WHY SO HARSH?

TEAM – ML_CODERS

IMT2020042 – Gousepeer Arella

IMT2020513 – Vishal Cheeti

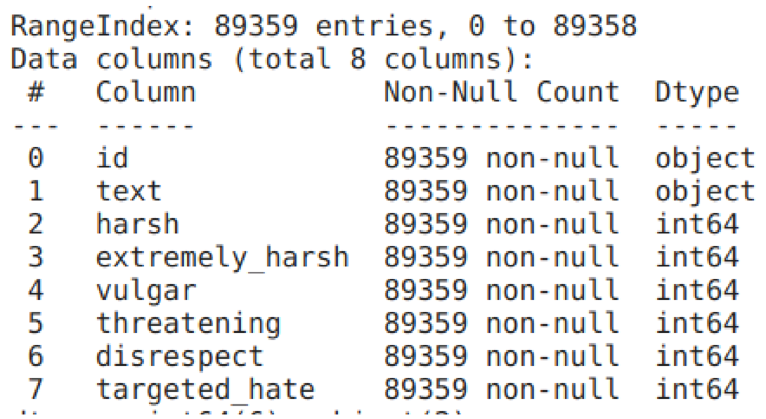
Preprocessing and EDA

Given a comment, classify it into 6 different classes of harsh. They are -

1) harsh
2) extremely harsh
3) vulgar
4) threatening
5) disrespect
6) targeted hate

consider a comment, “You useless piece of trash! you are such an asshole that you deserve to rot in the gutter alongside sewage.” That is certainly harsh! So, the label for this would be [1 1 0 0 1 1]. Translating to the comment being harsh, extremely harsh, disrespectful, and targeted hate.

Info of the dataset:



```
RangeIndex: 89359 entries, 0 to 89358
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               89359 non-null  object
 1   text             89359 non-null  object
 2   harsh            89359 non-null  int64
 3   extremely_harsh  89359 non-null  int64
 4   vulgar           89359 non-null  int64
 5   threatening      89359 non-null  int64
 6   disrespect       89359 non-null  int64
 7   targeted_hate    89359 non-null  int64
```

- We have checked for the null values and missing values; we didn’t find any of them.

Expanding Contractions:

- Contractions is the shortened form of words like **don’t** stands for **do not, aren’t** stands for **are not.** We need to expand this contraction in data for better data analysis.
- Import contractions library and apply contractions.fix to expand the contracted words

LowerCase:

- If the text is in same case it is very easy for machine to interpret the words because the lowercase and uppercase are treated differently by the machine.
- So, we need to make the text in same case and the most preferred case is lowercase to avoid such problems.

Handling Emoticons:

- Machine cant read emojis and emoticons, so we convert the emoticons into their respective mood text like smile, cry etc

Removing Punctuations:

- There are total of 32 main punctuations that should be taken care of.
- By using the string module with a regular expression to replace a punctuation with an empty string.

Removing words and words containing digits:

- If there exists a word containing the digits, or just existence of digits then we remove such words as they do not provide any required information to us regarding the sensitivity of the text.

Removing Stopwords:

- Stopwords are the most commonly occurring words in a text which do not provide any valuable information.
- stopwords like they, there, this, where etc are some of the stopwords. NLTK library is a common library that is used to remove stopwords and include approximately 180 stopwords which it removes. If we want to add any new word to a set of words then it is easy using the add method.
- Here we added new words like http,https,www.

Removing whitespaces:

- Most of the time text data contain extra spaces or while performing the above preprocessing techniques more than one space is left between the text so we need to control this problem
- We used regular expression to solve this problem.

Lemmetization:

- Comments are now tokenized and then lemmetized (Removing -ing,.. suffixes). Parts-ofSpeech Tagging is also done alongside lemmetization. This is the final step of pre-processing.

Feature Extraction

Word Vectoriser using TF-IDF

- strip_accents = 'unicode'

Remove accents and perform other character normalization during the preprocessing step.

'ascii' is a fast method that only works on characters that have an direct ASCII mapping.

'unicode' is a slightly slower method that works on any characters.

- analyzer = 'word'

Whether the feature should be made of word or character.

- stop_words = 'english'

Remove all the stop words of english

- ngram_range = (1, 2)
- ngram is the set of words together.Range (1,2) signifies that we take single words while tokenization or we take asset of twe words together.
- The lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min_n <= n <= max_n will be used. For example, an ngram_range of (1,1) means only unigrams, (1,2) means unigrams and bigrams, and (2,2)means only bigrams. Only applies if the analyser is not callable.
- min_df = 2,
- max_df = 0.5

Character Vectoriser using TF-IDF

- strip_accents = 'unicode'
-
- analyzer = 'char',then stop_words hyperparameter holds no significance and this is pretty logical.
- ngram_range = (2, 6)
- min_df = 2
- max_df = 0.5

Stacking

- We have stacked the both features of word and character

```
print(train_dfjoin.shape)
```

Logistic Regression

Naïve bayes classifier

Random Forest Classifier

XG Boost

SGD Classifier

```
[illegible]
```

Results: