STACKOVERFLOW TAG PREDICTION

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Supervisor:

Radhika Mamidi

Team Members:

Amit Sarkar(2019201040)
Akash Kumar(2010201046)

Table of Contents

1. Introduction

This project focusing on classifying questions asked on QnA forums, here we worked on Stack-Overflow(SO). Our main aim is to automatically classify a newly added question so that it can be assign it to some active user who have good rating for answering similar question on same topics.

For this purpose we use various Machine Learning(ML) models and will try to predic the tag of a question. Here, we mainly compare various ML algorithms models and tried various combination of parameters inorder to improve the accuracy of SO tag prediction.

2. Problem Statement

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

3. SCOPE OF THE PROJECT

This is a natural language processing and machine learning project, we will cover different types of evaluation metrics such as Accuracy, Confusion Matrix, Precision, Recall, Specificity, F1 score.

4. Costs

Hardware costs include the PC to interface with the server and cloud storage and other peripheral hardware required at both the server site as well as the reader site.

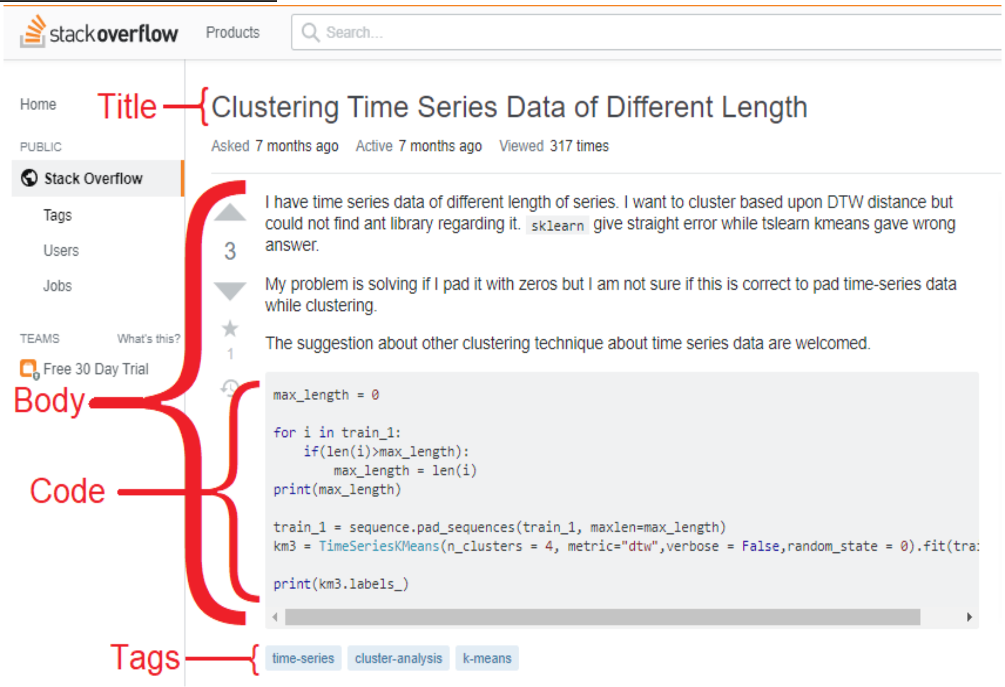
All additional costs have been development, software and programming time.

5. Data:

5.1 Data :

Each data item is a Question in SO forum. Below we is a pictorial view how the data is stored or presented in SO website. We can also find the relevant fields available in a data item. 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).

Pictorial View of Data in SO website:



Clustering Time Series Data of Different Length

Asked 7 months ago Active 7 months ago Viewed 317 times

I have time series data of different length of series. I want to cluster based upon DTW distance but could not find ant library regarding it. `sklearn` give straight error while tslearn kmeans gave wrong answer.

My problem is solving if I pad it with zeros but I am not sure if this is correct to pad time-series data while clustering.

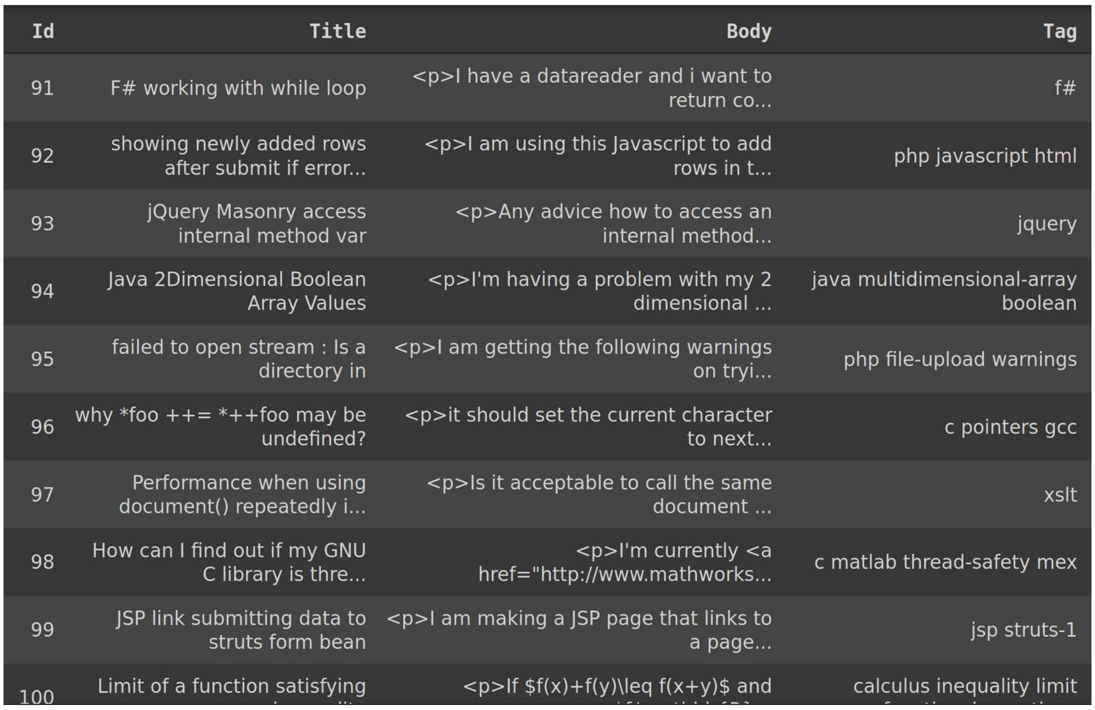
The suggestion about other clustering technique about time series data are welcomed.

```
max_length = 0

for i in train_1:
    if(len(i)>max_length):
        max_length = len(i)
print(max_length)

train_1 = sequence.pad_sequences(train_1, maxlen=max_length)
km3 = TimeSeriesKMeans(n_clusters = 4, metric="dtw",verbose = False,random_state = 0).fit(tra

print(km3.labels_)
```

time-series cluster-analysis k-means

fig: Pictorial View of Data in SO website.

