**Semantic Analysis of Comments**

**Profanity Vs. Free speech**

Writing behind the screen give us courage that usually is missing from our face to face interactions with others. Not seeing or knowing your audiences mitigate the fear of embarrassment or foul reputation. Our virtual identification can be totally untraceable to us, so we talk and behave without the fear of judgement and/ or consequences.

This freedom can easily take a wrong turn and create an unhealthy or even poisonous environment. In such an environment people easily lose sense of security or respect and stop expressing their thoughts and opinions and silence the chance of having a healthy and productive discussion in virtual environments.

The need for fostering a safe and healthy environment for conversations is so crucial yet challenging that some platforms rather limit or completely shut down user comments if they can’t facilitate a safe environment.

With all that said, we can see it’s very important for platforms to be able to protect their users from unwanted contents.

In this project, the challenge is to build model(s) that are capable of detecting the correct label (Offensive or non-offensive) for the comments in the dataset.

**Challenges:**

This dataset like any other real-world data is messy and far from perfect. Below is a list if problems needed to get addressed during the wrangling and pre-processing the text:

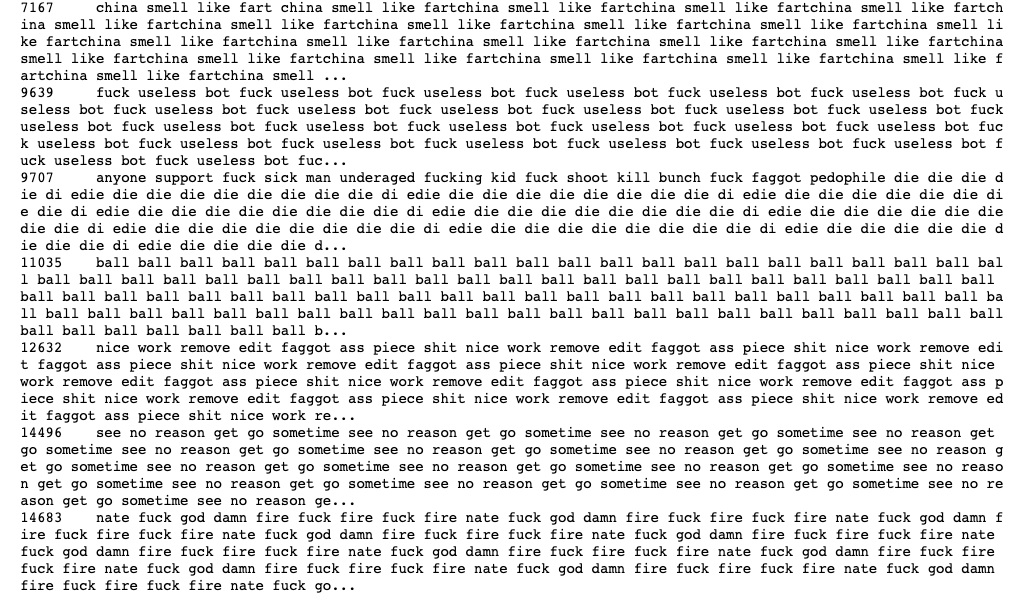
1- Spelling errors

2- Use of special characters such as $, %, !, @, /…

3- Mixed use of Lower and Upper cases in words and sentences

4- Text message abbreviations like JK (joking), JP (just playing), JT (just teasing) ,2morow(tomorrow), 2night(tonight), AAP(always a pleasure), AAF( as a friend), AAR8(at any rate), AAYF(always your friend), ACC(anyone can come), ATM(At the moment)can’t be decoded

5- Outliers: Comments longer than 500 words were considered as 500. Exploring them showed interestingly these comments are multiple copies of one phrase or sentence.



**Data Overview:**

The dataset in this project was obtained from kaggle website for one of their competitions, listed as “[Toxic Comment Classification Challenge](https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data)”.

Each comment has 6 labels, listed as:

* toxic
* severe\_toxic
* obscene
* threat
* insult
* identity\_hate

Scope of this project is to categorize comments into two classes: “Offensive” and “Non-Offensive”. Comments that are marked true for any of the labels listed above will be classified ad “Offensive” and if it hasn’t marked for any of the labels will be classified as “Non-Offensive”.

*Disclaimer: the dataset for this competition contains text that may be considered profane, vulgar, or offensive.*

