

 Malignant-Comments-Classifier

Submitted by:

Ruchita Parmar

**ACKNOWLEDGMENT**

1. <https://conversationai.github.io/>
2. <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>
3. <https://www.aclweb.org/anthology/P12-2018>
4. <https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html>

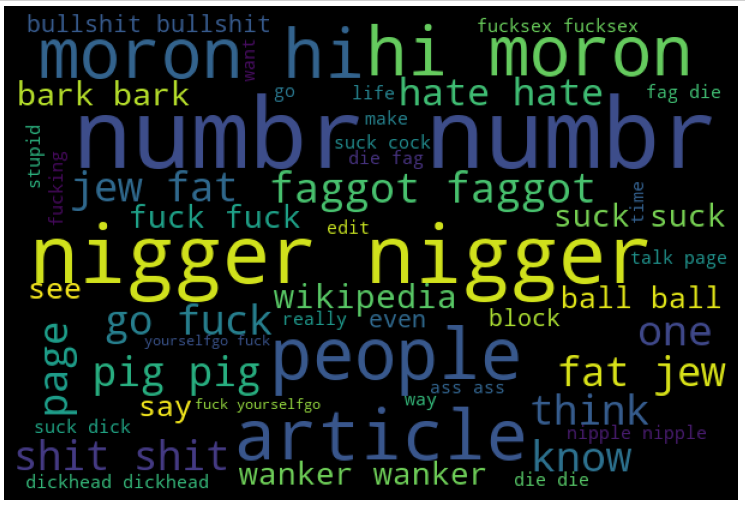
**INTRODUCTION**

**Problem Statement**

The goal is to create a classifier model that can predict if input text is inappropriate (toxic). 1. Explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments. 2. Create a baseline score with a simple logistic regression classifier. 3. Explore the effectiveness of multiple machine learning approaches and select the best for this problem. 4. Select the best model and tune the parameters to maximize performance. 5. Build a the final model with the best performing algorithm and parameters

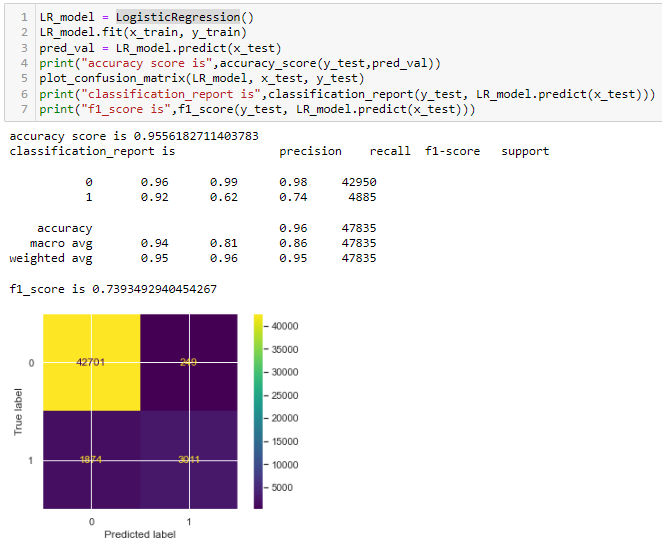
Conceptual Background of the Domain Problem

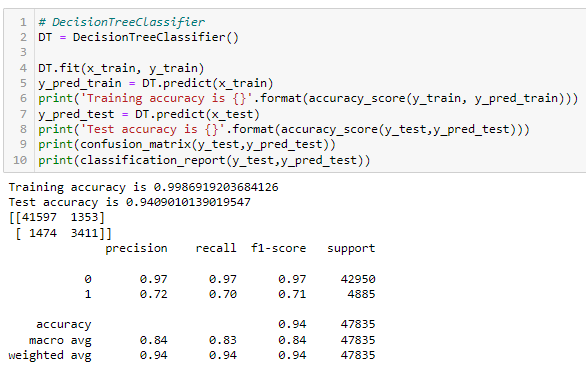
The background for the problem originates from the multitude of online forums, where-in people participate actively and make comments. As the comments some times may be abusive, insulting or even hate-based, it becomes the responsibility of the hosting organizations to ensure that these conversations are not of negative type. The task was thus to build a model which could make prediction to classify the comments into various categories. Consider the following examples :