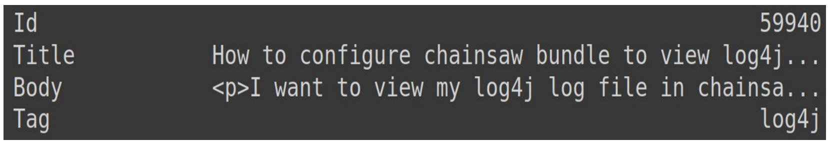
Pictorial View of Data after extracted and stored in pandas dataframe:



Id	Title	Body	Tag
91	F# working with while loop	<p>I have a datareader and i want to return co...	f#
92	showing newly added rows after submit if error...	<p>I am using this Javascript to add rows in t...	php javascript html
93	jQuery Masonry access internal method var	<p>Any advice how to access an internal method...	jquery
94	Java 2Dimensional Boolean Array Values	<p>I'm having a problem with my 2 dimensional ...	java multidimensional-array boolean
95	failed to open stream : Is a directory in	<p>I am getting the following warnings on tryi...	php file-upload warnings
96	why *foo ++= *++foo may be undefined?	<p>it should set the current character to next...	c pointers gcc
97	Performance when using document() repeatedly i...	<p>Is it acceptable to call the same document ...	xslt
98	How can I find out if my GNU C library is thre...	<p>I'm currently <a href="http://www.mathworks...	c matlab thread-safety mex
99	JSP link submitting data to struts form bean	<p>I am making a JSP page that links to a page...	jsp struts-1
100	Limit of a function satisfying	<p>If $f(x)+f(y)\leq f(x+y)$ and	calculus inequality limit

fig:Pictorial View of Data after extracted and stored in pandas dataframe

Single Data Point stored in pandas:



```
Id                                                   59940
Title        How to configure chainsaw bundle to view log4j...
Body         <p>I want to view my log4j log file in chainsa...
Tag                                                  log4j
```

5.2 Source of Data:

We used the SO data uploaded in a “Kaggle Competition” Even organized by FaceBook on 2012. It has seperate Test and Train data in csv format and each csv file is zipped seperately to reduce the size.
Reference Link : https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/

5.3 Format of Data:

Each data item is a question in SO and hence consists of 'Id', 'Title','Body','Tag' fields.
Train Data Format: consists of 'Id', 'Title','Body','Tag'
Test Data Format: 'Id', 'Title','Body' (as we have to predict the ‘Tag’)

Inside a body it might content <code> and <html> tags – consists of code and html formated text respectively. But these do required in our prediction - so we eliminate these.

Field Description:

- Id - Unique identifier for each question
- Title - The question's title
- Body - The body of the question
- Tags - The tags associated with the question and multiple tags might be present with ‘space’ seperation.

5.4 Size of Data:

Size of total data (train data + test data) – we have around 10GB data.
Number of data item (train data + test data) - 17 lakhs (Approx.)

Size of Train data - 7.3 GB
Number of data item in Train data - 12.6 lakhs

Size of Test data – 2.4GB
Number of data item in Test data - 4.5 lakh

6. Risks:

1. Kernel Crash - Since the amount of data is huge and we mostly use our ordinary day-to-day use laptop, with configuration 8 GB RAM, 512GB SSD SECONDARY MEMORY, its bit difficult to execute the ML program as it will crash definitely.

We use Google Colab, free, version, that provides 12 GB RAM and GPU.
But again with 12.6 lakhs data the kernel failed and it works best with 3 lakhs of data . And for simple less costly algorithms like LinearSVC it can runs upon 6 lakhs of data at max.

2. Since Google Colab allowing upto 15GB of space – it becomes difficult to keep intermediate temporary files

7. Library Used:

7.1. For data extraction, data processing and other internal works we used csv, pandas dataframe, numpy. Hence use the below libraries accordingly:

```
import csv
import pandas as pd
import numpy as np
```

7.2 For plotting and graphical representation of data we used matplot and sns.

```
import seaborn as sns
import matplotlib.pyplot as plt
```

7.3.For NLP related preprocessing we used python3 NLTK library.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
```

7.4 For other data prepocessing, vectorizing and other utility we used python3 SKLEARN library. For example:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import learning_curve
```

7.5. For Metrics and quantify the output result we used SKLEARN.

```
import make_scorer
from sklearn import metrics
from sklearn.metrics import f1_score,precision_score,recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import hamming_loss
from sklearn.model_selection import ShuffleSplitfrom sklearn.metrics
```

7.6 For standerd ML algorithms we used SKLEARN and other algorithm specific libraries.

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC
from sklearn.linear_model import Perceptron
```

8. Concept Used:

Main Concepts used:

- Mathematics in Data Science (Statistical Techniques)
- Regular Expression
- Sparse Matrix
- Natural Language Processing (Word Tokenizer, stop words Removing, Stemming, N-gram)
- Multi-Label Classification
- Machine Learning (BOW, Logistic Regression, SVM, One Vs Rest Classifier, Grid Search)

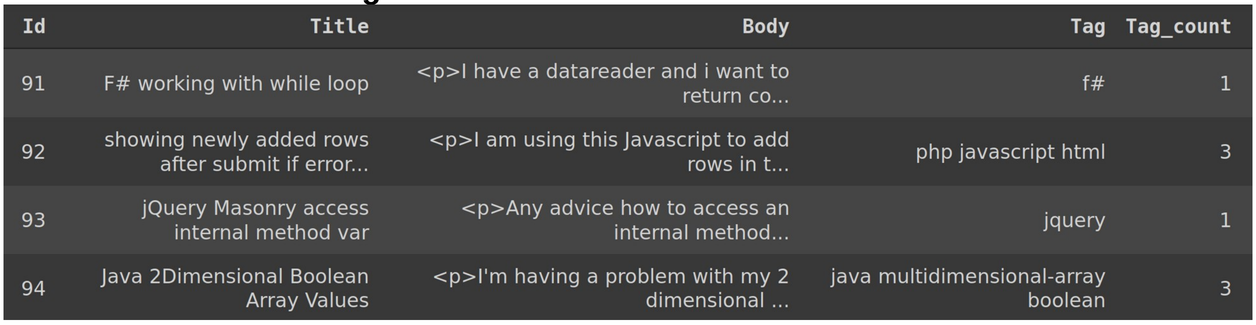
9. Strategy:

- Stage 1: Data Analysis -Tag Analysis And Ploting .
- Stage 2: Data Cleaning.
- Stage 3: Other Preporcessing and Merging
- Stage 4: Tag Reduction
- Stage 5: Splitting Data – split data into Train and Test set.
- Stage 6:
    - 6.1 Featurizing data with TF-IDF vectorizer.
    - 6.2 Featurizing Labels with TF-IDF vectorizer.
- Stage 7: Apply Various ML Algorithms – with default parameters.
- Stage 8: Select few specific ML Algorithms from stage 7 and apply Grid-Search

9.1 Data Analysis:

To get insight and know the data better we perform few processing and plotting. And this analysis help us to drive process the next operations as well as prediction stage.

9.1.1 count number of Tags:

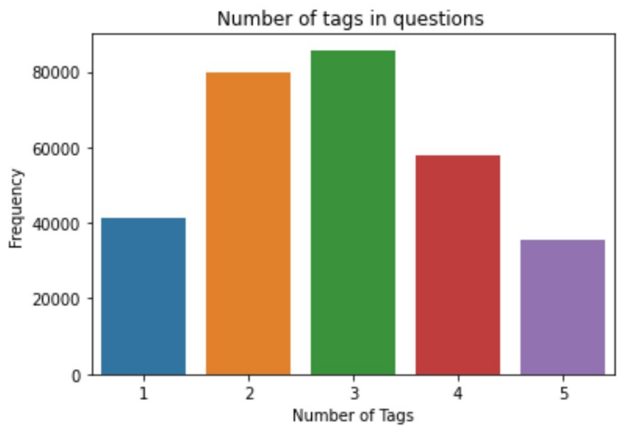


Id	Title	Body	Tag	Tag_count
91	F# working with while loop	<p>I have a datareader and i want to return co...	f#	1
92	showing newly added rows after submit if error...	<p>I am using this Javascript to add rows in t...	php javascript html	3
93	jQuery Masonry access internal method var	<p>Any advice how to access an internal method...	jquery	1
94	Java 2Dimensional Boolean Array Values	<p>I'm having a problem with my 2 dimensional ...	java multidimensional-array boolean	3

9.1.2 Find Max, Min and Average count of Tags for each of the records:

```
Maximum number of tags in a question:  5
Minimum number of tags in a question:  1
Average number of tags in a question:  2.88705
```

9.1.3 Frequency Plotting of Tags:



9.1.4 Frequency of Tag distributions:

Here we tried to see how often a tag can appears in questions. For this purpose we use frequency distribution plot. This will help us to select top most tags during prediction process.

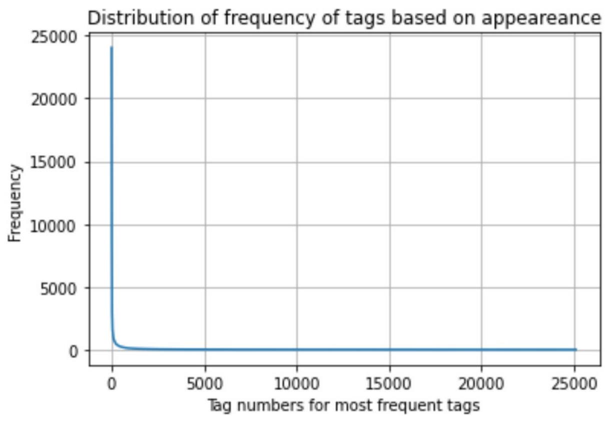


fig: Frequency Distribution of all tags

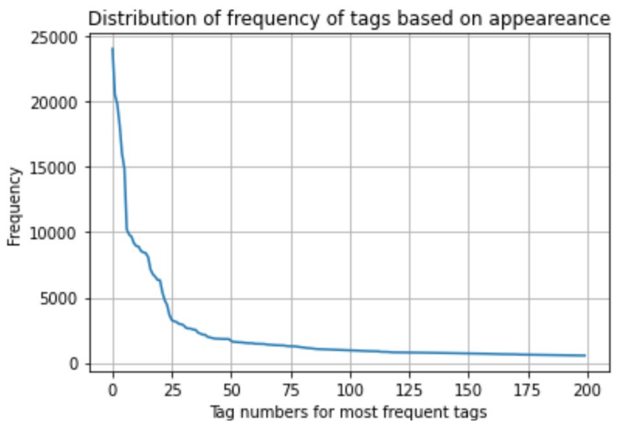


fig: Frequency Distribution of Top 200 Tags

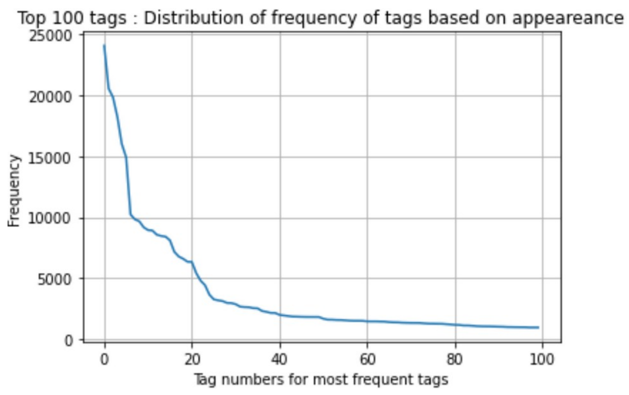


fig: Frequency Distribution of Top 100 Tags

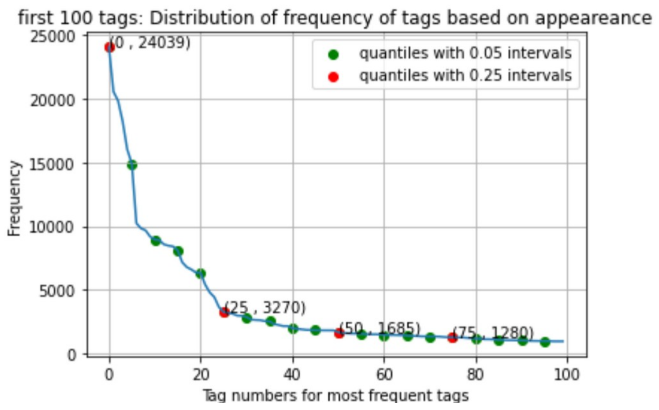


fig: Frequency Distribution of Top 100 Tags with percentile counts

9.1.5. Frequency of Top 20 tags:

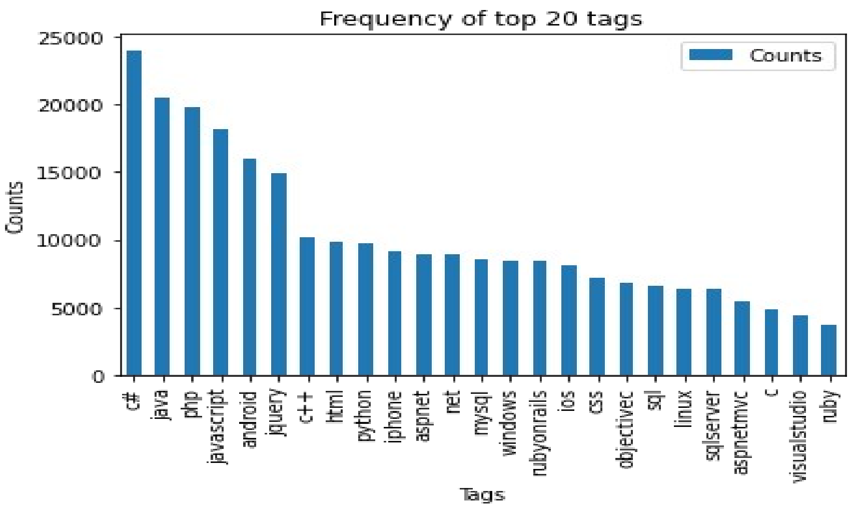


fig: Frequency of Top 20 tags

9.1.6 Tag Visulization using Word Cloud:

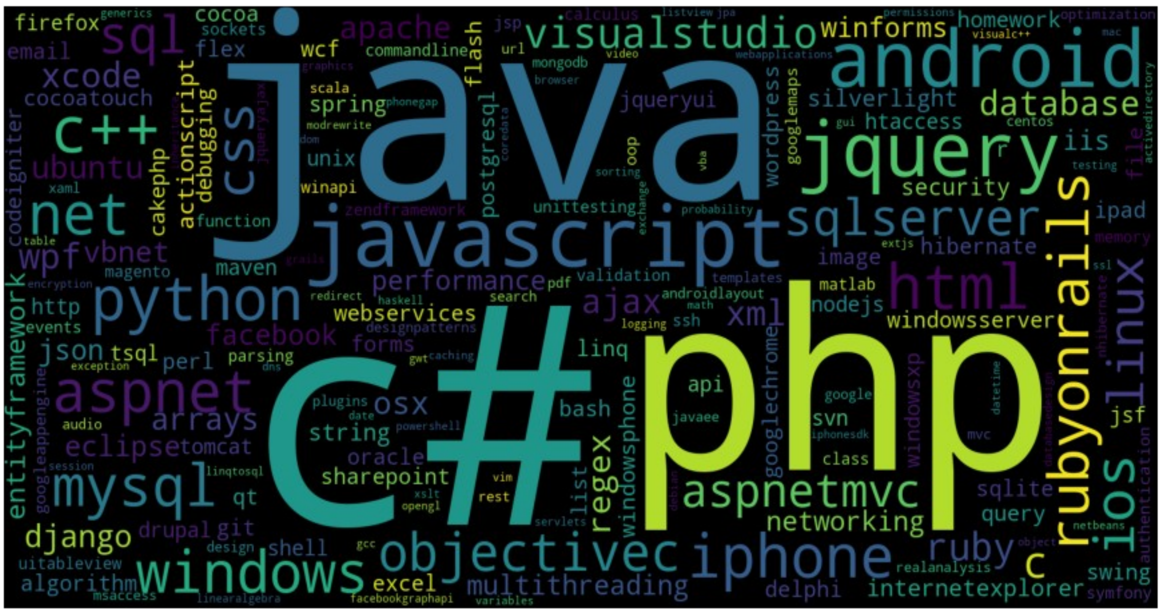


fig: Tag Visualization Using Word Cloud

9.1.7 Number of Questions Covered by a Tag Partially:

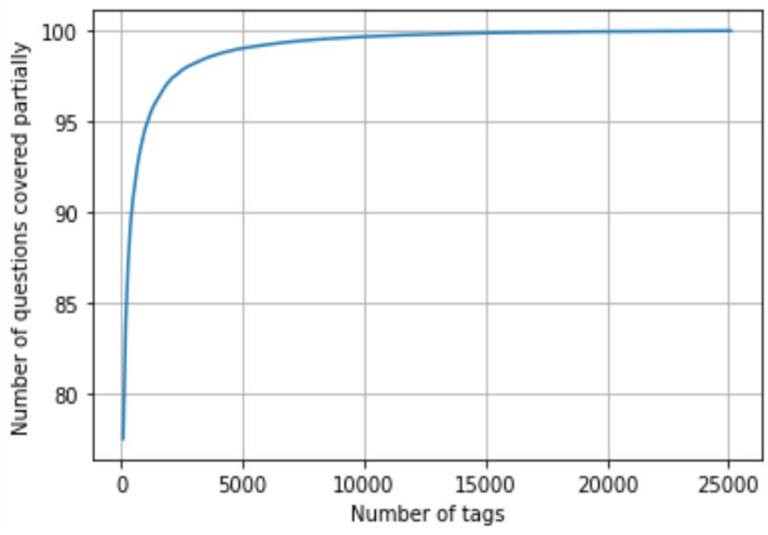


fig:Number of Questions Covered by a Tag Partially we can see from the plot calculation:

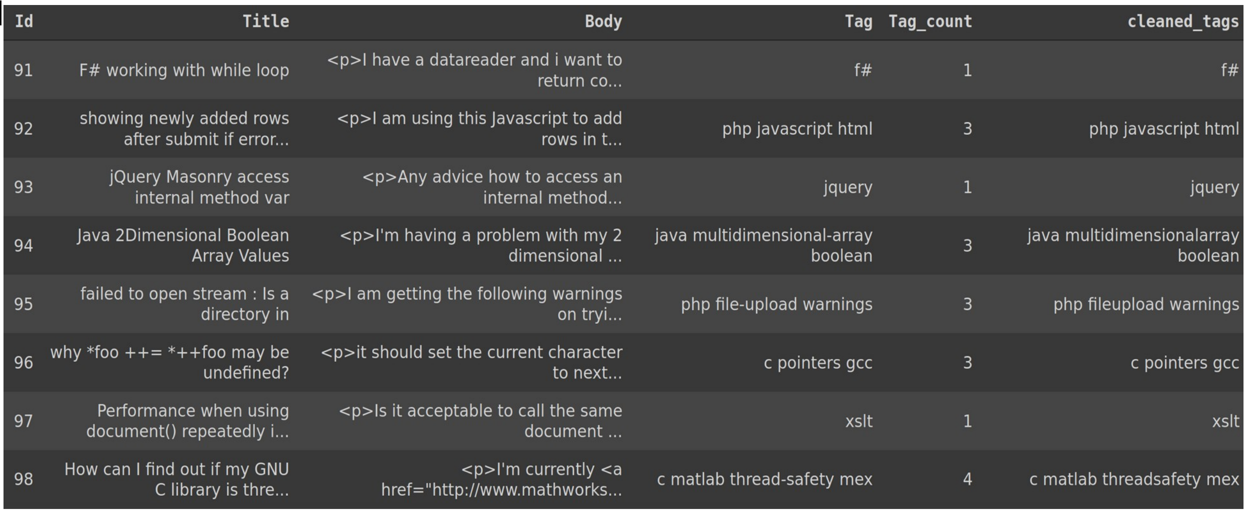
Questions covered by 100 tags partially: 94.601 %
Number of questions that are not covered by 100 tags : 16196 out of 300000

so if we consider top 100 tags out of 25118 tags (0.398120869495979 % of total tags), we can cover almost 95% questions asked in SO.

9.2 Data Cleaning:

We follow below steps to clean the raw data and make data appropriate for next steps.
