TOXIC COMMENT CLASSIFIERS

NOPPHRAKORN KERDSAMUT

A SENIOR PROJECT SUBMITTED IN
PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF SCIENCE
(PHYSICS)
MAHIDOL UNIVERSITY INTERNATIONAL COLLEGE
MAHIDOL UNIVERSITY
2018

COPYRIGHT OF MAHIDOL UNIVERSITY

Senior Project

entitled

TOXIC COMMENT CLASSIFIERS

was submitted to the Mahidol University International College, Mahidol University
for the degree of Bachelor of Science (Physics)
on
April 2018

.....
Nopphrakorn Kerdsamut
Candidate

.....
Dr. Piti Ongmongkolkul, Ph.D.
Advisor

.....
Assoc. Prof. Pakorn Bovonsombat, Ph.D
Chair of Science Division
Mahidol University International College
Mahidol University

.....
Dr. Tara Chalermsongsak, Ph.D.
Program Director
Bachelor of Science in Physics
Mahidol University International College
Mahidol University

ACKNOWLEDGEMENTS

This project will absolutely not be completed without the continuous support and guidance from my advisor, Dr. Piti Ongmongkulkol. This project along with his guidance pushed me to learn something completely new and eye opening experience. I want to give a huge appreciation to everyone who are there during my breakdowns. I would also want to pay gratitude to all my Ajarns for always pushing me to become a better student. Lastly, this project is dedicated to my friends and especially my family whom always believe in me.

Nopphrakorn Kerdsamut

TOXIC COMMENT CLASSIFIERS.

NOPPHRAKORN KERDSAMUT 5680928 ICPY/B

B.Sc. (PHYSICS)

SENIOR PROJECT ADVISORS : PITI ONGMONGKOLKUL, (PHYSICS)

ABSTRACT

This project builds four different classifiers to predict the probabilities of different toxic comments. The categories of toxic comments being classified are: toxic, severe toxic, obscene, threat, insult and identity hate. The first model implements Naive Bayes which focuses on the frequency of the words to predict each probabilities. The second model uses logistic regression which uses result from Naive Bayes, proportion of capital letters and proportion of special characters to predict the probabilities. The third model also uses logistic regression, but this time it focuses on the frequency of the words. The fourth model is a boost of the third model. The model with the most effective performance, quantified by the area under the ROC curve of each categories, is the third model, logistic regression that focuses on frequency of each word.

KEY WORDS : MACHINE LEARNING, CLASSIFICATION

38 pages

CONTENTS

CHAPTER 1

INTRODUCTION

The internet has undoubtedly become a major platform for discussions. Discussing things you care about can be difficult, as people can anonymously harass you online. As a result, people avoid expressing themselves because of cyberbullying. In addition, online platforms are struggling to facilitate conversations without harassment, leading to completely disabling community comments.

This project tries to understand the behavior of negative comments and implement it on different types of classifiers to identify them. This can be extremely beneficial to the online community, as it can encourage people to exchange ideas without the fear of being abused.

This project is inspired from a challenge in `www.kaggle.com` and all the data are from `www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data`.

CHAPTER 2

LITERATURE REVIEW

Classifying problems is a central topic in machine learning. It involves teaching machines to learn from *features* of the data and (hopefully) classify such data in their correct category or *classes*. To build a classifier, we first need to *train* machine about the classes and features from our training data set. Then we can feed in new data into the machine and let it make predictions.

2.1 Naive Bayes

Naive Bayes is a classifier the is based from Bayes Theorem and naive (conditionally independence) assumption. Recalling conditional probability definition,

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{2.1}$$

similarly,

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \tag{2.2}$$

Rewriting eqn 2.2 as $P(A \cap B) = P(B|A) \times P(A)$ and plug it in equation 2.1, we have

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \tag{2.3}$$

This is Bayes Theorem.

Suppose we have a four *features*, x_1, x_2, x_3, x_4, to predict the outcome of the *class*, C. The conditional probability that we want to find is

$$P(C|x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1 \cap x_2 \cap x_3 \cap x_4 \cap C)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)} \tag{2.4}$$

The nominator term $P(x_1 \cap x_2 \cap x_3 \cap x_4 \cap C)$ can be written (from eqn 2.1) as:

$$= P(x_1|x_2 \cap x_3 \cap x_4 \cap C) \times P(x_2 \cap x_3 \cap x_4 \cap C)$$

We can keep rewriting the trailing term as

$$\begin{aligned}
&= P(x_1|x_2 \cap x_3 \cap x_4 \cap C) \times P(x_2 \cap x_3 \cap x_4 \cap C) \\
&= P(x_1|x_2 \cap x_3 \cap x_4 \cap C) \times P(x_2|x_3 \cap x_4 \cap C) \times P(x_3 \cap x_4 \cap C) \\
&= P(x_1|x_2 \cap x_3 \cap x_4 \cap C) \times P(x_2|x_3 \cap x_4 \cap C) \times P(x_3|x_4 \cap C) \times P(x_4 \cap C) \\
&= P(x_1|x_2 \cap x_3 \cap x_4 \cap C) \times P(x_2|x_3 \cap x_4 \cap C) \times P(x_3|x_4 \cap C) \times P(x_4|C) \times P(C)
\end{aligned} \tag{2.5}$$

Plugging it back into the eqn 2.4 we want, we get

$$P(C|x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1|x_2 \cap x_3 \cap x_4 \cap C) \times P(x_2|x_3 \cap x_4 \cap C) \times P(x_3|x_4 \cap C) \times P(x_4|C) \times P(C)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)} \tag{2.6}$$

When our features are independent of each other, we can apply *naive assumption* to multiple probabilities to simplify it as:

$$P(C|x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1|C) \times P(x_2|C) \times P(x_3|C) \times P(x_4|C) \times P(C)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)} \tag{2.7}$$

Similarly, we can derive the probability of an outcome to be $\neg C$,

$$P(\neg C|x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1|\neg C) \times P(x_2|\neg C) \times P(x_3|\neg C) \times P(x_4|\neg C) \times P(\neg C)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)} \tag{2.8}$$

Since the values for C is either true or false, the probability of eqn 2.7 and eqn 2.8 should add to 1. We can combine both equations as

$$\begin{aligned}
1 &= P(C|x_1 \cap x_2 \cap x_3 \cap x_4) + P(\neg C|x_1 \cap x_2 \cap x_3 \cap x_4) \\
1 &= \frac{1}{P(x_1 \cap x_2 \cap x_3 \cap x_4)} \Big[\big(P(x_1|C) \times P(x_2|C) \times P(x_3|C) \times P(x_4|C) \times P(C)\big) \\
&\quad + \big(P(x_1|\neg C) \times P(x_2|\neg C) \times P(x_3|\neg C) \times P(x_4|\neg C) \times P(\neg C)\big) \Big]
\end{aligned} \tag{2.9}$$

$$P(x_1 \cap x_2 \cap x_3 \cap x_4) = \big(P(x_1|C) \times P(x_2|C) \times P(x_3|C) \times P(x_4|C) \times P(C)\big) + \big(P(x_1|\neg C) \times P(x_2|\neg C) \times P(x_3|\neg C) \times P(x_4|\neg C) \times P(\neg C)\big) \tag{2.10}$$

We can plug in eqn 2.10 into our denominator of eqn 2.7.

$$P(C|x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1|C) \times P(x_2|C) \times P(x_3|C) \times P(x_4|C) \times P(C)}{[\big(P(x_1|C) \times P(x_2|C) \times P(x_3|C) \times P(x_4|C) \times P(C)\big) + \big(P(x_1|\neg C) \times P(x_2|\neg C) \times P(x_3|\neg C) \times P(x_4|\neg C) \times P(\neg C)\big)]} \tag{2.11}$$

Now let's understand what each term means. $P(x_k|C)$ is the conditional probability of x_k given it is in class C. Similarly, $P(x_k|\neg C)$ is the conditional probability of x_k given it is in class $\neg C$. $P(C)$ is the probability of a element to be in class C and vice versa. This term is called the *prior* or the initial belief of the probability distribution. If we flip a fair coin, we know that $P(Heads)$ and $P(Tails)$ will each be 0.5. However, suppose we don't know the probability distribution of our data, then we should believe our training data to be a decent depiction of the probability distribution.

2.2 Logistic Regression

Logistic Regression is a classifying method for analyzing dataset with one or more independent variables that determine their binary outcomes. Often times, when a data point is in a certain *class*, it usually has a distinguishable *features* compared to a data point that is in a different *class*. For instance, students passed a tough exam would studied and did their homework relatively more than students who failed the exam. We can plot each features on x,y axis as

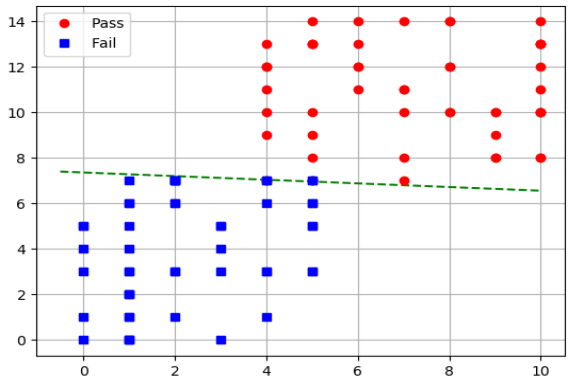


Figure 2.1: Students pass/failing an exam

The green linear line is a decision boundary that separate the class of a data point. Logistic regression is used to determine the coefficients of a decision boundary that will classify the class of a data point.

2.2.1 Notation

Suppose we hypothesized a set of features that will affect the outcome of a data. We can write the set of features and their coefficients as a vector and their classes for a data point as

$$\begin{aligned} \vec{x}^{(i)} &= (1, x_1, x_2, x_3, ..., x_n) \\ \vec{w}^{(i)} &= (w_0, w_1, w_2, w_3, ..., w_n) \\ y^{(i)} &\in \{-1, 1\} \end{aligned} \tag{2.12}$$

where the superscript i indicate a data point and the subscript n indicate the different features. Notice $x_0 = 1$ is used to get the constant term w_0.

Our goal is to find the probability of a data point d to be class $+1$ given their features $\vec{x}^d$. However, a probability is a real number between 0 and 1. $\vec{x}^d$ with arbitrary features and coefficients will have its values that ranges from $-\infty$ to ∞. We need a function to transform $(-\infty, \infty)$ to $(0, 1)$.

2.2.2 Logistic Function

Logistic Function has the equation

$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{e^{-s}+1} \tag{2.13}$$

where it transforms the range of $(-\infty \to \infty)$ to $(0, 1)$.

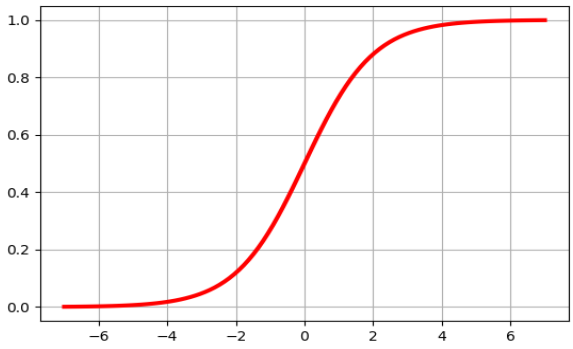


Figure 2.2: Logistic Function

The logistic function also has the property $1-\theta(s) = \theta(-s)$, which will be useful when we use it hypothesize the probability for data points with class -1.

We can use the logistic function to hypothesize the probability of a data point by

$$\begin{aligned} P(y|\vec{x}) &= \begin{cases} \theta(\vec{w}\cdot\vec{x}) & y=+1 \\ 1-\theta(\vec{w}\cdot\vec{x}) & y=-1 \end{cases} \\ &= \begin{cases} \theta(\vec{w}\cdot\vec{x}) & y=+1 \\ \theta(-(\vec{w}\cdot\vec{x})) & y=-1 \end{cases} \\ &= \theta(y(\vec{w}\cdot\vec{x})) \end{aligned} \tag{2.14}$$

In order to make a hypothesis for a data point, we need two things: features of the data and the coefficient vector $\vec{w}$ for each features.

2.2.3 Cost Function

Now we need to find a nice $\vec{w}$ to generalize our data. The Likelihood function will be used to find $\vec{w}$. Likelihood is the probability of a event that has already occurred

will occur again. It is given by

$$\begin{aligned}\mathcal{L} &= \prod_{i=1}^{N} P(y^{(i)}|\vec{x}^{(i)}) \\ &= \prod_{i=1}^{N} \theta(y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}))\end{aligned} \tag{2.15}$$

By using the likelihood function, we want to find $\vec{w}$ to maximize the $\mathcal{L}$ so that it generalizes all data. Furthermore, we can turn a maximizing problem into a minimizing problem by rewriting the equation as

$$\begin{aligned}\ln\mathcal{L}(\vec{w}) &= \frac{1}{N}\sum_{i=1}^{N} \ln\theta(y^{(i)}(\vec{w}\cdot\vec{x}^{(i)})) \\ -\ln\mathcal{L}(\vec{w}) &= -\frac{1}{N}\sum_{i=1}^{N} \ln\theta(y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}))\end{aligned} \tag{2.16}$$

$$\text{cost}(\vec{w}) = -\frac{1}{N}\sum_{i=1}^{N} \ln\theta(y^{(i)}(\vec{w}\cdot\vec{x}^{(i)})) \tag{2.17}$$

Let's examine how eqn 2.17 works.

A data of class $+1$ will have probability $\theta(\vec{w}\cdot\vec{x})$ close to one. This also means that $\ln(\theta)$ will be close to zero. Our cost is at minimum. However, if a data of class $+1$ is horribly classified, meaning it has $\theta(\vec{w}\cdot\vec{x})$ close to 0, then $\ln(\theta)$ goes to $-\infty$. This means the cost function heavily penalize data that are misclassified in order to find a suitable $\vec{w}$. This is the same for class -1 data points.