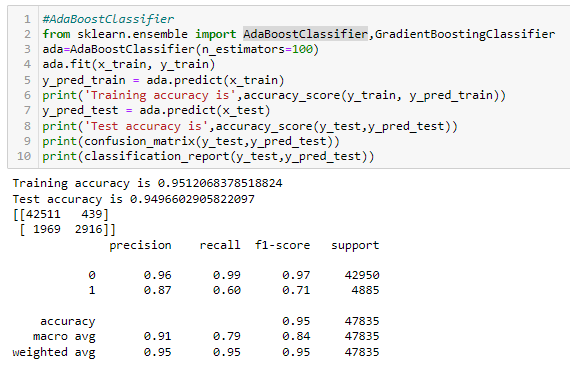


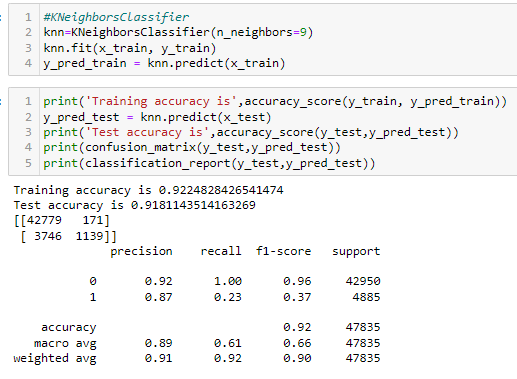
**Analytical Problem Framing**

Mathematical/ Analytical Modeling of the Problem

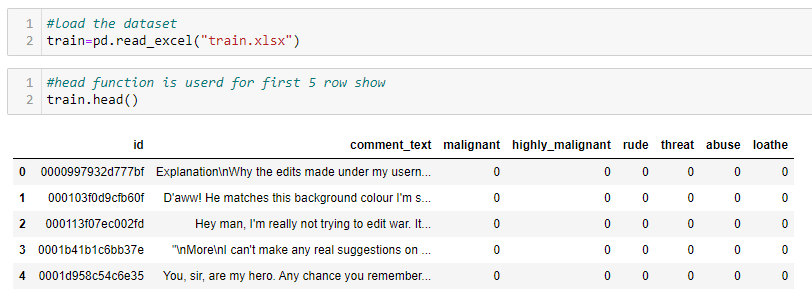


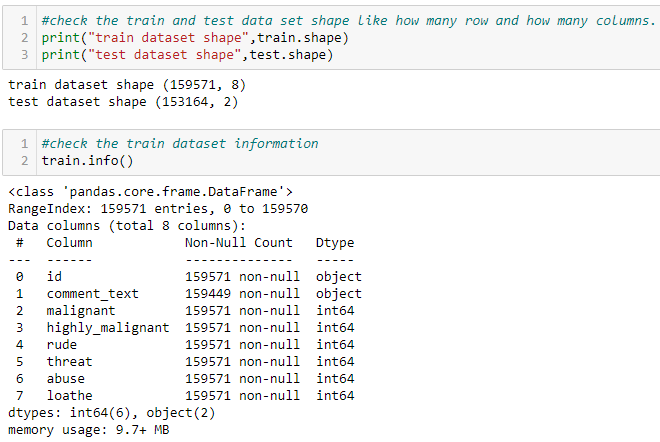


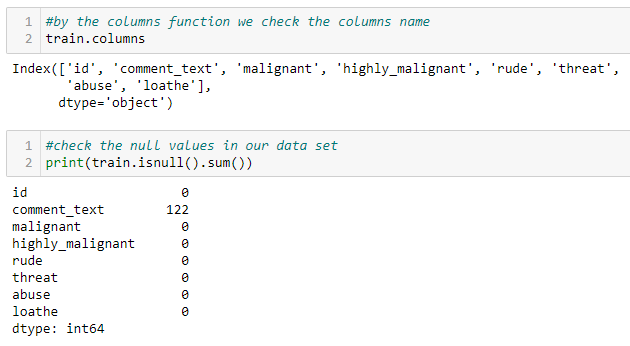


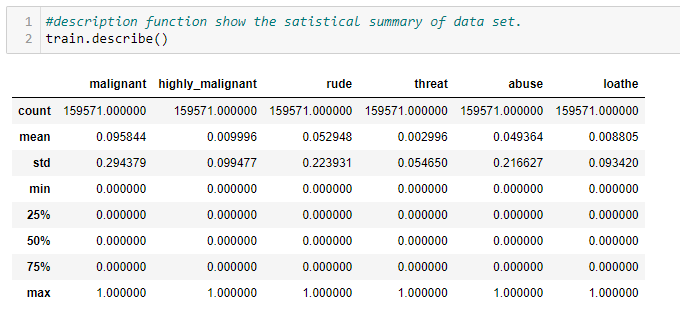


* Data Sources and their Description









* Data Preprocessing Done

The following steps were taken to process the data:

**Checking for missing values:**

First and foremost, after importing the training and test data into the pandas dataframe, I decided to check for missing values in the downloaded data. Using the “isnull” function on both the training and test data, I discovered that there were no missing records and therefore, I moved on to the next step of my project

**A string without all punctuations to be prepared**:

1. The string library contains punctuation characters. This is imported and all numbers are appended to this string. Our comment\_text field contains strings such as won't, didn't, etc. which contain apostrophe character('). To prevent these words from being converted to wont or didn't, the character ' represented as \' in escape sequence notation is replaced by empty character in the punctuation string.

2. Make trans( intab, outtab) function is used. It returns a translation table that maps each character in the intab into the character at the same position in the out tab string.

**→ updating the list of stop words**:

1. Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive or negative impact on our statement like “is, this, us, etc.”

2. Single letter words if existing or created due to any pre-processing step do not convey any useful meaning and so they can be directly removed. Hence letters from b to z, will be added to the list of stop words imported directly.

→ **Stemming and Lemmatizing**:

1. The process of converting inflected/derived words to their word stem or the root form is called stemming. Many similar origin words are converted to the same word e.g. words like "stems", "stemmer", "stemming", "stemmed" as based on "stem".

2. Lemmatizing is the process of grouping together the inflected forms of a word so they can be analysed as a single item. This is quite similar to stemming in its working but differs since it depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document

3. The wordnet library in nltk will be used for this purpose. Stemmer and Lemmatizer are also imported from nltk.

**→ Splitting dataset into Training and Testing**:

1. Since the system was going out of memory using train\_test\_split, I had jumbled all the indexes in the beginning itself.

2. The shuffle function defined here performs the task of assigning first 2/3rd values to train and remaining 1/3rd values to the test set.

Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

**Model/s Development and Evaluation**

* Identification of possible problem-solving approaches (methods)

Describe the approaches you followed, both statistical and analytical, for solving of this problem.

* Testing of Identified Approaches (Algorithms)

DecisionTreeClassifier

KNeighborsClassifier

SVC

LogisticRegression

AdaBoostClassifier

* Evaluate selected models

The model has been trained, tested, and optimized using training and test subsets of the data. I will use an unseen holdout subset of the data to evaluate the model.

The F1 Score on the holdout data is 0.8072.

Because its performance is similar to the results obtained in the previous stage, I can confidently say that the model will generalize well to unseen data. Because it is a real world dataset with a huge variety of comments discussing a diverse range of topics and covering situations from informative posts to flame wars, this is probably one of the better scenarios for training a model on text.

The text vectorization strategy using Scikit-Learn’s TfidfVectorizer() class makes the model immune to unseen features, as they will be ignored.

