10	24
426	0.00	0.33	0.01	3
427	0.07	0.30	0.11	94
428	0.27	0.26	0.27	151
429	0.34	0.71	0.46	63
430	0.02	0.25	0.04	40
431	0.23	0.43	0.30	49
432	0.01	0.03	0.01	34
433	0.27	0.78	0.40	37
434	0.10	0.35	0.15	34
435	0.06	1.00	0.11	1
436	0.03	0.17	0.05	29
437	0.06	0.18	0.09	50
438	0.00	0.00	0.00	104
439	0.00	0.00	0.00	29
440	0.06	0.35	0.11	23
441	0.03	0.09	0.04	46
442	0.05	0.15	0.07	39
443	0.17	0.34	0.23	56
444	0.18	0.47	0.27	80
445	0.03	0.10	0.05	30
446	0.04	0.17	0.06	30
447	0.13	0.30	0.18	37
448	0.02	0.05	0.03	39
449	0.07	0.16	0.10	83
450	0.06	0.39	0.10	23
451	0.06	0.44	0.10	9
452	0.06	0.23	0.10	44
453	0.25	0.17	0.20	166
454	0.19	0.66	0.30	32
455	0.06	0.26	0.10	53
456	0.14	0.63	0.23	41
457	0.05	0.12	0.07	78
458	0.13	0.19	0.16	133
459	0.05	0.20	0.08	25
460	0.11	0.34	0.16	103
461	0.10	0.21	0.14	53
462	0.07	0.30	0.12	30
463	0.03	0.50	0.05	4
464	0.06	0.08	0.07	66
465	0.03	0.28	0.05	47
466	0.06	0.26	0.10	31
467	0.14	0.49	0.22	35
468	0.01	0.04	0.02	50
469	0.34	0.35	0.34	155
470	0.06	0.26	0.10	27
471	0.36	0.50	0.42	58
472	0.03	0.08	0.05	159

```

1 start = datetime.now()
2 classifier_2 = OneVsRestClassifier(LogisticRegression(penalty='l1', solver='liblinear'), n_jobs=1)
3 classifier_2.fit(x_train_multilabel_bow, y_train)
4 predictions_2 = classifier_2.predict(x_test_multilabel_bow)
5 print("Accuracy :", metrics.accuracy_score(y_test, predictions_2))
6 print("Hamming loss ", metrics.hamming_loss(y_test, predictions_2))
7
8
9 precision = precision_score(y_test, predictions_2, average='micro')
10 recall = recall_score(y_test, predictions_2, average='micro')
11 f1 = f1_score(y_test, predictions_2, average='micro')
12
13 print("Micro-average quality numbers")
14 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
15
16 precision = precision_score(y_test, predictions_2, average='macro')
17 recall = recall_score(y_test, predictions_2, average='macro')
18 f1 = f1_score(y_test, predictions_2, average='macro')
19
20 print("Macro-average quality numbers")

```



```
21 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
22
23 print (metrics.classification_report(y_test, predictions_2))
24 print("Time taken to run this cell :", datetime.now() - start)
```



```

Accuracy : 0.2359
Hamming loss 0.002942057142857143
Micro-average quality numbers
Precision: 0.6777, Recall: 0.5207, F1-measure: 0.5889
Macro-average quality numbers
Precision: 0.4122, Recall: 0.3021, F1-measure: 0.3397
precision    recall    f1-score   support

```

0	0.98	0.97	0.97	42802
1	0.42	0.26	0.32	1764
2	0.56	0.38	0.46	942
3	0.34	0.19	0.25	6539
4	0.48	0.41	0.45	2540
5	0.65	0.54	0.59	2156
6	0.66	0.50	0.57	1990
7	0.31	0.17	0.22	611
8	0.55	0.24	0.34	324
9	0.76	0.78	0.77	2335
10	0.48	0.40	0.44	736
11	0.40	0.29	0.34	1199
12	0.55	0.40	0.46	842
13	0.54	0.36	0.43	770
14	0.61	0.56	0.58	975
15	0.37	0.21	0.27	523
16	0.40	0.27	0.32	143
17	0.71	0.56	0.62	428
18	0.71	0.53	0.61	552
19	0.67	0.55	0.60	708
20	0.81	0.61	0.69	89
21	0.48	0.25	0.33	219
22	0.43	0.27	0.33	662
23	0.23	0.13	0.17	1965
24	0.50	0.28	0.36	1046
25	0.44	0.29	0.35	488
26	0.47	0.31	0.37	458
27	0.52	0.38	0.44	1193
28	0.68	0.48	0.56	1492
29	0.23	0.16	0.19	354
30	0.49	0.41	0.45	159
31	0.47	0.29	0.36	157
32	0.31	0.19	0.24	740
33	0.53	0.40	0.46	1337
34	0.57	0.32	0.41	226
35	0.66	0.49	0.57	535
36	0.46	0.28	0.35	315
37	0.33	0.20	0.25	167
38	0.74	0.64	0.68	1074
39	0.44	0.26	0.33	336
40	0.49	0.37	0.42	312
41	0.22	0.13	0.16	682
42	0.74	0.57	0.65	265
43	0.77	0.61	0.68	478
44	0.63	0.36	0.46	598
45	0.45	0.27	0.34	1054
46	0.51	0.30	0.38	253
47	0.27	0.17	0.21	936
48	0.32	0.23	0.27	133
49	0.38	0.18	0.24	152
50	0.43	0.37	0.40	943
51	0.17	0.11	0.13	243
52	0.43	0.28	0.34	135
53	0.98	0.90	0.94	969
54	0.24	0.13	0.17	801
55	0.58	0.64	0.61	50
56	0.55	0.36	0.44	142
57	0.49	0.27	0.35	125
58	0.24	0.13	0.17	39
59	0.38	0.36	0.37	36
60	0.51	0.33	0.40	205
61	0.29	0.21	0.24	354
62	0.28	0.20	0.24	266
63	0.58	0.17	0.57	178

63	0.90	0.68	0.77	163
64	0.53	0.46	0.49	46
65	0.36	0.20	0.26	269
66	0.78	0.74	0.76	133
67	0.32	0.20	0.24	577
68	0.23	0.09	0.13	212
69	0.82	0.66	0.73	120
70	0.29	0.16	0.21	258
71	0.19	0.12	0.14	354
72	0.72	0.58	0.64	159
73	0.50	0.27	0.35	89
74	0.37	0.29	0.32	125
75	0.26	0.20	0.23	94
76	0.52	0.36	0.43	456
77	0.42	0.38	0.40	712
78	0.29	0.11	0.16	299
79	0.28	0.17	0.21	53
80	0.43	0.32	0.37	275
81	0.53	0.23	0.32	623
82	0.53	0.39	0.45	75
83	0.12	0.07	0.09	27
84	0.62	0.34	0.44	230
85	0.49	0.36	0.41	177
86	0.45	0.33	0.38	30
87	0.60	0.69	0.64	655
88	0.14	0.10	0.12	126
89	0.50	0.37	0.42	422
90	0.15	0.04	0.06	130
91	0.82	0.86	0.84	451
92	0.29	0.06	0.11	77
93	0.13	0.05	0.07	461
94	0.71	0.38	0.50	104
95	0.18	0.09	0.12	454
96	0.39	0.37	0.38	345
97	0.44	0.44	0.44	125
98	0.52	0.33	0.41	144
99	0.20	0.06	0.09	279
100	0.49	0.18	0.26	99
101	0.78	0.62	0.69	553
102	0.52	0.24	0.32	123
103	0.14	0.08	0.10	542
104	0.21	0.14	0.17	542
105	0.33	0.11	0.16	118
106	0.52	0.40	0.45	73
107	0.27	0.14	0.19	191
108	0.29	0.12	0.17	180
109	0.48	0.27	0.35	121
110	0.28	0.17	0.21	41
111	0.14	0.06	0.08	254
112	0.19	0.13	0.15	146
113	0.60	0.36	0.45	279
114	0.33	0.17	0.22	245
115	0.49	0.21	0.29	102
116	0.56	0.39	0.46	469
117	0.20	0.06	0.09	248
118	0.37	0.24	0.29	98
119	0.56	0.47	0.51	105
120	0.23	0.11	0.15	164
121	0.48	0.31	0.37	95
122	0.48	0.27	0.35	208
123	0.59	0.39	0.47	85
124	0.60	0.53	0.57	98
125	0.27	0.15	0.19	41
126	0.79	0.34	0.47	431
127	0.59	0.33	0.43	111
128	0.39	0.22	0.28	74
129	0.42	0.20	0.27	116
130	0.52	0.42	0.47	126
131	0.35	0.40	0.37	270
132	0.36	0.29	0.32	35
133	0.34	0.27	0.30	64
134	0.50	0.66	0.62	212