9.2.1 convert all letters to LOWER case.
9.2.2 ignore non ASCII char
9.2.3 eliminate Numeric characters
9.2.4 eliminate multiple space
9.2.5 eliminate junk characters: junk_chars = "[]{}.-"

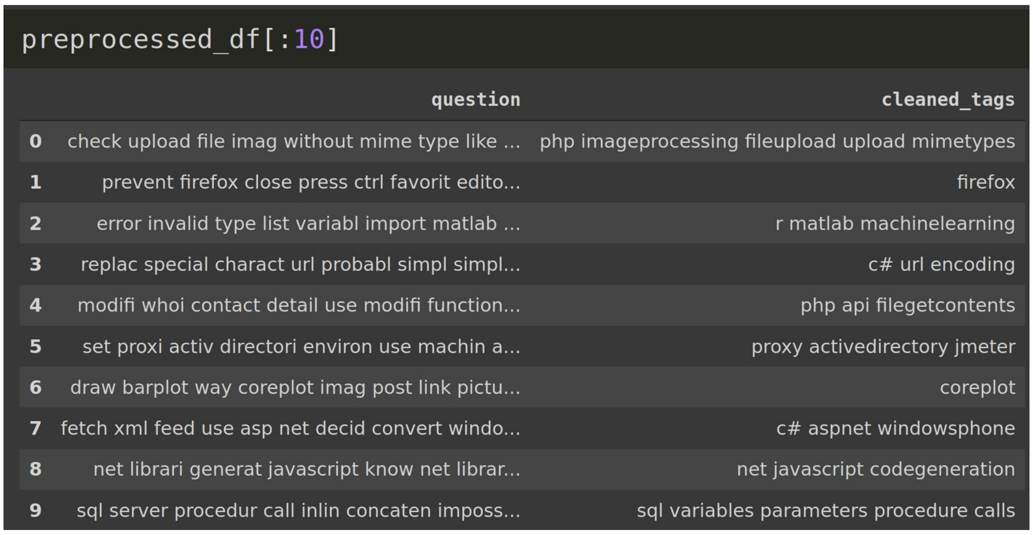
[clean data:]



Id	Title	Body	Tag	Tag_count	cleaned_tags
91	F# working with while loop	<p>I have a datareader and i want to return co...	f#	1	f#
92	showing newly added rows after submit if error...	<p>I am using this Javascript to add rows in t...	php javascript html	3	php javascript html
93	jQuery Masonry access internal method var	<p>Any advice how to access an internal method...	jquery	1	jquery
94	Java 2Dimensional Boolean Array Values	<p>I'm having a problem with my 2 dimensional ...	java multidimensional-array boolean	3	java multidimensionalarray boolean
95	failed to open stream : Is a directory in	<p>I am getting the following warnings on tryi...	php file-upload warnings	3	php fileupload warnings
96	why *foo ++= *++foo may be undefined?	<p>it should set the current character to next...	c pointers gcc	3	c pointers gcc
97	Performance when using document() repeatedly i...	<p>Is it acceptable to call the same document ...	xslt	1	xslt
98	How can I find out if my GNU C library is thre...	<p>I'm currently <a href="http://www.mathworks...	c matlab thread-safety mex	4	c matlab threadsafety mex

9.3 Other preprocessing and merging:

We focused on content of the question hence we create a new field “question” that merge cleaned and preprocessed “Title” and “Body” data stored in the dataframe:



```
preprocessed_df[:10]
```

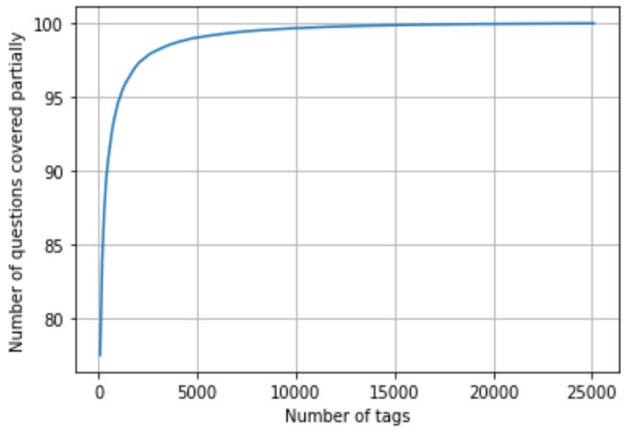
	question	cleaned_tags
0	check upload file imag without mime type like ...	php imageprocessing fileupload upload mimetypes
1	prevent firefox close press ctrl favorit edito...	firefox
2	error invalid type list variabl import matlab ...	r matlab machinelearning
3	replac special charact url probabl simpl simpl...	c# url encoding
4	modifi whoi contact detail use modifi function...	php api filegetcontents
5	set proxi activ directori environ use machin a...	proxy activedirectory jmeter
6	draw barplot way coreplot imag post link pictu...	coreplot
7	fetch xml feed use asp net decid convert windo...	c# aspnet windowsphone
8	net librari generat javascript know net librar...	net javascript codegeneration
9	sql server procedur call inlin concaten imposs...	sql variables parameters procedure calls

9.4 Tag Reduction:

Total number of tags we found from 3 lakh records was 25112. But considering all the tags will give us extremely high time complexity. But we observed in the frequency distribution that only top 100 tags can cover almost 95% of the questions asked in the SO forum.

Hence, to overcome resource limitation and in the sametime classifing questions as much as possible we can just

- Questions covered by 1000 tags: 94.601 %
- Number of questions that are not covered by 100 tags: 16196 out of 300000



9.5 Splitting Data:

We split the preprocessed data into 80:20 ration to create train and test dataset.

9.6 Featurizing data with TF-IDF vectorizer:

TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction.

- The tf–idf is the product of two statistics, term frequency and inverse document frequency. There are various ways for determining the exact values of both statistics.
- A formula that aims to define the importance of a keyword or phrase within a document.

Here we apply tf-idf vectorization on train dataset and test dataset. We use train dataset to fit the vectorization and use this fir to transform both the traib and test data.

