CAPSTONE PROJECT #1
TOXIC COMMENTS

BY KING WONG
MAY 4th, 2019

<u>* WARNING ***</u>**
<u>REPORT CONTAINS EXPLICIT LANGUAGE</u>

"lol at all of these ***idiots*** *who don't know about musical genres. Oh yeah, you can change Rihanna's genre to R&B; because that isn't even the type music she makes. lmfao! Go take a hike,* ***retard****."*

1. INTRODUCTION

Background:
Many websites allow users to post and share their comments. This type of information sharing helps people connect and have better understanding of each other.

While some people share their opinions in a meaningful and respectful manner, some are opposite and their comments are negative, offensive and hurtful towards others. This type of negative comments devalues the communication channel and ruins the experience for others. Furthermore, it may even deter people from visiting or commenting on the website.

Objective:
Develop machine learning models to evaluate user comments (input) and predict the category of toxicity (output). Client can use this model to flag, filter and/or remove these toxic comments from their website posts.

Data Set:
The data set contains a large number of Wikipedia comments hosted in the below location.
< https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>

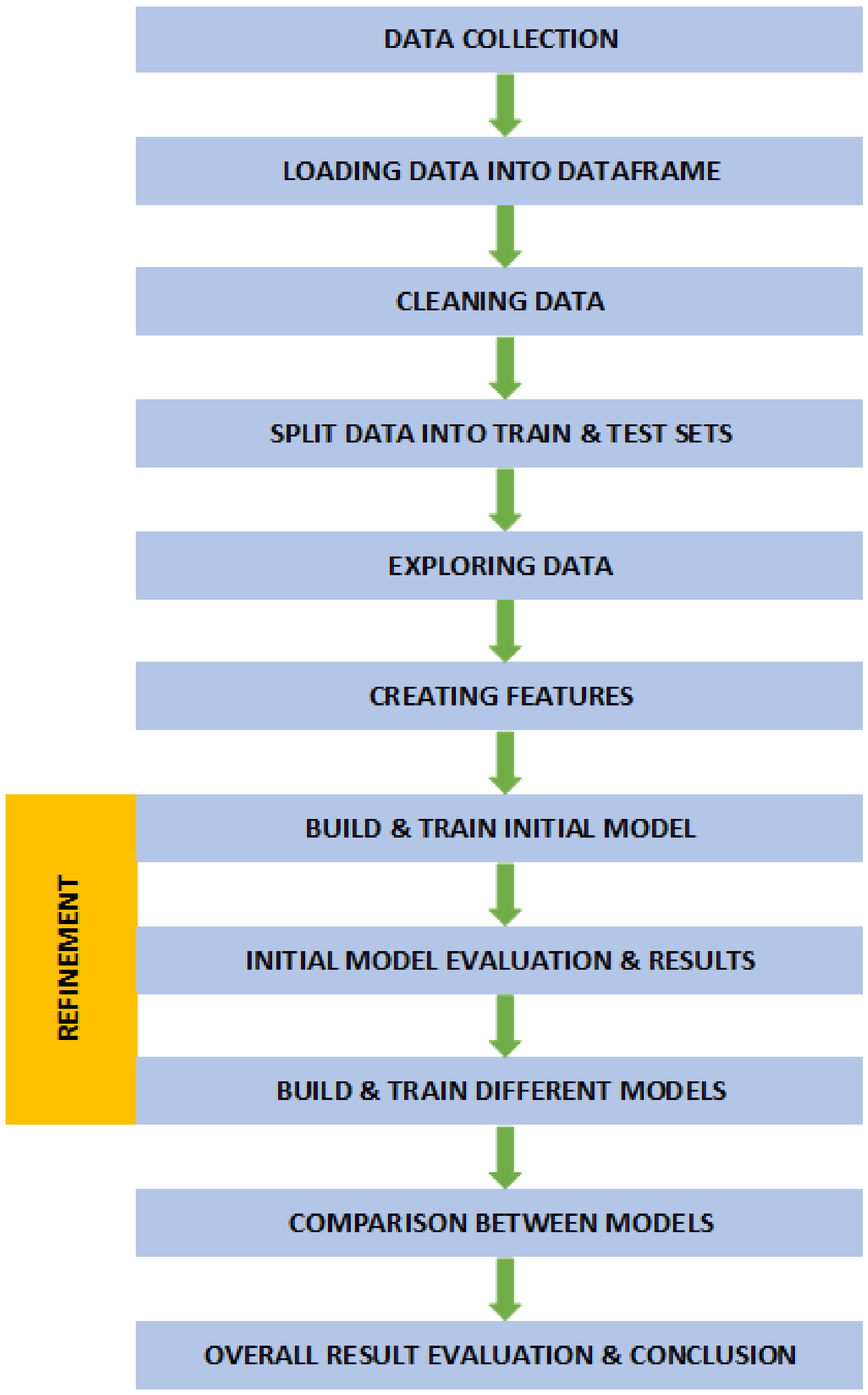
The original data set contains a training data set with [159571] comments and a test data set with [63978] comments. The comments in the training and test sets are labeled and the target labels / categories are as follow.