Eqn 2.17 can be used to find $\vec{w}$ to construct a decision boundary, however, it often tends to overfit the data. Overfit is where we try to separate exactly to all points of data in our training set that it does not generalize to other data.

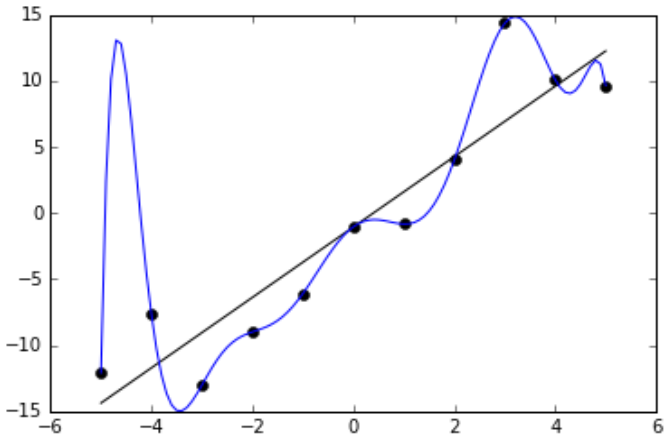


Figure 2.3: Sample of a overfit data

Figure 2.3 exemplifies the concept of overfitting. Even though the blue polynomial function perfectly fit the training data, but the linear function can better generalize unseen data.

In order to avoid overfitting, we should include regularization to our cost function.

$$\text{cost}(\vec{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left(\ln\theta(y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}))\right) + \lambda(\vec{w}\cdot\vec{w}) \tag{2.18}$$

Essentially the first term of eqn 2.18 tries its best to fit all the data, but the second term will avoid fitting too close to each data.

Once we have $\vec{w}$ that minimizes eqn 2.18, we can use it to find the probability of $+1$ class of unseen data by $\theta(\vec{x}\cdot\vec{w})$.

2.3 Boosting

Boosting is a ensemble of weak classifiers to form a strong classifier. Each classifier learns the mistake from the previous classifiers and tries to emphasize on the previous mistakes. Refer to the example in section 2.2 of students pass/fail a tough exam. Sometimes, students devote all their time studying rather than doing their homework, and vice versa. The introduction of these types of students are shown below.

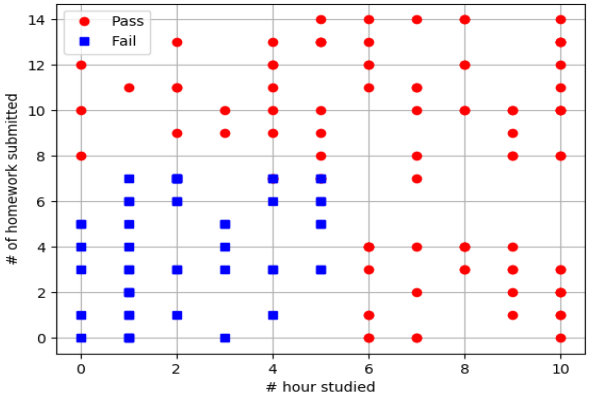


Figure 2.4: New students in the data

Now, we fit a weak classifier.

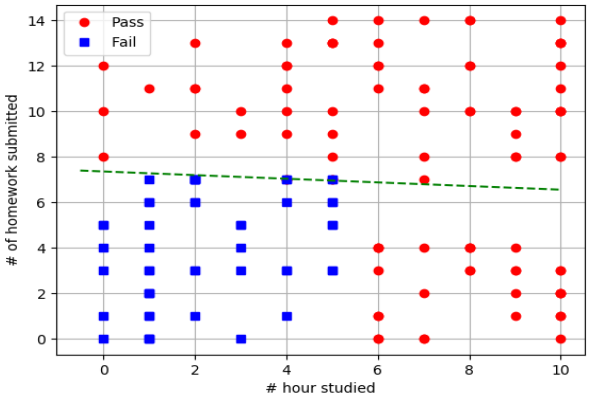


Figure 2.5: Iteration 1

Notice that there are many students who passed but were classified as failed. These are points we cannot ignore. To emphasize on these points, we can put a "weight" in these points. So in the next iteration, when we misclassify these points, along with the weights associated with it, the cost function will heavily penalize these particular data, resulting in a new set of ws.

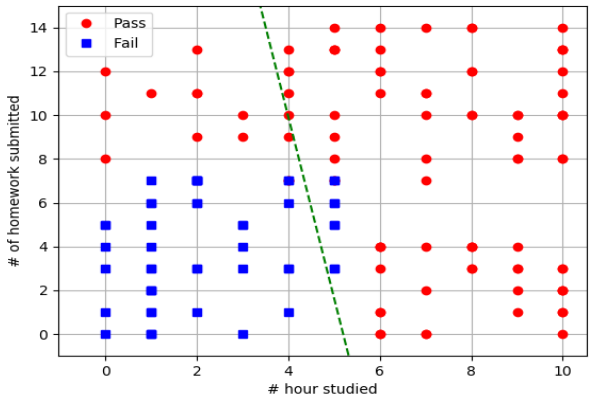


Figure 2.6: Iteration 2

Now, we fixed the previously misclassify data, however, the new $\vec{w}$ focuses heavily on these particular data, causing misclassfication somewhere else. If we combined the two decision boundaries from iteration 1 and iteration 2 focusing only where they are right, we have

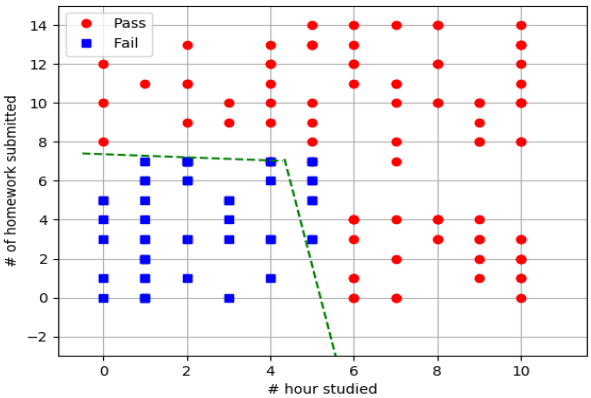


Figure 2.7: Combined iteration 1 and iteration 2

We can repeat the process until a precise decision boundary is achieved. This is an example of a type of boosting called *adaptive boosting* or *adaboost*.

The more iteration we perform adaboost, a more precise decision boundary will be achieved, however, this will cause overfitting as the decision boundary will not correctly classify on our test data. So the question is what should be the optimal number of iteration to train our machine? As we train model, the performance on the training

data will increase. In other words, the error or misclassified data in our training data will decrease. This also happens on our test data, however, at some iteration, the error will increase as we are overfitting the training data.

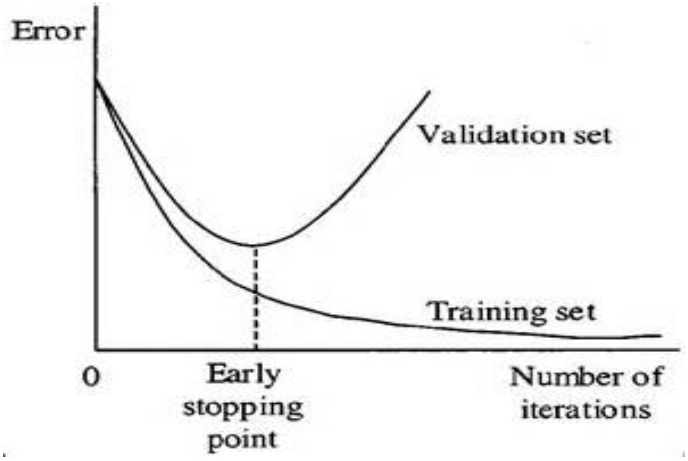


Figure 2.8: Sample error of train and test data on each iteration

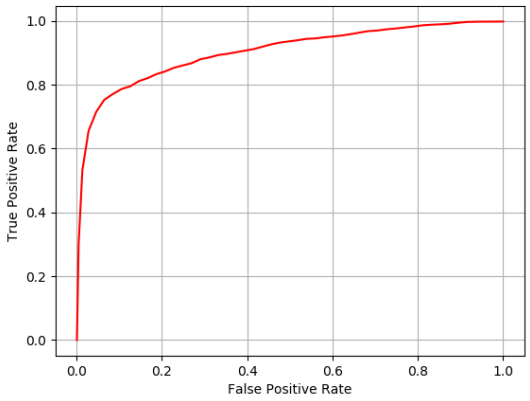
From the figure, we should train up to the iteration where the test error is at its lowest to avoid overfitting.

2.4 Receiver Operating Characteristic curve

Once we are done testing our classifiers on a dataset, how do we know that it is effective? All classifiers aforementioned will give us a probability for a data point. Suppose we have a data point that has the probability of 0.6. We might say that this data is more likely to be in class $+1$ than -1, because we set the cutoff threshold to be 0.5. However, sometimes we want to move the cutoff threshold to different values in various applications. For instance, we are running a preliminary check on people with a extremely contagious disease X. We have to be deeply concerned with people who actually does have disease X, but we misclassified him/her as not having disease X, because we don't want him/her to go outside and spread the disease. A data that is in class $+1$ but got misclassified being in class -1 are called *false negative*. On the other hand, we should care less about people who actually does not have disease X, but got misclassified as having one, because we can run more delicate tests to find out that he/she does not

have the disease. When a data with class -1 got misclassified as $+1$, they are called *false positive*. Data that are correctly classified in class $+1$ and -1 are called *true postive* and *true negative* respectively.

The receiver operating characteristic curve or ROC curve is a plot between *false positive rate* (FPR) and *true positive rate* (TPR) at different cutoff threshold. FPR and TPR can be found by $\text{FPR} = \frac{\text{false positive}}{\text{false positive + true negative}}$ and $\text{TPR} = \frac{\text{true positive}}{\text{true positive + false negative}}$. An example of ROC curve is shown below.



where each point on the line represent a different cutoff threshold. For example, if we set the cutoff threshold at 0, we classify all data of class $+1$ to be true positive and data of class -1 to be all false positive. Hence we get a point (1,1) on the ROC curve. As we lower the cutoff threshold, we classify less data as true positive and false positive, until at threshold 1.0 where we don't classify anything to be in positive class.

So how does an ROC curve visualize the performance of a classifier? In a good classifier, when we change our cutoff threshold, we should see a clear change between the ratio of FPR and TPR. However, in a terrible classifier, the ratio between FPR and TPR does not change much, which means the classifier is just randomly classifying data. The below figure shows the different ROC curves for different classifiers.

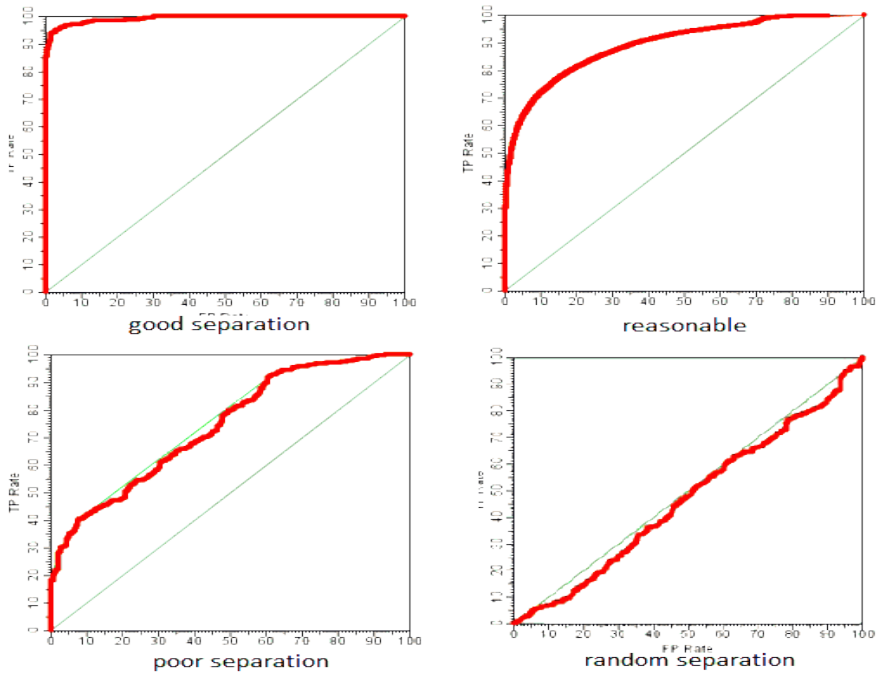


Figure 2.9: Different ROC curves

So the simplest way to determine if a classifier is a effective classifier is to find the area under the ROC curve. The better a classifier, the closer the area is to one. We will use this concept to evaluate the performance of our classifiers.

CHAPTER 3

TRAINING

3.1 Data Overview

The dataset for this project is downloaded from `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`. This dataset of comments from Wikipedia's talk page edits that are labeled by human raters for their toxic behavior. The below figure shows how the dataset is provided.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

The comments are given along with indicator of the 6 classes: toxic, severe_toxic, obscene, threat, insult, and identity_hate. We will separate this dataset into 80% training set and 20% validation set.

The comments length has a mean of 394.7120, standard deviation of 591.9907 and maximum of 5895. This means the length varies a lot.

Looking at example comments in each category:

toxic: `Fuck off You are NOT an administrator. You DON'T have the authority to tell me what to do.`

severe_toxic: `Since this is MY user page, I can say what I want: YOU GUYS ARE ALL A BUNCH OF MOTHERFUCKING ASS-EATERS AND I HOPE YOU ALL ROT IN HELL.`

obscene: What a motherfucking piece of crap those fuckheads for blocking us!

