

T1-21-22- AI 511
PROJECT REPORT

ASKREDDIT

December 25, 2021

Fahed Shaikh, Likhiteswar Modulla
IMT2019079, IMT2019511
Shaikh.Fahed@iiitb.ac.in,
Likhiteswar@iiitb.ac.in
IIIT Bangalore

1 Abstract

This section provides an overview about the project. Included with EDA, Text PreProcessing, Features, Performed Models and their respective scores.

2 Introduction

Here is a brief of AskReddit Project

- ***The goal of this project:***

Reddit is a less-well-known social media website built on anonymity and somewhat democratic principles. The probability of your post/comment "going viral" on Reddit is the same irrespective of whether you are a celebrity or if you're a commoner. Each community or "subreddit" has its own topics and rules enforced by subreddit moderators. The admins of Reddit only step in when there's a public controversy or when their Terms of Service are violated.

AskReddit is one of the most popular subreddits. Their official description: r/AskReddit is the place to ask and answer thought-provoking questions.

Of course, not all questions here are thought-provoking. Some are very obvious "troll" questions.

The moderators of AskReddit have provided us with a sample of all the questions they received last year. We have used Machine Learning to design a program, capable of automatically detecting troll questions so that they can be flagged and removed.

- ***The importance of the project:***

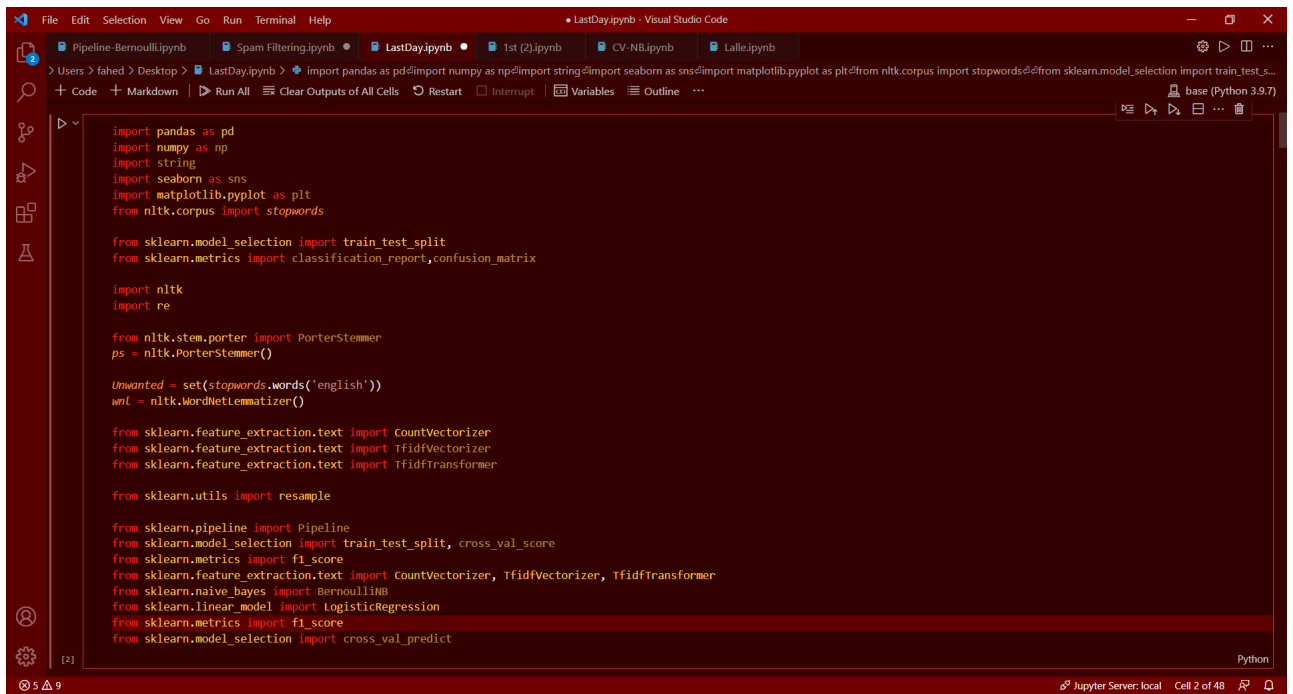
The moderators of AskReddit continuously strive to remove such questions, however, it is not possible to manually inspect every question and do this – thousands of questions are added every day.