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

This is a supervised learning and a multi-label classification problem. Features from the comments are extracted as predictors and the target labels as listed above are to be predicted.

• Methodology

Below is a flowchart illustrating the process of this study


DATA COLLECTION
LOADING DATA INTO DATAFRAME
CLEANING DATA
SPLIT DATA INTO TRAIN & TEST SETS
EXPLORING DATA
CREATING FEATURES
BUILD & TRAIN INITIAL MODEL
INITIAL MODEL EVALUATION & RESULTS
BUILD & TRAIN DIFFERENT MODELS
REFINEMENT
COMPARISON BETWEEN MODELS
OVERALL RESULT EVALUATION & CONCLUSION

2. LOADING AND EXPLORING DATA SET

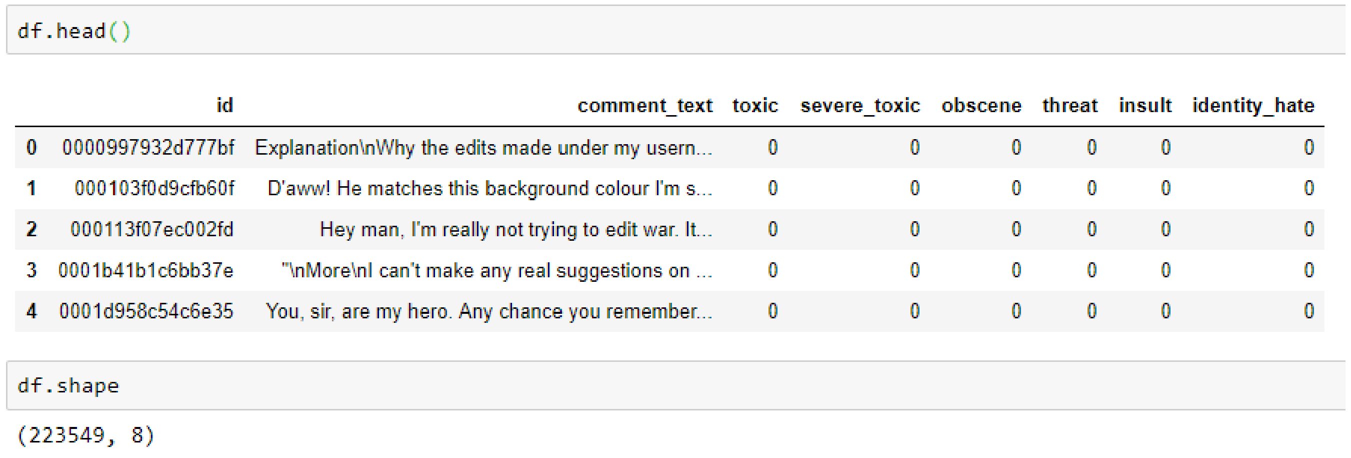
• Loading data into Pandas dataframe

The data is downloaded through web browser in zip file from the following location.
< https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>

The training data and test data sets are stored in .csv files. The data sets are loaded into Jupyter Notebook for review. The training data set contains a total of 159571 rows and 8 columns [159571x8].

The test data set is similar, but the comment ID and texts are in one file while the labels are in a separate file. Using Pandas, the information in the two files are combined into one dataframe. Based on further review, many comments are also assigned "-1" to the labels. According to the source these data are not used for scoring. For this project, these comment entries are removed, reducing the entries from [153164 x 8] to [63978 x 8].

