Intelligent Question Tagging: Leveraging Advanced Machine Learning Techniques for Automatic Stack Overflow Question Classification

Sean Rahman, Saber Ahmed, Rifat Bin Karim, Hadayet Ullah Razu
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Email: {190104103, 190104104, 190104105, 190104121}@aust.edu

***Abstract*—This research focuses on developing an autonomous tagging system for Stack Overflow, a popular platform for computer science and programming questions. Currently, users manually enter tags for their questions, which can be inaccurate or insufficient. To address this issue, we applied machine learning methods such as Logistic Regression, SGD Classifier, Multinomial NB, and Random Forest, along with essential data preprocessing techniques like HTML tag removal, punctuation removal, contraction, stop-word removal, and lemmatization. The workflow utilizes a dataset from Kaggle, which contains 10% of Stack Overflow questions. The evaluation of models shows that Logistic Regression is performing better than others. The SGD classifier also has a satisfactory performance. Finally, Multinomial NB is underperforming in comparison.**

***Index Terms*—autonomous tagging, stack overflow, logistic regression, sgd classifier, multinomial nb, and random forest**

I. Introduction

Stack Overflow, a widely used online forum, and question-and-answer platform, largely centers its attention on programming and software development [14]. Developers frequently utilize this platform as a means to seek assistance, exchange knowledge, and collaborate with fellow programmers. Stack Overflow is an online platform where individuals may seek assistance on computing-related issues by posting questions, while other members of the community can provide responses and provide potential solutions. Various programming languages, systems, libraries, and other resources may be the subject of the inquiry. There are many professional and amateur developers on the site. This platform uses tags to organize questions, and users can up-vote or down-vote replies based on how helpful they are. Both the original poster and additional readers can upvote comments they find helpful [15]. When a user asks a question, it is mandatory to provide a title, the details of your problem, the expected result, and add up to five tags [16]. These tags describe what the question is about. It is clear that the majority of the questions presented either need more tags or aren't tagged correctly and appropriately. Since there are so many tags, it might be difficult to manually sort through them all to discover the appropriate ones, which is why most users who ask questions avoid doing so. The goal of our study is to use machine learning to automatically label questions by guessing the right tags based on the title. For our study, 10% of an extensive collection of Stack Overflow questions will be taken as a sample [1]. This sample will be used to train and test the machine-learning model. The model will be trained based on the tags already attached to the questions, learning patterns, and connections between the content of the questions and the right tags. Once trained, the model can be used to anticipate tags for brand-new queries that haven't been asked before, potentially reducing time and increasing tagging accuracy. Once the model has been trained, it can be used to guess tags for new, unseen questions. This could save time and make the tagging process more accurate. Our study shows how machine learning and natural language processing can be used to improve the speed and accuracy of question tagging on Stack Overflow by automating the marking of questions. The results of the study can be used to improve how users interact with the platform and make it easier to organize and find useful information.

II. Dataset Description

The Stack Overflow question dataset is freely accessible to the public. The data set we used for our study was created between 2008 and 2016. For our study, we used two data sets and combined them. One dataset contains questions, and the other contains tags. The question dataset contains the title, body, creation date, closed date (if applicable), score, ID, and owner ID for all non-deleted Stack Overflow questions whose ID is a multiple of 10. It contains around 1.25 million rows of data. The data set for the tag has over 3.75 million rows. The tags dataset contains the tags and IDs corresponding to the questions, which creates a connection between the two datasets. We used this ID column to combine the datasets for our work. Initially, we used 10% of the data to conduct the entire study.

TABLE I
Tags Dataset

Id	Tag
80	flex
80	actionscript-3
80	air

TABLE II
QUESTION DATASET

Columns Name	Text
Id	80
OwnerUserId	26.0
CreationDate	2008-08-01T13:57:07Z
ClosedDate	NaN
Score	26
Title	SQLStatement.execute() - multiple queries in...
Body	<p>I've written a database generation script I...