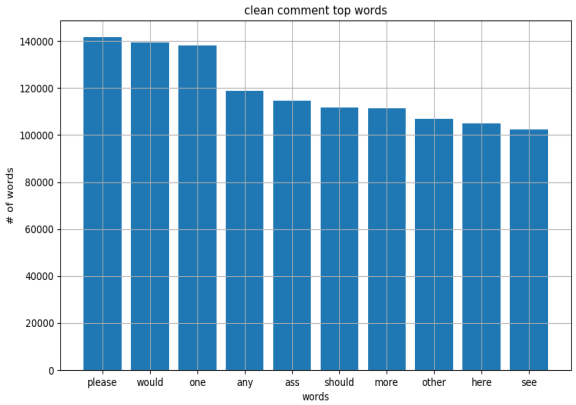
threat: hello moto hello flygayguy649 i hope u get a dick stuck up your ass i am a 13 year old kkk member. and i'll kick your ass so stay away and dont delete my pages. BITCH!!!

insult: Are you fucker mother fucker have nothing to do but block University computers. Go and suck cocks

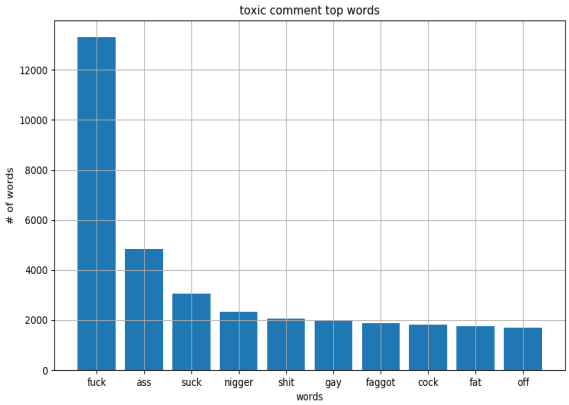
identity_hate: SmallThingy.... ... are you gay? Do you frequent the Hellfire Club? Do you shake your little tusch on the catwalk?

no tags: Oh, and the girl above started her arguments with me. She stuck her nose where it doesn't belong. I believe the argument was between me and Yvesnimmo. But like I said, the situation was settled and I apologized. Thanks,

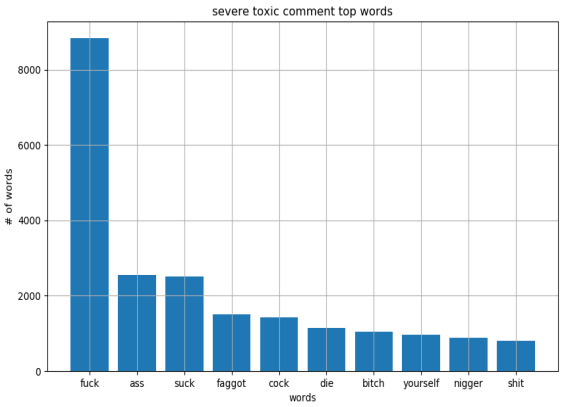
After going through sample of each category, one thing is certain, swear words definitely dominate in all category comments. Now let's take a look at words that appears most frequent in each categories.



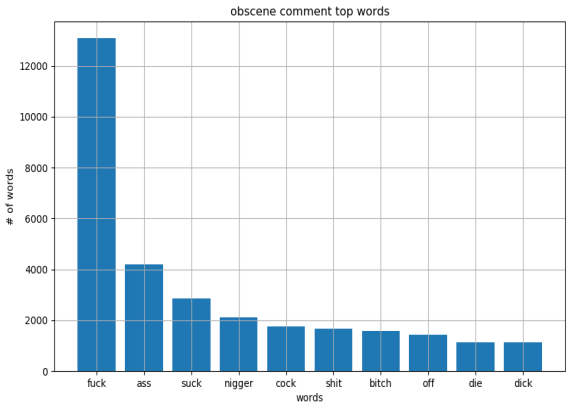
(a) Top words in clean comments



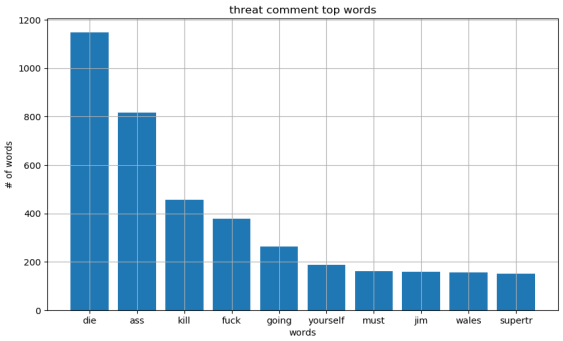
(b) Top words in toxic comments



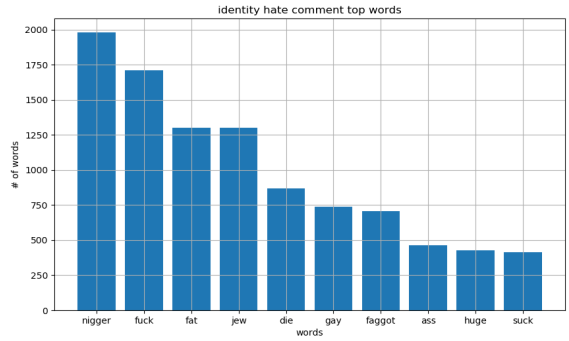
(c) Top words in severe toxic comments



(d) Top words in obscene comments



(e) Top words in threat comments



(f) Top words in identity hate comments

Figure 3.1: Top words in each categories

The frequency of swear words dominate comments that are tagged, thus we should use it as our feature.

3.2 Naive Bayes

Recall equation 2.11,

$$P(C|x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1|C) \times P(x_2|C) \times P(x_3|C) \times P(x_4|C) \times P(C)}{\left[\left(P(x_1|C) \times P(x_2|C) \times P(x_3|C) \times P(x_4|C) \times P(C)\right) + \left(P(x_1|\neg C) \times P(x_2|\neg C) \times P(x_3|\neg C) \times P(x_4|\neg C) \times P(\neg C)\right)\right]}$$

if we set each features as the word frequency, we can find the probability of a comment being toxic T by

$$= \frac{P(word_1|T) \times P(word_2|T) \times P(word_3|T)... \times P(word_n|T) \times P(T)}{\left[\big(P(word_1|T) \times P(word_2|T) \times P(word_3|T)... \times P(word_n|T) \times P(C)\big) + \big(P(word_1|\neg T) \times P(word_2|\neg T) \times P(word_3|\neg T)... \times P(word_n|\neg T) \times P(\neg T)\big)\right]} \tag{3.1}$$

So all we need to do is to keep track of the conditional probability of the word given it is toxic/nontoxic from the train data. For instance, $P(\texttt{fuck}|T) = 0.0428$ and $P(\texttt{fuck}|\neg T) = 6.43 \times 10^{-5}$. Note that common articles and conjunctions (a,are,is) are used more often in both categories, hence we should preprocess our comments. Once we are done training the classifier from our training data, we will have numerous $P(word|T)$ and $P(word|\neg T)$. To predict a probability of an unseen comment to be toxic, we just take each word in the comment and look up the conditional probabilities we keep track of and plug it in eqn 3.1.

3.3 Logistic Regression

In logistic regression classifiers, we can use multiple features to classify a data. For this classifier, I used result from naive bayes classifer as a feature, proportion of uppercase letters over the length of the comment, and the proportion of non-alphabet characters over the length of the comment.

$$\vec{x} = [\ 1,\ \text{Naive Bayes},\ \text{Uppercase letters},\ \text{Non-alphabets}\ ]$$

Then, I used the above features from the training dataset to find a $\vec{w}$ that minimizes the cost function (eqn 2.18). The corresponding $\vec{w}$ are:

$$\vec{w}_{\textbf{toxic}} = [\ -2.367,\ 7.024,\ 1.693,\ -1.130\ ]$$

$$\vec{w}_{\textbf{severe toxic}} = [\ -2.720,\ 7.705,\ 2.270,\ -0.492\ ]$$

$$\vec{w}_{\textbf{obscene}} = [\ -2.513,\ 7.293,\ 0.731,\ -0.419\ ]$$

$$\vec{w}_{\textbf{threat}} = [\ -2.625,\ 7.601,\ 0.552,\ -0.737\ ]$$

$$\vec{w}_{\textbf{insult}} = [\ -2.448,\ 7.071,\ 0.457,\ -0.718\ ]$$

$$\vec{w}_{\textbf{identity hate}} = [\ -2.593,\ 7.411,\ 0.740,\ -1.446\ ]$$

Let's take a moment to understand what each term in $\vec{w}$ means. We construct our decision boundary using coefficients from $\vec{w}$ by $y = w_0 + (w_1 \times x_1) + (w_2 \times x_2) + (w_3 \times x_3)$. This is the same as dot product which is also used to calculate the probability $\theta(\vec{w} \cdot \vec{x})$. Recall from logistic function that the bigger $\vec{w} \cdot \vec{x}$, the close we get to 1. Since all the terms in $\vec{x}$ are positive, $\vec{w}$ is analogous to putting importance factor to each term in $\vec{x}$. The bigger positive w_k means that feature corresponding to it is more likely to be associated with being **toxic**, and bigger negative w_k means the feature corresponding to it is likely to be associated with being **nontoxic**.

From the above $\vec{w}$s, the most dominant term is for feature that comes from Naive Bayes, since it did an decent job classifying toxic comments. We will use these $\vec{w}$ to classify unseen comments.

3.3.1 Vectorizing Words

Since we can put any features into logistic regression, we can try out a bunch different features and see which is associated with the class. In this logistic regression classifier, my $\vec{x}$ is the number of different words that appear.

$$\vec{x} = [\ 1,\ \#\text{ of } word_1,\ \#\text{ of } word_2,\ \#\text{ of } word_3,\ \#\text{ of } word_4,\ \ldots,\ \#\text{ of } word_k]$$

For this classifier, I used SCIKIT-LEARN, a machine learning library for python. Firstly, we want to count the number of different words that appear in every comments.

However, common words like a, am, are will be very present, hence carrying very little importance in classification. Therefore we need the "reweigh" the more important terms. Term frequency-inverse document frequency or tf-idf weighing is a method to redistribute the importance of words. Tf-idf is given by

$$F(t) = \text{term frequency} \times \log\left(\frac{N}{df+1}\right) \tag{3.2}$$

where term frequency is the number of words appear in the document, N is the total number of documents and df is the number of documents containing the term. The first term is just counting the number of the words, while the second checks if this term appears in other document in the sample. The more or less it is present in other documents, the more it dampens or boost the importance of the term.

Once I have a $\vec{x}$ for all train comments, I used it to find $\vec{w}$ such that it minimize our cost function (eqn 2.18). The follow words are the 3 top words with the highest w associated it.

toxic = fuck; fucking; idiot

severe toxic = fuck; fucking; fucker

obscene = fuck; fucking; bitch

threat = kill; die; burn

insult = idiot; asshole; bitch

identity hate = nigger; homosexual; gay

As expected, "fuck" would dominate in multiple categories. In threat, "kill", "die" and "burn" may inflict pain. The words in identity hate also makes sense as hate group usually advocate hatred towards ethnicity and gender.

3.4 Boosting

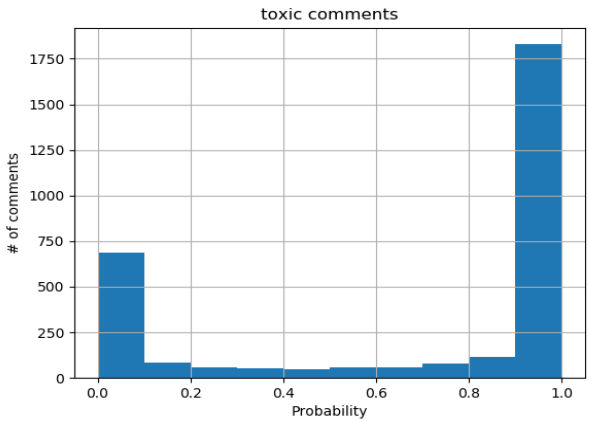
For this classifier, I used the same features as the vectorized words classifier, and the help from SCI-KIT LEARN to perform Adaboost.

CHAPTER 4

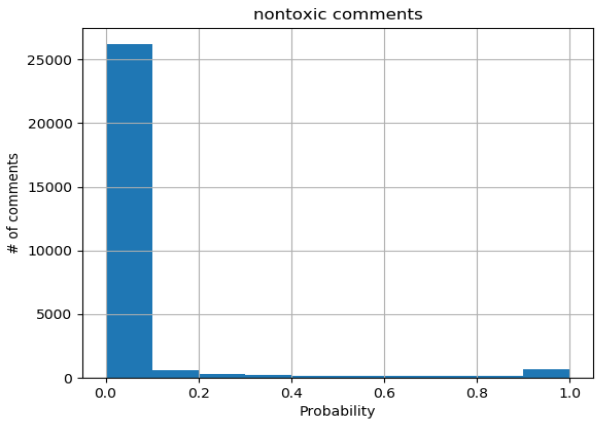
RESULTS

After we trained all the models, now we should see how well each model predict the probability of unseen comments. For this chapter, each category will have two histogram. One histogram is the probability of class positive, and the other is the probability of class negative. In an perfect classifier, we should see all 1s in the left graphs and a 0s on the right graphs. The ROC curve and its area will also be used to quantify its performance.