For this project, the provided training and testing data are combined into one [223549 x 8] dataframe. Train-test-split is used later to divide the training and testing sets.



```
df.head()
```

	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

```
df.shape
```

```
(223549, 8)
```

• First-look at the comments and labels

The first column contains the unique ID for each comment and the second column contains the raw texts / strings of the comments. The 3rd to 8th columns contain labels for each toxicity classification. For example, if a comment is classified as an insult, a value of "1" is assigned under column "insult" in that row. A comment can be assigned for more than label. If a comment does not have "1" assigned to any label, that comment is a non-toxic or clean comment.

COMMENT LABELED AS "0" = CLEAN COMMENT

COMMENT LABELED AS "1" = TOXIC COMMENT

Example:

This is clean comment with all labels as "0".

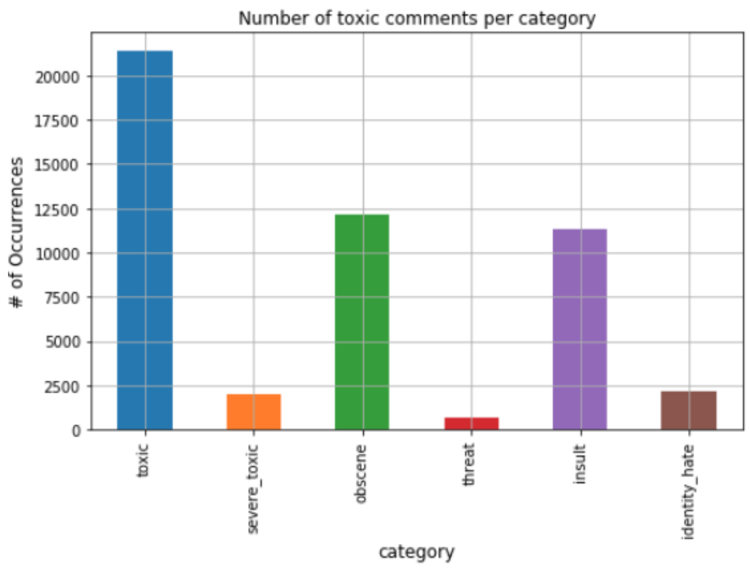
> *“There's no need to apologize. A Wikipedia article is made for reconciling knowledge about a subject from different sources, and you've done history studies and not archaeology studies, I guess. I could scan the page, e-mail it to you, and then you could ask someone to translate the page.”*

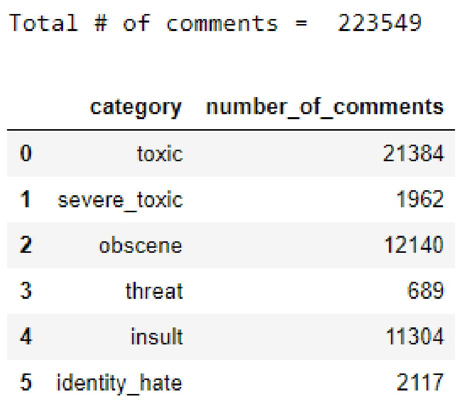
This is a toxic comment that has the categories “toxic”, “severe_toxic”, “obscene” and “threat” labeled as “1”.

> *“hey dickhole i'm gonna fuck you up assface so fuck off shitfucker*
> *-me*
>
> *Take this off again and i'm gonna whoop your ass so bad you'll be suffering and I'll be laughing hahahahahaha fuck you dickfucker”*

