**MY474 Applied Machine Learning for Social Science**

**Kaggle Competition Writeup**

**Preparing the dataset**

Load data and replace \n skip row sign with whitespace

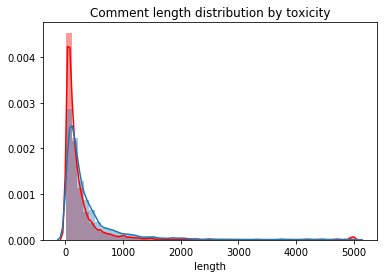
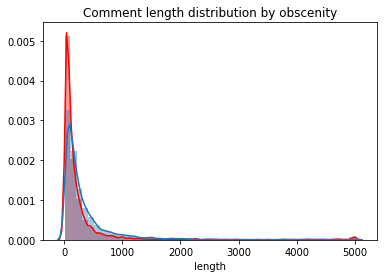
1. Create class object nlp that holds all the NLP algorithms and methods.
2. Create empty dataframe X that will hold all of the features.
3. Separate the two target variables from the training dataframe.

Create text corpus from train and test set together. All the relevant NLP methods will be done on both sets. In the end I'll separate them again for training the model and creating predictions.

## Exploratory Data Analysis and Feature Selection

## Comment length

By inspecting the length of comments grouped by the target variables groups we can see that it looks like toxic and obscene comments are usually shorter.

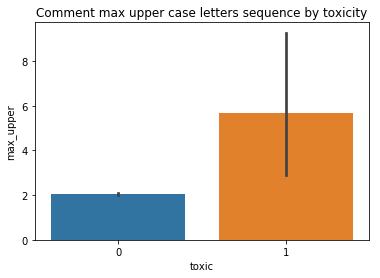
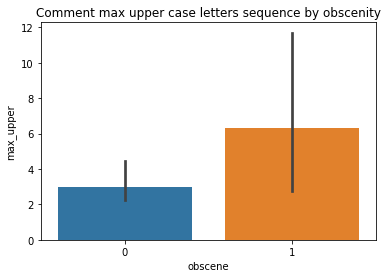


## Amount of upper-case letters

Inspecting the median upper-case letters amount we can see that toxic and obscene comments have less upper-case letters. I chose the median estimator to make the estimation more robust to outliers.

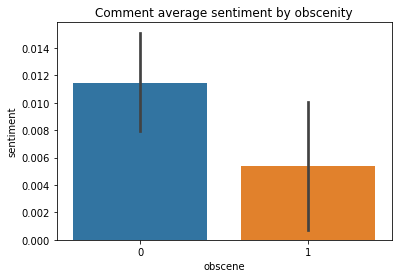
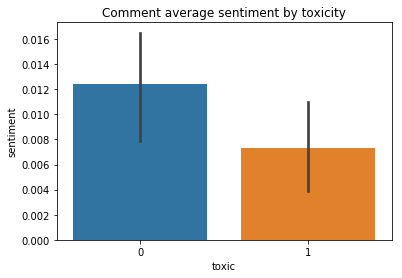
## C:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\10DA676.tmp C:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\9DBEB1F4.tmp Maximum sequence of upper-case letters

We can see that for both toxicity and obscenity, the average maximum sequence of upper-case letters is higher.



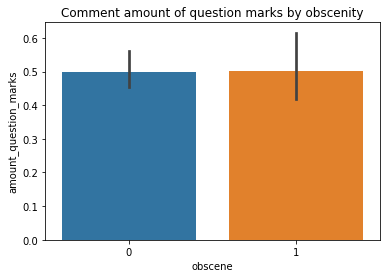
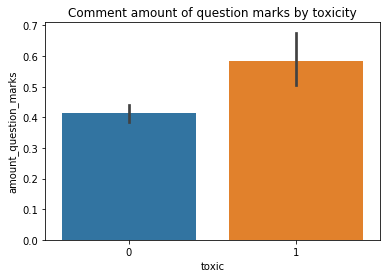
## Comment sentiment

