**Text Mining & Naïve-Bayes**

**Introduction to Dataset:**

* The dataset contains two columns namely Data & Labels. Data column has Blog text and Labels contains the theme of the blog
* There are 2000 data point and 20 unique Labels. Each there are 100 blogs of each labels.
* This means the dataset is balanced among the labels.

**Text Preprocessing:**

1. **Save the Blog Title/Subject:**

* The blog text contains many meta-data information. One such information is the Subject/Title of the blog.
* We have extracted the Subject of Blog from the meta-data and saved it to be used as a feature.

1. **Cleaning the text:**

* We removed all the meta-data info
* Removed all the unnecessary texts such as blog contributors’ names, references of previous discussions, etc
* Converted entire text to lower case
* Expanded the contractions (can’t to cannot, it’s to it is) from the text
* Remove all characters except spaces, letters and digits. (special characters, punctuations are removed here)
* Then, remove standalone numerical digits (like 1, 10, 34)
* Further, remove all the alpha numeric words (like A1, 12ABC, etc)
* Finally, remove all the excessive white space characters

Apply this cleaning to both the Blog Text as well as the subject Text that we saved earlier.

* Stop Words Removal : Remove all the Stop Words from both the text and subject.
* Merging Text & Subject : Concatenate the Text and Subject strings to form a single document. Keep a copy of separate Text and Subject as well.

**Vectorization and Classification:**

**1. Initial Tests:**

* I Vectorized the cleaned text documents of Text, Subject as well as Concatenated Text-Subject using TfidfVectorizer() with default parameters
* Then, I combined the vector matrix of Text & Subject to form a new Vector Matrix

Now we have 4 different vector matrix:

**Text Matrix**

**Subject Matrix**

**Text-Subject Combined Matrix**

**Concatenated Text-Subject Matrix**

* Then applied MultinomialNB() classifier with default parameters on all four. Results were as follows

**Features Accuracy**

only Text 67%

only Subject 59%

Text-Subject matrix combined 78%

Concatenated Text-Subject 75%

* Evidently, taking Text & Subject together gives a good boost to our classification.
* So, we chose to move forward to tuning of parameters where our features will be Concatenated Text-Subject

1. **Hyperparameter Tuning:**

* We tuned the parameters of TfidfVectorizer :

max\_df : maximum document frequency of tokens to be included in the vocabulary

min\_df: minimum document frequency required to be included in the vocabulary

max\_features: How many features will be there at max in the vocabulary.

ngram\_range: Defines the type of tokens in the vocabulary, unigrams, bigrams, trigrams, etc

use\_idf : Whether to use idf or not

smooth\_idf: whether to apply smoothing while calculating IDF for tokens

* Parameters of MultinomialNB() to be tuned :

alpha : value to be added for Laplacian smoothing

* The value set for these parameters were:

'tfidf\_\_max\_df': [0.75, 0.85, 1.0],

'tfidf\_\_min\_df': [1,2,5],

'tfidf\_\_ngram\_range': [(1,1),(1,2)],

'tfidf\_\_use\_idf': [True, False],

'tfidf\_\_smooth\_idf': [True, False],

'tfidf\_\_max\_features': [10000, 20000, 30000, 40000],

'classifier\_\_alpha':[0.01, 0.1, 2, 3, 5, 10, 100]

* After applying GridSearchCV over these parameters the best values for these parameters are

'classifier\_\_alpha': 0.1,

'tfidf\_\_max\_df': 0.75,

'tfidf\_\_max\_features': 20000,

'tfidf\_\_min\_df': 1,

'tfidf\_\_ngram\_range': (1, 2),

'tfidf\_\_smooth\_idf': True,

'tfidf\_\_use\_idf': True

1. **Finiding the final results**

* Using the above obtain values for parameters we vectorized our Concatenated Text-Subject dataset and then cross validated it to get the accuracy of 78%
* We vectorized the only Text and only Subject datasets suing the obtained parameters and combined their matrix. Over this matrix we got the cross-validated accuracy of 79%

**Final Discussion & Challenges:**

* The final results are not very satisfying. We are miss-classifying more than 20% of times.
* For Multiclass classifications where the labels are balanced accuracy is a good and sufficient metric.
* The text cleaning part was challenging. I Took help from online resources. I did all the text cleaning using the regular expressions. I didn’t know to write the regex patters for matching with unnecessary text. I took help from online for regex patters.
* Then, the parameter tuning did not have any substantial improvement in prediction.
* We need to do the text preprocessing more in depth for retaining the most significant words and removing unnecessary words. For this we need to have understanding of all the labels in the dataset.

**Sentiment Analysis**

**About the Data:**

* We have performed sentiment analysis on the cleaned blog texts. We did not include the subject in sentiment analysis.

**About the Model Used:**

* We have used the transformers model *“twitter-roberta-base-sentiment”* for sentiment analysis
* The model gives sentiment scores for (Negative, Neutral, Positive) sentiments in the text string.
* We converted these scores into probabilities using *softmax()* function

**About the Process:**

* Since the model we used has a token length limit of 512 tokens. We had to split our blog text into chunks of 512 tokens and then perform sentiment analysis on the chunks.
* Then we took the average of sentiment scores of all chunks to get the final sentiment scores of the entire blog text.
* For this process we create a function called *“sentiment\_analysis\_on\_chunks()”* . It returns sentiment scores and the predicted sentiment of an entire blog text.
* We iterated over the dataset of blogs and individually passed each blog into the above function