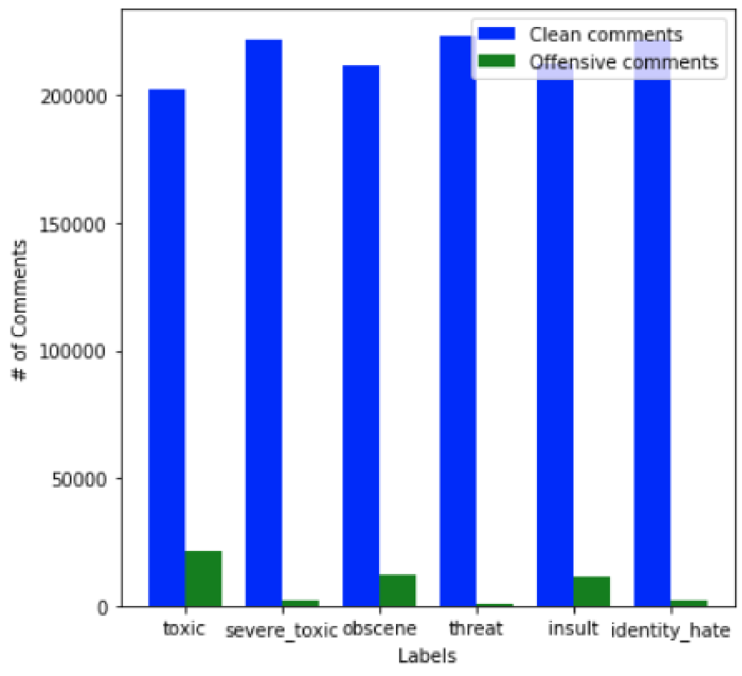
• Statistics of the data set.

Out of the 223549 total number of comments. The distribution of labeled comments for the 6 categories are shown below.




```
Total # of comments =  223549
```

	category	number_of_comments
0	toxic	21384
1	severe_toxic	1962
2	obscene	12140
3	threat	689
4	insult	11304
5	identity_hate	2117



```
Number of clean comments for each label: [202165, 221587, 211409, 222860, 212245, 221432]
Number of offensive comments for each label: [21384, 1962, 12140, 689, 11304, 2117]
Ratio of clean comments for each label: [0.90434312 0.9912234  0.94569423 0.9969179  0.9494339  0.99053004]
```

One can see that for each label, the number of comments labeled as toxic comments relative to the total number of comments are low, ranging from 1 to 10% depending on the category. In other words, most of the comments (90% to 99%) of the comments are clean comments.
There are relatively more comments labeled in categories ‘toxic’, ‘obscene’, and ‘insult’ than the rest.
One can consider this data set to be highly unbalanced.

3. EXPLORATORY DATA ANALYSIS

• Cleaning data

Preliminary study is performed on the dataset. Before doing that, cleaning is performed to the comment text of the training data set. The action items include:

• Convert all text to lower case (e.g. “Thomas” --> “thomas”)
• Convert contractions back to long form (e.g. “what’s --> “what is”)
• Remove non-alphabet and non-numbers (e.g. “!#$@”)
• Remove multi-space (e.g. “_ _“ --> “_“)

Example (see modifications in RED)

Before cleaning:

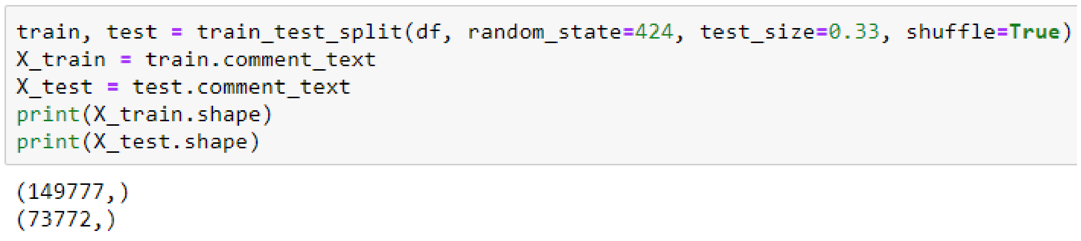
‘====Regarding edits made during August 9 2006 (UTC) to Oprah Winfrey====\nPlease do not replace Wikipedia pages or sections with blank content. It is considered vandalism. Please use the sandbox for any other tests you want to do. Take a look at the welcome page if you would like to learn more about contributing to our encyclopedia. Thanks. If this is an IP address, and it is shared by multiple users, ignore this warning if you did not make any unconstructive edits. don\'t talk email me ‘

After cleaning:

' regarding edits made during august utc to oprah winfrey please do not replace wikipedia pages or sections with blank content it is considered vandalism please use the sandbox for any other tests you want to do take a look at the welcome page if you would like to learn more about contributing to our encyclopedia thanks if this is an ip address and it is shared by multiple users ignore this warning if you did not make any unconstructive edits do not talk email me '

• Train-Test-Split

The full set of data [223549 x 8] is splitted to training set and test set at a 2:1 ratio and at a random state.



```
train, test = train_test_split(df, random_state=424, test_size=0.33, shuffle=True)
X_train = train.comment_text
X_test = test.comment_text
print(X_train.shape)
print(X_test.shape)
```

```
(149777,)
(73772,)
```

• Toxic Word Count

Secondly, a word count was performed for each toxic category to see what toxic words appear frequently in the toxic comments. Bag-of-words method was used for this.

