The intuition behind this challenge was to predict tag words for StackOverflow, MathOverflow and StackExchange posts.

***Step1: Data Preparation and Cleaning***

* The first step is to pull the data from the mongo DB. Since the data for this homework is large in compared to the computational capacity of the Aspen cluster, we are limiting the number of records to 10K. (Feature transformers were failing for record number greater than 10K)
* Once the data has been pulled, we would convert them into the spark data frames.
* The next step is to clean the data. We would remove the HTML tags from the body section.
* I have utilized “Beautiful Soup” library to remove the HTML tags from the Body column of the test and train data set.

***Step2: Feature Transformation***

* The first step in our process is to tokenize the body and the title column. Since we don’t need to ID column for train. We will drop this column and only select the relevant columns like “tokenizedTitle”, “tokenizedBody” and Tags
* Since we are dealing with huge text data, we need to ensure to remove the noise from the data to receive better performance. So, the next step is to to remove the stop words from the Title and the Body sections

Our final Data would like this.



***Step 3: Building a pipeline***

* Since, I wanted to use “OneVsRest” classifier in Scikit learn library, which would be able to predict multiple labels for each document in the corpus, I have converted my spark data frames into Pandas.
* The next step in the process is to create a ML pipeline using the following transformation

1. Count Vectorizer
2. TF-IDF
3. OneVsRest Classifier with LinearSVC algorithm

Count Vectorizer: It provided a sparse representation of the document over the vocabulary.

TF-IDF: To represent the importance of a term to a document in the corpus.

OneVsRest with LinearSVC: This method is utilized to classify the data into multiple data. Here we have used LinearSVC algorithm. We could try other algorithms like Logistic regression, due to infrastructure limitation on running the model, we are limiting to one algorithm.

Since we are predicting multiple labels, we are using a transformer ***MultiLabelBinarizer*** to binarize the Label column. We would get the predicted output also in vector format. We would then use the inverse transformation to get the predicted output in the text format.

* We would further, sanitize the labels by encoding the result into “UTF-8” format.

***Step 4: Calculate F1 metrics***

* Next, we calculate the F1 metrics using the Scikit library

**F1 score on predicted train data using parameter “weighted” =** 0.91959012867433809

**F1 score on predicted train data using parameter “micro” =** 0.89666515426497284