```
vectorization and use this fir to transform both the traib and test data.
vectorizer = TfidfVectorizer(min_df=0.00009,
                             max_features=20000,
                             tokenizer = lambda x: x.split(),
                             ngram_range=(1,3))
X_train_multilabel = vectorizer.fit_transform(X_train['question'])
X_test_multilabel = vectorizer.transform(X_test['question'])
```

Term frequency: Term frequency, tf(t,d), is the frequency of term t,

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

where ft,d is the raw count of a term in a document, i.e., the number of times that term t occurs in document d. There are various other ways to define term frequency.

Inverse document frequency: The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N: total number of documents in the corpus N = | D |
- | { d ∈ D : t ∈ d } | : number of documents where the term t appears (i.e., t f (t , d) ≠ 0). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to 1 + | { d ∈ D : t ∈ d } | .

Term frequency–Inverse document frequency: Then tf–idf is calculated as -

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf–idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf–idf closer to 0.

10. Machine Learning Algorithms:

Applying Various ML Algorithm and compare their data, we keep below fixed.

Size of data : 3,00,000
Max features=20000
Tags = 100
n_gram = 1,2,3

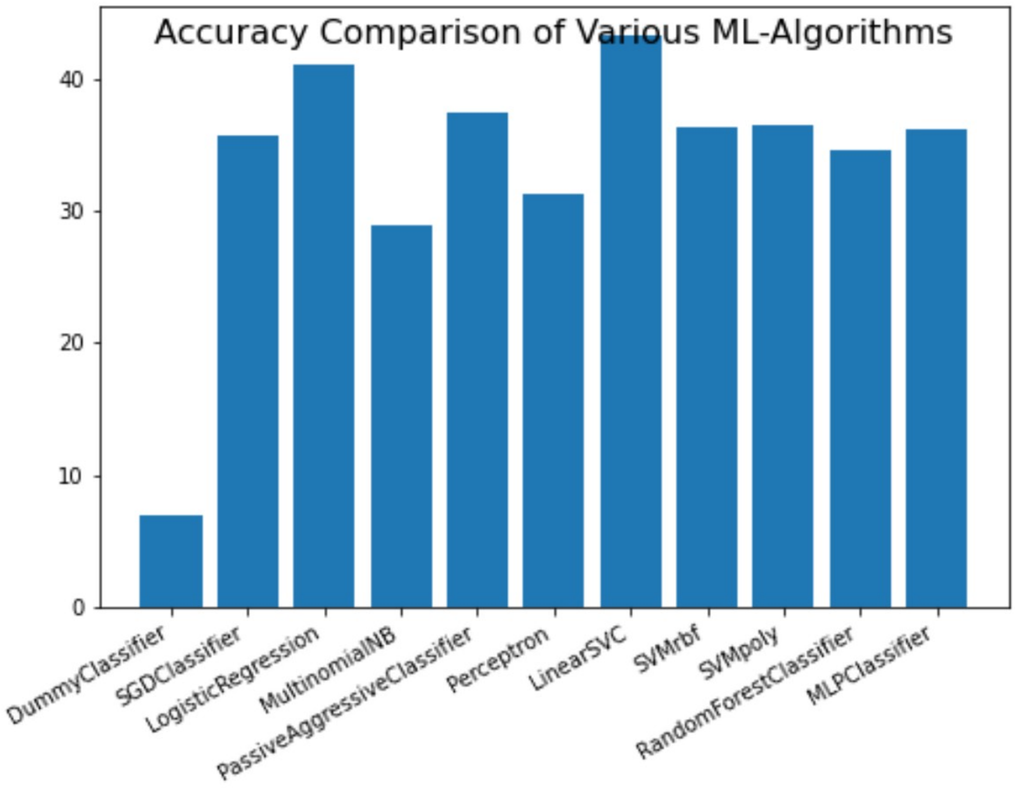
We used below ML algorithms:

1.Dummy Classifier
2.SGD Classifier
3.Logistic Regression
4.Multinomial NB
5.Passive Aggressive Classifier
6.Perceptron
7.Linear SVC
8.SVM - rbf
9.SVM - poly
10. Random Forest Classifier
11.MLP Classifier

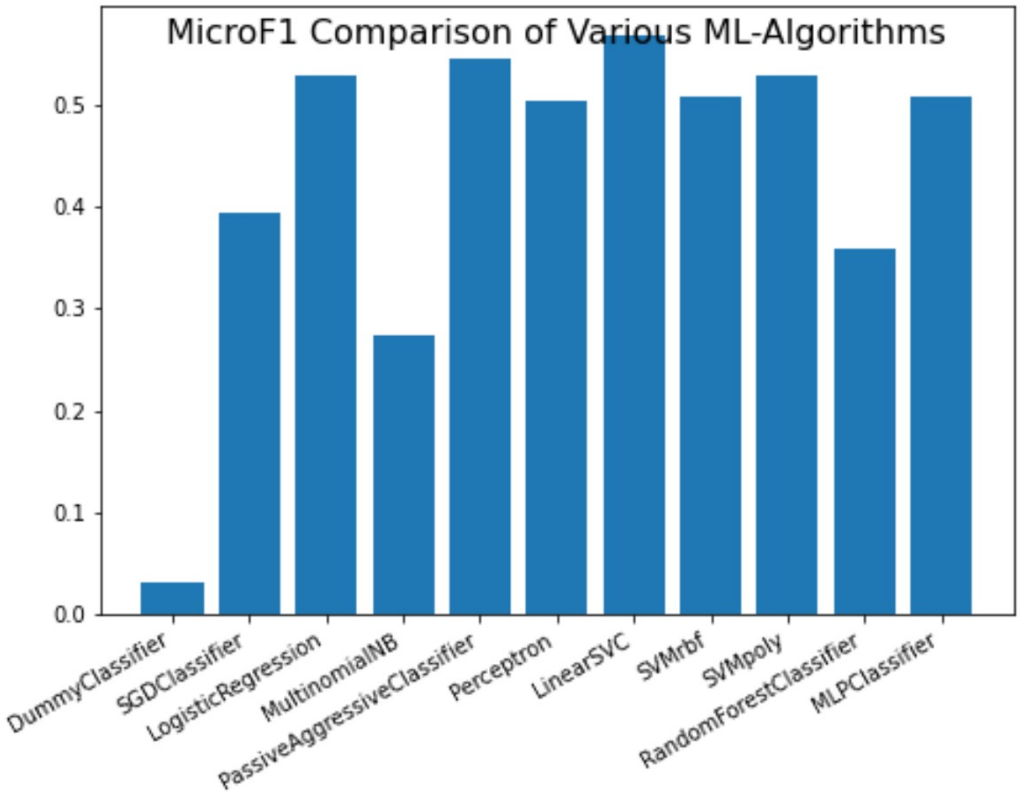
Table: showing the reading of different ML Algorithms performance on this dataset.

ML Algo	Best Accuracy (in percentage)	Macro F1	Micro F1	Hamming Loss	Train data Size	Test Data Size	No. of Tags
Dummy Classifier	6.889	0.012	0.032	0.024	240000	60000	100
SGD Classifier	35.7	0.262	0.393	0.0096	240000	60000	100
Logistic Regression	40.973	0.455	0.528	0.0087	240000	60000	100
Multinomial NB	28.913	0.109	0.273	0.0111	240000	60000	100
Passive Aggressive Classifier	37.443	0.513	0.544	0.0101	240000	60000	100
Perceptron	31.218	0.474	0.503	0.0121	240000	60000	100
Linear SVC	43.256	0.502	0.568	0.0084	240000	60000	100
Linear SVC	39.3	0.407	0.492	0.0091	24000	6000	100
SVM - rbf	36.27	0.452	0.508	0.00994	24000	6000	100
SVM - poly	36.5	0.455	0.528	0.0099	24000	6000	100
Random Forest Classifier	34.5	0.143	0.358	0.00999	24000	6000	100
MLP Classifier	36.2	0.450	0.508	0.00994	24000	6000	100

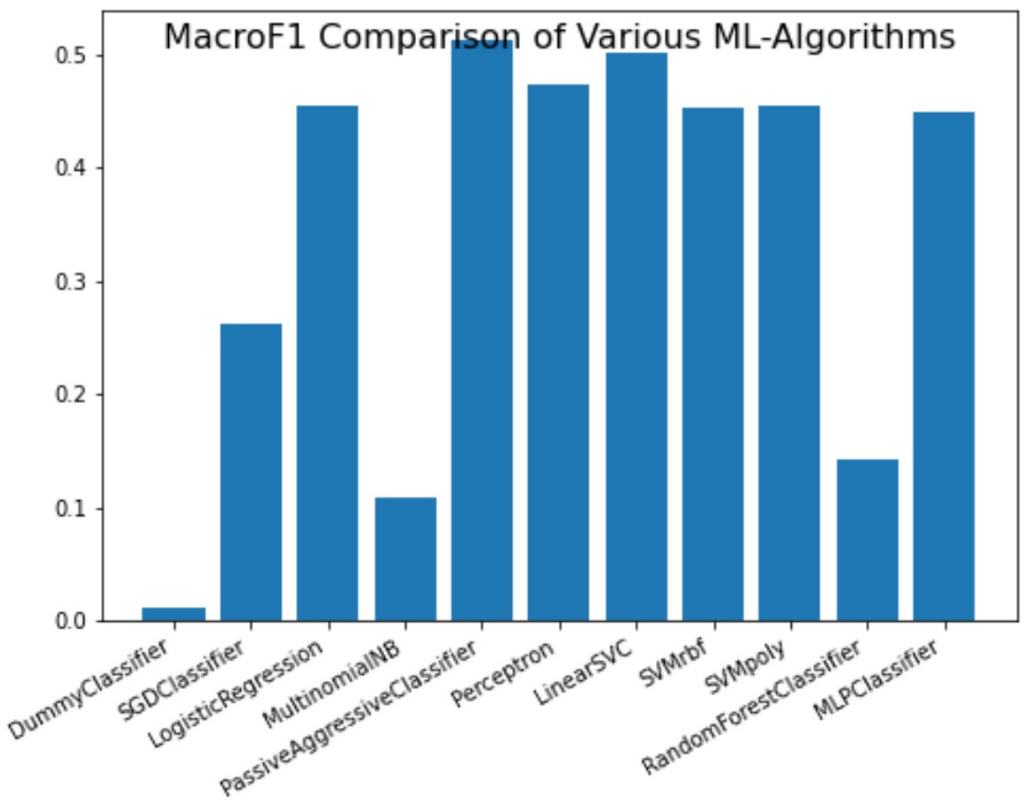
Plotting the above reading we get below diagrams:
Accuracy Comparison:



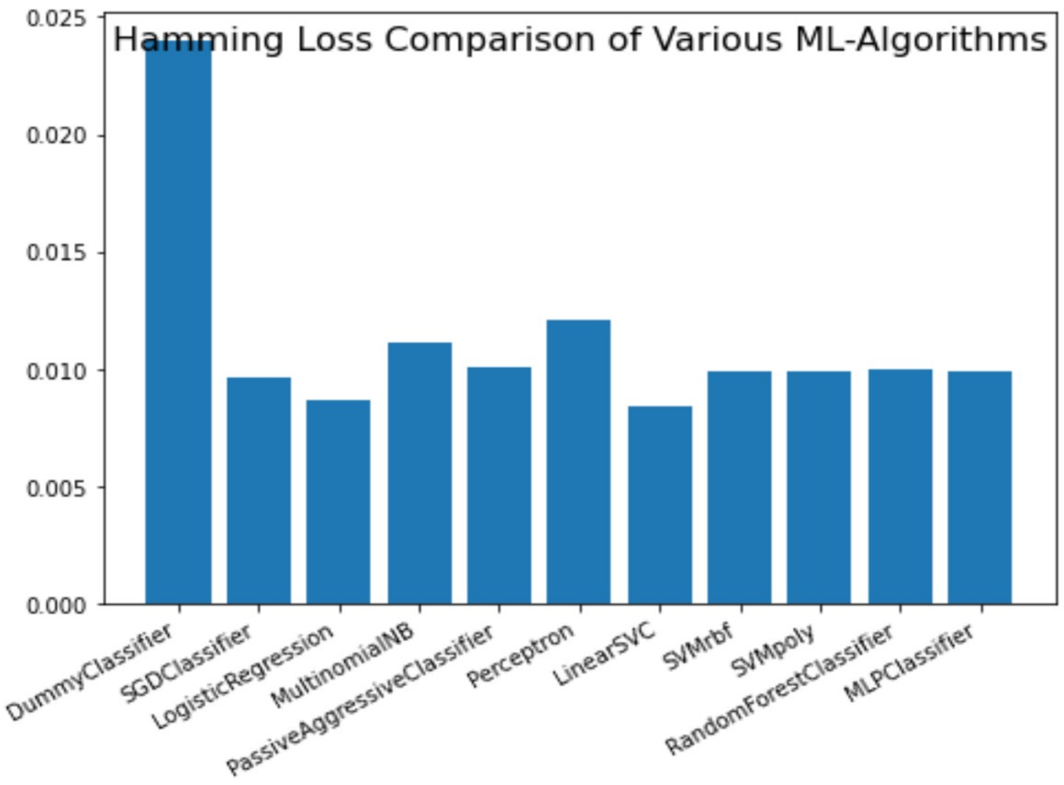
MicroF1 Comparison:



Macro Comparison:



Hamming Loss:



Plotting Accuracy, Macro-F1, Micro-F1, Hamming Loss altogether in the same plot.
To plot them together it was requied to scale the reading in the range 0 to 100.hence we first perform min-max scalling, followerd by multiplication with 100