The assumption here is that for each toxicity class, a generated bag-of-words from the comments would contain both toxic words and neutral words. By subtracting out the neutral words from the bag-of-words, only the toxic words remain.

{Toxic Words + Neutral Words } – { Neutral Words } = { Toxic Words }

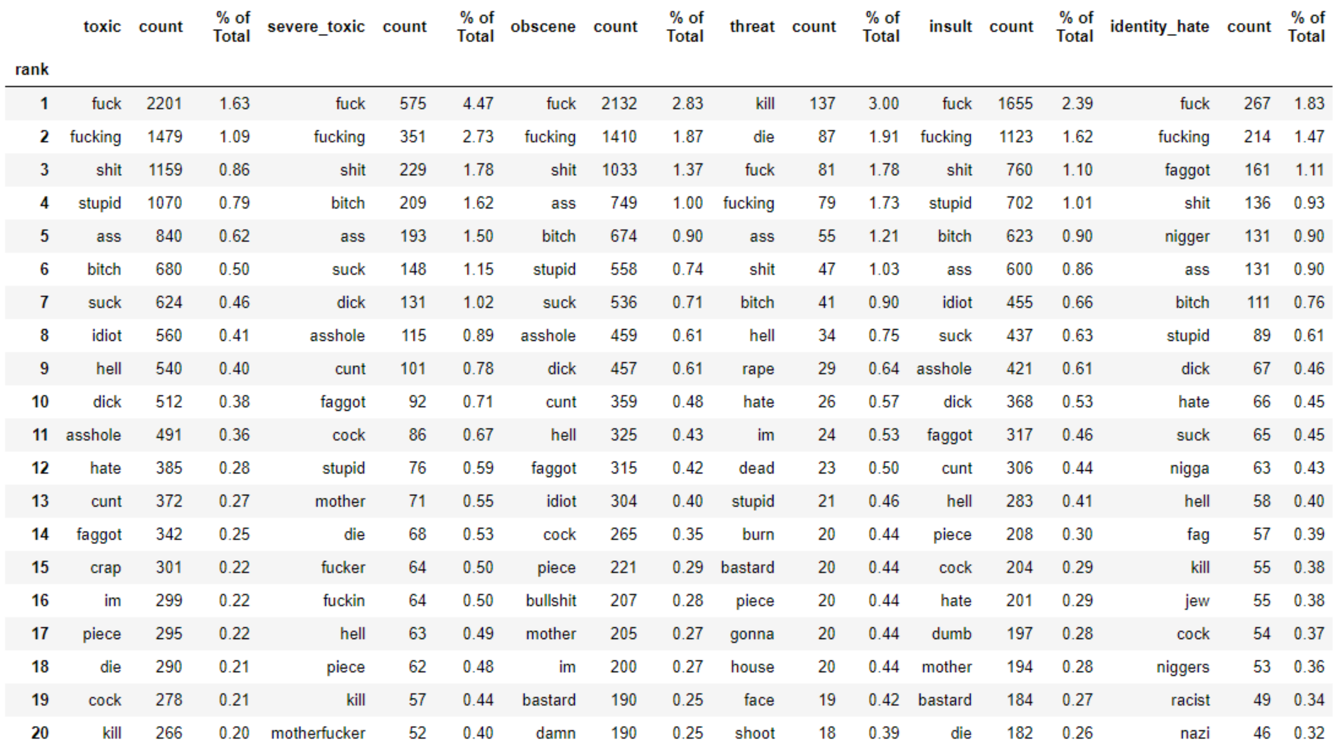
---------- Generated from comments in each toxicity class
---------- Generated from neutral comments

A bag-of-words for neutral words is generated from the clean comments that have no labels.

Six (6) bag-of-words, one for each toxicity label is generated from the toxic comments.