A visual representation of all the tags of the dataset is shown below:

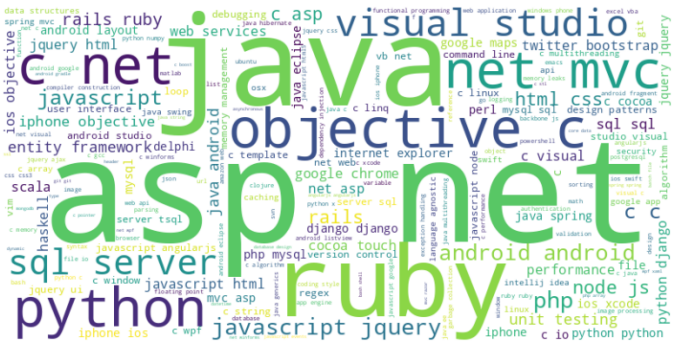


Fig. 1. Word Cloud of All Tags

III. RELATED WORK

Various tagging processes are used on different social sites. Some use the system to tag content or queries automatically; others let human users choose some tags manually from a list of suggestions, or human users may perform the entire manual process. (Jain and Lodhavia) [2] developed a method for an autonomous tagging system using machine learning techniques with some data preprocessing steps, like stemming, tokenization, and deleting stop words, which gave them an accuracy of 70% for Random Forests and 75% for k-Nearest Neighbors on a dataset collected from Kaggle comprised of 10% of Stack Overflow questions [1]. For their work, they extracted the top hundred tags by frequency from the dataset and balanced them on the basis of positive and negative examples to remove the biases.

(Gupta et. al) [3] used the same dataset collected from Kaggle [1]. They used the Logistic Regression Algorithm, Multinomial Naive Bayes, Stochastic Gradient Descent (SGD) Classifier, Support Vector Machine (SVM) Algorithm, and Logistic Regression Algorithm. The SGDClassifier, which provided them with the lowest hamming loss of 0.0096 and the highest Precision of 82.7% is the best model, in accordance with their methodology. Additionally, LinearSVC functioned admirably in their situation.

(Xia et.al) [5] proposed TagCombine, an automatic tag recommendation system that combines three components: multi-label ranking, a similarity-based ranking, and a tag-term ranking. They evaluated their process on 47,668 and 39,231 documents, with 437 and 243 tags, collected from StackOverflow and Freecode, respectively. TagCombine achieved recall@5 and recall@10 scores of 0.5964 and 0.7239, respectively, for StackOverflow, and for Freecode, it achieved scores of 0.6391 and 0.7773, respectively. In multi-label ranking, they used a learning algorithm for tag inference. Similarity-based ranking suggested tags from similar objects, while tag-term ranking used historical data to recommend tags based on object terms and affinity scores.

Though the authors of [1], [2], and [4], introduced an autonomous process, the authors of [4] introduced 'TagAssist', which suggests a set of appropriate tags and lets human users select tags. It uses previously tagged posts to recommend tags for new blog entries.

IV. PREPROCESSING

The dataset [1] that we utilized contains 1.25 million records. For tag preprocessing, we selected queries with a score higher than 5, resulting in 72950 rows of data. Then, after looking at the tag terms, we were able to compile a list of 224129 tags, from which we selected 14883 unique tags. From this unique tag list, we further focus on the top one hundred tags. By eliminating rows that do not include the top tags, we finally obtained 63167 rows to perform our preprocessing. In our work, we have applied most of the preprocessing steps in the "Body" and "Title" columns of the dataset [1]. The steps that we used for preprocessing the dataset are described below:

A. *HTML Tag Remover:*

Firstly, we removed any HTML tags [6] from our "Body" column. As these elements do not add any meaning to our "Body" column, we filtered them out using the lxml Python library [7]. For example:

Before removing HTML Tags from the "Body" column: <p> | *PostgreSQL is interesting in that it supports several languages for writing stored procedures. Which one do you use, and why?* <p> |

After removing HTML Tags from the "Body" column: *PostgreSQL is interesting in that it supports several languages for writing stored procedures. Which one do you use, and why?*

B. *Using Contractions:*

Secondly, we applied contractions using the Python contractions library to obtain the original and uncontracted versions of the words in the "Body" column. [8] Contractions involve connecting two words and eliminating certain letters with an apostrophe to shorten them into one. For example:

"can't" is a contraction of "cannot," while "you're" is a contraction of "you are."