**Text Representation with Feature Engineering:**

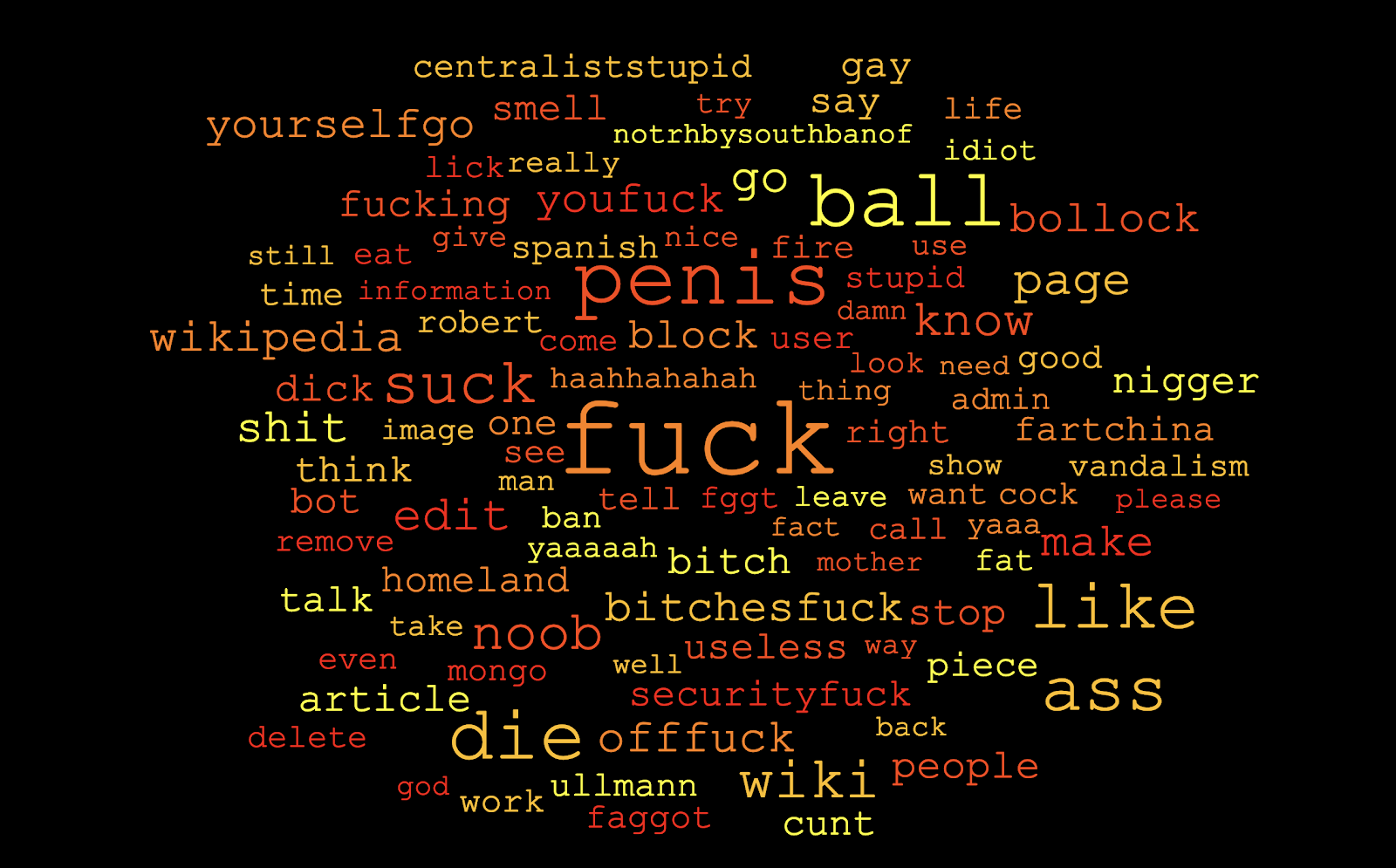
Feature engineering is known as the secret sauce to create a superior and better performing machine learning models.

Traditional Feature Engineering (Bag of Words):

Most frequent words in Non-Offensive comments:



Most frequent words in Offensive comments:



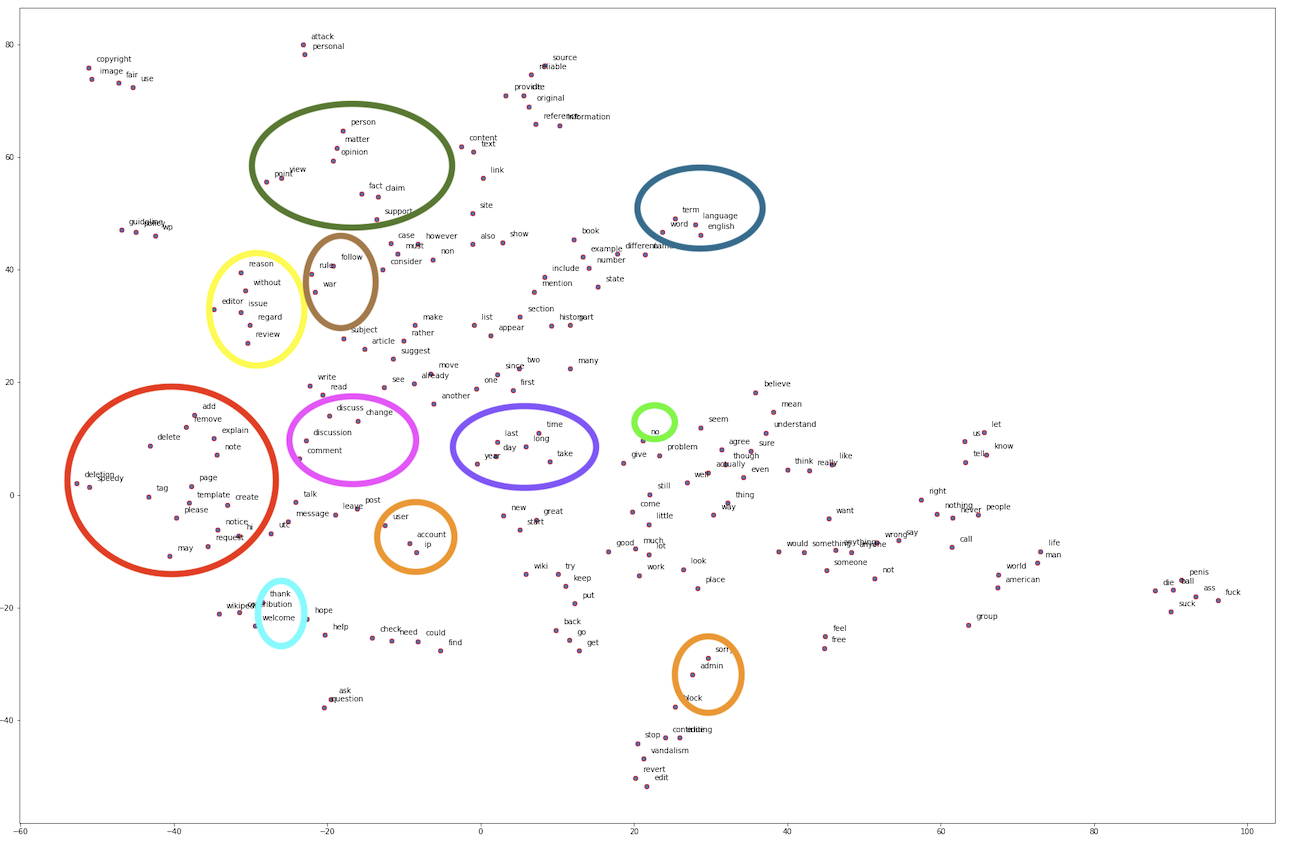
Traditional feature engineering for textual data includes Bag of Words and TF-IDF, while they are effective methods for extracting features from text, due to the inherent nature of the model being just a bag of unstructured words, we lose additional information like the semantics, structure, sequence and context around nearby words in each text document.