- ***Your main contributions to the project:***

We have analyzed the given Data Set and made possible assumptions and explored various methods to provide the best possible efficiency in classification. It is important that no Non-Troll question must go into the Troll question category and be removed because sometimes, important stuff can go waste.

We have tried Bagging and Boosting, Resampling and various attempts to use Word Embeddings have been made.

3 Useful Libraries



```
import pandas as pd
import numpy as np
import string
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

import nltk
import re

from nltk.stem.porter import PorterStemmer
ps = nltk.PorterStemmer()

Unwanted = set(stopwords.words('english'))
wnl = nltk.WordNetLemmatizer()

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

from sklearn.utils import resample

from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import f1_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_predict
```

4 Explanatory Data Analysis

Here is the overview of the Given DataSet:

- The size of the Data Set is *(653061, 3)*
- That is, *653061 Rows* of Questions with *3 columns*.
- Consisting the columns with names '*qid*', '*question_text*', '*target*'
- Target is a binary notation for implying whether the question is a troll or non-troll.
- Here's a preview of the Train Data Set:

<i>qid</i>	<i>question_text</i>	<i>target</i>
a3dee568776c08512c89	What is the role of Lua in Civ4?	0
bdb84f519e7b46e7b7bb	What are important chapters in Kannada for 10 ...	0
29c88db470e2eb5c97ad	Do musicians get royalties from YouTube?	0
3387d99bf2c3227ae8f1	What is the difference between Scaling Social ...	0
e79fa5038f765d0f2e7e	Why do elevators go super slow right before th...	0

- Portraying Samples

⇒ Sample Non Troll Questions:

<i>qid</i>	<i>question_text</i>	<i>target</i>
a3dee568776c08512c89	What is the role of Lua in Civ4?	0
bdb84f519e7b46e7b7bb	What are important chapters in Kannada for 10 ...	0
29c88db470e2eb5c97ad	Do musicians get royalties from YouTube?	0
3387d99bf2c3227ae8f1	What is the difference between Scaling Social ...	0
e79fa5038f765d0f2e7e	Why do elevators go super slow right before th...	0

⇒ Sample Troll Questions:

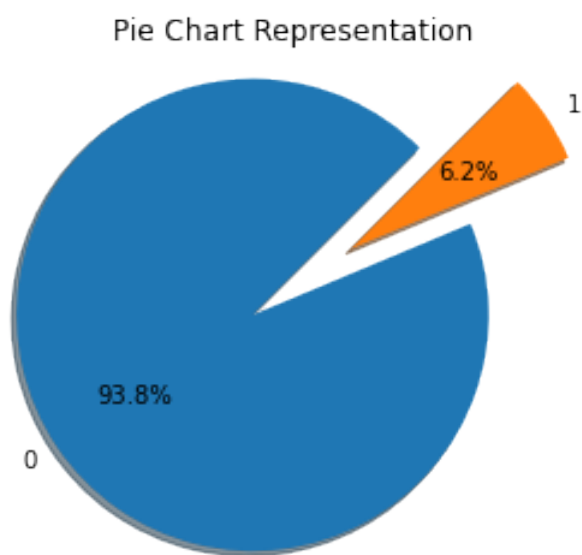
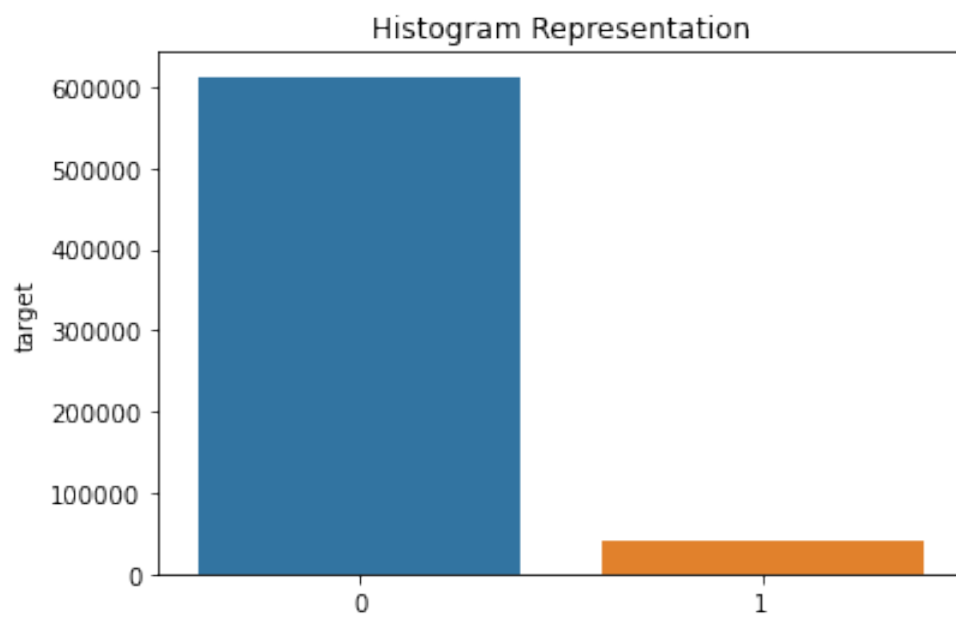
<i>qid</i>	<i>question_text</i>	<i>target</i>
19af3f158b9e37398746	What stupid things do Indians do when in your ...	1
684019b91e57e8354c98	Can I sue my parents for giving birth to me wh...	1
21176c01e8c4c1c75b35	What are your views about sexual relationship ...	1
7103dd22e720543e3362	You became an atheist, and after 2 years you f...	1
24355956c313869bb421	Why aren't we protesting for government contro...	1