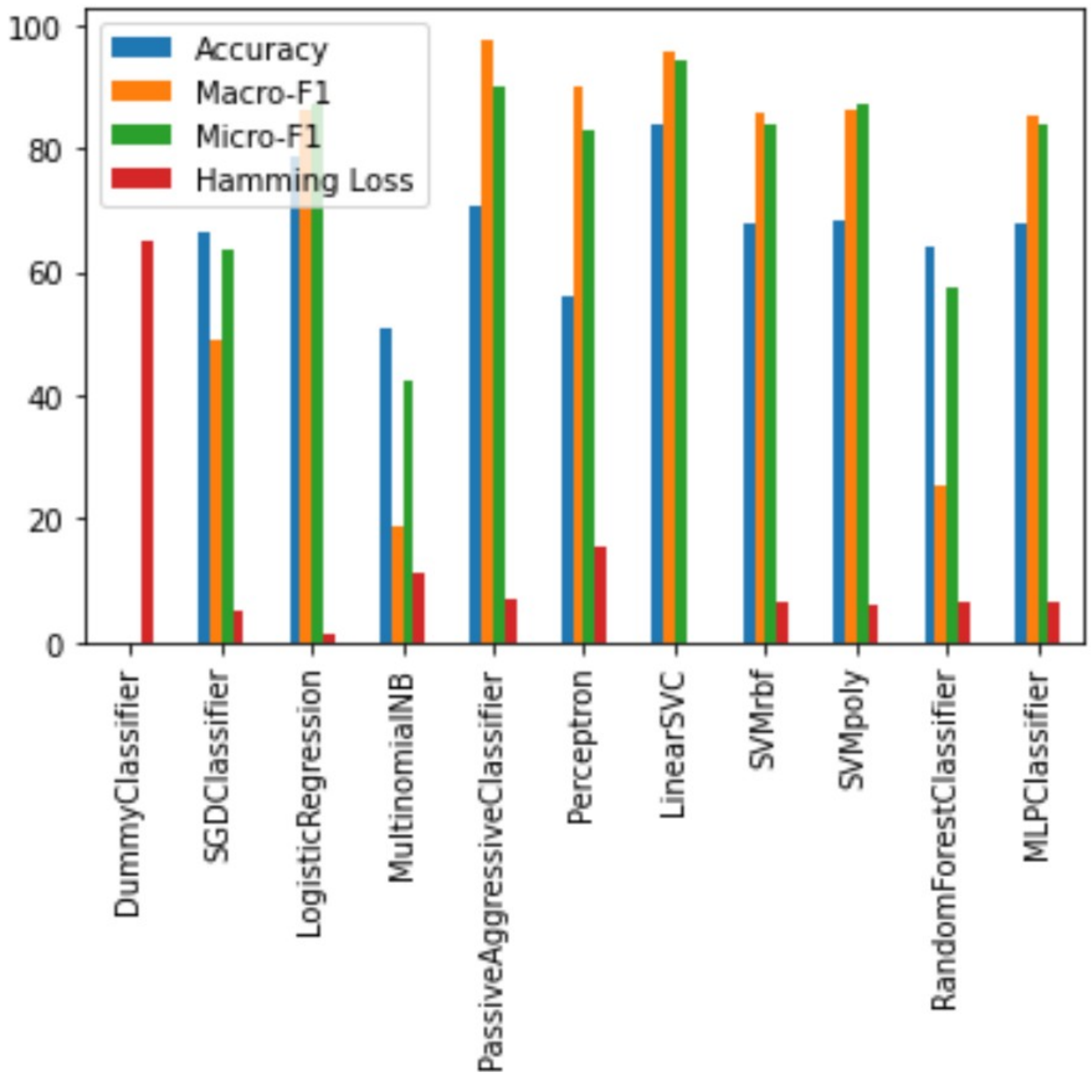


fig: Accuracy, Micro & Macro F1 and Hamming Loss plotting with respect to all ML algorithms

11. Observation:

1. SVM based algorithms perform relatively well compare to other ML algorithms.
2. NN algoritms did not perform well mainly because of two reason – (1) we were unable to provide large dataset to help NN predict. (2) NN algorithms were very costly interms of resources(RAM,CPU) and time. And hence not a good preference if we have resource limitation.
3. with increasing Accuracy, loss (here hamming loss) will decrease.
4. with increasing accureacy F1 value will increase.

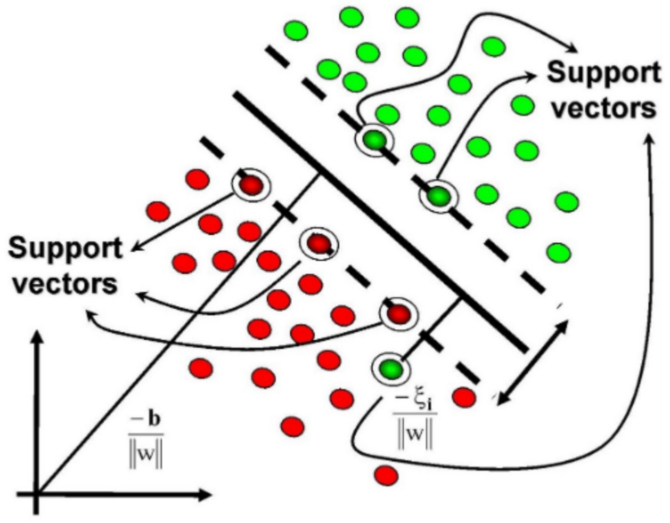
12.Analysis:

As we know for text classification NB and SVM performce well in general.

Naive Bayes (NB) must be performing some naive operations for classification, going down into the math of Naive Bayes, it assumes in the posterior probability that every element is Independent of each other provided the evidence. Well, it works surprisingly well for most of the texts where the corpus is small. (such as tweets, or quotes)

But what comes as a boon acts as imprecation for the algorithm since in most cases the assumption fails to hold its significance. The price paid for such ease is it cannot learn interactions between features because of the class-condition independence assumption made above which is rarely true in most real-world applications. Resulting in the loss of accuracy.

Support Vector Machine is considered to be one the best out of box model of all time. Because rather than taking a probabilistic approach SVM works on the geometric interpretation of the problems. The text being a high dimension problem fits right into its core since the model is independent of dimensions (ability to learn is independent of the dimensionality of the features space).



Also, due to preprocessing steps of text classification the corpus now contains very few irrelevant features and a good classifier would like to also take into account the interaction between the term to understand a “dense” concept which Naive Bayes fails due to its assumption but since SVM has the ability to plug a number of kernels which include Linear kernel can easily process on such predictors and separate out the independent term to max distance. (try to relate the cat example here).

Although, some recent deep learning studies improvement over the SVM performance for the same problem statement.

13. Hyper Parameter Tuning To Find Suitable Parameters:

After comparison and a proper analysis we further go through with the best performed ML algorithm, that is , the Linear SVC. We perform grid search along with 5-fold cross validation and get the best parameters for Linear SVC.

We also tried different svm functions with suitable degree(poly) and gamma value (rbf). But results were not as good as the Linear SVC – hence we stick with Linear SVC.