135	0.99	0.99	0.99	475
136	0.96	0.79	0.87	345
137	0.41	0.18	0.25	174
138	0.32	0.27	0.29	183
139	0.73	0.39	0.50	454
140	0.81	0.72	0.76	302
141	0.68	0.34	0.46	82
142	0.77	0.77	0.77	82
143	0.50	0.37	0.42	98
144	0.86	0.74	0.79	137
145	0.41	0.25	0.31	412
146	0.21	0.09	0.12	224
147	0.29	0.13	0.18	153
148	0.60	0.62	0.61	64
149	0.61	0.41	0.49	68
150	0.20	0.08	0.11	126
151	0.15	0.09	0.12	202
152	0.26	0.28	0.27	39
153	0.56	0.39	0.46	36
154	0.50	0.46	0.48	136
155	0.20	0.06	0.09	212
156	0.49	0.39	0.43	51
157	0.45	0.44	0.44	94
158	0.20	0.08	0.12	286
159	0.52	0.45	0.48	350
160	0.45	0.41	0.43	22
161	0.10	0.07	0.08	120
162	0.24	0.12	0.16	144
163	0.59	0.50	0.55	119
164	0.37	0.26	0.31	42
165	0.82	0.81	0.81	361
166	0.28	0.15	0.19	206
167	0.60	0.29	0.39	87
168	0.29	0.29	0.29	112
169	0.24	0.11	0.15	298
170	0.40	0.23	0.29	191
171	0.44	0.48	0.46	91
172	0.67	0.49	0.57	100
173	0.10	0.05	0.07	167
174	0.47	0.34	0.40	344
175	0.07	0.04	0.05	76
176	0.17	0.09	0.12	198
177	0.40	0.24	0.30	127
178	0.33	0.20	0.25	102
179	0.65	0.35	0.46	31
180	0.51	0.46	0.48	139
181	0.65	0.56	0.60	63
182	0.46	0.24	0.32	367
183	0.52	0.70	0.59	67
184	0.00	0.00	0.00	46
185	0.57	0.16	0.25	381
186	0.90	0.31	0.46	29
187	0.33	0.26	0.29	111
188	0.26	0.17	0.20	121
189	0.21	0.04	0.06	82
190	0.58	0.42	0.49	118
191	0.70	0.52	0.60	77
192	0.59	0.53	0.56	118
193	0.26	0.09	0.14	159
194	0.43	0.38	0.41	269
195	0.76	0.64	0.70	81
196	0.41	0.31	0.35	299
197	0.31	0.09	0.13	47
198	0.76	0.74	0.75	47
199	0.40	0.26	0.31	62
200	0.31	0.17	0.22	24
201	0.53	0.27	0.36	86
202	0.53	0.22	0.31	308
203	0.67	0.56	0.61	321
204	0.08	0.03	0.04	67
205	0.16	0.18	0.17	28
206	0.58	0.49	0.53	59
207	0.26	0.22	0.28	215

207	0.30	0.25	0.20	25
208	0.85	0.64	0.73	53
209	0.11	0.09	0.10	274
210	0.12	0.12	0.12	8
211	0.48	0.26	0.34	95
212	0.29	0.22	0.25	129
213	0.64	0.47	0.54	34
214	0.29	0.34	0.31	89
215	0.55	0.36	0.43	67
216	0.36	0.20	0.26	25
217	0.62	0.63	0.63	109
218	0.40	0.15	0.22	134
219	0.62	0.49	0.54	70
220	0.53	0.40	0.46	67
221	0.39	0.22	0.28	79
222	0.50	0.40	0.44	50
223	0.72	0.71	0.71	93
224	0.27	0.26	0.26	94
225	0.14	0.05	0.07	180
226	0.10	0.06	0.08	79
227	0.31	0.36	0.33	64
228	0.15	0.06	0.09	50
229	0.65	0.64	0.65	53
230	0.52	0.34	0.41	44
231	0.24	0.21	0.23	61
232	0.39	0.39	0.39	49
233	0.73	0.75	0.74	72
234	0.20	0.14	0.16	233
235	0.33	0.10	0.16	166
236	0.67	0.52	0.58	58
237	0.16	0.07	0.09	152
238	0.52	0.27	0.35	302
239	0.49	0.55	0.52	42
240	0.46	0.34	0.39	269
241	0.26	0.11	0.16	54
242	0.21	0.07	0.11	162
243	0.50	0.39	0.44	23
244	0.65	0.30	0.41	66
245	0.19	0.10	0.13	40
246	0.67	0.59	0.63	73
247	0.83	0.62	0.71	78
248	0.35	0.17	0.23	131
249	0.26	0.11	0.16	82
250	0.67	0.58	0.62	57
251	0.19	0.06	0.09	296
252	0.39	0.18	0.25	87
253	0.27	0.16	0.20	96
254	0.44	0.14	0.21	280
255	0.25	0.21	0.23	24
256	0.31	0.19	0.24	88
257	0.23	0.43	0.30	7
258	0.24	0.15	0.19	136
259	0.72	0.56	0.63	73
260	0.63	0.43	0.51	268
261	1.00	0.82	0.90	11
262	0.69	0.51	0.59	82
263	0.50	0.80	0.62	5
264	0.37	0.24	0.29	108
265	0.67	0.50	0.57	78
266	0.16	0.10	0.12	69
267	0.34	0.12	0.18	80
268	0.37	0.25	0.30	28
269	0.12	0.05	0.07	44
270	0.66	0.60	0.62	42
271	0.47	0.25	0.32	114
272	0.10	0.08	0.09	59
273	0.27	0.21	0.24	130
274	0.26	0.21	0.23	48
275	0.17	0.11	0.13	227
276	0.72	0.77	0.74	75
277	0.71	0.32	0.44	68
278	0.59	0.48	0.53	143
279	0.16	0.15	0.16	72

277	0.10	0.10	0.10	70
280	0.69	0.56	0.62	78
281	0.72	0.59	0.65	61
282	0.06	0.02	0.03	61
283	0.43	0.25	0.32	52
284	0.20	0.08	0.12	24
285	0.06	0.02	0.02	125
286	0.35	0.19	0.24	138
287	0.18	0.09	0.12	171
288	0.68	0.45	0.54	157
289	0.52	0.40	0.45	30
290	0.18	0.10	0.13	30
291	0.77	0.31	0.44	64
292	0.14	0.22	0.17	9
293	0.20	0.12	0.15	123
294	0.45	0.29	0.35	35
295	0.25	0.09	0.13	22
296	0.41	0.46	0.43	184
297	0.64	0.21	0.32	140
298	0.38	0.19	0.25	224
299	0.63	0.45	0.53	97
300	0.07	0.03	0.04	65
301	0.11	0.05	0.06	44
302	0.43	0.26	0.33	38
303	0.37	0.18	0.24	98
304	0.48	0.48	0.48	31
305	0.38	0.25	0.30	235
306	0.98	0.93	0.95	249
307	0.40	0.09	0.15	247
308	0.40	0.30	0.34	122
309	0.20	0.08	0.11	230
310	0.37	0.16	0.22	166
311	0.23	0.15	0.18	40
312	0.56	0.29	0.38	17
313	0.27	0.19	0.23	36
314	0.55	0.33	0.41	109
315	0.00	0.00	0.00	67
316	0.63	0.58	0.61	79
317	0.08	0.02	0.03	197
318	0.27	0.53	0.36	47
319	0.62	0.35	0.45	222
320	0.47	0.56	0.51	27
321	0.68	0.56	0.61	207
322	0.69	0.38	0.49	240
323	0.26	0.18	0.21	215
324	0.42	0.33	0.37	120
325	0.42	0.23	0.30	130
326	0.68	0.54	0.60	28
327	0.15	0.16	0.15	166
328	0.66	0.56	0.60	45
329	0.45	0.45	0.45	180
330	0.17	0.08	0.11	62
331	0.44	0.23	0.30	105
332	0.81	0.67	0.73	39
333	0.67	1.00	0.80	4
334	0.54	0.28	0.37	113
335	0.39	0.40	0.39	78
336	0.27	0.14	0.18	51
337	0.42	0.17	0.24	147
338	0.07	0.02	0.03	135
339	0.29	0.33	0.31	27
340	0.29	0.13	0.18	79
341	0.92	0.73	0.81	30
342	0.26	0.15	0.19	54
343	0.59	0.23	0.33	195
344	0.35	0.31	0.33	39
345	0.75	0.67	0.71	9
346	0.75	0.78	0.77	86
347	0.39	0.20	0.27	44
348	0.68	0.39	0.50	185
349	0.68	0.61	0.64	66
350	1.00	0.67	0.80	3
351	0.20	0.21	0.25	25