This creates enough motivation for us to explore more sophisticated methods which can capture this information and give us features which are vector representation of words, popularly known as embedding.

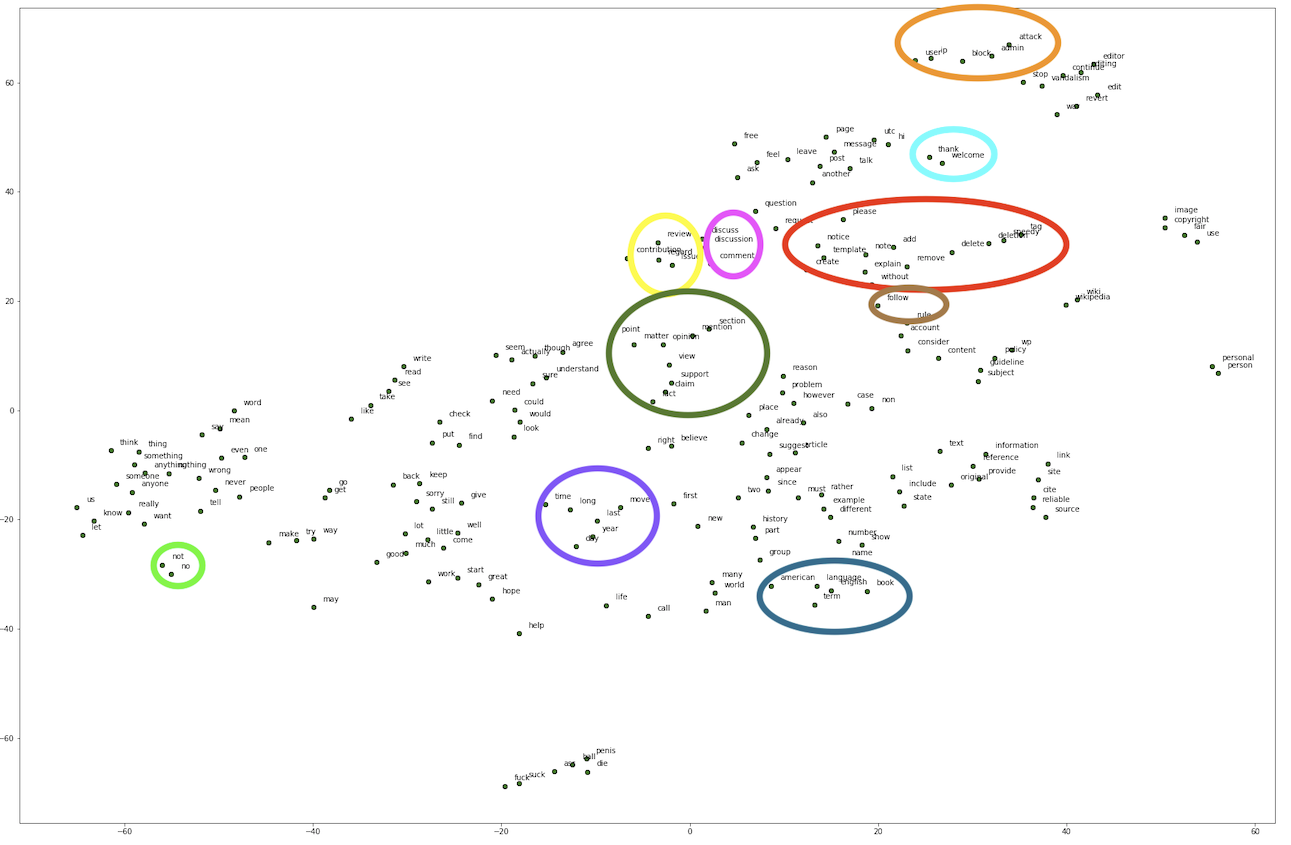
Techniques we explored in this project are:

* Word2Vec
* FastTex

**Word2vec:**



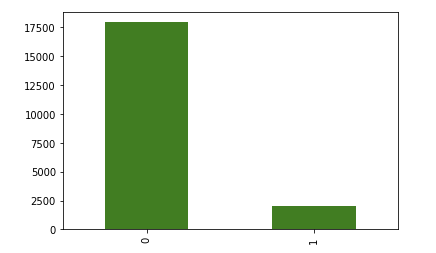
**FastText:**

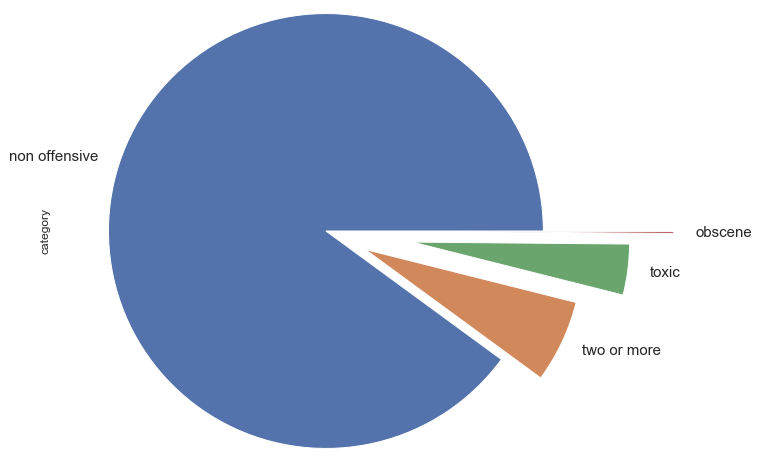


Comparing FastText with Word2vec shows slight improvement in clustering similar words.

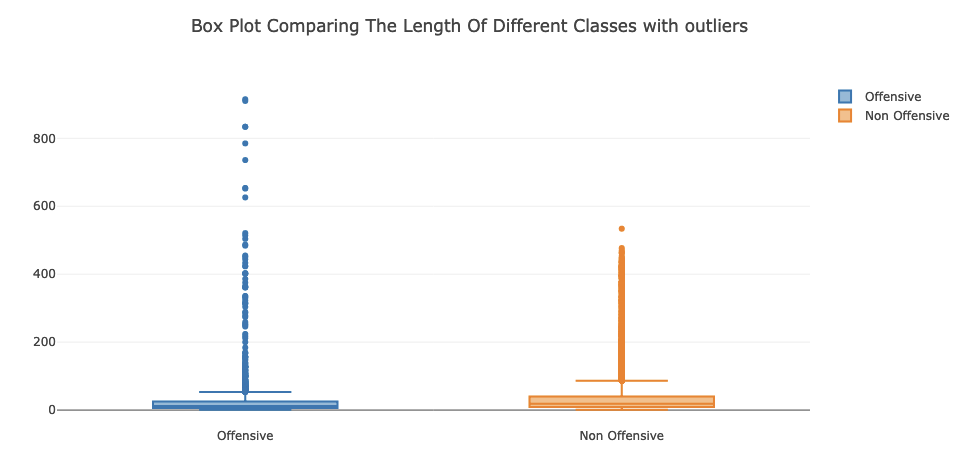
**Exploratory Data Analysis (EDA):**

Our dataset is imbalance, there is a huge gap between number of offensive texts and non-offensive texts.