- We also checked if there are any NULL values or empty spaces in the data set and here is the result:

<i>qid</i>	653061
<i>question_text</i>	653061
<i>target</i>	2
dtype: int64	

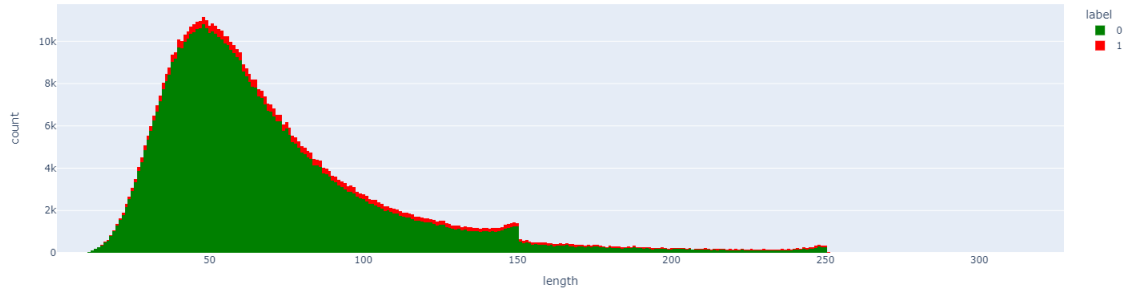
- We checked the number of Troll Questions Vs. Non-Troll Questions in the Train Data:
 - Troll Questions (Target = 1) = 612656
 - Non-Troll Questions (Target = 0) = 40405

- The Graphical Representation of the above data would look like:



This clearly shows the data set is highly imbalanced and needs to be balanced using Sampling.

- We have plotted a graph for Length of the question vs Number of questions.



Where, Green is the count of Non-Troll Questions Vs. Red shows the latter.

- We sorted the questions according to their lengths.