Therefore, we obtained the original and uncontracted versions of the words to preserve the integrity of the sentence.

C. *Lowercase:*

The computer handles lowercase and uppercase differently; if the text is in the same case, a machine can read the words with ease. For this reason, all the texts in the columns “Title”, ”Body” and “Tags” were made lowercase.

D. *Removing Punctuation:*

Another text pre-processing technique is removing punctuation. It can help simplify the text. For this reason, we removed punctuation like commas, periods, question marks, etc. from the ”Body” and “Title” columns using a pre-initialized string, which is used as a string constant in Python. It returns all sets of punctuation. [9]

E. *Lemmatization:*

Lemmatization reduces words to their simplest or base forms, which helps in increasing the accuracy of text analysis. [11] We applied WordNet lemmatizer [12] on our ”Body” and “Title” columns. For example:

Using lemmatization we can convert the word “programming” to “program”, which is a base form.

So, lemmatization can help to maintain the grammatical structure of sentences, which can help in text analysis.

F. *Eliminating Stopwords:*

Stop words are common terms like ”the,” ”and,” ”is,” etc. that appear frequently in text but don’t provide useful information for most text analysis procedures. For our next text preprocessing technique, we have removed these stopwords from the ”Body” and “Tags” columns using the list of stopwords from the NLTK module. [10]

G. *MultiLabelBinarizer:*

A data preprocessing tool called a MultiLabelBinarizer [13] is often used in machine learning to solve multi-label classification problems. Every instance (data point) in a multi-label classification might concurrently belong to many categories. The multi-label target labels are essentially converted into a binary matrix representation, where each column denotes a unique class and each row denotes an instance. The binary matrix’s appropriate entry is set to 1 if an instance belongs to a particular class, and 0 otherwise.

After augmentation, we resulted in 71264 rows, and many of the ”Tags” columns are multilabeled. In order to make this multi-label information usable for machine learning algorithms, Multilabel Binarizer is applied.

V. Feature Extraction

We are employing TF-IDF for feature extraction. The term TF-IDF stands for Term Frequency-Inverse Document Frequency. It helps to pull out important features. It is a numerical measure that is often used in information retrieval and text mining to figure out how important a word or phrase is in a document or group of documents. It is based on the idea that words that are used a lot in one document but not very often in the whole collection tend to be more important and give better information about what the document is about. TF-IDF brings together two important parts:

Frequency of a Term (TF): It counts how often a word is used in a text. This part gives more weight to words that appear more often in the document, thinking that they are more important to what it is about.

Inverse Document Frequency (IDF): This is a way to figure out how rare a term is in the whole group of documents. The logarithm of the ratio of the total number of documents to the number of documents that contain the term is used to figure it out. The IDF gives more weight to terms that aren’t used as often in the collection. This means that those terms are more useful or unique.

We have combined the ”Body” and ”Title” columns of the dataset [1] for our workflow. After that, all of the data was fitted to the vectorizer.

VI. Methodology

Steps we have followed for our work:

Step 1: Read the datasets Questions.csv and Tags.csv in two dataframes df_questions, and df_tag.

Step 2: Merge the rows in df_tag where there are the same Id’s.

Step 3: Merge the two dataframes according to Ids. Now the dataframe contains 1264126 data.

Step 4: Filter the new dataframe where the score is greater than 5. Now the dataframe contains 72950 rows of data.

Step 5: Drop unnecessary columns such as Id, Owner User Id, Creation Date, Closed Date, and Score. So the new dataframe contains “Title”, “Body” and “Tags” columns.

Step 6: Get the unique tags from the dataframe. We have 14883 unique tags.

Step 7: From this unique list of tags, prioritize the top one hundred tags. By eliminating rows without the top tags, 63167 rows were ultimately obtained from preprocessing.