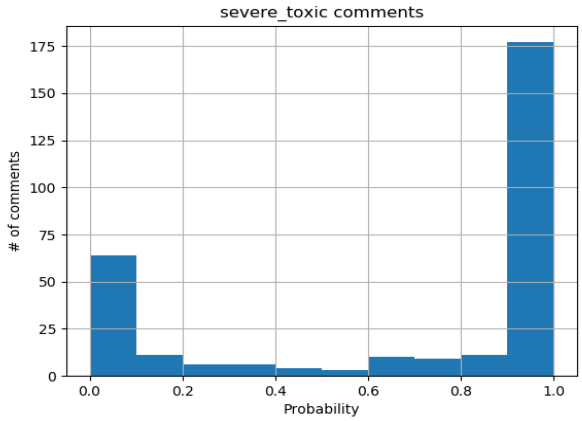
4.1 Naive Bayes



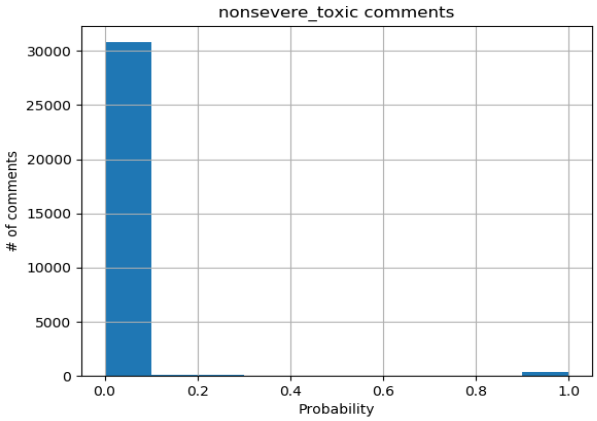
(a) Probabilities of toxic comments



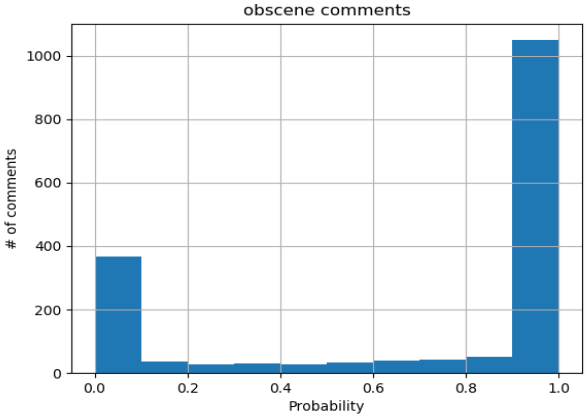
(b) Probabilities of nontoxic comments



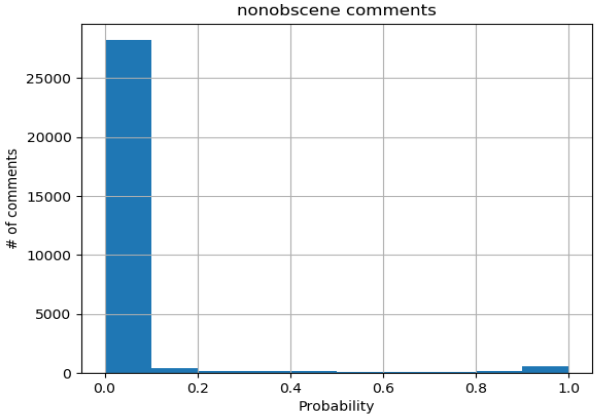
(c) Probabilities of severe toxic comments



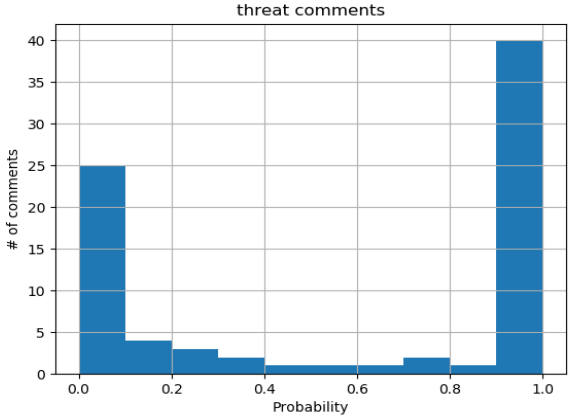
(d) Probabilities of non-severe toxic comments



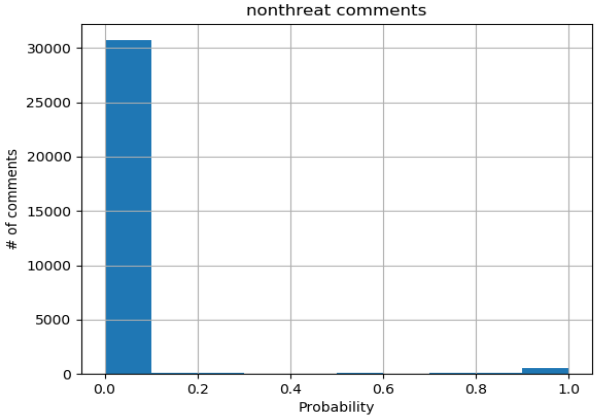
(e) Probabilities of obscene comments



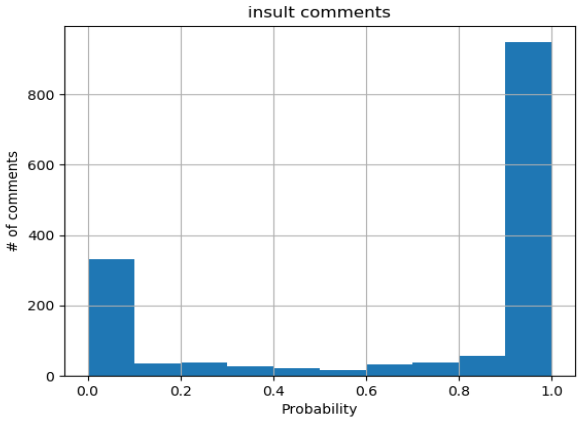
(f) Probabilities of non-obscene comments



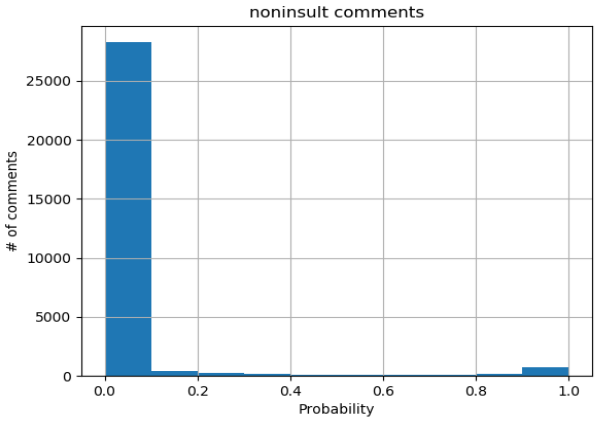
(g) Probabilities of threat comments



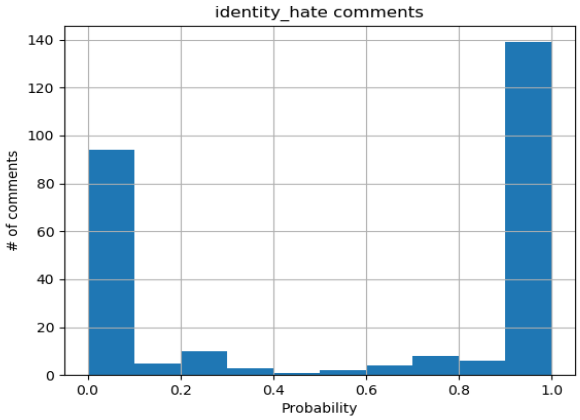
(h) Probabilities of non-threat comments



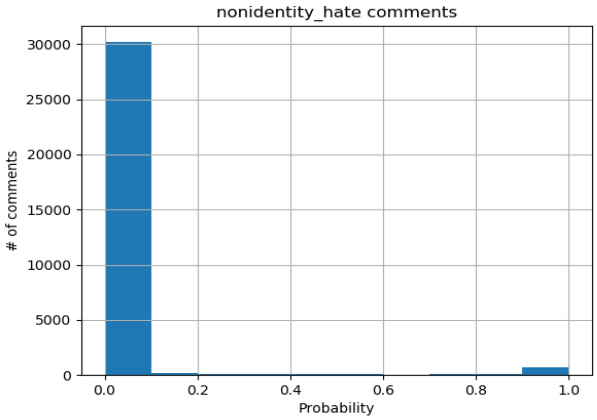
(i) Probabilities of insult comments



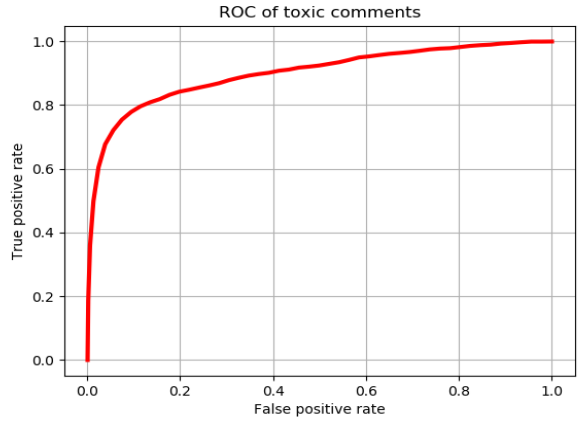
(j) Probabilities of non-insult comments



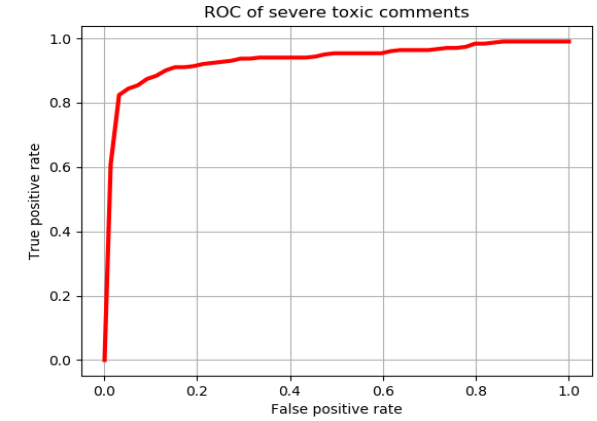
(k) Probabilities of identity hate comments



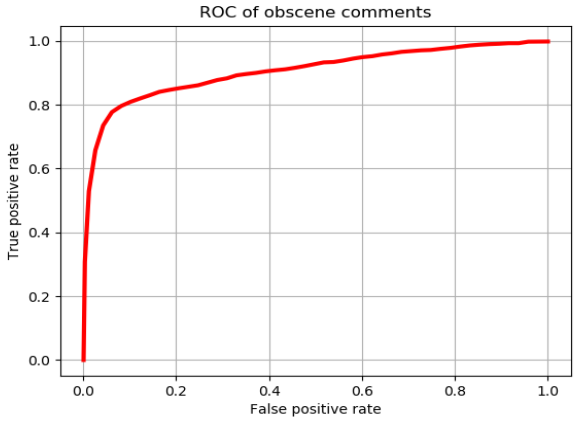
(l) Probabilities of non-identity hate comments