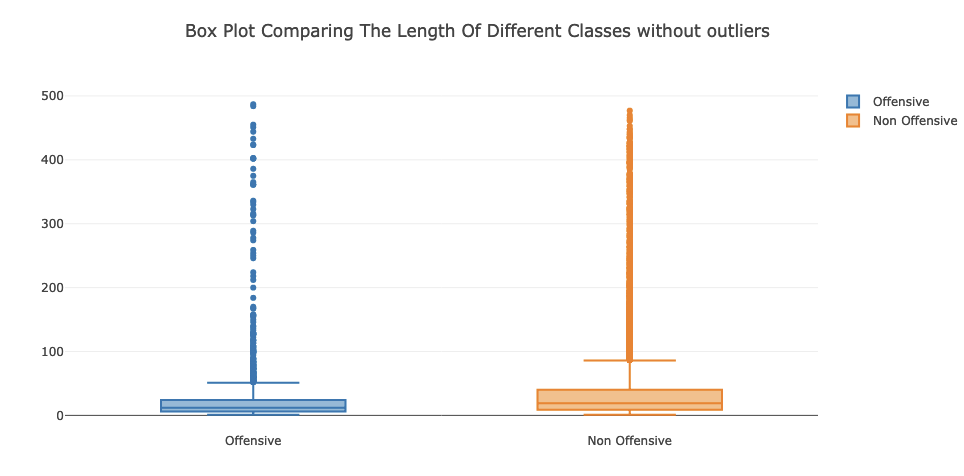




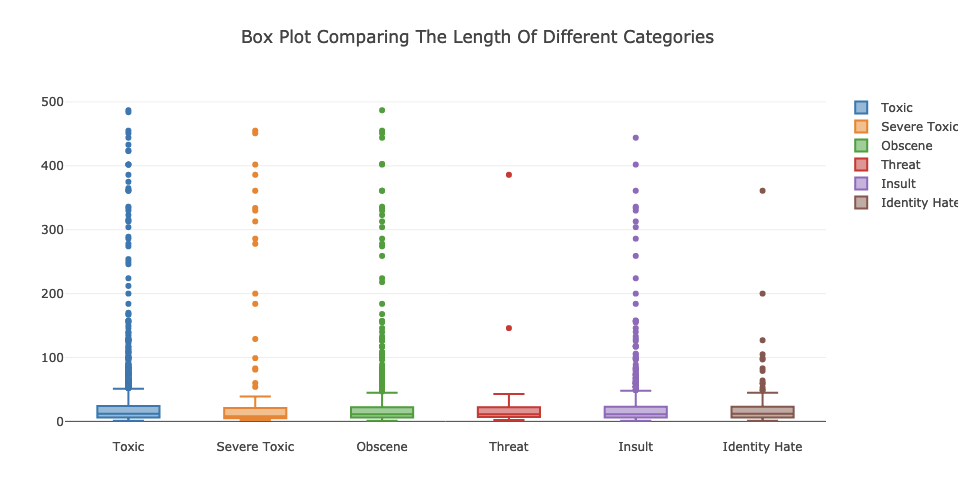
It appears Non-offensive comments are longer than Offensive comments:

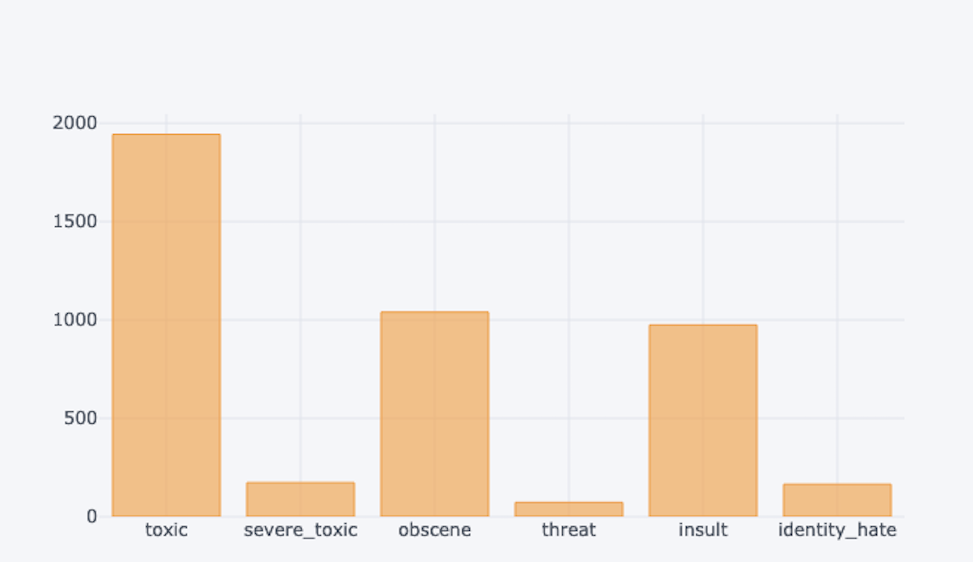


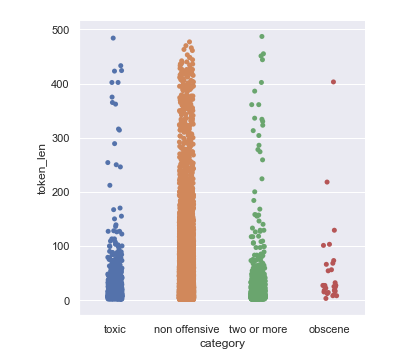
After removing outliers, Offensive text messages are still longer than Non-Offensive messages:



Comparing the length of different categories of offensive comments shows comments marked as “Threat”, “Identity Hate” and “Sever Toxic” are longer than other categories. We also see the number of comments tagged as these categories are much less than “Toxic”, “Obscene”, “Insult”.







**Model Building and predictions:**

As it was mentioned earlier, we have two main classes: “Offensive” and “Non-Offensive” and the main goal of this project is to find the best predictor to classify comment in the correct classes. Normally the first go to for evaluating machine learning models is accuracy, BUT that’s not the case for every case. Let’s pause here and have another look at our data.

Exploratory Data analysis showed there is a huge imbalance between comments in Offensive class and Non-Offensive class. When we deal with an imbalance dataset accuracy is not the most reliable metric for evaluating the classifier. Model can classify all the comments in the bigger class and results a high accuracy, yet not being a good predictor.

In this project like similar classification problem with imbalanced classes we need to evaluate models based on number of False Negatives and False Positive, basically it’s important for us to see how many comments were classified in the wrong classes.

The goal of this project is to detect and flag comments with offensive content. According to our business plan, these flagged comments will be sent to the review team for detailed review. For our business it’s important to catch as many offensive comments as possible, the cost of missing an offensive comment is more than the cost of further review. So, due to the nature of this problem our focus is on reducing number of False Negatives as much as possible.