Step 8: For preprocessing, the HTML tags are removed, the abbreviated forms of all letters are contracted, all punctuation is removed, words are lemmatized, and stopwords are eliminated.

Step 9: There are tags that only appear once in the dataframe. For this purpose, data augmentation is employed to obtain a balanced dataframe. These data are augmented twice in this instance. Now the expanded dataframe contains 8097 records.

Step 10: Concate the original dataframe and the augmented dataframe. So the final dataframe contains 71264 data.

Step 11: As some tags consist of an array of multiple distinct tags, a multi-level binarizer was applied to them.

Step 12: Apply TF-IDF to features. Here the features are the “Body” and “Title” columns.

Step 13: Split the entire dataset into train and test dataframes, where the train dataframe contains 70% of the total data and the test dataframe contains 30% of the total data.

Step 14: These train and test dataframes are used for model

fitting. As models, we employ Multinomial Naive Bayes, SGD Classifier, Logistic Regression, and Random Forest.

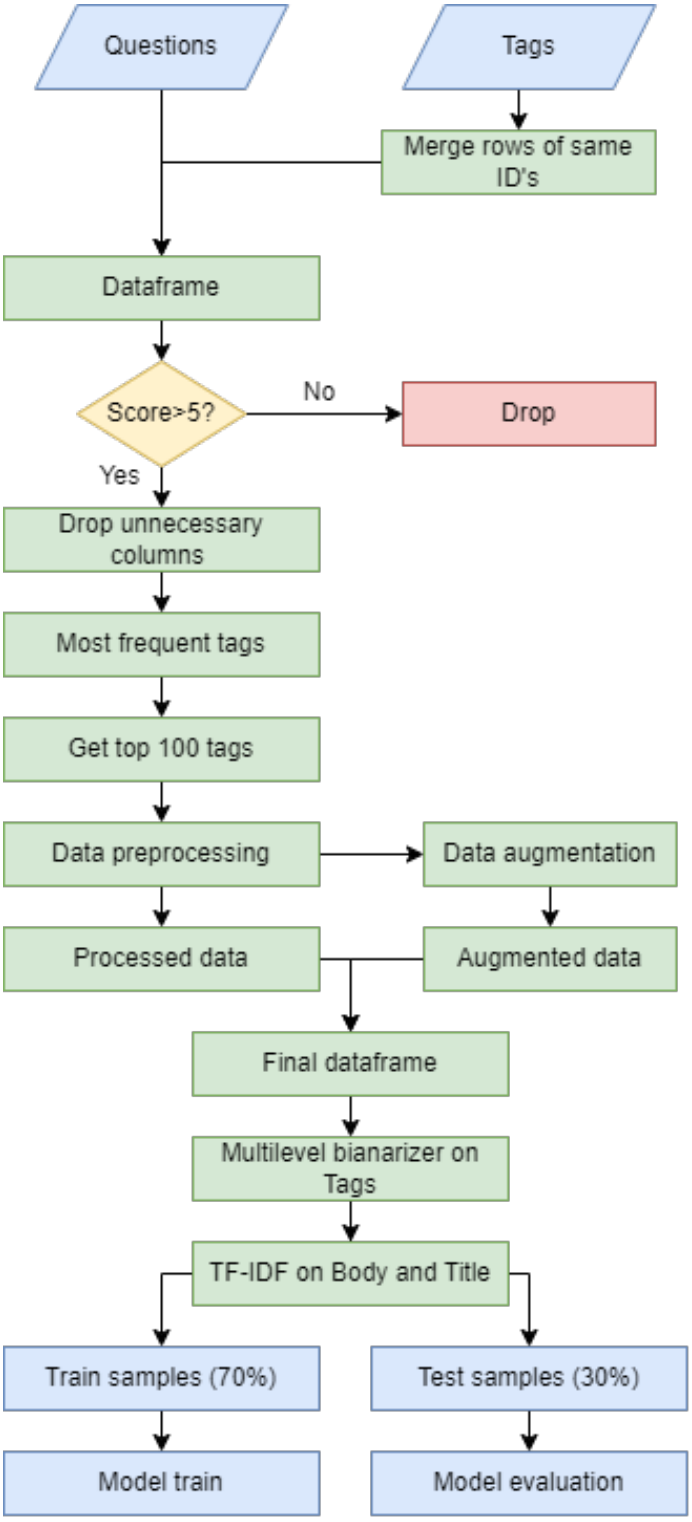


Fig. 2. Proposed Methodology

VII. Evaluation Metrics

For multi-label classification, accuracy may not necessarily be the most relevant metric because It doesn't take into account situations where numerous labels are valid for a single sample. For multi-label classification tasks, Precision, Recall, F1-score, and Hamming Loss are frequently more important metrics. [19], [24]

A. Precision

Precision measures how many of the predicted positive instances were actually positive. A high precision indicates that the model makes fewer false positive errors [17].

B. Recall

It measures how many of the actual positive instances were correctly predicted. So, it gives a measure of how accurately our model is able to identify the relevant data. A high recall indicates that the model makes fewer false negative errors [17].

C. F1-Score

In cases where we need both precision and recall, we use the F1-Score. It is a Harmonic mean of precision and recall. It is useful when there is a class imbalance. A good F1-Score indicates good Precision and a good Recall value [17].

D. Jaccard Index

The Jaccard similarity index compares the individuals in two sets to determine which individuals are shared and which individuals are distinct. It ranges from 0% to 100% and measures how similar the two sets of data are. The higher the percentage, the more similar the two populations are. The Jaccard score is frequently used in multi-label classification to assess the degree of overlap between expected and actual label sets [18].

E. Hamming Loss

It measures the fraction of labels that were wrongly predicted for each instance. It refers to the fraction of incorrect labels relative to the total number of labels. A lower Hamming loss indicates better performance [19].