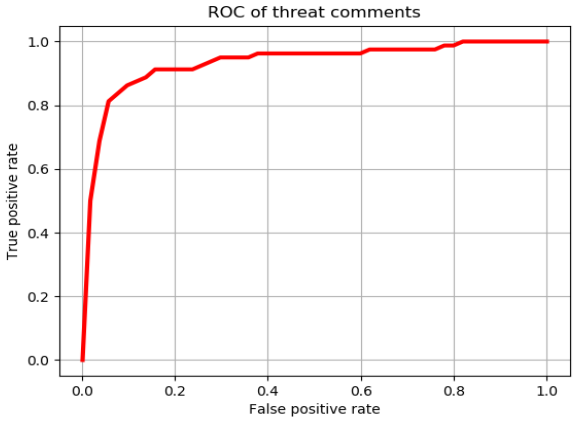
(m) AUC = 0.899



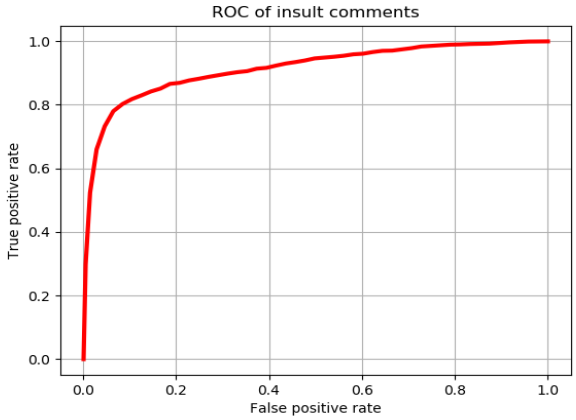
(n) AUC = 0.932



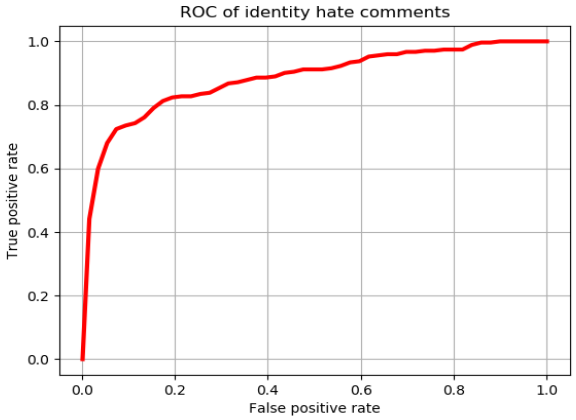
(o) AUC = 0.904



(p) AUC = 0.931

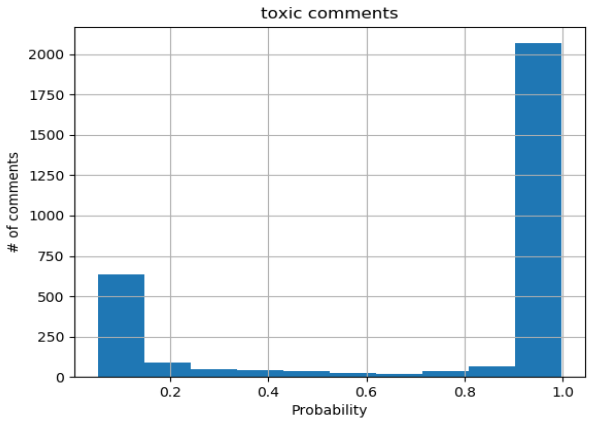


(q) AUC = 0.913

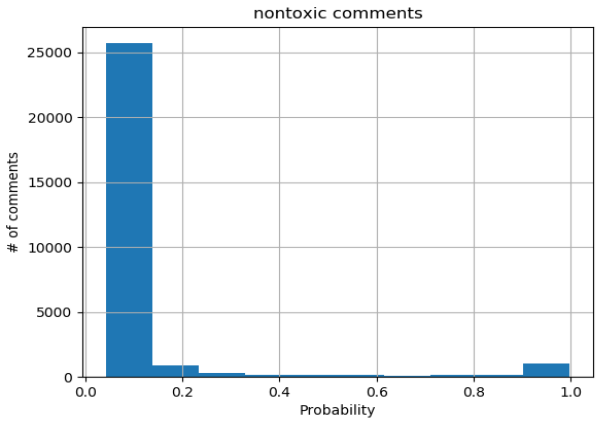


(r) AUC = 0.881

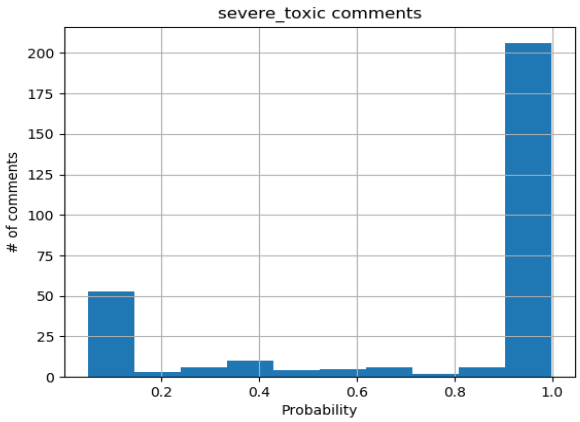
4.2 Logistic Regression



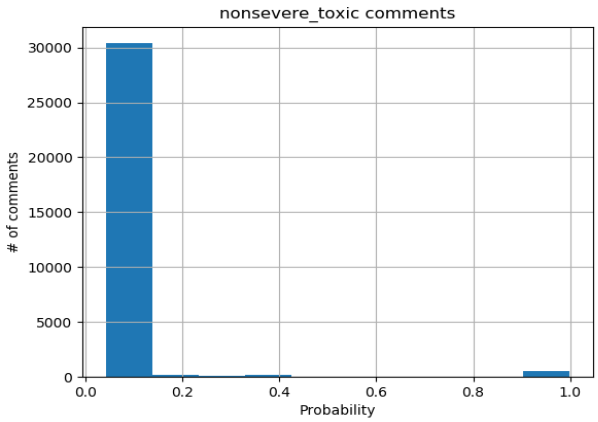
(a) Probabilities of toxic comments



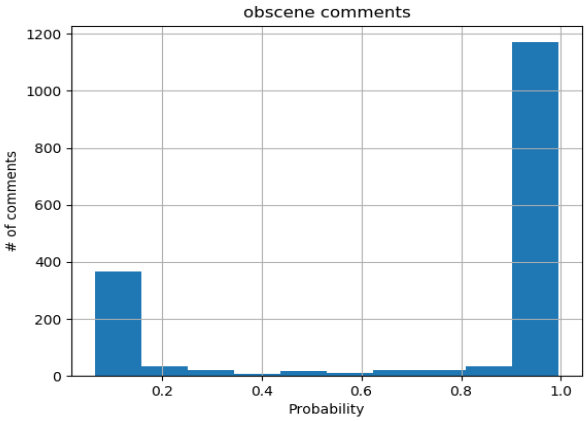
(b) Probabilities of nontoxic comments



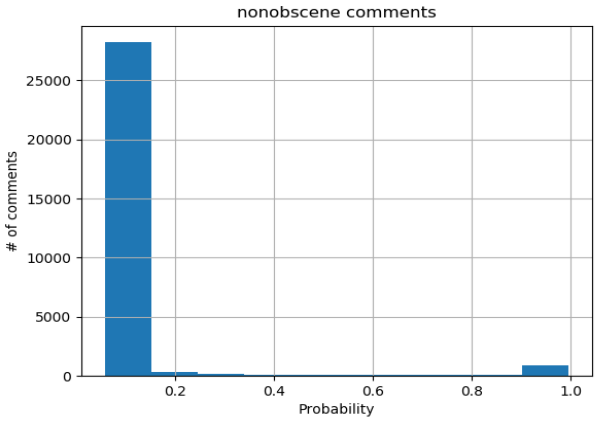
(c) Probabilities of severe toxic comments



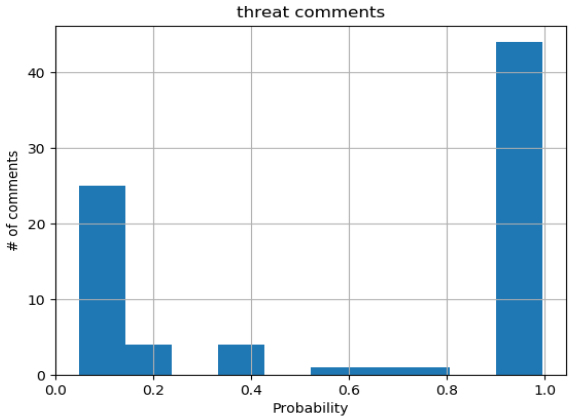
(d) Probabilities of non-severe toxic comments



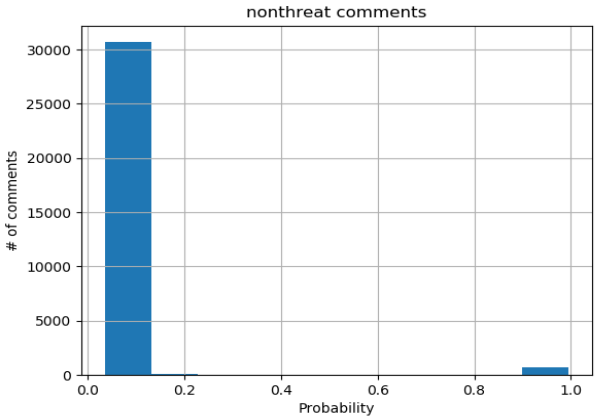
(e) Probabilities of obscene comments



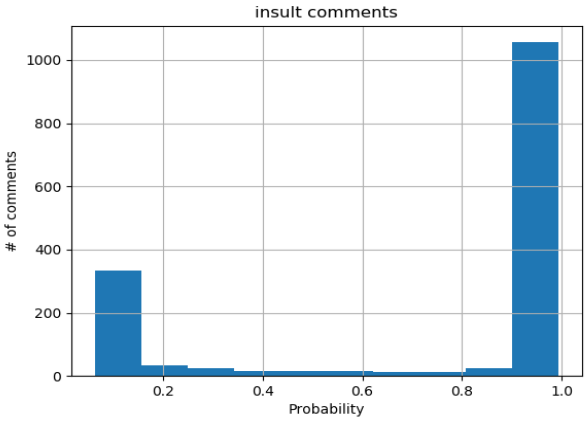
(f) Probabilities of non-obscene comments



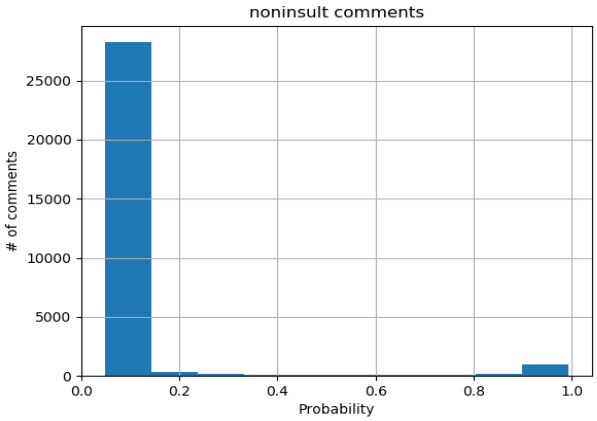
(g) Probabilities of threat comments



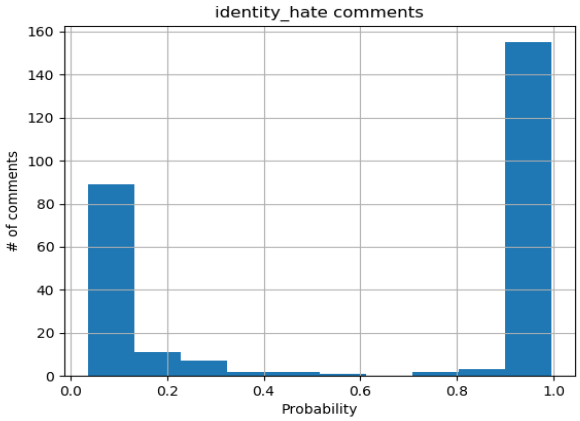
(h) Probabilities of non-threat comments



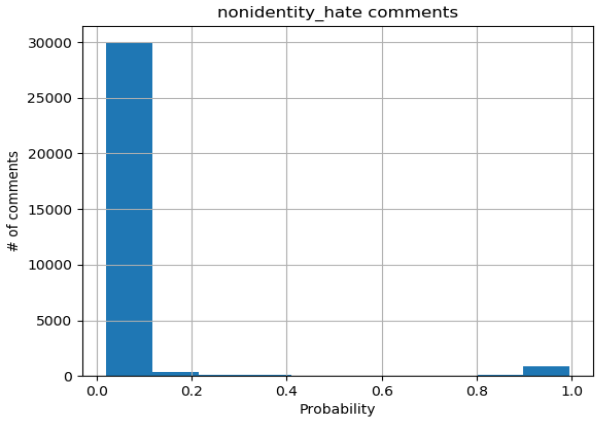
(i) Probabilities of insult comments



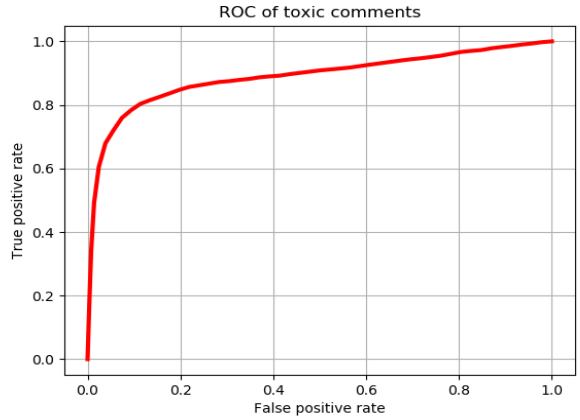
(j) Probabilities of non-insult comments



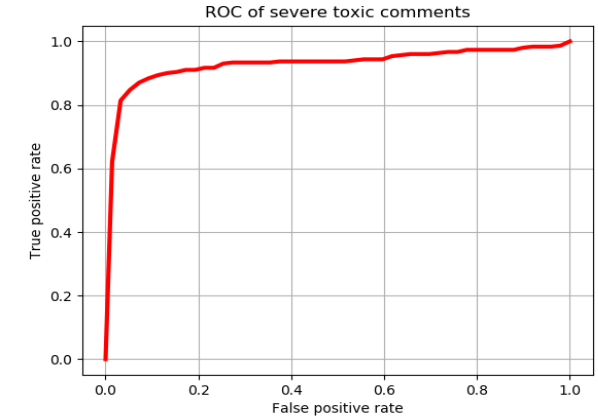
(k) Probabilities of identity hate comments



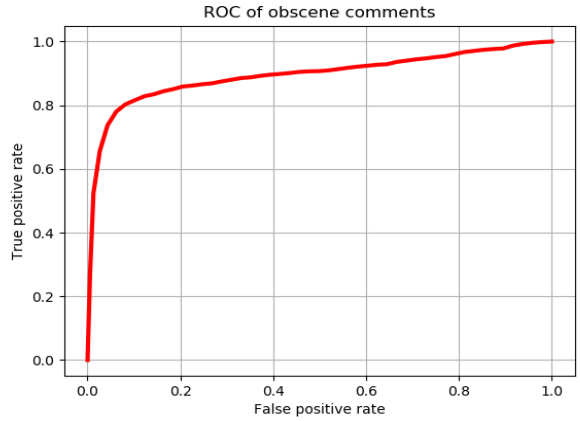
(l) Probabilities of non-identity hate comments