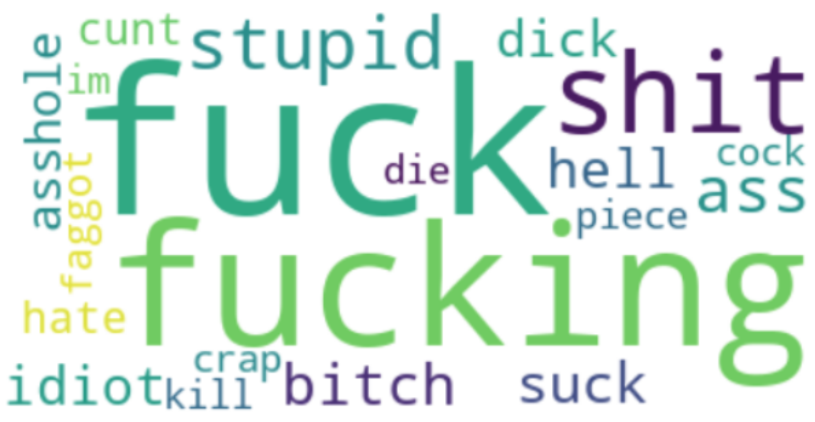
Note that some comments were labeled as having more than one type of toxicity. This aspect was ignored for this initial study for simplicity.

Below is a summary of the word count results, ranked based on the most frequently appeared toxic words for each toxicity class.

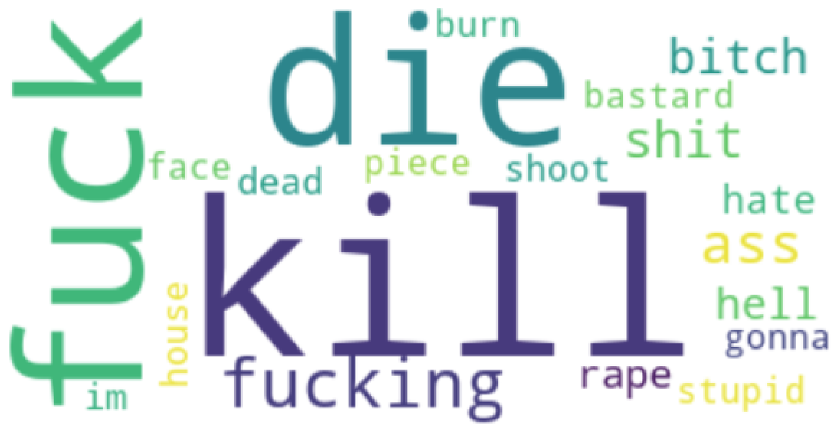


	toxic	count	% of Total	severe_toxic	count	% of Total	obscene	count	% of Total	threat	count	% of Total	insult	count	% of Total	identity_hate	count	% of Total
rank																		
1	fuck	2201	1.63	fuck	575	4.47	fuck	2132	2.83	kill	137	3.00	fuck	1655	2.39	fuck	267	1.83
2	fucking	1479	1.09	fucking	351	2.73	fucking	1410	1.87	die	87	1.91	fucking	1123	1.62	fucking	214	1.47
3	shit	1159	0.86	shit	229	1.78	shit	1033	1.37	fuck	81	1.78	shit	760	1.10	faggot	161	1.11
4	stupid	1070	0.79	bitch	209	1.62	ass	749	1.00	fucking	79	1.73	stupid	702	1.01	shit	136	0.93
5	ass	840	0.62	ass	193	1.50	bitch	674	0.90	ass	55	1.21	bitch	623	0.90	nigger	131	0.90
6	bitch	680	0.50	suck	148	1.15	stupid	558	0.74	shit	47	1.03	ass	600	0.86	ass	131	0.90
7	suck	624	0.46	dick	131	1.02	suck	536	0.71	bitch	41	0.90	idiot	455	0.66	bitch	111	0.76
8	idiot	560	0.41	asshole	115	0.89	asshole	459	0.61	hell	34	0.75	suck	437	0.63	stupid	89	0.61
9	hell	540	0.40	cunt	101	0.78	dick	457	0.61	rape	29	0.64	asshole	421	0.61	dick	67	0.46
10	dick	512	0.38	faggot	92	0.71	cunt	359	0.48	hate	26	0.57	dick	368	0.53	hate	66	0.45
11	asshole	491	0.36	cock	86	0.67	hell	325	0.43	im	24	0.53	faggot	317	0.46	suck	65	0.45
12	hate	385	0.28	stupid	76	0.59	faggot	315	0.42	dead	23	0.50	cunt	306	0.44	nigga	63	0.43
13	cunt	372	0.27	mother	71	0.55	idiot	304	0.40	stupid	21	0.46	hell	283	0.41	hell	58	0.40
14	faggot	342	0.25	die	68	0.53	cock	265	0.35	burn	20	0.44	piece	208	0.30	fag	57	0.39
15	crap	301	0.22	fucker	64	0.50	piece	221	0.29	bastard	20	0.44	cock	204	0.29	kill	55	0.38
16	im	299	0.22	fuckin	64	0.50	bullshit	207	0.28	piece	20	0.44	hate	201	0.29	jew	55	0.38
17	piece	295	0.22	hell	63	0.49	mother	205	0.27	gonna	20	0.44	dumb	197	0.28	cock	54	0.37
18	die	290	0.21	piece	62	0.48	im	200	0.27	house	20	0.44	mother	194	0.28	niggers	53	0.36
19	cock	278	0.21	kill	57	0.44	bastard	190	0.25	face	19	0.42	bastard	184	0.27	racist	49	0.34
20	kill	266	0.20	motherfucker	52	0.40	damn	190	0.25	shoot	18	0.39	die	182	0.26	nazi	46	0.32