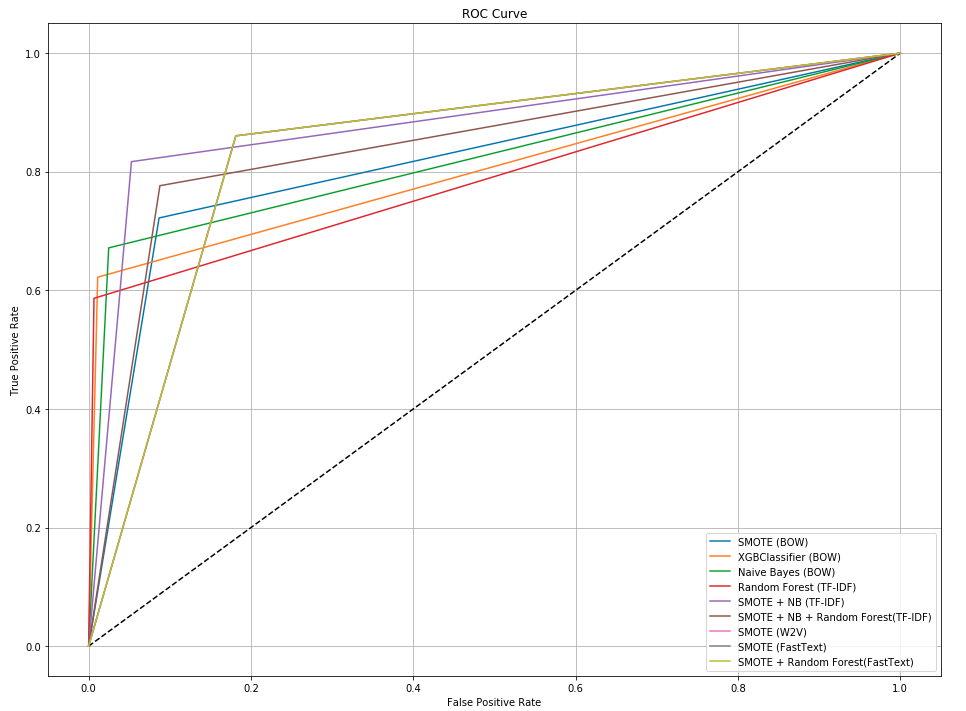
Recall is a metric that measure rate of true positive to the total number of predicted positives. Due to the nature of this problem it would be the most appropriate metric for evaluating performance of different models for this dataset.