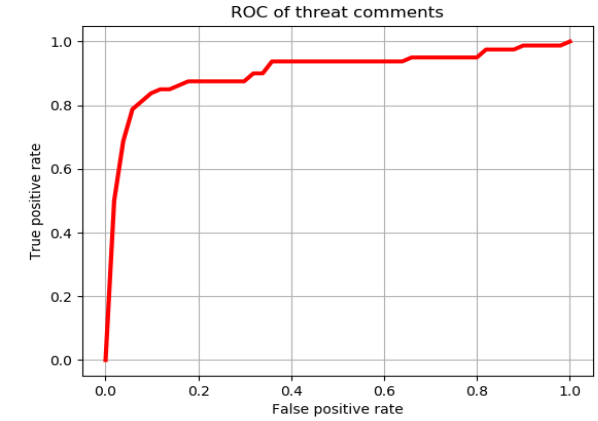
(m) AUC = 0.888



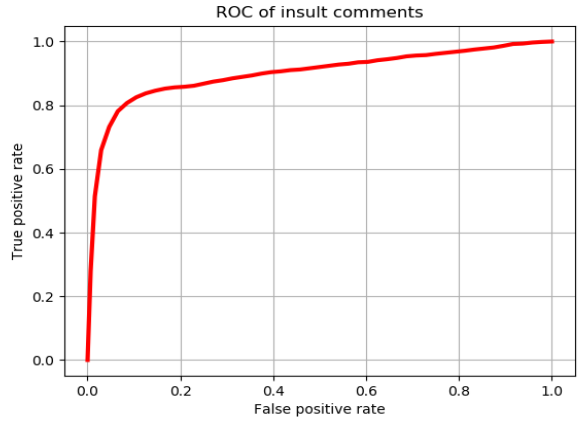
(n) AUC = 0.928



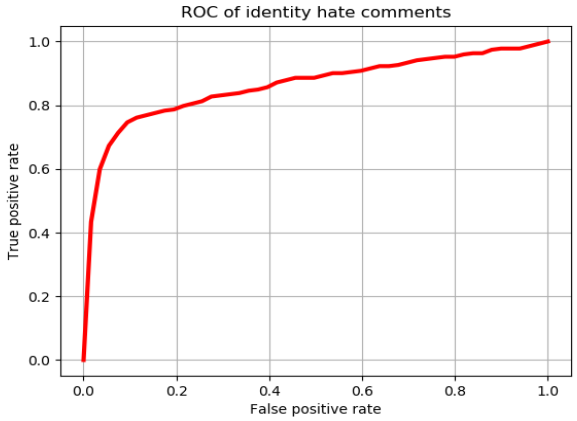
(o) AUC = 0.894



(p) AUC = 0.903

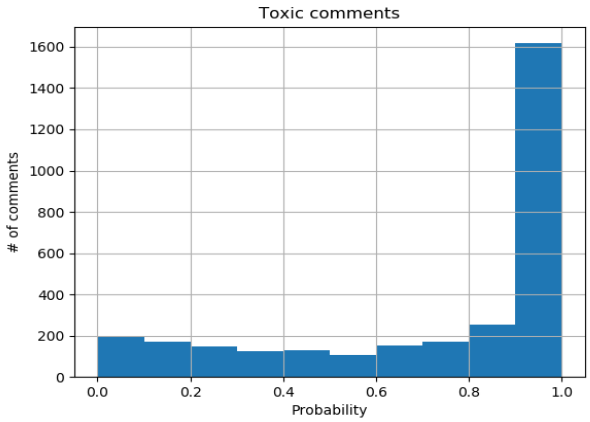


(q) AUC = 0.899

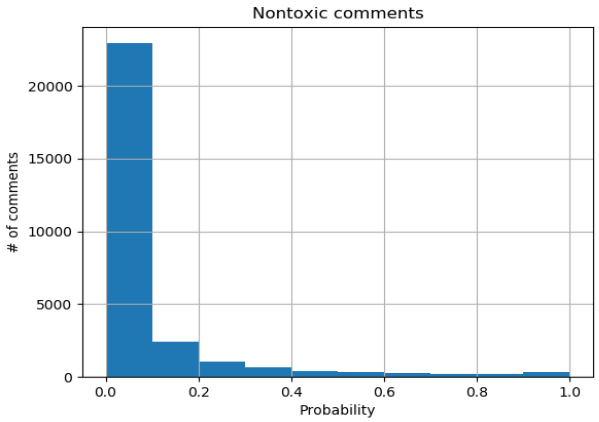


(r) AUC = 0.861

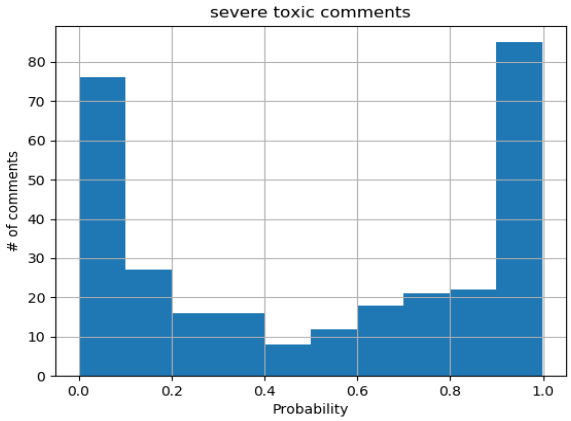
4.2.1 Vectorizing Words



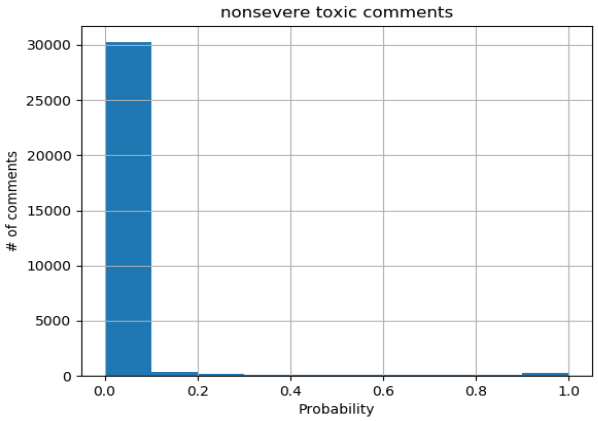
(a) Probabilities of toxic comments



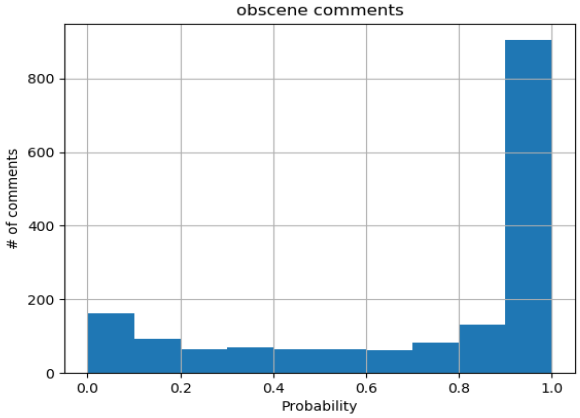
(b) Probabilities of nontoxic comments



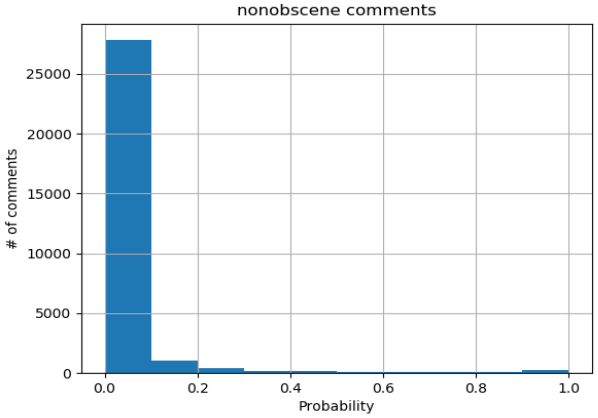
(c) Probabilities of severe toxic comments



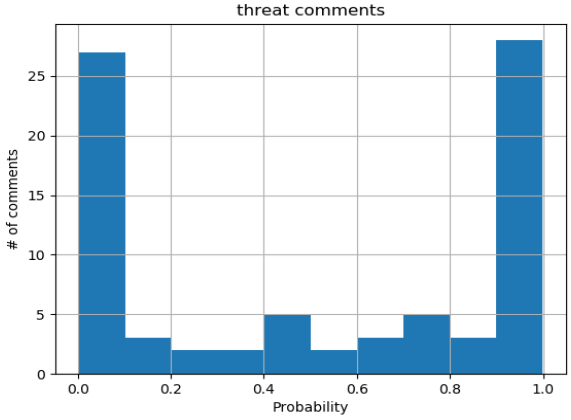
(d) Probabilities of non-severe toxic comments



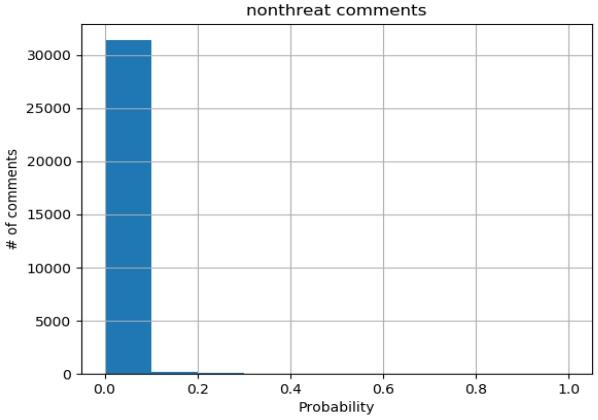
(e) Probabilities of obscene comments



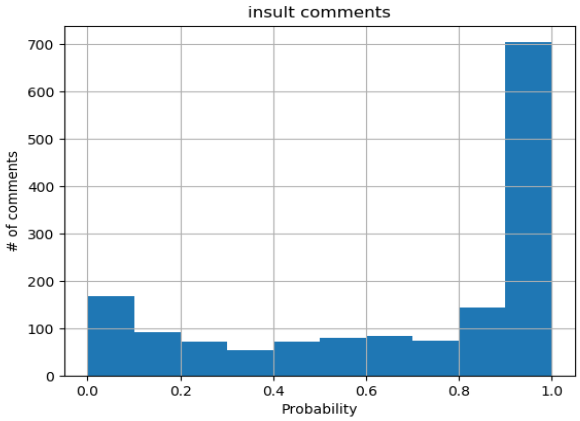
(f) Probabilities of non-obscene comments



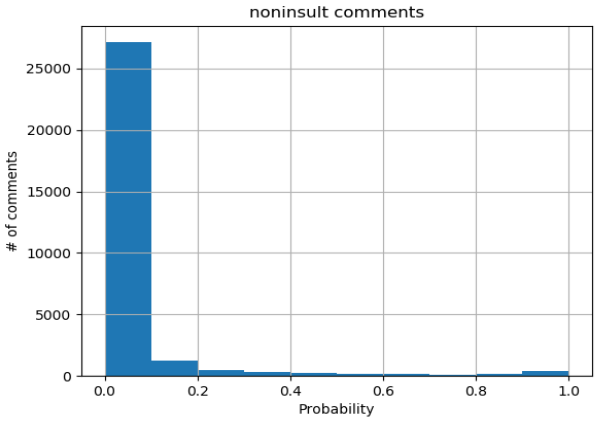
(g) Probabilities of threat comments



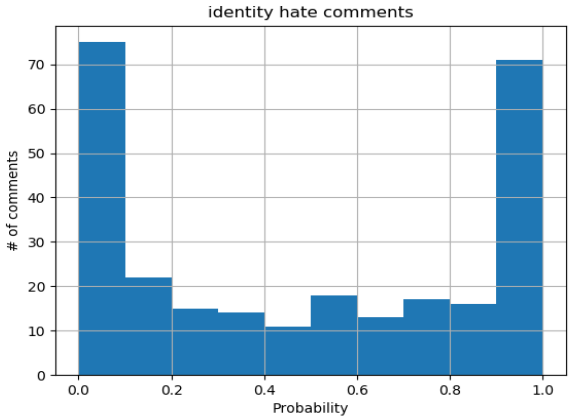
(h) Probabilities of non-threat comments



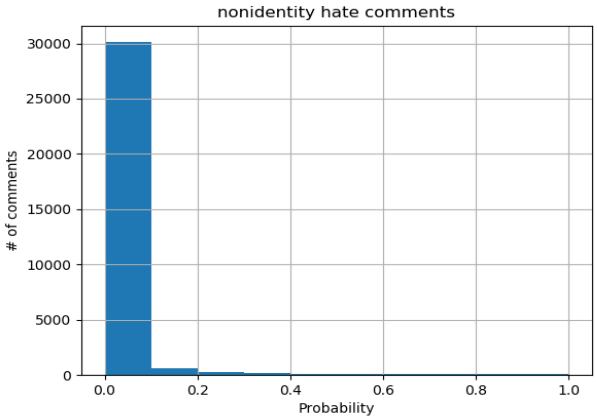
(i) Probabilities of insult comments



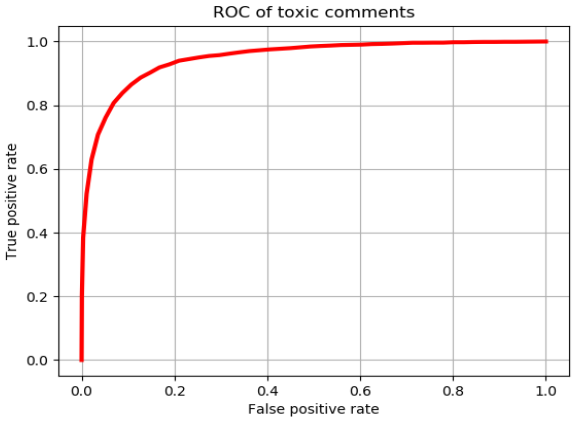
(j) Probabilities of non-insult comments



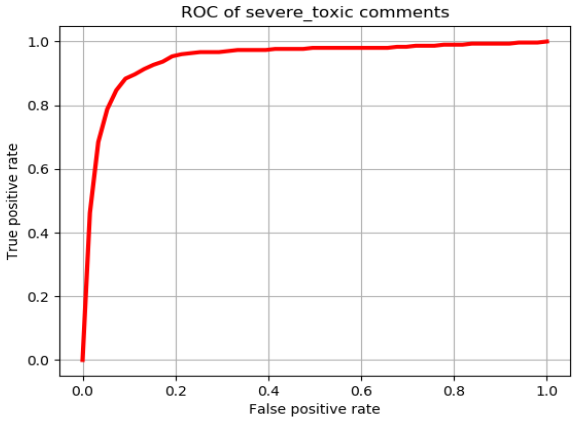
(k) Probabilities of identity hate comments



