



AI 511 - Machine Learning

Project Report - Ask Reddit

Team name: Bug Slayers

Divyanshi Rajput (IMT2019029) and Nikhil Agarwal (IMT2019060)

Task

To create a model capable of automatically detecting **troll questions** so that they can be flagged and removed.

The Wikipedia entry for Internet trolls/ trolling:

In internet slang, a troll is a person who posts inflammatory, insincere, digressive, extraneous, or off-topic messages in an online community (such as social media (Twitter, Facebook, Instagram, etc.), a newsgroup, forum, chat room, or blog), with the intent of provoking readers into displaying emotional responses, or manipulating others' perception. This is typically for the troll's amusement, or to achieve a specific result such as disrupting a rival's online activities or manipulating a political process.

Data Observation

Training dataset:

Total records - 653061

Feature columns: question_text

question id: qid

Class: Target

Test dataset:

Total records - 653061

Feature columns: question_text

question id: qid

EDA

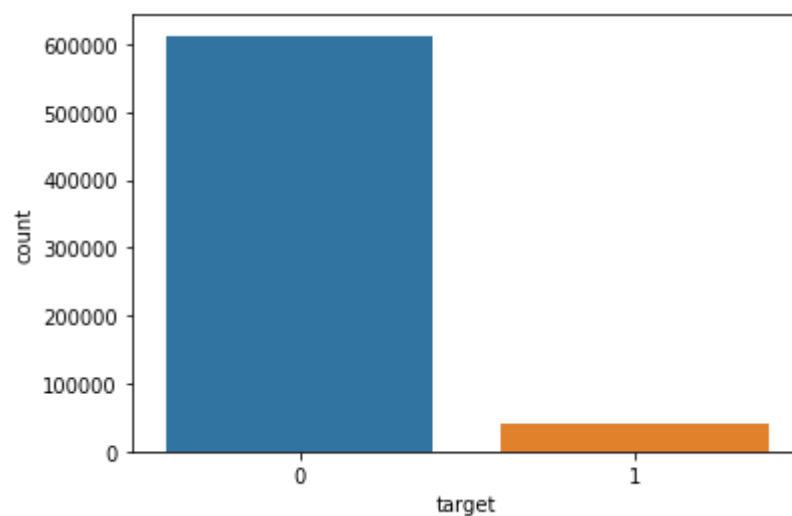
Target Class Distribution

Dataset contains less number of 1's i.e questions which are troll. So while training Models we need to account this difference (example we can give class weight parameter in logistic regression)



Target == 1: 40405

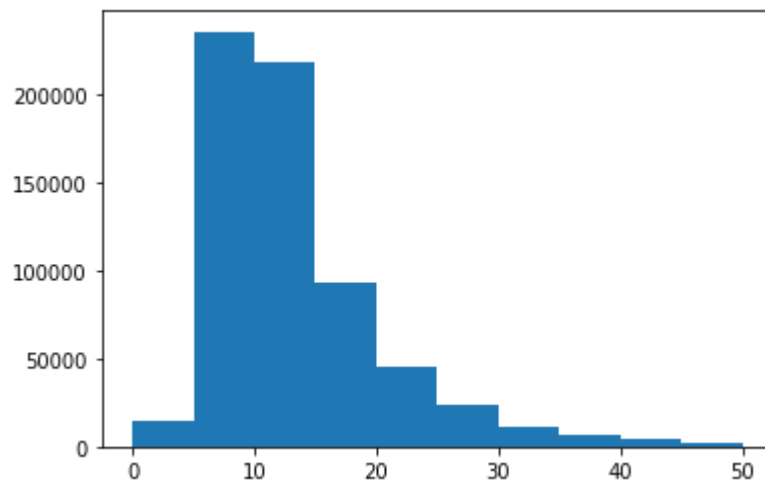
Target == 0: 612656



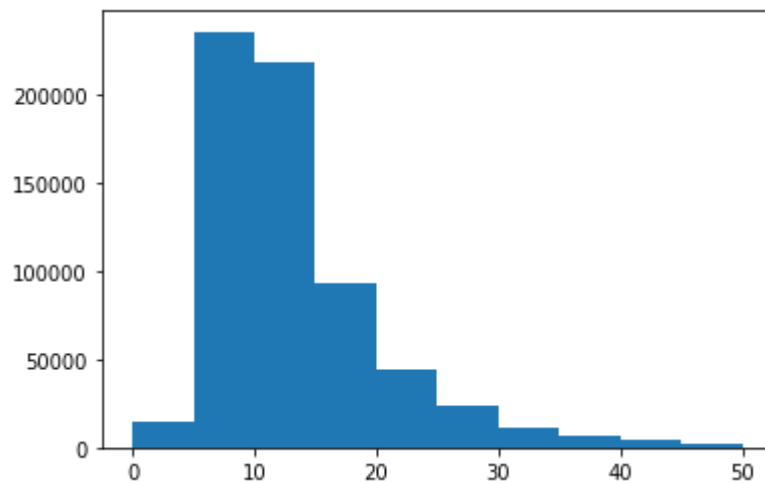
Number of words in a question vs Frequency

We get a very similar graph for train and test dataset. This means we can preprocess train and test dataset in a similar manner.