We can get an idea about how our model is performing using Aggregate metrics like macro, micro, weighted, and sampled avg. The weighted average is simply the average of the metric values for individual classes, weighted by the support of that class [24]. For the calculation of Precision, Recall, and F1-Score, we use the weighted average metric.

VIII. Models Description

A. Logistic Regression

Logistic regression is a widely utilized method in the field of Machine Learning, specifically falling within the category of Supervised Learning techniques. This method is employed to forecast the categorical dependent variable based on a specified collection of independent factors. Logistic regression is a statistical method used to predict the outcome of a dependent

variable that is categorical in nature. Consequently, the result must be a value that falls into a certain category or is discrete in nature. The binary nature of a variable can be represented by values such as Yes or No, 0 or 1, true or False, and so on. However, instead of providing precise values of 0 and 1, probabilistic values within the range of 0 to 1 are utilized [20].

B. *Stochastic Gradient Descent (SGD)*

Stochastic Gradient Descent (SGD) is a very effective method for training linear classifiers and regressors using convex loss functions, such as Support Vector Machines and Logistic Regression. This methodology is both straightforward and efficient. The Stochastic Gradient Descent (SGD) algorithm has demonstrated its effectiveness in addressing the challenges posed by large-scale and sparse machine learning issues often encountered in the fields of text categorization and natural language processing. Considering the limited availability of data, the classifiers incorporated in this module exhibit a high degree of scalability, enabling their application to problem domains encompassing training sets exceeding `10^5` instances and feature spaces exceeding `10^5` dimensions [21].

C. *Multinomial Naive Bayes*

The Multinomial Naive Bayes (MNB) technique is widely utilized in the field of Natural Language Processing (NLP) for the purpose of text categorization. This approach proves to be quite advantageous for dealing with issues that pertain to textual data including discrete attributes, such as the quantification of word frequencies. The MNB algorithm operates based on the principles of Bayes' theorem and makes the assumption that the features are conditionally independent given the class variable [22].

D. *Random Forest*

The Random Forest method is well recognized as a prominent machine learning approach within the realm of supervised learning. Machine learning (ML) may utilize this technique for both classification and regression tasks. The approach is grounded in the principle of ensemble learning, a methodology that involves aggregating several classifiers to address intricate problems and enhance the model's performance. An increased abundance of trees within a forest setting has been seen to positively correlate with enhanced accuracy and serves as a preventive measure against the issue of overfitting [23].