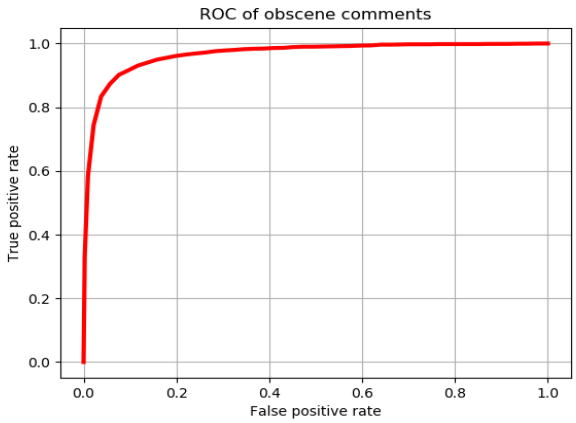
(l) Probabilities of non-identity hate comments



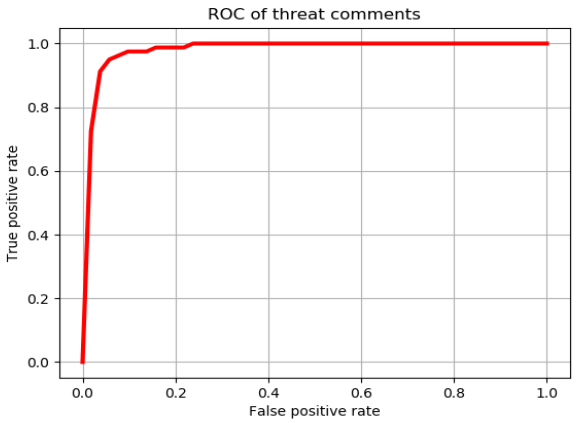
(m) AUC = 0.948



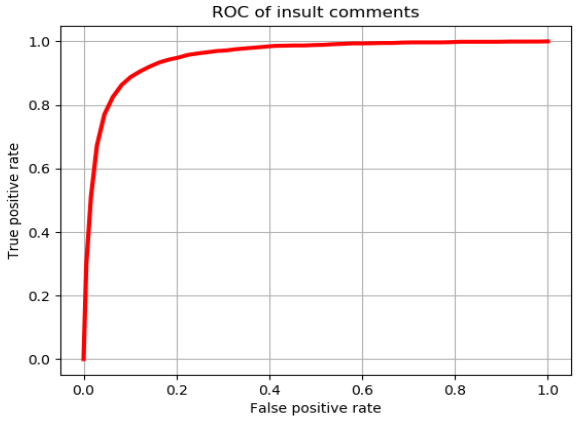
(n) AUC = 0.945



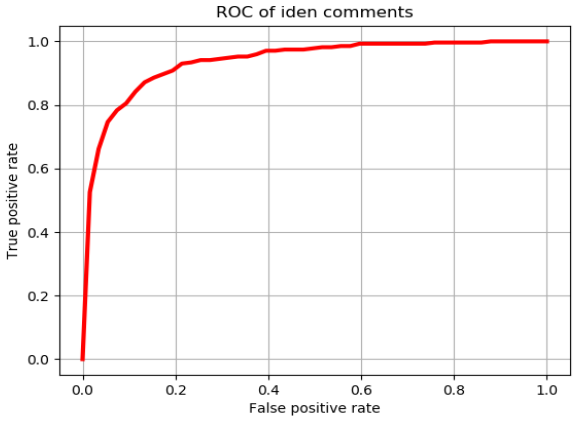
(o) AUC = 0.966



(p) AUC = 0.980



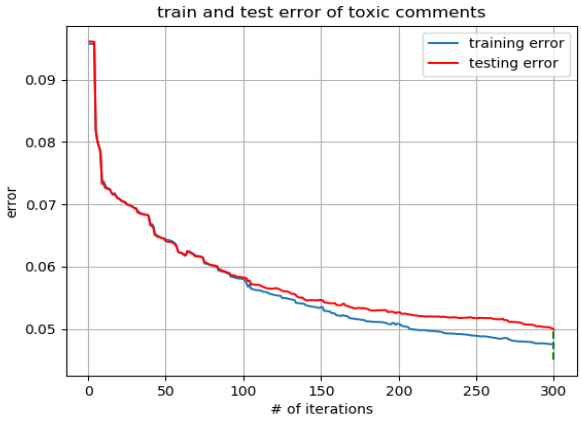
(q) AUC = 0.953



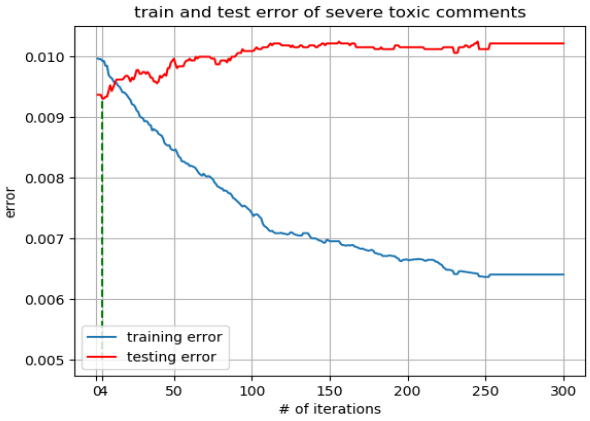
(r) AUC = 0.936

4.3 Boosting

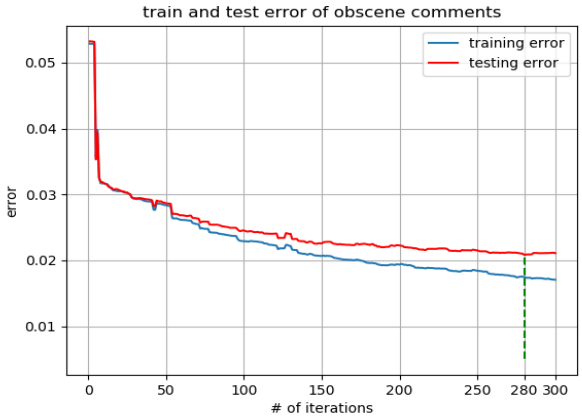
This section will include a plot of train and test error for each category and the number of iterations to train as represented by the dashed green line.


train and test error of toxic comments
training error
testing error
error
# of iterations

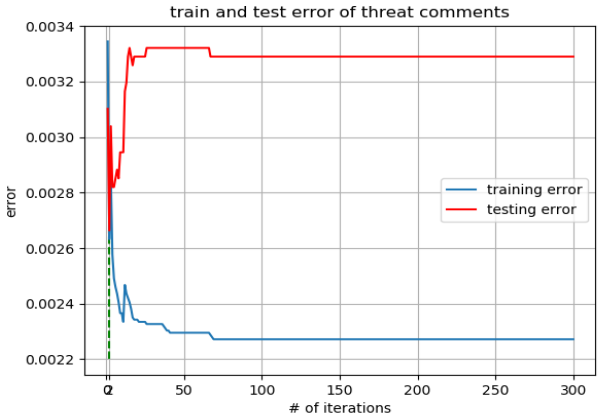
(a)


train and test error of severe toxic comments
training error
testing error
error
# of iterations

(b)


train and test error of obscene comments
training error
testing error
error
# of iterations

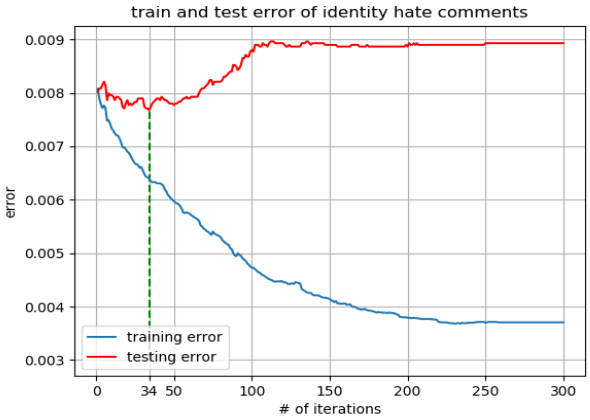
(c)


train and test error of threat comments
training error
testing error
error
# of iterations

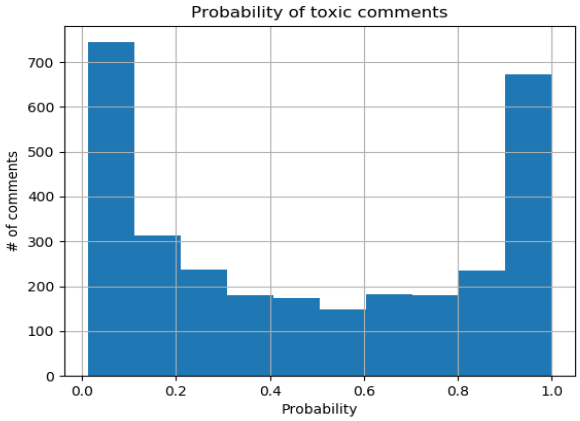
(d)


train and test error of insult comments
training error
testing error
error
# of iterations

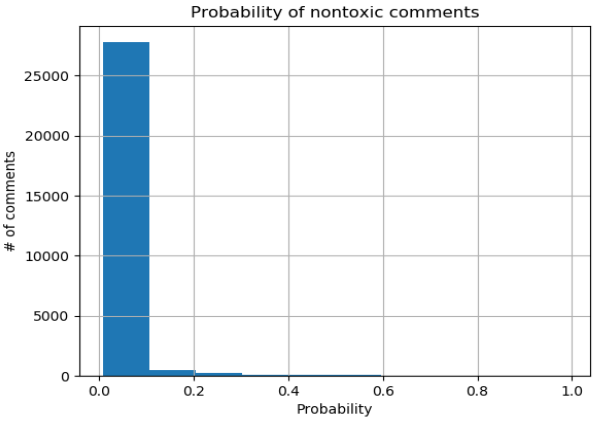
(e)


train and test error of identity hate comments
training error
testing error
error
# of iterations

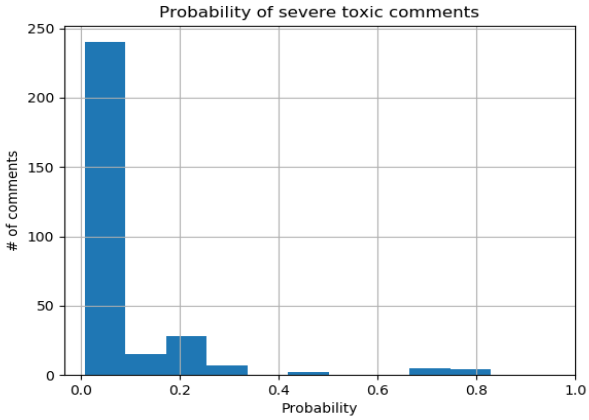
(f)



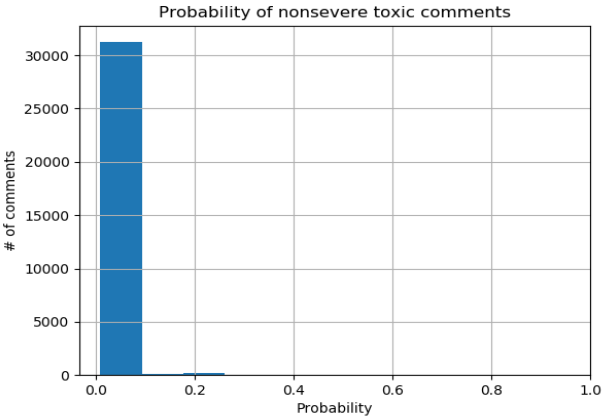
(g) Probabilities of toxic comments



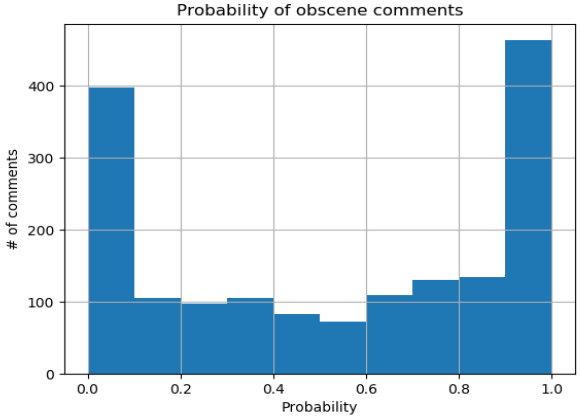
(h) Probabilities of nontoxic comments



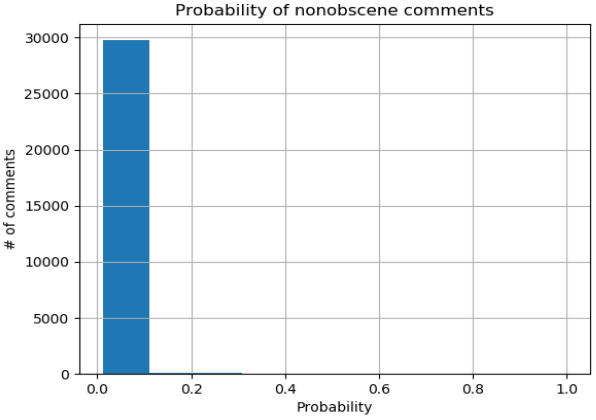
(i) Probabilities of severe toxic comments