352	0.57	0.42	0.48	216
353	0.41	0.38	0.40	42
354	0.67	0.33	0.44	6
355	0.06	0.33	0.10	3
356	0.22	0.14	0.17	14
357	0.34	0.35	0.35	31
358	0.32	0.06	0.10	204
359	0.12	0.01	0.02	211
360	0.38	0.19	0.25	184
361	0.29	0.20	0.24	108
362	0.06	0.02	0.03	54
363	0.21	0.18	0.19	56
364	0.21	0.10	0.14	97
365	0.29	0.21	0.24	72
366	0.17	0.08	0.11	12
367	0.64	0.41	0.50	185
368	0.18	0.06	0.09	193
369	0.27	0.12	0.16	34
370	0.57	0.34	0.43	164
371	0.62	0.72	0.67	18
372	0.22	0.11	0.14	65
373	0.60	0.30	0.40	20
374	0.00	0.00	0.00	29
375	0.64	0.42	0.51	71
376	0.14	0.12	0.13	164
377	0.37	0.34	0.35	185
378	0.19	0.21	0.20	24
379	0.24	0.17	0.20	52
380	0.21	0.12	0.15	57
381	0.20	0.07	0.10	59
382	0.23	0.09	0.13	117
383	0.58	0.54	0.56	39
384	0.57	0.53	0.55	125
385	0.35	0.14	0.20	130
386	0.66	0.58	0.62	74
387	0.71	0.63	0.67	35
388	0.48	0.62	0.54	21
389	0.35	0.24	0.28	175
390	0.19	0.09	0.12	54
391	0.00	0.00	0.00	29
392	0.35	0.11	0.17	63
393	0.28	0.32	0.30	34
394	0.81	0.66	0.72	38
395	0.20	0.20	0.20	15
396	0.29	0.20	0.24	10
397	0.33	0.18	0.24	49
398	0.51	0.20	0.29	169
399	0.29	0.24	0.26	33
400	0.45	0.35	0.39	84
401	0.72	0.58	0.64	31
402	0.38	0.42	0.40	24
403	0.67	0.36	0.47	187
404	0.40	0.33	0.36	6
405	0.50	0.30	0.38	33
406	0.21	0.24	0.22	17
407	0.20	0.10	0.13	21
408	0.42	0.16	0.23	62
409	0.20	0.06	0.10	78
410	0.70	0.50	0.58	147
411	0.30	0.26	0.28	31
412	0.00	0.00	0.00	14
413	0.41	0.12	0.18	103
414	0.74	0.64	0.69	36
415	0.58	0.59	0.58	68
416	0.07	0.05	0.05	43
417	0.45	0.38	0.41	73
418	0.73	0.60	0.65	62
419	0.04	0.02	0.03	97
420	0.03	0.02	0.03	42
421	0.63	0.46	0.54	41
422	0.07	0.05	0.06	38
423	0.00	0.00	0.00	26

424	0.14	0.08	0.10	13
425	0.42	0.33	0.37	24
426	0.10	0.33	0.15	3
427	0.37	0.23	0.29	94
428	0.38	0.22	0.28	151
429	0.60	0.54	0.57	63
430	0.09	0.05	0.06	40
431	0.53	0.41	0.46	49
432	0.00	0.00	0.00	34
433	0.78	0.76	0.77	37
434	0.36	0.29	0.32	34
435	1.00	1.00	1.00	1
436	0.13	0.07	0.09	29
437	0.20	0.18	0.19	50
438	0.00	0.00	0.00	104
439	0.00	0.00	0.00	29
440	0.33	0.22	0.26	23
441	0.05	0.02	0.03	46
442	0.12	0.08	0.09	39
443	0.57	0.30	0.40	56
444	0.54	0.49	0.51	80
445	0.09	0.03	0.05	30
446	0.11	0.10	0.11	30
447	0.32	0.30	0.31	37
448	0.19	0.10	0.13	39
449	0.21	0.14	0.17	83
450	0.47	0.39	0.43	23
451	0.40	0.67	0.50	9
452	0.38	0.20	0.26	44
453	0.48	0.36	0.41	166
454	0.64	0.56	0.60	32
455	0.37	0.25	0.30	53
456	0.52	0.59	0.55	41
457	0.18	0.09	0.12	78
458	0.29	0.11	0.16	133
459	0.25	0.16	0.20	25
460	0.51	0.37	0.43	103
461	0.53	0.15	0.24	53
462	0.34	0.33	0.34	30
463	0.11	0.25	0.15	4
464	0.11	0.06	0.08	66

▼ LINEAR SVM

468	0.00	0.00	0.00	50
-----	------	------	------	----

```

1 from sklearn.model_selection import GridSearchCV
2 from scipy.stats import randint as sp_randint
3 from sklearn.model_selection import cross_val_score
4 from sklearn.linear_model import SGDClassifier
5
6
7 LinearSVM = OneVsRestClassifier(SGDClassifier(loss='hinge',penalty='l2',random_state=None, class_weight=None), n_j
8 LinearSVM.fit(x_train_multilabel_bow, y_train)
9 predictions_svm = LinearSVM.predict(x_test_multilabel_bow)
10 print("Accuracy :",metrics.accuracy_score(y_test, predictions_svm))
11 print("Hamming loss ",metrics.hamming_loss(y_test,predictions_svm))
12
13
14 precision = precision_score(y_test, predictions_svm, average='micro')
15 recall = recall_score(y_test, predictions_svm, average='micro')
16 f1 = f1_score(y_test, predictions_svm, average='micro')
17
18 print("Micro-average quality numbers")
19 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
20
21 precision = precision_score(y_test, predictions_svm, average='macro')
22 recall = recall_score(y_test, predictions_svm, average='macro')
23 f1 = f1_score(y_test, predictions_svm, average='macro')

```



```
24
25 print("Macro-average quality numbers")
26 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
27
28 print (metrics.classification_report(y_test, predictions_svm))
29 print("Time taken to run this cell :", datetime.now() - start)
```



Accuracy : 0.2384
 Hamming loss 0.002897085714285714
 Micro-average quality numbers
 Precision: 0.7029, Recall: 0.4923, F1-measure: 0.5790
 Macro-average quality numbers
 Precision: 0.4318, Recall: 0.2714, F1-measure: 0.3184