IX. Results and Analysis

We have used 4 models to classify the data. They are Logistic Regression, SGD Classifier, Multinomial Naive Bayes, and Random Forest Classifier. To evaluate the performances of the models we have measured Precision, Recall, F1-Score, Hamming loss, and Jaccard Index. Each of the models performed differently to classify the tags. As a result evaluation metrics result for models are different.

From the following table, we can see that the Random Forest classifier has the highest precision, which is about 94%. Then comes SGD Classifier. Maintaining a precision of about 85%. Logistic Regression is providing good precision(82%) than Multinomial NB but performs slightly less than SGD Classifier.

TABLE III
Performance Score of Evaluation Metrices

	Logistic Regression	**SGD Classifier**	**Multinomial NB**	**Random Forest**
Precision	82%	85%	77%	94%
Recall	61%	51%	33%	44%
F1- Score	70%	62%	44%	59%
Hamming Loss	0.89%	0.99%	1.35%	1.01%
Jaccard Index	54.07%	46.99%	29.52%	43.11%

For Recall, Logistic Regression is providing a 61% score, which is the highest among all. Then comes SGD Classifier, scoring about 51%. Random Forest is giving 44% and at last, Multinomial NB is performing lowest, providing about 33% score.

The logistic regression model is giving highest F1-Score. The score is about 70%. SGD Classifier and Random Forest Classifier are performing well, Scoring about 62% and 59% respectively. Multinomial NB is giving a score of about 44%.

For Hamming Loss, Logistic Regression has the best score among others. It's about 0.89%. Then comes SGD Classifier, scoring about 0.99%. The score for Multinomial NB and Random Forest Classifier is quite similar but Random Forest is performing slightly better.

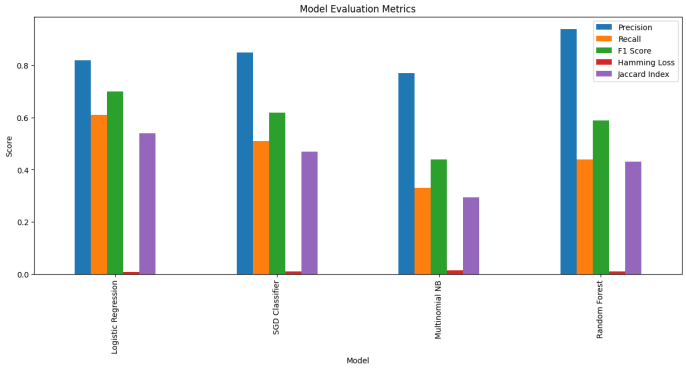
For Jaccard Index, the Logistic Regression model is performing better, scoring about 54.07%. SGD Classifier and Random Forest Classifier score is quite close. But SGD Classifier is performing better. Scoring about 46.99% and Random Forest scoring 43.11%. Multinomial NB is performing worse, Scoring about 29.52%.

The evaluation of models shows that Logistic Regression is performing better than others. The SGD classifier also has a satisfactory performance. Finally, Multinomial NB is underperforming in comparison.

X. Conclusion

Using models based on machine learning, our primary objective is to develop an automated system for query tagging. We completed the workflow by performing various steps and utilizing four models. Specifically, Logistic Regression, SGD Classifier, Multinomial Naive Bayes, and Random Forest Classifier. In consequence, the precision, recall, F1-score, Hamming loss, and Jaccard Index are distinct for each model. Which assists in determining which model performs better. Based on our observations, we've determined that Logistic Regression performs better. SGD Classifier performs marginally better than others, but not as well as Logistic Regression. Multinomial NB is performing poorly comparatively. We have

also encountered some difficulties in our task. When we attempted to execute the SVM model, it took too long, so we had to suspend execution. Due to this, we were unable to include the evaluation performance in a comparative analysis. Finally, we can say that our workflow will aid in detecting the user's tagging according to the title and query more precisely and automatically.