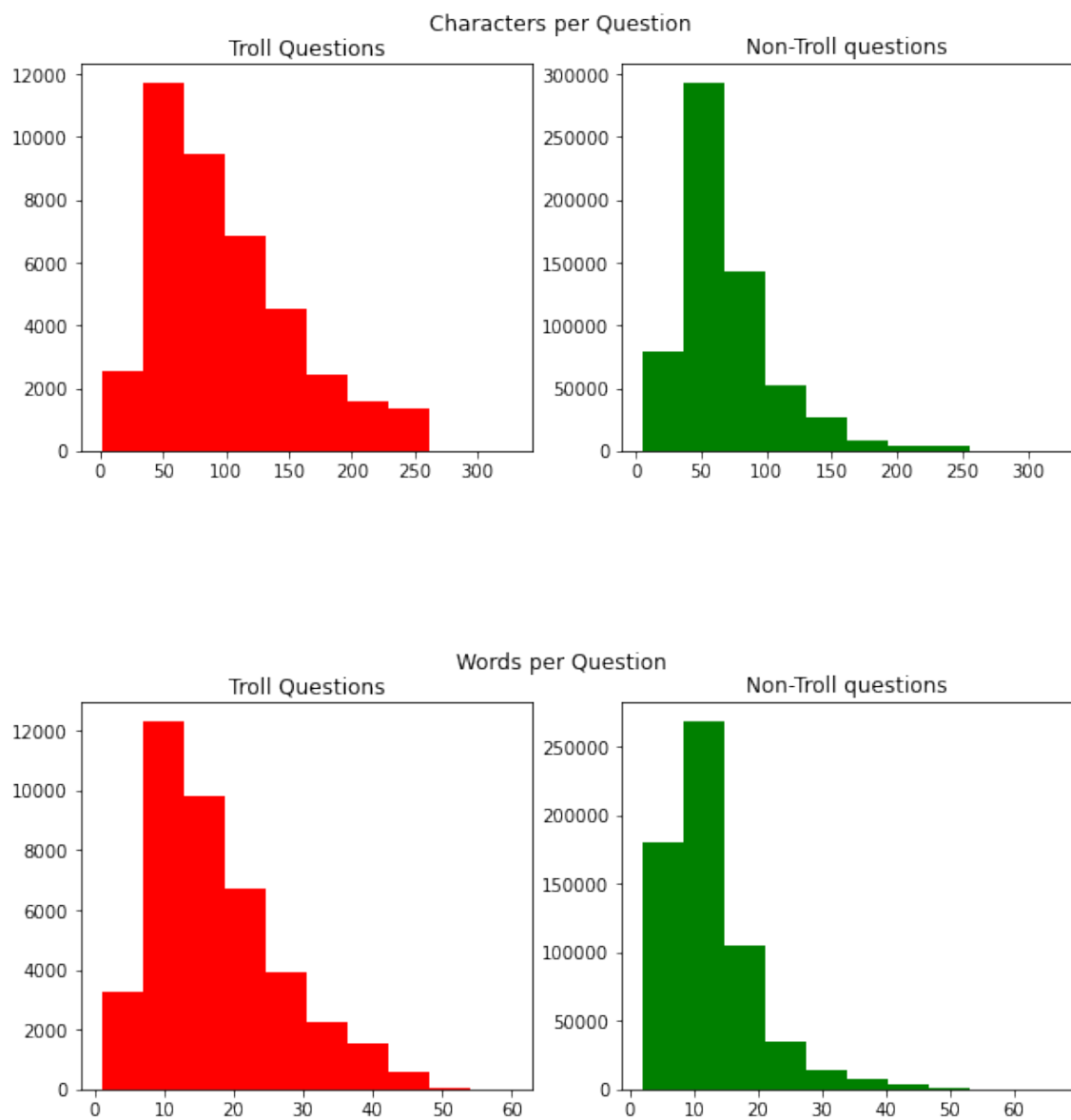
<i>qid</i>	<i>question_text</i>	<i>target</i>	<i>length</i>
527aac2ce6f12f789fe5	”	1	1
3a9ae962f1094242e36f	If	1	3
0f5a41d6752d5d667895	Is	1	3
2cfd7dec2231e47afd6c	I 12?	1	5
0c2a113858db20e0a4db	Quora:	1	7
4a5c932c3b57957e71c8	Islam:	1	7
6adc80c68b1f75e4540e	India:	1	7
1e52e57a821c597eee0c	Dowry:	1	7
83d01336b3406133723e	Bye Bye?	1	8
955bcd9278b7810cd39a	Incest:	1	8

- We calculated the average number of words and characters per question.

<i>Target</i>	<i>Mean Of Words</i>	<i>Mean Of Characters</i>
0	17.283059027348102	98.07437198366539
1	12.505327622678958	68.86373593011413

This shows us that the Troll Questions generally consist lesser words and characters when compared to Non-Troll Questions. We deduced that most of the troll questions are short.

- Let's talk about the count of words and characters present in a question.



- Clearly, most of the troll questions consist lesser words/characters.

5 Text Pre-Processing

The given '*question_text*' contains numerous excessive words and complex stuff which makes it harder for the model to predict. Hence, we perform '*Text Pre-Processing*'

- '*Tokenization*'

This process split the sequence of strings into words. It removes all the punctuations from the text data and gives words of text which is called tokens. Tokens are free of numbers, symbols, emojis and anything other than alphabets. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph.

What is the role of Lua in Civ4? → [What, is, the, role, of, Lua, in, Civ]

- '*Lower Casing*'

This process makes sure all the alphabets used in the text are in Lower Case so that there doesn't occur any disparity while processing it.

[What, is, the, role, of, Lua, in, Civ] → [what, is, the, role, of, lua, in, civ]

- '*Removing Stop Words*'

A stop word is a commonly used word (such as 'the', 'a', 'an', 'in'). We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

[what, is, the, role, of, lua, in, civ] → [role, lua, civ]

- '*Stemming*'

Stemming is a technique used to extract the base form of the words by removing affixes from them. It is just like cutting down the branches of a tree to its stems. For example, the stem of the words eating, eats, eaten is eat. Stemming reduces the size of the index and increases retrieval accuracy.