	precision	recall	f1-score	support
0	0.98	0.95	0.96	42802
1	0.36	0.23	0.28	1764
2	0.55	0.32	0.40	942
3	0.36	0.17	0.23	6539
4	0.46	0.37	0.41	2540
5	0.64	0.53	0.58	2156
6	0.64	0.49	0.56	1990
7	0.29	0.18	0.22	611
8	0.43	0.25	0.32	324
9	0.76	0.79	0.78	2335
10	0.44	0.38	0.41	736
11	0.37	0.25	0.29	1199
12	0.49	0.36	0.42	842
13	0.52	0.30	0.38	770
14	0.55	0.53	0.54	975
15	0.32	0.17	0.23	523
16	0.38	0.25	0.30	143
17	0.64	0.54	0.59	428
18	0.71	0.50	0.59	552
19	0.62	0.53	0.57	708
20	0.84	0.60	0.70	89
21	0.47	0.18	0.26	219
22	0.41	0.24	0.30	662
23	0.25	0.12	0.16	1965
24	0.49	0.27	0.35	1046
25	0.42	0.26	0.32	488
26	0.42	0.27	0.33	458
27	0.53	0.38	0.45	1193
28	0.69	0.42	0.52	1492
29	0.24	0.13	0.17	354
30	0.43	0.36	0.39	159
31	0.33	0.24	0.28	157
32	0.32	0.22	0.26	740
33	0.55	0.40	0.46	1337
34	0.57	0.30	0.39	226
35	0.69	0.44	0.54	535
36	0.48	0.18	0.27	315
37	0.28	0.15	0.20	167
38	0.76	0.59	0.67	1074
39	0.39	0.21	0.28	336
40	0.43	0.35	0.39	312
41	0.24	0.16	0.19	682
42	0.81	0.54	0.65	265
43	0.76	0.63	0.68	478
44	0.67	0.25	0.36	598
45	0.53	0.25	0.34	1054
46	0.49	0.28	0.36	253
47	0.31	0.13	0.18	936
48	0.33	0.20	0.25	133
49	0.31	0.10	0.15	152
50	0.45	0.24	0.31	943
51	0.19	0.09	0.12	243
52	0.41	0.30	0.35	135
53	0.98	0.74	0.85	969
54	0.23	0.04	0.07	801
55	0.62	0.58	0.60	50
56	0.60	0.30	0.40	142
57	0.39	0.24	0.30	125
58	0.12	0.08	0.09	39
59	0.44	0.33	0.38	36
60	0.51	0.27	0.36	205
61	0.31	0.19	0.23	354
62	0.27	0.15	0.19	266
63	0.66	0.28	0.48	178

63	0.88	0.58	0.78	178
64	0.85	0.63	0.72	163
65	0.54	0.43	0.48	46
66	0.39	0.15	0.22	269
67	0.72	0.74	0.73	133
68	0.34	0.09	0.14	577
69	0.15	0.04	0.06	212
70	0.77	0.67	0.71	120
71	0.25	0.14	0.18	258
72	0.17	0.19	0.18	354
73	0.72	0.57	0.63	159
74	0.44	0.25	0.32	89
75	0.48	0.29	0.36	125
76	0.32	0.26	0.29	94
77	0.56	0.28	0.37	456
78	0.48	0.27	0.35	712
79	0.25	0.07	0.11	299
80	0.18	0.15	0.16	53
81	0.54	0.27	0.36	275
82	0.53	0.11	0.19	623
83	0.55	0.37	0.44	75
84	0.10	0.07	0.08	27
85	0.63	0.39	0.48	230
86	0.51	0.32	0.39	177
87	0.50	0.33	0.40	30
88	0.61	0.50	0.55	655
89	0.18	0.12	0.14	126
90	0.49	0.38	0.43	422
91	0.08	0.02	0.03	130
92	0.83	0.80	0.82	451
93	0.19	0.05	0.08	77
94	0.08	0.00	0.01	461
95	0.68	0.40	0.51	104
96	0.19	0.09	0.12	454
97	0.39	0.41	0.40	345
98	0.43	0.40	0.42	125
99	0.54	0.39	0.45	144
100	0.27	0.11	0.15	279
101	0.47	0.09	0.15	99
102	0.79	0.55	0.65	553
103	0.71	0.20	0.32	123
104	0.22	0.07	0.10	542
105	0.17	0.05	0.08	542
106	0.32	0.14	0.20	118
107	0.58	0.40	0.47	73
108	0.29	0.14	0.18	191
109	0.30	0.12	0.17	180
110	0.47	0.17	0.25	121
111	0.39	0.22	0.28	41
112	0.12	0.02	0.04	254
113	0.17	0.10	0.13	146
114	0.66	0.34	0.45	279
115	0.35	0.12	0.18	245
116	0.51	0.23	0.31	102
117	0.57	0.36	0.44	469
118	0.23	0.03	0.06	248
119	0.38	0.23	0.29	98
120	0.57	0.41	0.48	105
121	0.18	0.05	0.08	164
122	0.39	0.28	0.33	95
123	0.47	0.37	0.42	208
124	0.53	0.35	0.42	85
125	0.65	0.51	0.57	98
126	0.23	0.07	0.11	41
127	0.79	0.39	0.52	431
128	0.57	0.29	0.38	111
129	0.41	0.27	0.33	74
130	0.44	0.22	0.30	116
131	0.54	0.31	0.39	126
132	0.37	0.37	0.37	270
133	0.35	0.20	0.25	35
134	0.41	0.23	0.30	64
135	0.62	0.65	0.61	212

135	0.92	0.93	0.87	475
136	0.93	0.75	0.83	345
137	0.41	0.18	0.25	174
138	0.34	0.28	0.31	183
139	0.76	0.27	0.39	454
140	0.86	0.64	0.73	302
141	0.62	0.39	0.48	82
142	0.74	0.80	0.77	82
143	0.42	0.35	0.38	98
144	0.87	0.64	0.73	137
145	0.48	0.21	0.30	412
146	0.24	0.10	0.14	224
147	0.26	0.08	0.12	153
148	0.62	0.64	0.63	64
149	0.54	0.38	0.45	68
150	0.18	0.07	0.10	126
151	0.15	0.07	0.10	202
152	0.21	0.15	0.18	39
153	0.55	0.33	0.41	36
154	0.69	0.43	0.53	136
155	0.21	0.04	0.07	212
156	0.50	0.43	0.46	51
157	0.49	0.48	0.49	94
158	0.18	0.07	0.10	286
159	0.49	0.49	0.49	350
160	0.41	0.32	0.36	22
161	0.15	0.07	0.10	120
162	0.24	0.14	0.18	144
163	0.59	0.55	0.57	119
164	0.50	0.19	0.28	42
165	0.85	0.73	0.79	361
166	0.35	0.14	0.20	206
167	0.60	0.17	0.27	87
168	0.39	0.26	0.31	112
169	0.33	0.10	0.15	298
170	0.45	0.15	0.23	191
171	0.55	0.44	0.49	91
172	0.88	0.50	0.64	100
173	0.22	0.07	0.10	167
174	0.47	0.41	0.44	344
175	0.14	0.04	0.06	76
176	0.25	0.09	0.13	198
177	0.42	0.19	0.26	127
178	0.38	0.22	0.28	102
179	0.67	0.26	0.37	31
180	0.57	0.47	0.52	139
181	0.74	0.41	0.53	63
182	0.47	0.25	0.32	367
183	0.49	0.64	0.55	67
184	0.00	0.00	0.00	46
185	0.54	0.11	0.18	381
186	0.56	0.17	0.26	29
187	0.31	0.16	0.21	111
188	0.22	0.12	0.15	121
189	0.17	0.01	0.02	82
190	0.50	0.36	0.42	118
191	0.79	0.34	0.47	77
192	0.61	0.53	0.57	118
193	0.38	0.07	0.12	159
194	0.44	0.24	0.31	269
195	0.78	0.58	0.67	81
196	0.44	0.20	0.27	299
197	0.09	0.02	0.03	47
198	0.79	0.70	0.74	47
199	0.36	0.16	0.22	62
200	0.00	0.00	0.00	24
201	0.60	0.30	0.40	86
202	0.52	0.15	0.24	308
203	0.77	0.45	0.57	321
204	0.00	0.00	0.00	67
205	0.21	0.21	0.21	28
206	0.51	0.44	0.47	59
207	0.22	0.19	0.21	215