Toxic and obscene comments tend to have less positive sentiment score

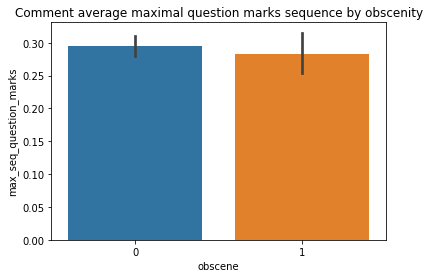
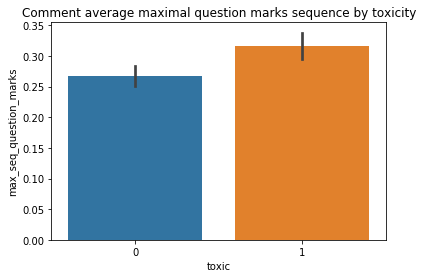


## Comment amount of question marks

Toxic comments have more question marks. Obscene comments don't have more question marks than non-obscene comments.



## Comment maximal question marks sequence

This feature's average values are different for toxic/non-toxic comments but for obscene/not obscene the average values are similar.

## 

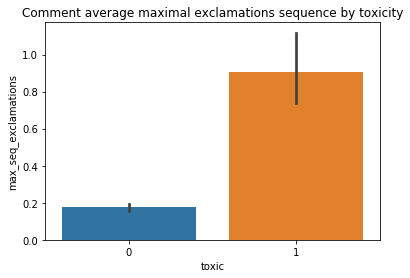
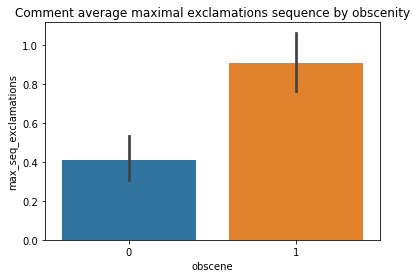
## Comment amount of exclamations[¶](http://localhost:8888/notebooks/Documents/GitHub/MY474_kaggle_competition/MY474_Kaggle_competition_notebook.ipynb#Comment-amount-of-exclamations)

We can see that for both toxic and obscene comments, the amount of exclamations is much bigger.

## 

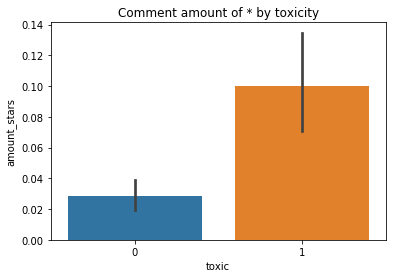
## C:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\833177BE.tmpC:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\775E6230.tmp

## Comment maximal sequence of exclamations

It is clear to see that for toxic and obscene comments the maximal sequence of exclamations is higher.

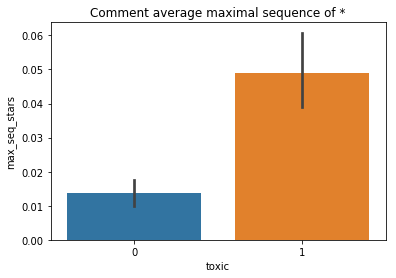
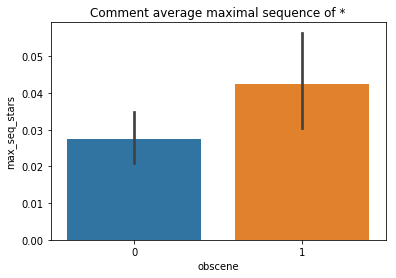
## 

## Comment amount of \*

For both toxic and obscene comments, the average amount of stars is higher.

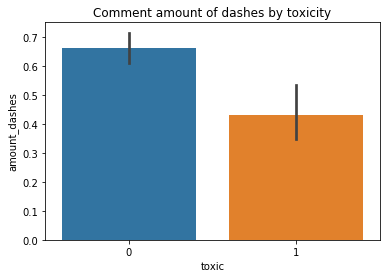
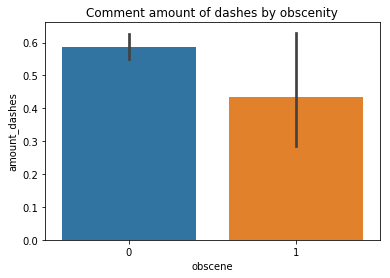
## C:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\528318D6.tmp

## Comment maximal sequence of \*

For both toxic and obscene comments, the average maximal sequence of \* is higher.