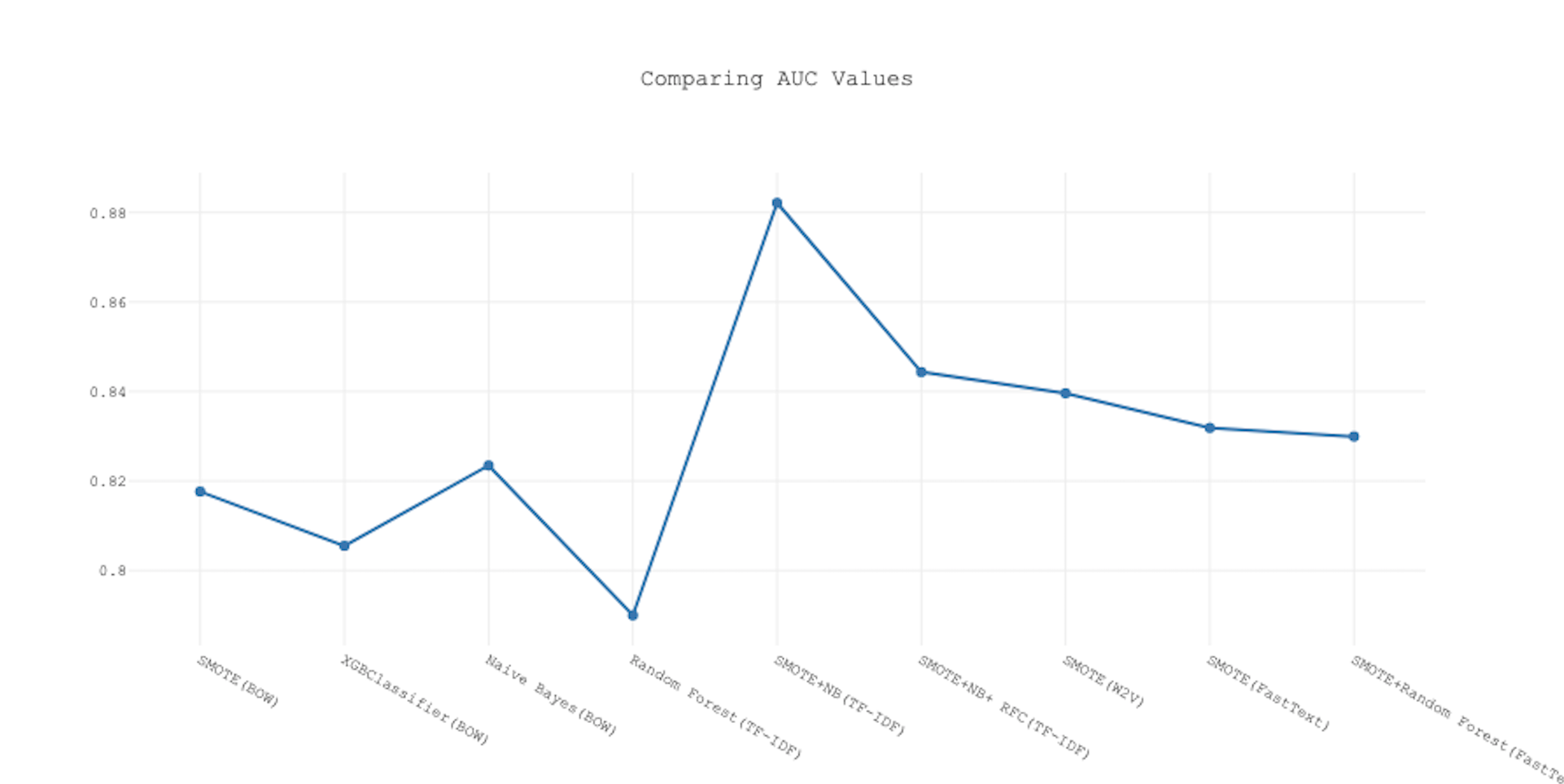
**Traditional Machine Learning:**

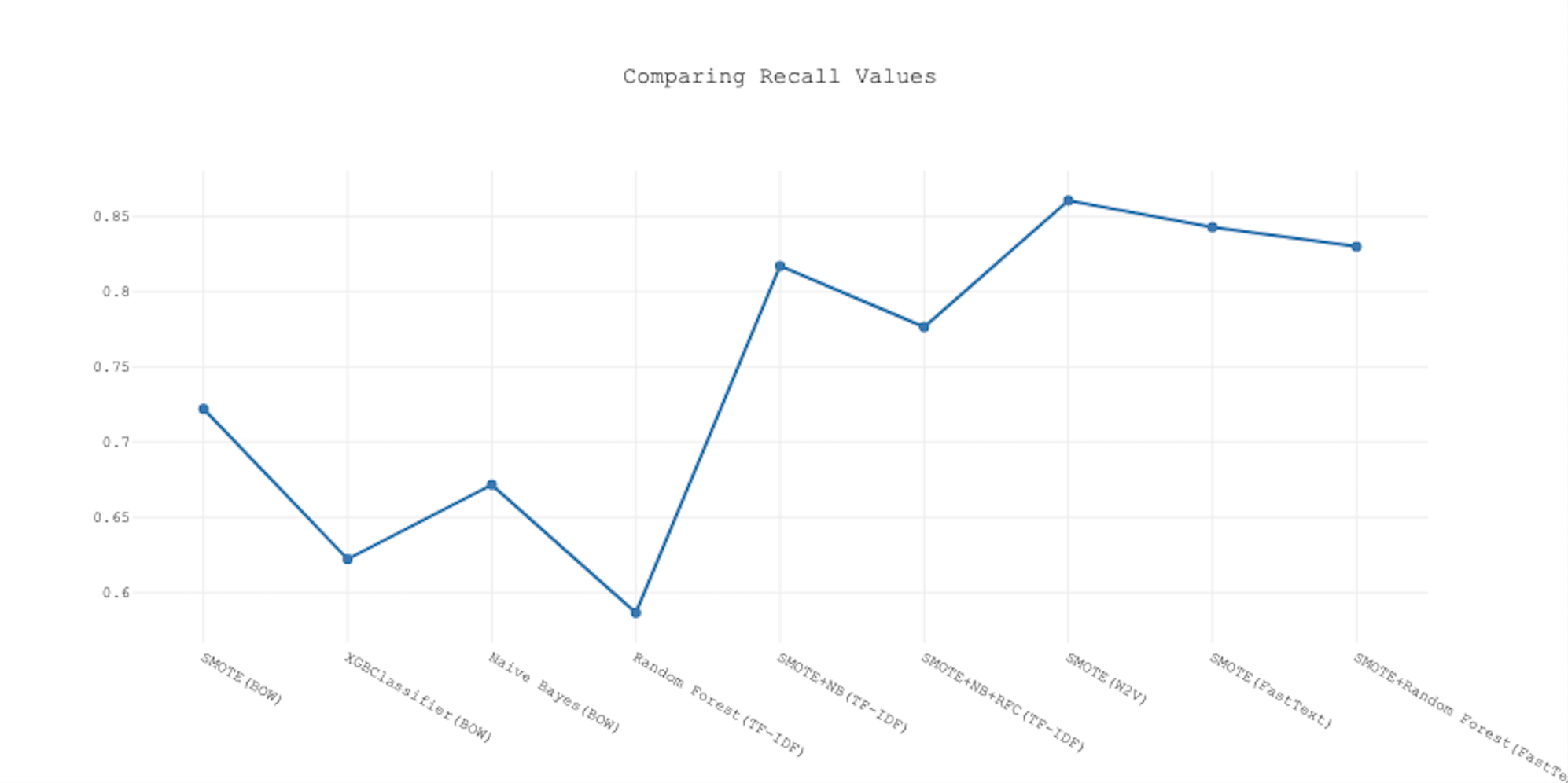
In order to find the most optimized model, we started with training some traditional machine learning algorithms such as “Logistic Regression, SMOTE, Naive Bayes, Random Forest and XGBoost” and tuned the hyper-parameters for each model via RandomizedSearchCV/ GridSearchCV to train the best classifier for the model. The best results were obtained from SMOTE followed by Naive Bayes.

The next step was to stack the optimized models and trained them on Bag Of Words, TF-IDF, Word2Vec, and FastText to see if we can get a better result from combined models. Logistic Regression-SMOTE trained on Word2vec appeared to have the least number of False Negatives and we experienced 15% improvement in Recall value.

