

MACHINE LEARNING - PROJECT REPORT

YAHOO TROLL QUESTION DETECTION

TEAM - Kuch_toh_karunga

CHAITHANYA REDDY IMT2020054

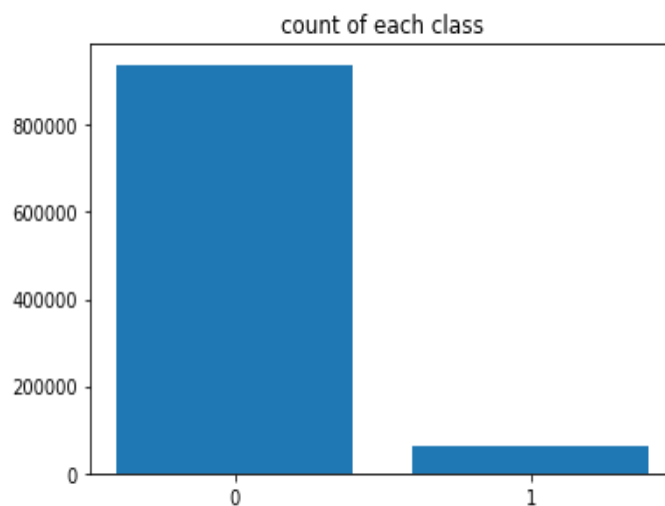
SAMARTH GATTU - IMT2020062

Problem Statement

Detecting troll questions. Classify the question as '1' if it is a troll, otherwise classify it as '0' if it is not a troll.

Data Analysis

The given training set has 1 million questions and the test dataset has 0.3 million questions. Data contains letters, numbers, punctuation marks, and special characters. Data also contains few words of different languages (other than english - like arabic, japanese, etc). It contains HTML tags and URLs. It contains both uppercase and lowercase letters. Sentences contain contractions (I'm here instead of I am here). There is a lot of bias in target value: most of the target values are '0' (not trolls).



Preprocessing steps:

- Removed URL/HTML tags (https://)
- Removed all special characters and numbers (#,%,...)
- Expanded contractions (We'll → We will)
- Changed all the letters to lowercase
- Tokenized the sentences. (breaking sentences into words)
- Removed stop words. (Insignificant words in troll detection)
- Lemmatization (converting every word into its base form using parts of speech)
- Corrected spelling mistakes.
- A new column is added for Profanity of sentences (for every sentence we calculated profanity- index/offensiveness of the sentence)

Vectorizer

The process of converting words into the numbers is called vectorization and we use vectorizer for this process. We used Count-vectorizer and TF-IDF (term frequency - inverse document frequency) to vectorize the dataset. Tried different values of maxfeatures and ngrams. ngram(1,2) and ngram(1,3) gave the best f1_score.

Sampling

The given dataset has bias in the target value. Number of 0's are 10 times more than the number of 1's in the training dataset. This bias in the training dataset may influence the ml models to ignore the minority class which is a problem because typically the minority class on which predictions are most important. To overcome this problem we can use the sampling techniques.

- Over Sampling

This method randomly selects samples from the minority class and duplicates these randomly selected samples.

- Under Sampling

This is exactly the opposite of over sampling. This method randomly selects and removes the samples from the majority class.

Training Models:

- Logistic Regression
- Perceptron
- XGBoost
- Ridge Classifier
- KNN

For each of the above models we have tried both Count vectorization and tf-idf vectorization.

Model	Count-Vectorizer	Tf-Idf
Logistic Regression	0.5892587486048191	0.5237924682767089
Perceptron	0.49960274804879184	0.46995762928173973
Ridge Classifier	0.5182186234817814	0.558688819039066
Xg-Boost	0.5020454545454546	0.5006962007331419

Logistic Regression gave the best f1_score when compared to other classification models. Count vectorizer gave a better f1_score for all the classification models when compared to other vectorizers.