## 

## Comment amount of dashes

Toxic and Obscene comments have less dashes.

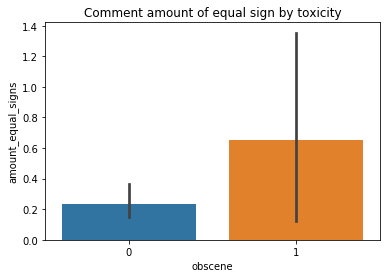
## 

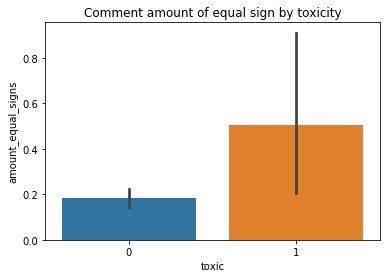
## Comment maximal sequence of dashes

Toxic and obscene comments have shorter maximal sequence of dashes on average.

## C:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B8D0B21A.tmpC:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\7300C72C.tmp

## Comment amount of equal sign

Toxic and obscene comments have more equal signs on average



After I finished inspecting and counting all punctuation in the comments, I am now removing it in order to proceed with the rest of the analysis.

In the next function I am performing a few more clean-up procedures on the comments' text:

1. Turn all text to lower case. This is necessary for avoiding duplications in word processing.
2. Split all sentences to words for the rest of the analysis.
3. Remove i.p addresses - Wikipedia comments tend to have i.p addresses as words, I remove them using ReGex.
4. Remove comments with less than two words - these carry almost no information hence I will discard them.

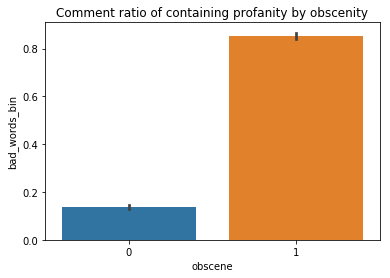
I remove the stopwords from the corpus since they contain no valuable information.

Here I will be using one of two methods - either Lemmatization or Stemming. This is done for reducing words to their stem or lemmas in order to decrease the feature space and sparsity. Lemmatization is better but more computationally and time expensive therefore I will only use it in the final model.

I add bigrams of each comment. This technique helps add context to some of the words.

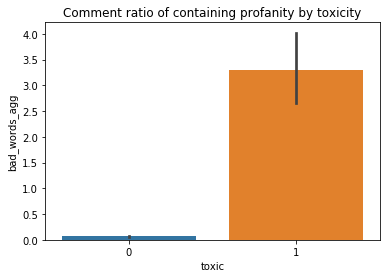
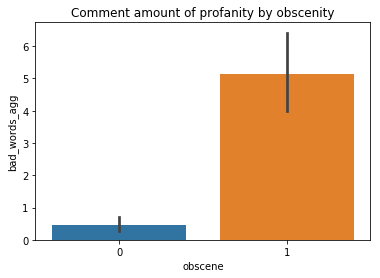
I am using a list of profanity words I have found on <https://github.com/RobertJGabriel/Google-profanity-words/blob/master/list.txt> to identify swear words in each comment. I create two different features - the 'bin' one is binary and the 'agg' one is aggregating the amount of swear words

## Ratio of comments that include profanity

It is clear to see that both toxic and obscene comments have a higher ratio of profanity

## C:\Users\natan\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\4A1A5604.tmp

## Amount of profanity

In this case we get the same results - toxic and obscene results have a higher amount of profanity

## 

## Gensim dictionary and filtering extremes

For the algorithms I am planning to use next, I will need to create a Gensim dictionary. This is a dictionary of all the words and bigrams I have in the corpus. I filter extreme word cases - if a word/bigram appears in less than 11 comments, this is not enough observations for the algorithm to make conclusions therefore I remove them. I also remove words that appear in more than 80% of the corpus since these also don't contain any unique information that will help classifying.