Below is the visual comparison of the models:







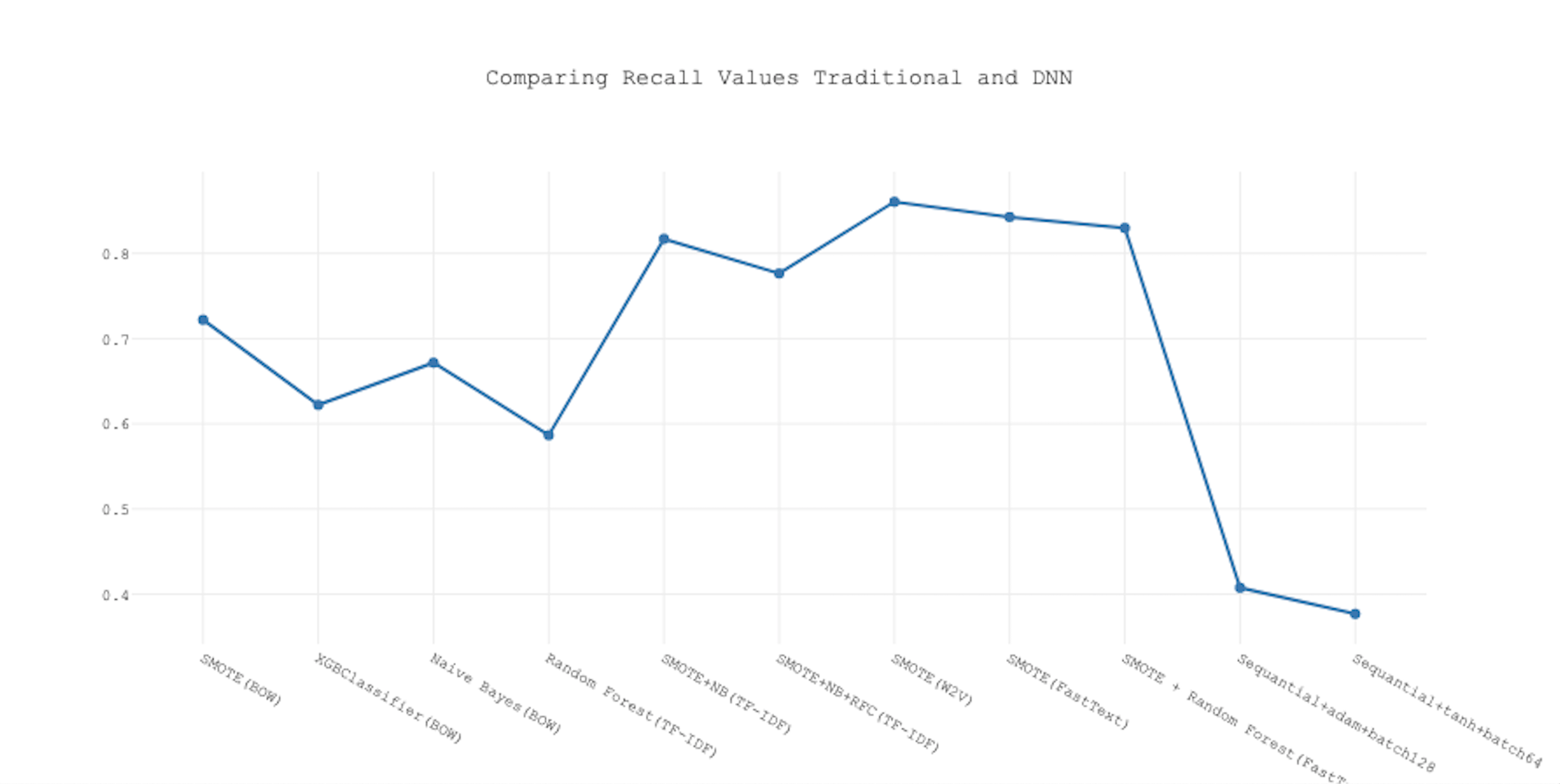
As you see in the presented graphs Logistic Regression -SMOTE trained on Word2Vec has the highest Recall value.

**Deep Neural Networks:**

In pursuit of the most optimized model, we created a Sequential model with 6 hidden layers and trained it on the averaged word vector of features from word2vec model. We created different versions of the model by choosing different:

* Optimizers: “adam” and “tanh”
* “Batch\_size” and “epochs”
* Neurons for each layer: 64 , 128

And trained each version on our data to find the most optimized model. Even though the overall accuracy was around 93%, but the high number of False Negatives showed our model and different variation of it are not good classifiers compare to some of traditional models.

****

**Conclusions:**

* In order to evaluate the performance of imbalanced datasets it’s not enough to rely solely on accuracy, depending on the problem we should focus on False Negatives or False Positives. In this project goal is reduce the number of False Negatives as much as possible or get the highest possible value for recall.
* After Optimizing different machine learning models such as Logistic Regression, SMOTE, Naive Bayes and Random Forest via RandomSearch CV and GridSearchCV we took one step further and combined the optimized models to see if we possibly can improve the value of Recall metric. Logistic Regression - SMOTE trained on Word2Vec generated the best possible Recall value of almost 86% and only misclassified 140 offensive comments as non-offensive.