* Key Metrics for success in solving problem under consideration

In order to be able to evaluate the performance of each algorithm, several metrics are defined accordingly, and are discussed briefly in this section

**Confusion Matrix**: It is very informative performance measures for classification tasks. Ci,j an element of matrix tells how many of items with label i are classified as label j. Ideally we are looking for diagonal Confusion matrix where no item is miss-classified. The matrix in Figure 1 is a good representation for our binary classification. Positive (P) represents toxic label and n (negative) represents non-toxic label

**Accuracy:** This metric measures how many of the comments are labeled correctly. However, in our data set, where most of comments are not toxic, regardless of performance of model, a high accuracy was achieved

recision := T P + T N/ N’+ P’

**Precision and Recall**: Precision and recall in were designed to measure the model performance in its ability to correctly classify the toxic comments. Precision explains what fraction of toxic classified comments are truly toxic, and Recall measures what fraction of toxic comments are labeled correctly.

precision := T P/ P Recall := T P /P’

**F Score:** Both Precision and Recall are important for checking the performance of the model. However, implementing a more advanced metric that combines both Precision and Recall together is quite informative and applicable . In this equation, setting β = 1 leads equation to return harmonic mean of Precision and Recall.

FN=False negative

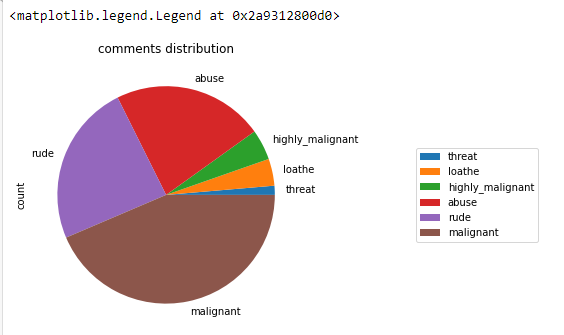
TP=True positive

FP=False positive

TN=True Negative

Visualizations

Data Exploration This dataset contains 159,571 comments from Wikipedia. The data consists of one input feature, the string data for the comments, and six labels for different categories of toxic comments: 'malignant', 'highly malignant', 'rude', 'threat', 'abuse', 'loathe'. The figure on the following page contains a breakdown of how the labels are distributed throughout the dataset, including overlapping data. As you can see in the breakdown, while most comments with other labels are also toxic, not all of them are. Only “Malignant” is clearly a subcategory of “toxic.” And it’s not close enough to be a labelling error. This suggests that “toxic” is not a catch-all label, but rather a subcategory in itself with a large amount of overlap. Because of this, I’m going to create a seventh label called “any label” to represent overall toxicity of a comment. From here on in, I’m going to refer to any labelled comments as toxic, and the specific “Malignant” label (along with other labels) in quotation marks.



**CONCLUSION**

Key Findings and Conclusions of the Study

As a general conclusion for this project, it was very rewarding for us to take part in this competition because it allowed us to use what we had learned in class directly onto real-life problems, but also to make our own research, to investigate and to try to come up with new ideas when we did not find anything that would suit what we wanted to achieve. It was for all of us an interesting first step into the world of research, and it provided us a hands-on experience of what can be done with machine learning techniques in general.

Learning Outcomes of the Study in respect of Data Science

This project allowed me to work with different type of deep learning models and additionally, I was able to implement them on a Natural Language Processing use-case. The various data pre-processing and feature engineering steps in the project made me cognizant of the efficient methods that can be used to clean textual data. I understood the working of various deep-learning models. I got introduced to the concepts of word embedding and the advantages of using pre-trained word embedding

Scope for Future Work

The current project predicts the type or toxicity in the comment. We are planning to add the following features in the future:

→ Analyse which age group is being malignant towards a particular group or brand.

→ Add feature to automatically sensitize words which are classified as toxic.

→ automatically send alerts to the concerned authority if threats are classified as severe.

→ Build a feedback loop to further increase the efficiency of the model.

→ Handle mistakes and short forms of words to get better accuracy of the result.

Thank you