207	0.55	0.15	0.27	27
208	0.65	0.58	0.61	53
209	0.06	0.01	0.01	274
210	0.25	0.12	0.17	8
211	0.53	0.38	0.44	95
212	0.30	0.25	0.27	129
213	0.58	0.44	0.50	34
214	0.35	0.30	0.33	89
215	0.51	0.37	0.43	67
216	0.42	0.20	0.27	25
217	0.67	0.53	0.59	109
218	0.45	0.10	0.16	134
219	0.60	0.47	0.53	70
220	0.63	0.36	0.46	67
221	0.50	0.18	0.26	79
222	0.57	0.40	0.47	50
223	0.75	0.71	0.73	93
224	0.33	0.22	0.27	94
225	0.10	0.06	0.07	180
226	0.22	0.09	0.13	79
227	0.35	0.38	0.36	64
228	0.25	0.04	0.07	50
229	0.72	0.55	0.62	53
230	0.55	0.27	0.36	44
231	0.18	0.26	0.21	61
232	0.38	0.51	0.43	49
233	0.82	0.75	0.78	72
234	0.23	0.17	0.19	233
235	0.34	0.06	0.10	166
236	0.84	0.45	0.58	58
237	0.12	0.01	0.02	152
238	0.61	0.23	0.33	302
239	0.49	0.48	0.48	42
240	0.56	0.42	0.48	269
241	0.25	0.07	0.11	54
242	0.20	0.02	0.03	162
243	0.53	0.39	0.45	23
244	0.78	0.38	0.51	66
245	0.14	0.05	0.07	40
246	0.69	0.49	0.58	73
247	0.86	0.55	0.67	78
248	0.28	0.13	0.18	131
249	0.37	0.13	0.20	82
250	0.86	0.56	0.68	57
251	0.21	0.01	0.03	296
252	0.49	0.20	0.28	87
253	0.25	0.15	0.18	96
254	0.52	0.26	0.34	280
255	0.36	0.17	0.23	24
256	0.38	0.14	0.20	88
257	0.30	0.43	0.35	7
258	0.26	0.12	0.16	136
259	0.76	0.47	0.58	73
260	0.62	0.51	0.56	268
261	0.82	0.82	0.82	11
262	0.67	0.43	0.52	82
263	0.33	0.40	0.36	5
264	0.36	0.25	0.30	108
265	0.80	0.47	0.60	78
266	0.24	0.09	0.13	69
267	0.41	0.11	0.18	80
268	0.57	0.29	0.38	28
269	0.22	0.05	0.08	44
270	0.71	0.57	0.63	42
271	0.41	0.29	0.34	114
272	0.15	0.07	0.09	59
273	0.43	0.20	0.27	130
274	0.23	0.19	0.21	48
275	0.28	0.10	0.14	227
276	0.77	0.61	0.68	75
277	0.65	0.38	0.48	68
278	0.66	0.51	0.57	143
279	0.26	0.15	0.19	72

277	0.25	0.15	0.15	75
280	0.81	0.44	0.57	78
281	0.75	0.49	0.59	61
282	0.00	0.00	0.00	61
283	0.69	0.21	0.32	52
284	0.25	0.12	0.17	24
285	0.11	0.02	0.03	125
286	0.35	0.17	0.23	138
287	0.17	0.05	0.07	171
288	0.72	0.31	0.43	157
289	0.62	0.33	0.43	30
290	0.17	0.07	0.10	30
291	0.74	0.31	0.44	64
292	0.15	0.22	0.18	9
293	0.36	0.13	0.19	123
294	0.55	0.34	0.42	35
295	0.20	0.05	0.07	22
296	0.40	0.44	0.42	184
297	0.63	0.29	0.40	140
298	0.39	0.10	0.16	224
299	0.80	0.44	0.57	97
300	0.12	0.03	0.05	65
301	0.14	0.05	0.07	44
302	0.50	0.32	0.39	38
303	0.45	0.26	0.32	98
304	0.52	0.42	0.46	31
305	0.32	0.25	0.28	235
306	0.99	0.76	0.86	249
307	0.31	0.23	0.26	247
308	0.47	0.29	0.36	122
309	0.24	0.06	0.10	230
310	0.40	0.13	0.19	166
311	0.19	0.10	0.13	40
312	0.44	0.24	0.31	17
313	0.27	0.19	0.23	36
314	0.53	0.39	0.44	109
315	0.00	0.00	0.00	67
316	0.72	0.68	0.70	79
317	0.19	0.06	0.09	197
318	0.16	0.47	0.23	47
319	0.68	0.32	0.44	222
320	0.63	0.44	0.52	27
321	0.70	0.57	0.63	207
322	0.74	0.29	0.41	240
323	0.32	0.06	0.09	215
324	0.47	0.29	0.36	120
325	0.44	0.16	0.24	130
326	0.82	0.64	0.72	28
327	0.12	0.15	0.13	166
328	0.76	0.49	0.59	45
329	0.51	0.48	0.49	180
330	0.23	0.11	0.15	62
331	0.28	0.25	0.26	105
332	0.96	0.56	0.71	39
333	0.50	1.00	0.67	4
334	0.61	0.40	0.48	113
335	0.64	0.29	0.40	78
336	0.21	0.08	0.11	51
337	0.40	0.18	0.25	147
338	0.00	0.00	0.00	135
339	0.30	0.37	0.33	27
340	0.12	0.01	0.02	79
341	0.95	0.63	0.76	30
342	0.25	0.06	0.09	54
343	0.59	0.20	0.30	195
344	0.30	0.33	0.31	39
345	0.86	0.67	0.75	9
346	0.78	0.65	0.71	86
347	0.31	0.11	0.17	44
348	0.76	0.28	0.40	185
349	0.83	0.53	0.65	66
350	0.33	0.33	0.33	3
351	0.52	0.29	0.27	25

352	0.51	0.31	0.39	216
353	0.52	0.33	0.41	42
354	0.67	0.33	0.44	6
355	0.07	0.67	0.12	3
356	0.20	0.07	0.11	14
357	0.42	0.35	0.39	31
358	0.39	0.09	0.14	204
359	0.00	0.00	0.00	211
360	0.48	0.17	0.26	184
361	0.29	0.18	0.22	108
362	0.00	0.00	0.00	54
363	0.31	0.18	0.23	56
364	0.32	0.09	0.14	97
365	0.41	0.17	0.24	72
366	0.00	0.00	0.00	12
367	0.60	0.22	0.32	185
368	0.20	0.02	0.03	193
369	0.17	0.06	0.09	34
370	0.57	0.27	0.37	164
371	0.67	0.67	0.67	18
372	0.29	0.15	0.20	65
373	0.42	0.25	0.31	20
374	0.00	0.00	0.00	29
375	0.52	0.34	0.41	71
376	0.12	0.01	0.01	164
377	0.53	0.14	0.22	185
378	0.12	0.12	0.12	24
379	0.26	0.25	0.25	52
380	0.08	0.02	0.03	57
381	0.29	0.03	0.06	59
382	0.33	0.02	0.03	117
383	0.65	0.56	0.60	39
384	0.65	0.44	0.53	125
385	0.27	0.12	0.16	130
386	0.66	0.58	0.62	74
387	0.66	0.54	0.59	35
388	0.52	0.57	0.55	21
389	0.49	0.13	0.20	175
390	0.00	0.00	0.00	54
391	0.00	0.00	0.00	29
392	0.46	0.10	0.16	63
393	0.33	0.38	0.36	34
394	0.75	0.63	0.69	38
395	0.50	0.20	0.29	15
396	0.25	0.20	0.22	10
397	0.33	0.12	0.18	49
398	0.67	0.05	0.09	169
399	0.33	0.30	0.32	33
400	0.56	0.45	0.50	84
401	0.80	0.52	0.63	31
402	0.44	0.46	0.45	24
403	0.72	0.16	0.26	187
404	0.50	0.33	0.40	6
405	0.58	0.33	0.42	33
406	0.17	0.12	0.14	17
407	0.50	0.05	0.09	21
408	0.32	0.10	0.15	62
409	0.19	0.04	0.06	78
410	0.67	0.48	0.56	147
411	0.35	0.23	0.27	31
412	0.00	0.00	0.00	14
413	0.69	0.19	0.30	103
414	0.88	0.61	0.72	36
415	0.51	0.59	0.54	68
416	0.16	0.07	0.10	43
417	0.44	0.37	0.40	73
418	0.89	0.52	0.65	62
419	0.00	0.00	0.00	97
420	0.08	0.05	0.06	42
421	0.73	0.39	0.51	41
422	0.33	0.05	0.09	38
423	0.50	0.02	0.05	26