Results and observations:

OVERFITTING

Oversampling and Undersampling were giving very high f1_score on training data (more than 90%) but gave 58-62% f1_score on test_data.

PREPROCESSING STEPS

All the preprocessing steps described above gave less f1_score when compared to the model which is trained on plane text. So, we tried testing without preprocessing the dataset.

Logistic Regression	Count-Vectorizer
With preprocessing, multiple ML models with default parameters (logistic regression, ridge regression, perceptron, XgBoost)	Best – 0.60158 (logistic regression)
With preprocessing (including sampling), ngram_range(1,3), classweights(0.15,0.85)	0.61122
Without preprocessing (only countvectorizer), ngram_range(1,2), classweights(0.15,0.85)	0.64395 (BEST)
Without preprocessing (only countvectorizer), ngram_range(1,2), classweights(0.10,0.90)	0.63813
Without preprocessing (only countvectorizer), ngram_range(1,2), balanced weights	0.60623

Logistic Regression:

Since logistic regression gave a very high f1_score on both pre-processed and plain text when compared with other models, we continued training in various ways fixing logistic regression as the final/best classification model.

Fixed count-vectorization as the final vectorizer. Tried count vectorization in multiple methods using various ngram ranges (ex: ngram(1,2) bigram, ngram(1,3) trigram, etc.)

Parameters used for logistic regression:

1. class weights (balanced class weights and manual assignment of weights)
2. predict() and predict_proba() for classifying a given sentence.

Logistic Regression	Count-Vectorizer
With preprocessing, multiple ML models with default parameters (logistic regression, ridge regression, perceptron, xgboost)	Best - 0.60158 (logistic regression)
With preprocessing (including sampling), ngram_range(1,3), classweights(0.15,0.85)	0.61122
Without preprocessing (only countvectorizer), ngram_range(1,2), classweights(0.15,0.85)	0.64395 (BEST)
Without preprocessing (only countvectorizer), ngram_range(1,2), classweights(0.10,0.90)	0.63813
Without preprocessing (only countvectorizer), ngram_range(1,2), balanced weights	0.60623

Hyper parameter Tuning:

We used a tool called “optuna” which is a framework for automating the optimization process of the hyper parameters. This tool is very similar to the gridCV. Optuna automatically searches and finds the optimal hyper parameters values by trial and error.

First observation:

We tried hyper parameter tuning on the parameters:

1. ngram()
2. class_weights()

The result showed high f1_scores for ngram(1,2) and ngram(1,3).

```

76 finished with value: 0.6095695482034863 and parameters: {'zero_weight': 0.17983704240438395, 'one_weight': 0.39834578055150544, 'ng_first': 1, 'ng_second': 3}. Best is trial 72 with value: 0.6419743782969
77 finished with value: 0.6297709923664122 and parameters: {'zero_weight': 0.05989772827518757, 'one_weight': 0.5454337155779243, 'ng_first': 1, 'ng_second': 2}. Best is trial 72 with value: 0.6419743782969
78 finished with value: 0.4075556078615982 and parameters: {'zero_weight': 0.9650023418559937, 'one_weight': 0.4857858802701961, 'ng_first': 2, 'ng_second': 2}. Best is trial 72 with value: 0.6419743782969
79 finished with value: 0.6408329837141005 and parameters: {'zero_weight': 0.12474118121437315, 'one_weight': 0.5610201118049515, 'ng_first': 1, 'ng_second': 2}. Best is trial 72 with value: 0.6419743782969
80 finished with value: 0.6348386038308519 and parameters: {'zero_weight': 0.14802790269587135, 'one_weight': 0.6022433809223758, 'ng_first': 1, 'ng_second': 3}. Best is trial 72 with value: 0.6419743782969
81 finished with value: 0.633291715332877 and parameters: {'zero_weight': 0.13215536080602493, 'one_weight': 0.42595244811301836, 'ng_first': 1, 'ng_second': 2}. Best is trial 72 with value: 0.6419743782969
82 finished with value: 0.6420049523523674 and parameters: {'zero_weight': 0.11730015838439879, 'one_weight': 0.5736501379989539, 'ng_first': 1, 'ng_second': 2}. Best is trial 82 with value: 0.6420049523523
83 finished with value: 0.6320785331818919 and parameters: {'zero_weight': 0.17837284799553715, 'one_weight': 0.5685697040149952, 'ng_first': 1, 'ng_second': 2}. Best is trial 82 with value: 0.6420049523523
84 finished with value: 0.6152617841240596 and parameters: {'zero_weight': 0.05003730371401667, 'one_weight': 0.6353228851985259, 'ng_first': 1, 'ng_second': 2}. Best is trial 82 with value: 0.6420049523523
85 finished with value: 0.6186056550830995 and parameters: {'zero_weight': 0.23945860107441358, 'one_weight': 0.5292353133066244, 'ng_first': 1, 'ng_second': 2}. Best is trial 82 with value: 0.6420049523523
86 finished with value: 0.62592360676268 and parameters: {'zero_weight': 0.20923850259710208, 'one_weight': 0.5650010588325503, 'ng_first': 1, 'ng_second': 2}. Best is trial 82 with value: 0.642004952352367