NLTK has PorterStemmer class with the help of which we can easily implement Porter Stemmer algorithms for the word we want to stem. This class knows several regular word forms and suffixes with the help of which it can transform the input word to a final stem. The resulting stem is often a shorter word having the same root meaning.
royalties

'royalties' → 'royalti'
'elevators' → 'elev'

- **'Lemmatizing'**

is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors. For example, lemmatization would correctly identify the base form of 'caring' to 'care', whereas, stemming would cutoff the 'ing' part and convert it to car.

'Caring' → Lemmatization → 'Care'
'Caring' → Stemming → 'Car'

Also, sometimes, the same word can have multiple different 'lemma's. So, based on the context it's used, you should identify the 'part-of-speech' (POS) tag for the word in that specific context and extract the appropriate lemma.

After all the above processing, we get '***clean_text***' which we would use for our vectorizer and other model uses.

<i>qid</i>	<i>question_text</i>	<i>target</i>
a3dee568776c08512c89	role lua civ	0
bdb84f519e7b46e7b7bb	import chapter kannada ic	0
29c88db470e2eb5c97ad	musician get royalti youtub	0
3387d99bf2c3227ae8f1	differ scale social enterpris social franchis	0
e79fa5038f765d0f2e7e	elev go super slow right door open	0

6 Sampling

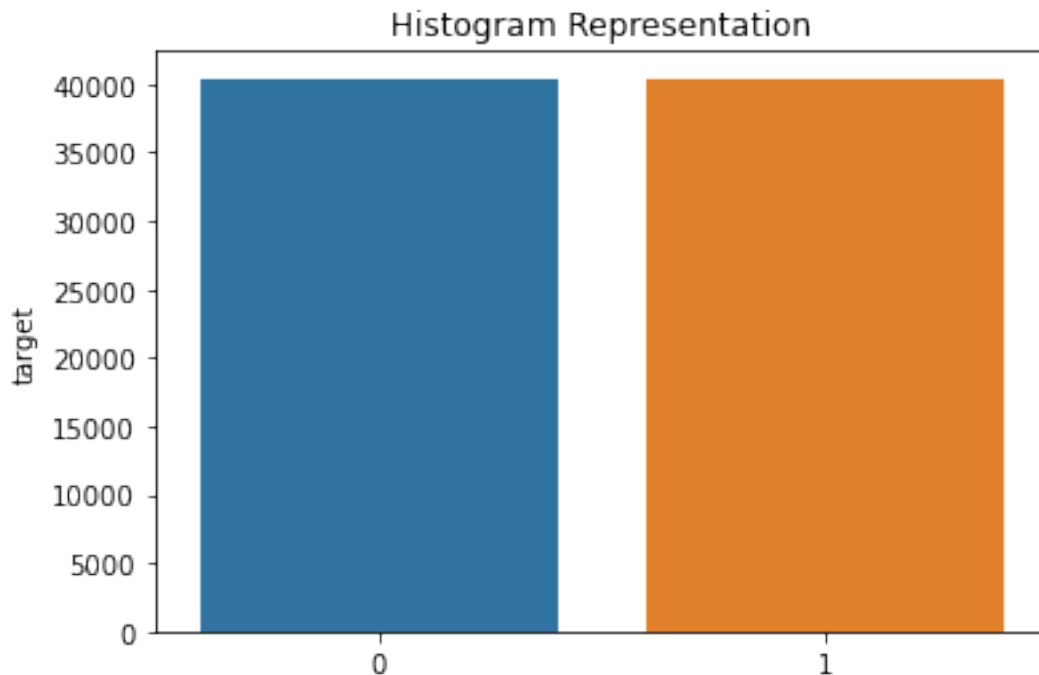
Since we know that the given data is Highly Imbalanced, We use Sampling to deal with it. Sampling makes it easier for the training of the model and it gets simpler as there are equal opportunities for both the target values to make the model experienced.

We have used Undersampling, we made a DataFrame *'Ones'* with all questions corresponding to target value of one. And similarly, *'Zeroes'* contain all the questions whose target value is 0.

```
balanced_df = pd.concat([resample(zeroes, replace=True,  
                                n_samples=len(ones)), ones])
```

We have used the *resample()* function on *'Zeroes'* by restricting it to length of *'Ones'* and *concatenating* the final result to the Array *'Ones'*. This way, now our *'balanced_df'* contains equal number of Troll and Non-Troll Questions.