424	0.20	0.08	0.11	13
425	0.33	0.12	0.18	24
426	0.00	0.00	0.00	3
427	0.40	0.22	0.29	94
428	0.37	0.16	0.22	151
429	0.73	0.59	0.65	63
430	0.25	0.07	0.12	40
431	0.47	0.31	0.37	49
432	0.00	0.00	0.00	34
433	0.87	0.70	0.78	37
434	0.39	0.32	0.35	34
435	1.00	1.00	1.00	1
436	0.14	0.03	0.06	29
437	0.17	0.14	0.15	50
438	0.00	0.00	0.00	104
439	0.00	0.00	0.00	29
440	0.20	0.04	0.07	23
441	0.00	0.00	0.00	46
442	0.27	0.10	0.15	39
443	0.61	0.30	0.40	56
444	0.61	0.35	0.44	80
445	0.14	0.07	0.09	30
446	0.10	0.07	0.08	30
447	0.33	0.24	0.28	37
448	0.62	0.13	0.21	39
449	0.19	0.10	0.13	83
450	0.67	0.26	0.38	23
451	0.31	0.56	0.40	9
452	0.47	0.16	0.24	44
453	0.55	0.10	0.17	166
454	0.68	0.47	0.56	32
455	0.29	0.11	0.16	53
456	0.60	0.51	0.55	41
457	0.14	0.08	0.10	78
458	0.25	0.05	0.09	133
459	0.33	0.12	0.18	25
460	0.50	0.17	0.26	103
461	0.62	0.09	0.16	53
462	0.38	0.30	0.33	30
463	1.00	0.25	0.40	4
464	0.17	0.05	0.07	66
465	0.25	0.11	0.15	47
466	0.33	0.19	0.24	31

1

469	0.59	0.21	0.31	155
-----	------	------	------	-----

▼ LOGISTIC REGRESSION FOR HYPER PARAMETER TUNING

```

1 from sklearn.model_selection import GridSearchCV
2 from scipy.stats import randint as sp_randint
3 from sklearn.model_selection import cross_val_score
4 from sklearn.linear_model import SGDClassifier
5
6 start = datetime.now()
7
8 grid_params = [{ 'estimator__C' : [10**x for x in range(-4,3)]]}
9 logistic_regression = OneVsRestClassifier(LogisticRegression(penalty='l1', solver='liblinear'))
10 clf=GridSearchCV(logistic_regression,grid_params,scoring='f1_micro',cv=3)
11 clf.fit(x_train_multilabel_bow, y_train)
12 predictions_2 = clf.predict(x_test_multilabel_bow)
13 print("Accuracy :",metrics.accuracy_score(y_test, predictions_2))
14 print("Hamming loss ",metrics.hamming_loss(y_test,predictions_2))
15
16
17 precision = precision_score(y_test, predictions_2, average='micro')
18 recall = recall_score(y_test, predictions_2, average='micro')
19 f1 = f1_score(y_test, predictions_2, average='micro')

```



```
19 f1 = f1_score(y_test, predictions_2, average='micro')
20
21 print("Micro-average quality numbers")
22 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
23
24 precision = precision_score(y_test, predictions_2, average='macro')
25 recall = recall_score(y_test, predictions_2, average='macro')
26 f1 = f1_score(y_test, predictions_2, average='macro')
27
28 print("Macro-average quality numbers")
29 print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))
30
31 print(metrics.classification_report(y_test, predictions_2))
32 print("Time taken to run this cell :", datetime.now() - start)
```



Accuracy : 0.23587142857142857

Hamming loss 0.0029417714285714285

Micro-average quality numbers

Precision: 0.6778, Recall: 0.5207, F1-measure: 0.5890

Macro-average quality numbers

Precision: 0.4123, Recall: 0.3021, F1-measure: 0.3397

	precision	recall	f1-score	support
0	0.98	0.97	0.97	42802
1	0.42	0.26	0.32	1764
2	0.56	0.38	0.46	942
3	0.34	0.19	0.25	6539
4	0.48	0.41	0.45	2540
5	0.65	0.54	0.59	2156
6	0.66	0.50	0.57	1990
7	0.31	0.17	0.22	611
8	0.55	0.24	0.34	324
9	0.76	0.78	0.77	2335
10	0.48	0.40	0.44	736
11	0.40	0.29	0.34	1199
12	0.55	0.40	0.46	842
13	0.54	0.36	0.43	770
14	0.61	0.56	0.58	975
15	0.37	0.21	0.27	523
16	0.40	0.27	0.32	143
17	0.71	0.56	0.62	428
18	0.71	0.53	0.61	552
19	0.67	0.55	0.60	708
20	0.81	0.61	0.69	89
21	0.48	0.25	0.33	219
22	0.43	0.27	0.33	662
23	0.23	0.13	0.17	1965
24	0.50	0.28	0.36	1046
25	0.44	0.29	0.35	488
26	0.47	0.31	0.37	458
27	0.52	0.38	0.44	1193
28	0.68	0.48	0.56	1492
29	0.23	0.16	0.19	354
30	0.49	0.41	0.45	159
31	0.48	0.29	0.36	157
32	0.31	0.19	0.24	740
33	0.54	0.40	0.46	1337
34	0.57	0.32	0.41	226
35	0.66	0.49	0.57	535
36	0.46	0.28	0.35	315
37	0.33	0.20	0.25	167
38	0.74	0.64	0.68	1074
39	0.44	0.26	0.33	336
40	0.48	0.37	0.42	312
41	0.22	0.13	0.16	682
42	0.74	0.57	0.65	265
43	0.77	0.61	0.68	478
44	0.63	0.36	0.46	598
45	0.45	0.27	0.34	1054
46	0.50	0.30	0.37	253
47	0.27	0.17	0.21	936
48	0.32	0.23	0.27	133
49	0.38	0.18	0.24	152
50	0.43	0.37	0.40	943
51	0.17	0.11	0.13	243
52	0.43	0.28	0.34	135
53	0.98	0.90	0.94	969
54	0.24	0.13	0.17	801
55	0.58	0.64	0.61	50
56	0.56	0.36	0.44	142
57	0.49	0.27	0.35	125
58	0.24	0.13	0.17	39
59	0.38	0.36	0.37	36
60	0.51	0.33	0.40	205
61	0.29	0.21	0.24	354
62	0.28	0.20	0.24	266
63	0.58	0.17	0.27	178

63	0.90	0.69	0.78	163
64	0.53	0.46	0.49	46
65	0.36	0.20	0.26	269
66	0.78	0.73	0.75	133
67	0.32	0.20	0.24	577
68	0.23	0.09	0.13	212
69	0.82	0.66	0.73	120
70	0.29	0.16	0.21	258
71	0.19	0.12	0.14	354
72	0.72	0.58	0.64	159
73	0.50	0.27	0.35	89
74	0.37	0.29	0.32	125
75	0.26	0.20	0.23	94
76	0.53	0.36	0.43	456
77	0.42	0.38	0.40	712
78	0.29	0.11	0.16	299
79	0.28	0.17	0.21	53
80	0.43	0.32	0.37	275
81	0.53	0.23	0.32	623
82	0.53	0.39	0.45	75
83	0.12	0.07	0.09	27
84	0.62	0.34	0.44	230
85	0.49	0.36	0.41	177
86	0.45	0.33	0.38	30
87	0.60	0.69	0.64	655
88	0.14	0.10	0.12	126
89	0.50	0.37	0.42	422
90	0.15	0.04	0.06	130
91	0.82	0.86	0.84	451
92	0.29	0.06	0.11	77
93	0.13	0.05	0.07	461
94	0.71	0.38	0.50	104
95	0.18	0.09	0.12	454
96	0.39	0.37	0.38	345
97	0.44	0.44	0.44	125
98	0.52	0.33	0.41	144
99	0.20	0.06	0.09	279
100	0.49	0.18	0.26	99
101	0.78	0.62	0.69	553
102	0.52	0.24	0.32	123
103	0.14	0.08	0.10	542
104	0.21	0.14	0.17	542
105	0.33	0.11	0.16	118
106	0.52	0.40	0.45	73
107	0.27	0.14	0.19	191
108	0.29	0.12	0.17	180
109	0.47	0.27	0.35	121
110	0.28	0.17	0.21	41
111	0.14	0.06	0.08	254
112	0.19	0.13	0.15	146
113	0.60	0.36	0.45	279
114	0.33	0.17	0.22	245
115	0.49	0.21	0.29	102
116	0.56	0.39	0.46	469
117	0.20	0.06	0.09	248
118	0.37	0.24	0.29	98
119	0.56	0.47	0.51	105
120	0.23	0.11	0.15	164
121	0.48	0.31	0.37	95
122	0.48	0.27	0.35	208
123	0.59	0.39	0.47	85
124	0.60	0.53	0.57	98
125	0.27	0.15	0.19	41
126	0.79	0.34	0.47	431
127	0.59	0.33	0.43	111
128	0.39	0.22	0.28	74
129	0.42	0.20	0.27	116
130	0.52	0.42	0.47	126
131	0.35	0.40	0.37	270
132	0.36	0.29	0.32	35
133	0.34	0.27	0.30	64
134	0.58	0.66	0.67	712