Train dataset

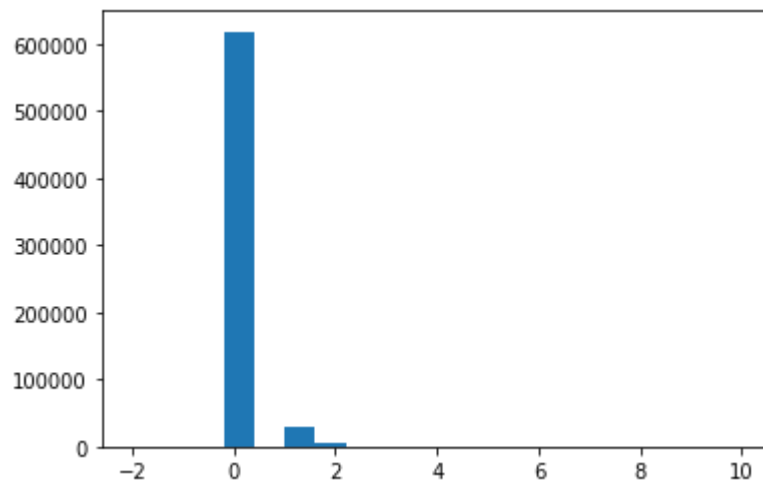


Test dataset



Number of numerics vs Frequency

Only a very small portion of dataset has numerics, so it would be better to remove all the numerical values during preprocessing.



Preprocessing

We performed the following cleaning steps to the 'question_text' column of a combine dataset (train + test). *We combined the dataset to get more corpus to feed Count and Tfidf Vectorizer.*

- **Word tokenization:** To remove unnecessary characters, punctuations.
- **Stop word removal:** Removed stop words which are a part of the sklearn.feature_extraction.text package. The text is also now converted to **lowercase**.
- Removed **numeric values** using Regex Tokenizer
- **Lemmatization** using WordNetLemmatizer

After performing this round of cleaning, we checked for null values in the 'question_text' column and handle them accordingly.

Sample preprocessed Text

- What is the role of Lua in Civ4? → role lua civ
- What are important chapters in Kannada for 10 ICSE 2018? → important chapter kannada icse
- Do musicians get royalties from YouTube? → musician royalty youtube
- What is the difference between Scaling Social Enterprises and Social Franchising? → difference scaling social enterprise social franchising

Approach - Bag of Words

With the combined corpus, we fitted SKLearn CountVectorizer with different parameters.

n-grams = { (1,1), (1,2), (1,3), (1,4) } (uni-grams, bi-grams, tri-grams)

min_df = 2

The result from TfidfVectorizer was less than or equal to that of CountVectorizer. Therefore, we used CountVectorizer.

Approach - Word Embeddings

We also tried to implement word embeddings using word2vec. But vectorising each and every sentence was taking a lot of processing time (~ 1 hr for 1 lakh rows), so we decided to not go with this approach.

Model Selection

We implemented the following models:

- Gaussian Naive Bayes
- Logistic Regression
- Support Vector Classifier
- XGBClassifier
- MLPClassifier

The best result was obtained from Logistic Regression model. Later on, we hyper tuned its parameters to obtain better scores (results).

Results - Top 3 submissions

1

Preprocessing - lowercase, remove numbers, punctuation and stop words, lemmatization

CountVectorizer(ngram_range = (1,3), min_df = 2, strip_accents = 'ascii')

LogisticRegression(class_weight = {0:1, 1:1.5}, max_iter=1000, solver="lbfgs", penalty="l2")

Predict Probabiliy **Threshold** = 0.20

2

Preprocessing - convert text to lowercase

CountVectorizer(ngram_range = (1,3), min_df = 2, strip_accents = 'ascii')

LogisticRegression(C= 0.4, max_iter=1000, solver="lbfgs", penalty="l2")

Predict Probabiliy **Threshold** = 0.37

3

Preprocessing - None

CountVectorizer(ngram_range = (1,4), min_df = 2)


LogisticRegression(C= 0.4, max_iter=1000, solver="lbfgs", penalty="l2")

Predict Probabiliy **Threshold** = 0.42

Submissions - Colab Links

1st


Google Colaboratory

 <https://colab.research.google.com/drive/1XeBS826P9na03j7WkBKy6O6kPXNSN6R9?usp=sharing>



2nd


Google Colaboratory

 <https://colab.research.google.com/drive/1F-nb7UFpLSQa0aJP4TqPTR0Q6je8CpnX?usp=sharing>



3rd

Google Colaboratory

 <https://colab.research.google.com/drive/1om6KHbP7l3LJVhGRuaO9KDV63YpvyYgy?usp=sharing>



Slides - Drive link



Ask Red https://docs.google.com/presentation/d/1OnHBbeJp4_r2ZbD4dtMT_diVGp4FIJ_iQ5cQVi9-BjE/edit?usp=drivesdk