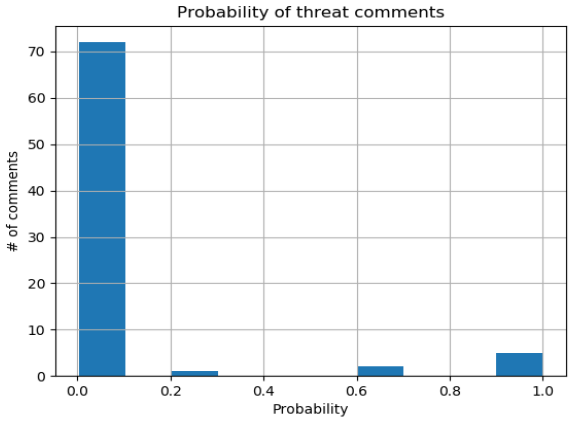
(j) Probabilities of non-severe toxic comments



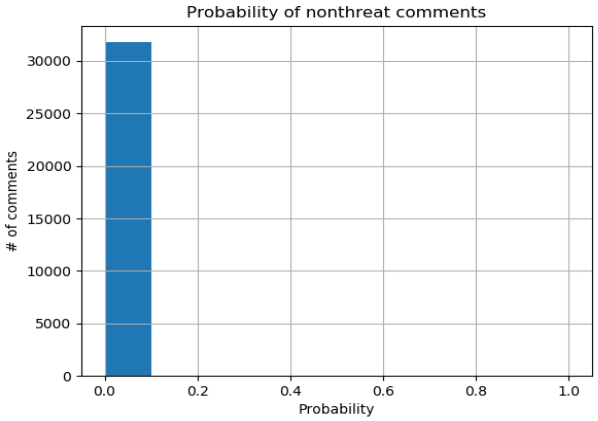
(k) Probabilities of obscene comments



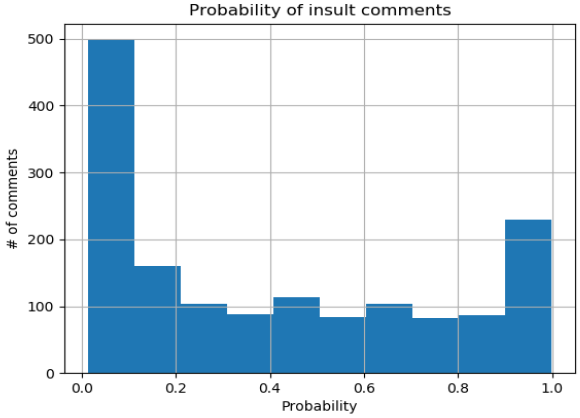
(l) Probabilities of non-obscene comments



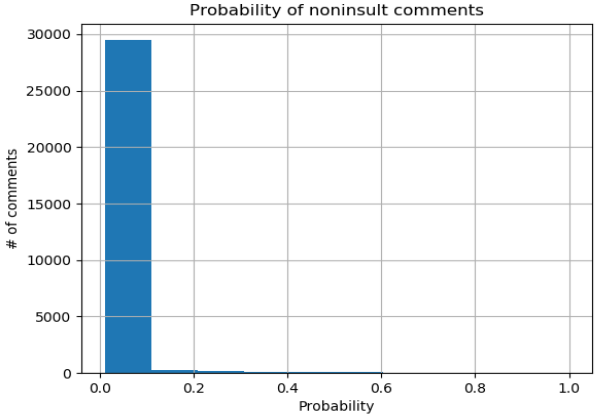
(m) Probabilities of threat comments



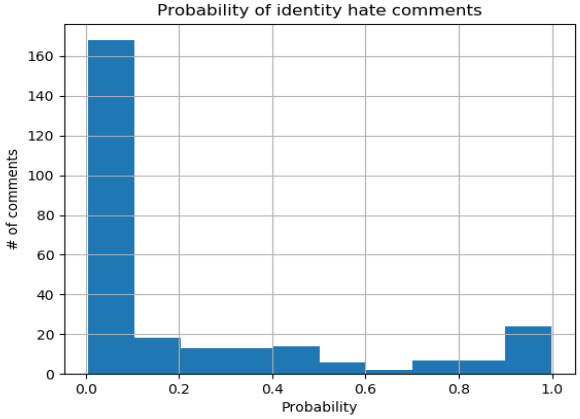
(n) Probabilities of non-threat comments



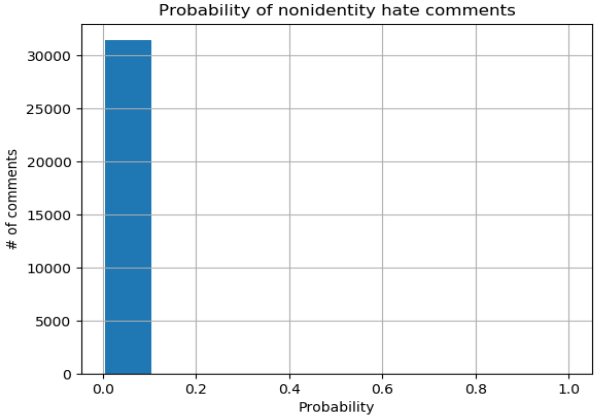
(o) Probabilities of insult comments



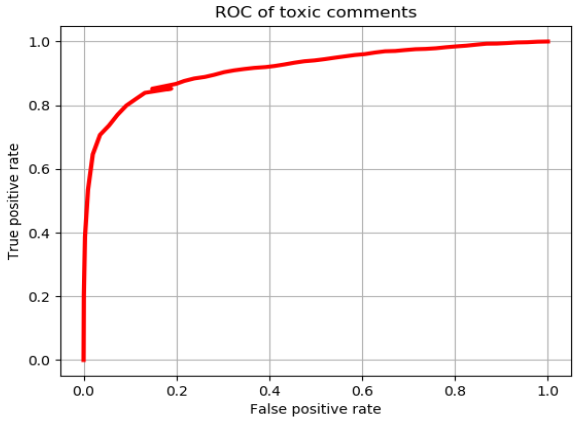
(p) Probabilities of non-insult comments



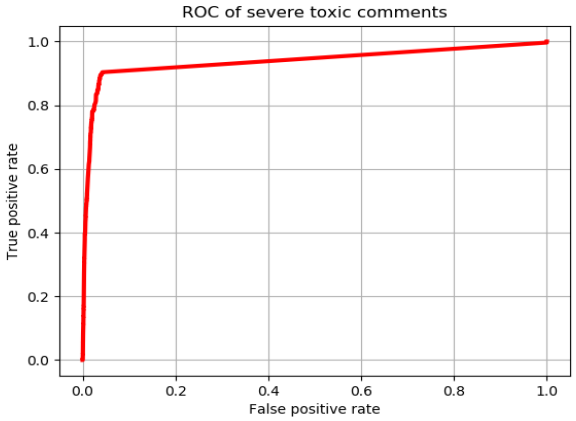
(q) Probabilities of identity hate comments



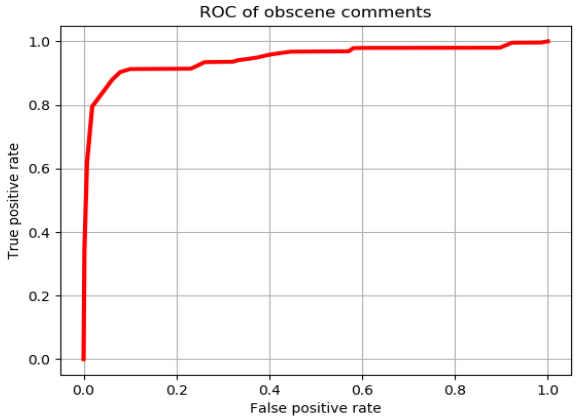
(r) Probabilities of non-identity hate comments



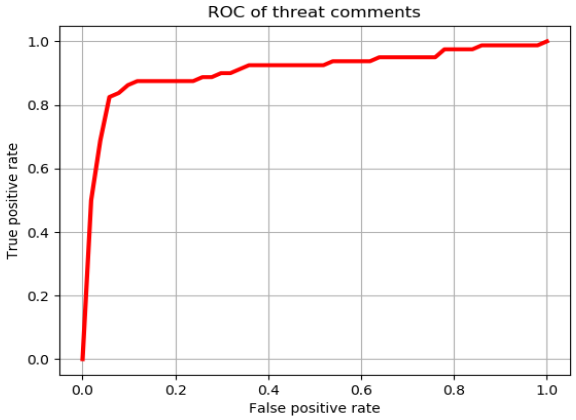
(s) AUC = 0.915



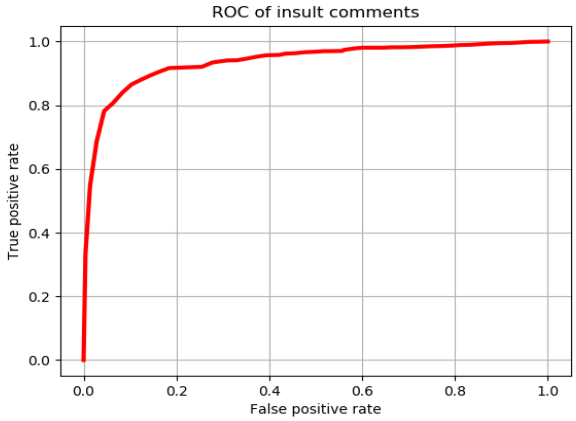
(t) AUC = 0.938



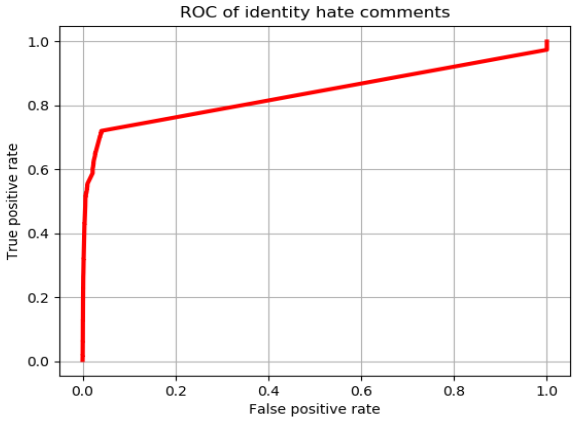
(u) AUC = 0.947



(v) AUC = 0.711



(w) AUC = 0.937



(x) AUC = 0.836

CHAPTER 5

CONCLUSION

Area under ROC curve				
	Naive Bayes	Logistic Regression	Vectorized Words	Boosting
Toxic	0.899	0.888	0.947	0.915
Severe Toxic	0.932	0.928	0.945	0.938
Obscene	0.904	0.894	0.966	0.947
Threat	0.931	0.903	0.980	0.711
Insult	0.913	0.899	0.953	0.937
Identity Hate	0.881	0.861	0.936	0.836
Average	0.910	0.894	0.955	0.881

Overall, logistic regression where we vectorize words are the most effective in classifying toxic comments. It produces relatively better area under the ROC curve for all categories. Other models may perform worse as these comments may be trying to avoid the features we are focusing, or the huge imbalance of class from the dataset. We have a greater number of clean comments than toxic comments, where when we classify them, we emphasize on correctly classify clean comments than toxic comments.

From all the histograms from all classifiers, it is clear that none are perfect. It is worth to look at some of the comments that are false negative, since we don't want these comments to disrupt any discussion online.

```
Y-ou are a qu-eer Su-ck my c-ock fa-ggot
```

```
What are you talking about?  When did I use my adminship in any argument?  You
are being ridiculous.  - (formerly ) talk
```

```
It's abrevashun!  It makes stuff shorter, U f***er!  I'll Nswer dat queshton,
```

`now f*** off  let me do it!  Skymac207 5:00, 6 August 2007 (UTC)`

`What a ridiculous suggestion.  If you don't think certain flags are British then why would it be relevant for the United Kingdom?`

These types of comments tries to censor swear words or they're just a expression of frustration. These comments may easily avoid being classified as they avoid swear words, or uses the connotation of the word to express their toxicity. In other words, machine learning focuses on syntax rather than semantics.

REFERENCES

[1] Fabian P., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. (2011),"Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Reseach*, 12, 2825-2830.

[2] McCromick C.(2013), "Adaboost Tutorial", McCromick. Retrieved from `http://mccormickml.com/2013/12/13/adaboost-tutorial/`

[3] Kluyver T., Ragan-Kelly B., Pérez F., Granger B.,Bussonnier M., Frederic J., Kelley K., Hamrick J., Grout J., Corlay S., Ivanov P., Avila D., Abdalla S., Willing C. (2016), "Jupyter Notebooks – a publishing format for reproducible computational workflows". *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, IOS Press, 87-90.

[4] Ongmongkolkul P.(2016), "Naive Bayes [Lecture notes on ICCS 478]". *Mahidol University.*

[5] Ongmongkolkul P.(2016),"Logistic Regression [Lecture notes on ICCS 478]." *Mahidol University.*

[6] Ongmongkolkul P.(2016),"Overfitting and Regularization [Lecture notes on ICCS 478]." *Mahidol University.*

[7] Freund Y., Schapire R. (1997),"A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", *Journal of Computer and System Sciences*, 55(1), 119-139.

[8] Y'Barbo D.(2010),"Ways to improve the accuracy of Naive Bayes" *stackOverflow*. Retrieved from `https://stackoverflow.com/questions/3473612/ways-to-improve-the-accuracy-of-a-naive-bayes-classifier`

BIOGRAPHY

NAME	Nopphrakorn Kerdsamut
DATE OF BIRTH	9th October 1995
PLACE OF BIRTH	Samut Prakarn, Thailand
INSTITUTIONS ATTENDED	Thai-Chinese International School, 2001-2014 Upper Secondary School (Grade 12) Mahidol University, 2014-2018 Bachelor of Science (Physics)
HOME ADDRESS	99/225 Moo 8, Srinakarin Rd, Amphur Muang, Samut Prakarn, Thailand, 10270
E-MAIL	nopphrakorn.k@gmail.com