135	0.96	0.79	0.87	345
136	0.96	0.79	0.87	345
137	0.41	0.18	0.25	174
138	0.32	0.27	0.29	183
139	0.73	0.39	0.50	454
140	0.81	0.72	0.76	302
141	0.68	0.34	0.46	82
142	0.77	0.77	0.77	82
143	0.50	0.37	0.42	98
144	0.86	0.74	0.79	137
145	0.42	0.25	0.31	412
146	0.21	0.09	0.12	224
147	0.29	0.13	0.18	153
148	0.60	0.62	0.61	64
149	0.61	0.41	0.49	68
150	0.20	0.08	0.11	126
151	0.15	0.09	0.12	202
152	0.26	0.28	0.27	39
153	0.56	0.39	0.46	36
154	0.50	0.46	0.48	136
155	0.20	0.06	0.09	212
156	0.49	0.39	0.43	51
157	0.46	0.44	0.45	94
158	0.20	0.08	0.12	286
159	0.52	0.45	0.48	350
160	0.45	0.41	0.43	22
161	0.10	0.07	0.08	120
162	0.24	0.12	0.16	144
163	0.59	0.50	0.55	119
164	0.37	0.26	0.31	42
165	0.82	0.81	0.81	361
166	0.28	0.15	0.19	206
167	0.60	0.29	0.39	87
168	0.29	0.29	0.29	112
169	0.24	0.11	0.15	298
170	0.40	0.23	0.29	191
171	0.44	0.48	0.46	91
172	0.67	0.49	0.57	100
173	0.10	0.05	0.07	167
174	0.47	0.34	0.40	344
175	0.07	0.04	0.05	76
176	0.17	0.09	0.12	198
177	0.40	0.24	0.30	127
178	0.33	0.20	0.25	102
179	0.65	0.35	0.46	31
180	0.51	0.46	0.48	139
181	0.65	0.56	0.60	63
182	0.46	0.24	0.32	367
183	0.51	0.70	0.59	67
184	0.00	0.00	0.00	46
185	0.57	0.16	0.25	381
186	0.90	0.31	0.46	29
187	0.33	0.26	0.29	111
188	0.26	0.17	0.20	121
189	0.21	0.04	0.06	82
190	0.58	0.42	0.49	118
191	0.70	0.52	0.60	77
192	0.59	0.53	0.56	118
193	0.26	0.09	0.14	159
194	0.43	0.38	0.41	269
195	0.77	0.65	0.71	81
196	0.41	0.31	0.35	299
197	0.31	0.09	0.13	47
198	0.76	0.74	0.75	47
199	0.40	0.26	0.31	62
200	0.31	0.17	0.22	24
201	0.53	0.27	0.36	86
202	0.53	0.22	0.31	308
203	0.67	0.56	0.61	321
204	0.08	0.03	0.04	67
205	0.16	0.18	0.17	28
206	0.58	0.49	0.53	59
207	0.26	0.22	0.28	215

207	0.30	0.25	0.20	27
208	0.85	0.64	0.73	53
209	0.11	0.09	0.10	274
210	0.12	0.12	0.12	8
211	0.48	0.26	0.34	95
212	0.29	0.22	0.25	129
213	0.64	0.47	0.54	34
214	0.29	0.34	0.31	89
215	0.55	0.36	0.43	67
216	0.36	0.20	0.26	25
217	0.62	0.63	0.63	109
218	0.40	0.15	0.22	134
219	0.62	0.49	0.54	70
220	0.53	0.40	0.46	67
221	0.39	0.22	0.28	79
222	0.50	0.40	0.44	50
223	0.72	0.72	0.72	93
224	0.27	0.26	0.26	94
225	0.14	0.05	0.07	180
226	0.10	0.06	0.08	79
227	0.31	0.36	0.33	64
228	0.15	0.06	0.09	50
229	0.65	0.64	0.65	53
230	0.52	0.34	0.41	44
231	0.24	0.21	0.23	61
232	0.39	0.39	0.39	49
233	0.73	0.75	0.74	72
234	0.20	0.14	0.16	233
235	0.33	0.10	0.16	166
236	0.67	0.52	0.58	58
237	0.15	0.07	0.09	152
238	0.52	0.27	0.35	302
239	0.49	0.55	0.52	42
240	0.46	0.34	0.39	269
241	0.26	0.11	0.16	54
242	0.21	0.07	0.11	162
243	0.50	0.39	0.44	23
244	0.65	0.30	0.41	66
245	0.19	0.10	0.13	40
246	0.67	0.59	0.63	73
247	0.84	0.62	0.71	78
248	0.35	0.17	0.23	131
249	0.26	0.11	0.16	82
250	0.67	0.58	0.62	57
251	0.19	0.06	0.09	296
252	0.39	0.18	0.25	87
253	0.27	0.16	0.20	96
254	0.44	0.14	0.21	280
255	0.25	0.21	0.23	24
256	0.31	0.19	0.24	88
257	0.23	0.43	0.30	7
258	0.24	0.15	0.19	136
259	0.72	0.56	0.63	73
260	0.63	0.43	0.51	268
261	1.00	0.82	0.90	11
262	0.69	0.51	0.59	82
263	0.50	0.80	0.62	5
264	0.37	0.24	0.29	108
265	0.67	0.50	0.57	78
266	0.16	0.10	0.12	69
267	0.34	0.12	0.18	80
268	0.37	0.25	0.30	28
269	0.12	0.05	0.07	44
270	0.66	0.60	0.62	42
271	0.47	0.25	0.32	114
272	0.09	0.08	0.09	59
273	0.27	0.21	0.24	130
274	0.26	0.21	0.23	48
275	0.17	0.11	0.13	227
276	0.72	0.77	0.74	75
277	0.71	0.32	0.44	68
278	0.59	0.48	0.53	143
279	0.16	0.15	0.16	72

277	0.10	0.10	0.10	70
280	0.69	0.56	0.62	78
281	0.73	0.59	0.65	61
282	0.06	0.02	0.03	61
283	0.43	0.25	0.32	52
284	0.20	0.08	0.12	24
285	0.06	0.02	0.02	125
286	0.35	0.19	0.24	138
287	0.18	0.09	0.12	171
288	0.68	0.45	0.54	157
289	0.52	0.40	0.45	30
290	0.18	0.10	0.13	30
291	0.77	0.31	0.44	64
292	0.14	0.22	0.17	9
293	0.20	0.12	0.15	123
294	0.45	0.29	0.35	35
295	0.25	0.09	0.13	22
296	0.41	0.46	0.43	184
297	0.64	0.21	0.32	140
298	0.38	0.19	0.25	224
299	0.63	0.45	0.53	97
300	0.07	0.03	0.04	65
301	0.11	0.05	0.06	44
302	0.43	0.26	0.33	38
303	0.37	0.18	0.24	98
304	0.48	0.48	0.48	31
305	0.38	0.25	0.30	235
306	0.98	0.93	0.95	249
307	0.40	0.09	0.15	247
308	0.40	0.30	0.34	122
309	0.20	0.08	0.11	230
310	0.37	0.16	0.22	166
311	0.23	0.15	0.18	40
312	0.56	0.29	0.38	17
313	0.27	0.19	0.23	36
314	0.55	0.33	0.41	109
315	0.00	0.00	0.00	67
316	0.63	0.58	0.61	79
317	0.08	0.02	0.03	197
318	0.27	0.53	0.36	47
319	0.62	0.35	0.45	222
320	0.47	0.56	0.51	27
321	0.68	0.56	0.61	207
322	0.69	0.38	0.49	240
323	0.26	0.18	0.21	215
324	0.42	0.33	0.37	120
325	0.42	0.23	0.30	130
326	0.68	0.54	0.60	28
327	0.15	0.16	0.15	166
328	0.66	0.56	0.60	45
329	0.45	0.45	0.45	180
330	0.17	0.08	0.11	62
331	0.44	0.23	0.30	105
332	0.81	0.67	0.73	39
333	0.67	1.00	0.80	4
334	0.55	0.28	0.37	113
335	0.39	0.40	0.39	78
336	0.27	0.14	0.18	51
337	0.42	0.17	0.24	147
338	0.07	0.02	0.03	135
339	0.29	0.33	0.31	27
340	0.29	0.13	0.18	79
341	0.92	0.73	0.81	30
342	0.26	0.15	0.19	54
343	0.59	0.23	0.33	195
344	0.35	0.31	0.33	39
345	0.75	0.67	0.71	9
346	0.75	0.78	0.77	86
347	0.39	0.20	0.27	44
348	0.68	0.39	0.50	185
349	0.68	0.61	0.64	66
350	1.00	0.67	0.80	3
351	0.20	0.21	0.25	25