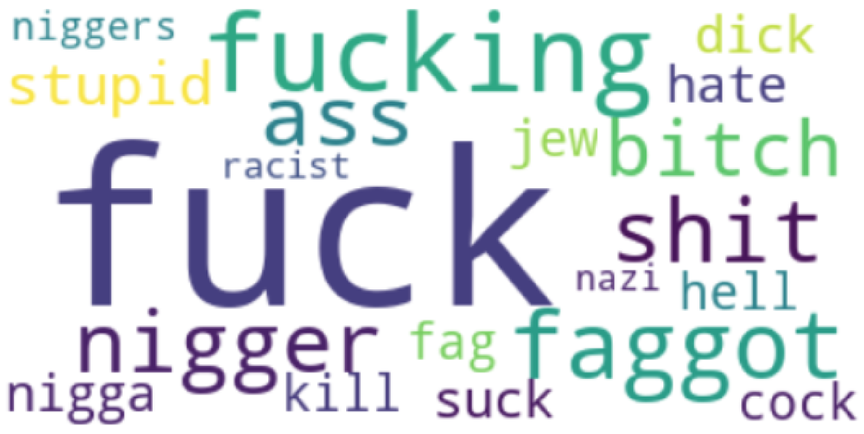
One can see that the toxicity classes share many similar words. Below is a word cloud generated for the ‘toxic’ category to illustrate the proportions.



Note that in the category ‘threat’, words such as ‘kill’ and ‘die’ are ranked high at the top.



In the category ‘identity_hate’, words such as ‘faggot’, ‘nigger’ are ranked high at the top.



The exploratory data analysis results seem to make sense.

4. BUILDING IN-DEPTH PREDICTION MODELS

Binary Techniques

• Multinomial Naive Bayes (MNB) model

Let’s begin by using a binary technique, which treats each toxicity class as an individual class assuming no correlation with other classes.

CountVectorizer is used to tokenize the comments and convert them into a matrix of token counts. Parameters of min_df =1, single word (1-gram) and English stop-words are used.

Using OneVsRestClassifier, all (6) subsets, one with each label, can be analyzed at once. The model is embedded in the pipeline function with the vectorizer for process.

```
NB_pipeline_t = Pipeline([
                ('vect', CountVectorizer(min_df=1, ngram_range=(1, 1),stop_words='english')),
                ('clf', OneVsRestClassifier(MultinomialNB(fit_prior=True, class_prior=None)))])

NB_pipeline_t.fit(X_train, train[categories].values)

prediction_NB = NB_pipeline_t.predict(X_test)
```

```
print('For all (6) labels:')
print('\t accuracy = {:.2%}'.format(accuracy_score(test[categories].values, prediction_NB)))
print('\t hamming_loss = {:.2%}\n'.format(hamming_loss(test[categories].values, prediction_NB)))
```

```
For all (6) labels:
         accuracy = 89.95%
         hamming_loss = 2.82%
```

If we consider it a success only when the predictions for all (6) labels are correct (see below graphic for example), then this model does not performing too well since the data is highly unbalanced with most of the comments being non-toxic / clean comments, as described in Section 3.