```
svc = OneVsRestClassifier(LinearSVC())
CV_svc = model_selection.GridSearchCV(
                                        estimator=svc,
                                        param_grid=param_grid,
                                        cv= 5,
                                        verbose=10)
CV_svc.fit(X_train_multilabel, y_train)
```

After analysis we find,the Linear SVC giving its best performance(here, accuracy) when 'C' parameter lies between the range of 0.65 to 0.7.

When going with C parameter = 0.65 we get the best accuracy score. And its reading mentioned below:

```
Accuracy : 0.43256666666666665
Macro f1 score : 0.501832360150235
Micro f1 scoore : 0.567803603355797
Hamming loss : 0.00838
```

Detail readings of top 3 Parameters:

C - Value	Accuracy	Macro F1 Score	Micro F1 Score	Hamming Loss
0.65	43.257	0.5018	0.5678	0.0083800
0.70	43.253	0.5040	0.5686	0.0083817
1.0	43.077	0.5091	0.5704	0.0084467

14.Limitation:

1. resource Limitation: Althought we had 1.2 million records but we hardly used 0.3 million because of resource constraints.

Kernel Crash - Since the amount of data is huge and we mostly use our ordinary day-to-day use laptop, with configuration 8 GB RAM, 512GB SSD SECONDARY MEMORY, its bit difficult to execute the ML program as it will crash definitely.

We use Google Colab, free, version, that provides 12 GB RAM and GPU.
With the available free space of maximum 15GB – it also a big issue to put all the extracted data together in the same drive and it requires deleting temporary files generated during the process.

2. Time Limitation: it requires enormouse time to run the program with full data and for all tags. Especially for NN, RF, SVM RBF and POLY algorithms we hardly able to use 30K records because even for 0.3 million records it was taking >6 hours to train.

For other other algorithms we used 0.3 million records with top 100 tags.

15. Conclusion:

In this project we perform few well know **NLP** tactics like n gram, lemmataion, regular expression , junk char remove and stopword remove etc and preprocess the raw data.

We also perform **data analysis** and get through a detail analysis of underlying data. Exploratory data analysis was performed on the data set at first that gave many hints about the nature of data.This analysis was a vital step as it helps us to choose ML algorithm and its relevant parameters. And also help to analyse the prediction.

It was found that out of the 42K tags(over whole 1.2 million records) that were originally in the data set and 25K used tags(over 0.3 million records that were actually used), keeping the **100** in consideration and discarding the rest would still correspond to more than **95% of the original information**.

Further, The **OneVsRestClassifier** was trained for the data with logistic regression with the default values except for the regularization that was **L1** in this case.

The micro **f1 score** it yielded was approximately **0.57** and the precision and recall were also fair values.

Then, for the same model, some **hyperparameter tuning** was performed, with different values of C(1000,100,10,5,1,0.9,0.8,0.70.65,0.6,0.5,0.3,0.25) with **GridSearchCV** yielded 0.65 to be the best value. Which then yielded F1-measure(micro):0.57. At last, **Linear SVC** was used tried that gave the values of performance metrics as F1-measure: 0.57.

16. Appendices:

SO - Stackoverflow
ML – Machine Learning
Algo - Algorithms
Pred - Prediction
NB – Naive Baise
SVM – Support Vector Machine
SVC - Support Vector Classification
Poly - Polynomal
RBF - Radial Basis Function
RF – Randon Forest
MLP – Multi Layer Percepton
NN – Neural Network
DP – Deep Learning

17. References:

1. https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34
2. www.stackoverflow.com
3. www.kaggle.com
4. colab.research.google.com
5. https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/