352	0.57	0.42	0.48	216
353	0.41	0.38	0.40	42
354	0.67	0.33	0.44	6
355	0.06	0.33	0.10	3
356	0.22	0.14	0.17	14
357	0.34	0.35	0.35	31
358	0.31	0.06	0.10	204
359	0.12	0.01	0.02	211
360	0.38	0.19	0.25	184
361	0.29	0.20	0.24	108
362	0.06	0.02	0.03	54
363	0.21	0.18	0.19	56
364	0.21	0.10	0.14	97
365	0.29	0.21	0.24	72
366	0.17	0.08	0.11	12
367	0.64	0.41	0.50	185
368	0.18	0.06	0.09	193
369	0.27	0.12	0.16	34
370	0.56	0.34	0.42	164
371	0.62	0.72	0.67	18
372	0.22	0.11	0.14	65
373	0.60	0.30	0.40	20
374	0.00	0.00	0.00	29
375	0.64	0.42	0.51	71
376	0.14	0.12	0.13	164
377	0.37	0.34	0.35	185
378	0.19	0.21	0.20	24
379	0.24	0.17	0.20	52
380	0.21	0.12	0.15	57
381	0.20	0.07	0.10	59
382	0.23	0.09	0.13	117
383	0.58	0.54	0.56	39
384	0.57	0.53	0.55	125
385	0.35	0.14	0.20	130
386	0.66	0.58	0.62	74
387	0.71	0.63	0.67	35
388	0.48	0.62	0.54	21
389	0.35	0.24	0.28	175
390	0.19	0.09	0.12	54
391	0.00	0.00	0.00	29
392	0.35	0.11	0.17	63
393	0.28	0.32	0.30	34
394	0.81	0.66	0.72	38
395	0.20	0.20	0.20	15
396	0.29	0.20	0.24	10
397	0.33	0.18	0.24	49
398	0.51	0.20	0.29	169
399	0.29	0.24	0.26	33
400	0.45	0.35	0.39	84
401	0.72	0.58	0.64	31
402	0.38	0.42	0.40	24
403	0.67	0.36	0.47	187
404	0.40	0.33	0.36	6
405	0.50	0.30	0.38	33
406	0.21	0.24	0.22	17
407	0.20	0.10	0.13	21
408	0.42	0.16	0.23	62
409	0.20	0.06	0.10	78
410	0.70	0.50	0.58	147
411	0.30	0.26	0.28	31
412	0.00	0.00	0.00	14
413	0.41	0.12	0.18	103
414	0.74	0.64	0.69	36
415	0.58	0.59	0.58	68
416	0.07	0.05	0.05	43
417	0.45	0.38	0.41	73
418	0.73	0.60	0.65	62
419	0.04	0.02	0.03	97
420	0.03	0.02	0.03	42
421	0.63	0.46	0.54	41
422	0.07	0.05	0.06	38
423	0.00	0.00	0.00	26

424	0.14	0.08	0.10	13
425	0.42	0.33	0.37	24
426	0.10	0.33	0.15	3
427	0.37	0.23	0.29	94
428	0.38	0.22	0.28	151
429	0.60	0.54	0.57	63
430	0.09	0.05	0.06	40
431	0.53	0.41	0.46	49
432	0.00	0.00	0.00	34
433	0.78	0.76	0.77	37
434	0.36	0.29	0.32	34
435	1.00	1.00	1.00	1
436	0.13	0.07	0.09	29
437	0.20	0.18	0.19	50
438	0.00	0.00	0.00	104
439	0.00	0.00	0.00	29
440	0.33	0.22	0.26	23
441	0.05	0.02	0.03	46
442	0.12	0.08	0.09	39
443	0.57	0.30	0.40	56
444	0.54	0.49	0.51	80
445	0.09	0.03	0.05	30
446	0.11	0.10	0.11	30
447	0.32	0.30	0.31	37
448	0.19	0.10	0.13	39
449	0.21	0.14	0.17	83
450	0.47	0.39	0.43	23
451	0.40	0.67	0.50	9
452	0.38	0.20	0.26	44
453	0.48	0.36	0.41	166
454	0.64	0.56	0.60	32
455	0.37	0.25	0.30	53
456	0.52	0.59	0.55	41
457	0.18	0.09	0.12	78
458	0.29	0.11	0.16	133
459	0.25	0.16	0.20	25
460	0.51	0.37	0.43	103
461	0.53	0.15	0.24	53
462	0.34	0.33	0.34	30
463	0.11	0.25	0.15	4
464	0.11	0.06	0.08	66
465	0.25	0.17	0.20	47
466	0.32	0.19	0.24	31
467	0.55	0.51	0.53	35
468	0.00	0.00	0.00	50
469	0.63	0.44	0.52	155
470	0.29	0.15	0.20	27
471	0.64	0.48	0.55	58
472	0.18	0.08	0.11	159
473	0.29	0.15	0.20	27
474	0.56	0.28	0.37	115
475	0.26	0.20	0.22	142

▼ CONCLUSION USING PRETTY TABLE

479	0.30	0.13	0.25	65
-----	------	------	------	----

```
1 pip install -U PTable
```

```
Collecting PTable
```

```
  Downloading https://files.pythonhosted.org/packages/ab/b3/b54301811173ca94119eb474634f120a49cd370f257d1aae5a4
```

```
Building wheels for collected packages: PTable
```

```
  Building wheel for PTable (setup.py) ... done
```

```
  Created wheel for PTable: filename=PTable-0.9.2-cp36-none-any.whl size=22908 sha256=411947626798f9652c176229b
```

```
  Stored in directory: /root/.cache/pip/wheels/22/cc/2e/55980bfe86393df3e9896146a01f6802978d09d7ebcba5ea56
```

```
Successfully built PTable
```

```
Installing collected packages: PTable
```

```
Successfully installed PTable-0.9.2
```

492	0.41	0.22	0.29	73
-----	------	------	------	----

```
1 from prettytable import PrettyTable
```

```
2 x= PrettyTable()
```



```

3 x.title = "TF-IDF Featurization"
4 x.field_names = ["Algorithm" , "Macro F1 Score Precision", "Micro F1 Score", "Hamming Loss"]
5 x.add_row(["Logistic Regression with OneVsRest",0.5473,0.7216,0.002708])
6 print(x)
7 y= PrettyTable()
8 y.add_row(["Logistic Regression with OneVsRest",0.4122,0.6777,0.002942])
9 y.add_row(["Linear SVM with OneVsRest",0.4318,0.7029,0.00289])
10 y.add_row(["Logistic Regression for Hyperparameter Tuning GridSearch",0.4123,0.6778,0.00290])
11 y.title = "CountVectorizer BIGRAM"
12 y.field_names = ["Algorithm" , "Macro F1 Score Precision", "Micro F1 Score", "Hamming Loss"]
13 print(y)

```

TF-IDF Featurization			
Algorithm	Macro F1 Score Precision	Micro F1 Score	Hamming Loss
Logistic Regression with OneVsRest	0.5473	0.7216	0.002708

CountVectorizer BIGRAM			
Algorithm	Macro F1 Score Precision	Micro F1 Score	Hamming Loss
Logistic Regression with OneVsRest	0.4122	0.6777	0.002942
Linear SVM with OneVsRest	0.4318	0.7029	0.00289
Logistic Regression for Hyperparameter Tuning GridSearch	0.4123	0.6778	0.00290

CONCLUSION : LOGISTIC REGRESSION WITH ONEVSREST HAVE GOOD PRECISION : 0.6777 AS MICRO F1 SCORE