REFERENCES

[1] Auto Tagging Stack Overflow Questions. Available: www.kaggle.com/code/younday/auto-tagging-stack-overflow-questions/input. [Accessed: June 10, 2023.]

[2] Automatic Question Tagging using k-Nearest Neighbors and Random Forest, Virik Jain, Jash Lodhavia, Department of Computer Engineering, VJTI, Mumbai, India

[3] Tagging Stack-Overflow Questions using Supervised Machine Learning Techniques, Ms. Garima Gupta, Disha Sharma, Harshit Aggarwal, Ishan Agarwal, MAIT, Delhi

[4] TagAssist: Automatic Tag Suggestion for Blog Posts, Sanjay C. Sood, Kristian J. Hammond, Sara H. Owsley, Larry Birnbaum, Northwestern University

[5] Xia X, Lo D,Wang X, Zhou B. Tag recommendation in software information sites. In Proc. the 10th Working Conference on Mining Software Repositories, May 2013, pp.287-296. https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=3020&context=sis_research. [Accessed: August 10,2023.]

[6] HTML Tags. Available: javatpoint.com/html-tags. [Accessed: June 10, 2023.]

[7] lxml - XML and HTML with Python. Available: lxml.de/. [Accessed: June 10, 2023.]

[8] NLP – Expand contractions in Text Processing. Available: geeksforgeeks.org/nlp-expand-contractions-in-text-processing/. [Accessed: June 10, 2023.]

[9] string.punctuation in Python. Available: geeksforgeeks.org/string-punctuation-in-python/. [Accessed: June 10, 2023.]

[10] NLTK stop words. Available: pythonspot.com/nltk-stop-words/. [Accessed: July 10, 2023.]

[11] Stemming and Lemmatization in Python. Available: datacamp.com/tutorial/stemming-lemmatization-python. [Accessed: June 10, 2023.]

[12] nltk.stem.wordnet documentation Available: nltk.org/_modules/nltk/stem/wordnet.html. [Accessed: June 10, 2023.]

[13] sklearn.preprocessing.MultiLabelBinarizer. Available: scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html. [Accessed: June 10, 2023.]

[14] StackOverflow. Available: stackoverflow.com/. [Accessed: June 10, 2023.]

[15] Tour StackOverflow. Available: stackoverflow.com/tour. [Accessed: June 10, 2023.]

[16] Ask Questions StackOverflow. Available: stackoverflow.com/questions/ask. [Accessed: June 10, 2023.]

[17] Precision, Recall, F1-Score. Available: analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/. [Accessed: August 10, 2023.]

[18] Jaccard Index. Available: statisticshowto.com/jaccard-index/. [Accessed: August 10, 2023.]

[19] Multi-Label Classification in Python. Available: multi-label-classification-in-python-empowering-machine-learning-with\\-versatility. [Accessed: August 10, 2023.]

[20] Logistic Regression in Machine Learning. Available: javatpoint.com/logistic-regression-in-machine-learning. [Accessed: August 10, 2023.]

[21] Stochastic Gradient Descen. Available: scikit-learn.org/stable/modules/sgd.html#:~:text=Stochastic%20Gradient%20Descent%20(SGD)%20is,Vector%20Machines%20and%20Logistic%20Regression.. [Accessed: August 10, 2023.]

[22] Applying Multinomial Naive Bayes to NLP Problems. Available: geeksforgeeks.org/applying-multinomial-naive-bayes-to-nlp-problems/. [Accessed: August 10, 2023.]

[23] Random Forest Algorithm. Available: javatpoint.com/machine-learning-random-forest-algorithm. [Accessed: August 10, 2023.]

[24] Evaluating Multi-label Classifiers. Available: towardsdatascience.com/evaluating-multi-label-classifiers-a31be83da6ea#:~:text=Accuracy%20doesn't%20tell%20the%20whole%20story&text=Class%20red%20has%20the%20majority,their%20proportion%20is%209%3A1%20.&text=That%20would%20mean%20that%20given,would%20belong%20to%20class%20blue%20. [Accessed: August 10, 2023.]