- 40405 Troll Questions
- 40405 Non-Troll Questions



7 Implementing Bag Of Words

We have made two lists, *train_troll_words* and *train_non_troll_words* which contain all the words in troll questions and in non troll questions respectively.



Now, we build two bags such that, each word common in both the set has a value given by it's count in that set divided by size of the set. This simply tells us where the weightage of a given word is more. If it has higher weightage in `train_troll_words`, the word contributes to the Troll option for the corresponding question and vice-versa.

```

1  FRAC_SPAM_TEXTS = balanced_df.target.mean()
2
3  #get all words from spam and non-spam datasets
4  train_spam_words = ' '.join(balanced_df[balanced_df.target == True].question_text).split(' ')
5  train_non_spam_words = ' '.join(balanced_df[balanced_df.target == False].question_text).split(' ')
6
7  common_words = set(train_spam_words).intersection(set(train_non_spam_words))
8
9
10 train_spam_bow = dict()
11 for w in common_words:
12     train_spam_bow[w] = train_spam_words.count(w) / len(train_spam_words)
13
14 train_non_spam_bow = dict()
15 for w in common_words:
16     train_non_spam_bow[w] = train_non_spam_words.count(w) / len(train_non_spam_words)
17
18
19 valid_words = [w for w in t if w in train_spam_bow]
20
21 #get the probabilities of each valid word showing up in spam and non-spam BOW
22 spam_probs = [train_spam_bow[w] for w in valid_words]
23 non_spam_probs = [train_non_spam_bow[w] for w in valid_words]
24
25
26 spam_score = sum([np.log(p) for p in spam_probs]) + np.log(FRAC_SPAM_TEXTS)
27 non_spam_score = sum([np.log(p) for p in non_spam_probs]) + np.log(1-FRAC_SPAM_TEXTS)

```

8 Vectorization

CountVectorizer tokenizes(tokenization means dividing the sentences in words) the text along with performing very basic preprocessing. It removes the punctuation marks and converts all the words to lowercase. The vocabulary of known words is formed which is also used for encoding unseen text later. An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document

```
vectorizer = CountVectorizer(ngram_range = (1, 3))  
X = vectorizer.fit_transform(balanced_df.question_text.tolist())
```

Combination of words sometimes are more meaningful. Let's say we have words 'sunny' and 'day', 'sunny day' combined makes more sense. This is bigram. We can also use character level and word level n-grams. `ngram_range=(1,2)` specifies we want to consider both unigrams(single words) and bigrams(a combination of 2 words).

Tf-Idf vectorization has been notably reducing the score, hence did not use it in the final code.

9 Training Model

Used the *train_test_split* from *sklearn.model_selection* to split the Train Data Set into two parts, for training and testing.

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2,  
random_state= 0)
```

10 Pipelining

A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model. Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment

We tried the concept of Pipelining with parameters of

```
pipeline_model = Pipeline([('vect', CountVectorizer()),  
                           ('tfidf', TfidfTransformer()),  
                           ('clf', MultinomialNB())])
```

But later removed Pipeline because the dataset showed better results without using the 'TfidfTransformer()'. Hence, implemented only 'CountVectorizer'.

11 Model Selection

We have tried various models like:

1. Logistic Regression
2. Naive Bayes Classifier (Both MultinomialNB and BernoulliNB)
3. Decision Trees Classifier
4. SVM
5. SGD
6. Bag of Words algorithm

Tried all these with possible combinations of Pipelining and Tf-Idf

Out of which, Logistic Regression with optimum parameters showed more promising result and a highest score of **0.61439**. And the bag of words algorithm stood second with a fBeta score of **0.56974**.

12 All Results

<i>Method</i>	SCORE
Logistic Regression with Sampling	0.61439
Bag Of Words No Sampling	0.57796
Naive Bayes Without Tf-Idf	0.56463
Random Forest Without sampling	0.52906
Bernoulli with Pipeline, CV and Tf-Idf	0.51685
Logistic with Pipeline, CV and Tf-Idf	0.50585

13 Contributions

EDA and Vectorization done by Likhiteshwar. Text Pre-processing, Sampling, Bagging, Embedding, Various models like Logistic Regression, Naive Bayes Classifier, Decision Trees Classifier, SVM, SGD done by Fahed Shaikh.