## Tf-Idf algorithm

I use the Gensim dictionary to create a Tf-Idf table. This is short for term frequency–inverse document frequency. It is a table that holds a value for each word in each document. I chose this algorithm over other alternatives because it is very useful in classification problems. It considers the weight of the word relative to both the specific document and to all of the corpus.

## Latent Dirichlet allocation

I use a topic modelling algorithm for assigning one of 20 topics to each comment. I suspect that some topics are more sensitive and therefore will have more toxic and obscene comments.

I turn the Tf-Idf object to matrix that can be appended to the features' matrix X.

## Scaling the data

Some of the algorithms I will be using require the feature space X to be standardized, so I use SKLearn's standard scaler for scaling X. I do this excluding the tfidf matrix which already has a different kind of standardization.

I join the tfidf table with the rest of the features and separate it to train and test matrices.

I will now be trying out a few different Machine Learning algorithms for predicting the test set. I will be using cross validated F1 score estimates which is the metric used for evaluating performance in this specific competition.

## Logistic Regression

This is a classification algorithm which I found very useful because it allows to add a penalty to the function. I used the l1 penalty parameter (Lasso) because it allows for some coefficients of the model to be totally zeroed out. This is particularly useful for this kind of feature space because of its huge size of predictors. Because we want to generalize our model, which is very complicated due to the number of features we would like to get rid of some of the features that are less useful or correlated with other features. This helps to lower the model's variance and improves the test score performance. I will use the CV results to pick the C tuning parameter which is tuning the regularization strength.

Here are the results of the cross validated error for the toxic target variable:

| **c\_value** | **cv\_f1\_score** |
| --- | --- |
| 0.1 | 0.772727 | |
| 2.0 | 0.885949 | |
| 3.0 | 0.886442 | |
| 4.0 | 0.887637 | |
| 5.0 | 0.887201 | |
| 6.0 | 0.885627 | |

And here are the results for the obscenity target variable:

| **c\_value** | **cv\_f1\_score** |
| --- | --- |
| 0.1 | 0.779277 |
| 0.5 | 0.805336 |
| 2.0 | 0.807040 |
| 2.5 | 0.808841 |
| 3.0 | 0.806664 |
| 3.5 | 0.810189 |
| 4.0 | 0.807755 |

## Random Forest Classifier

This classifier performs very well due to its ability to lower variance by creating many different uncorrelated decision trees. I will produce random values for some of the tuning parameters for getting the best CV F1 score. I chose a random grid search of the parameters because searching over all possible values can be very expensive computationally and time consuming. I chose the max depth of the trees to be no more than 75 because I don't want the tree to be too complicated and overfit the data. I chose 150 trees to be the maximal number of trees possible for calculation because any more than that will be computationally very expensive. I also chose the class\_weight to be balanced due to the fact that for at least one of the target variables the classes are not even in size.

Here are the results for the toxicity target variable:

| **cv\_f1\_score** | **max\_depth** | **n\_estimators** | **max\_features** |
| --- | --- | --- | --- |
| 0.865457 | 99.0 | 132.0 | auto |
| 0.859909 | 131.0 | 285.0 | 0.25 |
| 0.860589 | 97.0 | 265.0 | 0.5 |
| 0.861168 | 72.0 | 71.0 | 0.5 |
| 0.863154 | 119.0 | 92.0 | auto |

And here are the results for the obscenity target variable:

| **cv\_f1\_score** | **max\_depth** | **n\_estimators** | **max\_features** |
| --- | --- | --- | --- |
| 0.811807 | 19.0 | 155.0 | 0.8 |
| 0.812379 | 130.0 | 235.0 | 0.8 |
| 0.811713 | 142.0 | 193.0 | 0.8 |
| 0.815711 | 41.0 | 167.0 | 0.8 |
| 0.808827 | 88.0 | 52.0 | 0.8 |