```

Second observation:

From the first observation it is clear that we can only train models using ngram(1,2) and ngram(1,3). So, we modified the hyper parameters to:

1. classweights()
2. predict_proba(): (useful for setting up probability threshold for classifying a data point)

```
155 : finished with value: 0.638800785379852 and parameters: {'zero_weight': 0.07974127401848945, 'one_weight': 0.33545965812108103, 'probability': 0.4330696538751615} Best is trial 95 with value: 0.63966584
156 : finished with value: 0.6378428280186175 and parameters: {'zero_weight': 0.07650702462129144, 'one_weight': 0.3104139685927387, 'probability': 0.4493706504238292} Best is trial 95 with value: 0.63966584
157 : finished with value: 0.6378933566433566 and parameters: {'zero_weight': 0.05619500027269439, 'one_weight': 0.2702413987242175, 'probability': 0.43421000426197376} Best is trial 95 with value: 0.63966584
158 : finished with value: 0.6311734052501825 and parameters: {'zero_weight': 0.10091224743672526, 'one_weight': 0.34171653045869865, 'probability': 0.4676319924117845} Best is trial 95 with value: 0.63966584
159 : finished with value: 0.6395661929693345 and parameters: {'zero_weight': 0.08173447284168495, 'one_weight': 0.3614293798113334, 'probability': 0.42833311699403354} Best is trial 95 with value: 0.63966584
160 : finished with value: 0.6353192837902076 and parameters: {'zero_weight': 0.11275699611210588, 'one_weight': 0.365020665579366, 'probability': 0.42922346184869103} Best is trial 95 with value: 0.63966584
161 : finished with value: 0.6397368615566561 and parameters: {'zero_weight': 0.0843683326353982, 'one_weight': 0.39255923965226237, 'probability': 0.42105959647081376} Best is trial 159 with value: 0.63973686
162 : finished with value: 0.6269240976645436 and parameters: {'zero_weight': 0.05126615339515267, 'one_weight': 0.3984613783536707, 'probability': 0.41734944783904526} Best is trial 159 with value: 0.63973686
163 : finished with value: 0.6376922196359751 and parameters: {'zero_weight': 0.08764235223670484, 'one_weight': 0.382074780274622, 'probability': 0.4447294478877793} Best is trial 159 with value: 0.63973686
164 : finished with value: 0.6350164872273647 and parameters: {'zero_weight': 0.1154642027553905, 'one_weight': 0.3273950658918494, 'probability': 0.41120038430322736} Best is trial 159 with value: 0.63973686
165 : finished with value: 0.6382731725013693 and parameters: {'zero_weight': 0.06992291863906721, 'one_weight': 0.36263252249866684, 'probability': 0.4286484686348746} Best is trial 159 with value: 0.63973686
166 : finished with value: 0.5953630999395995 and parameters: {'zero_weight': 0.4319746029659072, 'one_weight': 0.38973582181157423, 'probability': 0.40232821322141293} Best is trial 159 with value: 0.63973686
167 : finished with value: 0.6352653190145294 and parameters: {'zero_weight': 0.0920720860673035, 'one_weight': 0.3445761680172483, 'probability': 0.4567987045160609} Best is trial 159 with value: 0.63973686
168 : finished with value: 0.6384455763063572 and parameters: {'zero_weight': 0.10237123647945298, 'one_weight': 0.4230076295460571, 'probability': 0.41670279998824694} Best is trial 159 with value: 0.63973686
169 : finished with value: 0.6291473125414732 and parameters: {'zero_weight': 0.12012599609975551, 'one_weight': 0.2966977820205888, 'probability': 0.43589356278604711} Best is trial 159 with value: 0.63973686
```

Using this hyper parameter tuning on the following parameters:

1. ngram()
2. class_weights()
3. predict_proba()









Final Model










We have finalized our parameter values (approximately) as:

1. ngram(1,3)
2. class_weights(0:0.117,1:0.685)
3. predict_proba(>0.425) for being a troll

Logistic Regression ngram(1,3)	f1_score
With preprocessing (including sampling), ngram_range(1,3), classweights(0.15,0.85)	0.61122
classweights(0.117,0.753) predict_proba(>0.63) [without preprocessing]	0.62797
classweights(0.117,0.753) predict_proba(>0.45) [without preprocessing]	0.64861
classweights(0.117,0.685) predict_proba(>0.425) [without preprocessing]	0.64902 (BEST)

Kaggle submission:

	13_0.475_0.11730015838439879_0.7116501379989539.csv Complete · samarth gattu · 3d ago	0.6384	0.64673	<input type="checkbox"/>
	13_0.415_0.11730015838439879_0.7116501379989539.csv Complete · samarth gattu · 3d ago	0.63697	0.64693	<input type="checkbox"/>
	13_0.44_0.11730015838439879_0.7516501379989539.csv Complete · samarth gattu · 3d ago	0.63685	0.64844	<input type="checkbox"/>
	13_0.485_0.11730015838439879_0.8016501379989539.csv Complete · samarth gattu · 3d ago	0.6379	0.64723	<input type="checkbox"/>
	13_0.465_0.11730015838439879_0.8016501379989539.csv Complete · samarth gattu · 3d ago	0.63744	0.64841	<input type="checkbox"/>
	13_0.43_0.11730015838439879_0.8016501379989539.csv Complete · samarth gattu · 3d ago	0.63701	0.64649	<input type="checkbox"/>
	13_0.445_0.11730015838439879_0.8016501379989539.csv Complete · samarth gattu · 3d ago	0.63662	0.64839	<input type="checkbox"/>
	13_0.445_0.11730015838439879_0.8216501379989539.csv Complete · samarth gattu · 4d ago	0.63695	0.6481	<input type="checkbox"/>

	13_0.4262_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63771	0.64875	<input type="checkbox"/>
	13_0.425_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63762	0.64902	<input type="checkbox"/>
	13_0.423_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63747	0.64866	<input type="checkbox"/>
	13_0.43_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63727	0.6484	<input type="checkbox"/>
	13_0.42_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63719	0.64871	<input type="checkbox"/>
	13_0.42_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63737	0.64825	<input type="checkbox"/>
	13_0.428_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63769	0.64864	<input type="checkbox"/>
	13_0.425_0.11730015838439879_0.6856501379989539.csv Complete · samarth gattu · 2d ago	0.63762	0.64902	<input type="checkbox"/>
	13_0.42_0.11730015838439879_0.6716501379989539.csv Complete · samarth gattu · 2d ago	0.63771	0.64873	<input type="checkbox"/>