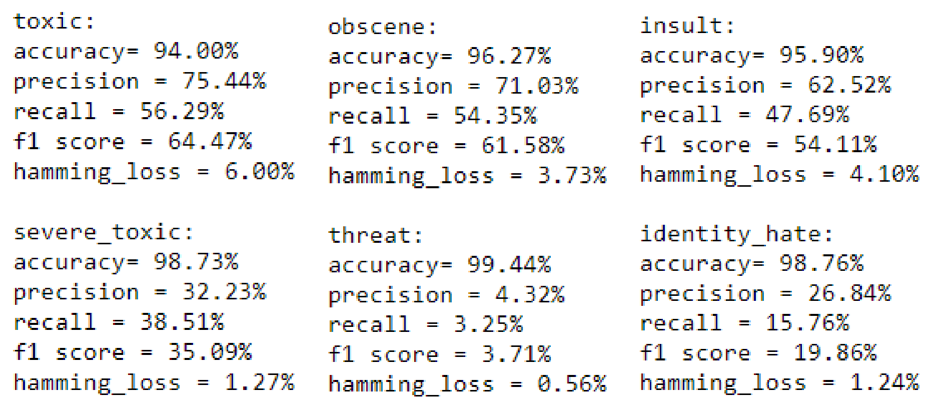
	Label	**toxic**	**severe_toxic**	**obscene**	**insult**	**threat**	**identity_ hate**	
Case 1	**ACTUAL**	1	0	0	1	1	0	**FAILURE**
	PREDICTION	1	0	0	0	0	0	
Case 2	**ACTUAL**	1	0	0	1	1	0	**SUCCESS**
	PREDICTION	1	0	0	1	1	0	

If one is to guess all clean comments (labels contain all 0’s) as a baseline, the accurary would already be at 89.82%. The model is not doing any better.

```
print('Accuracy if one guesses all clean comments = {:.2%}'.format
      (accuracy_score(test[categories].values, np.zeros((X_test.shape[0], 6)))))
```

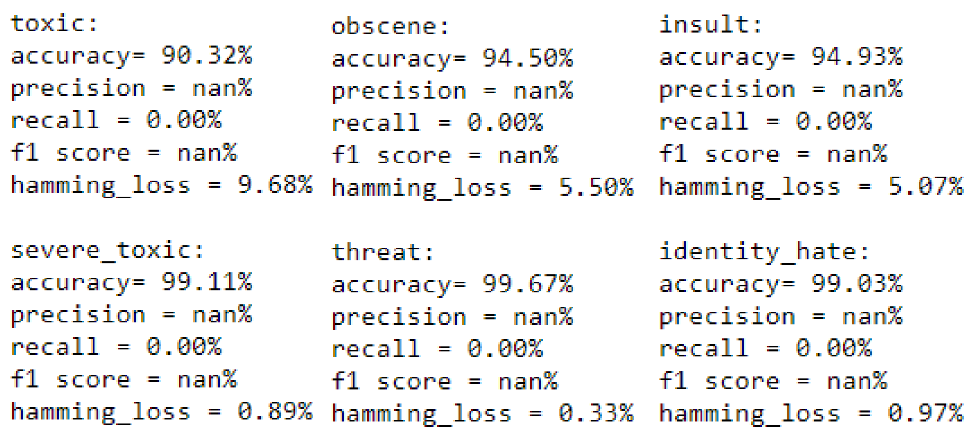
```
Accuracy if one guesses all clean comments = 89.82%
```

Let’s look at the results for the individual labels for the MNB model,



```
toxic:                     obscene:                   insult:
accuracy= 94.00%           accuracy= 96.27%           accuracy= 95.90%
precision = 75.44%         precision = 71.03%         precision = 62.52%
recall = 56.29%            recall = 54.35%            recall = 47.69%
f1 score = 64.47%          f1 score = 61.58%          f1 score = 54.11%
hamming_loss = 6.00%       hamming_loss = 3.73%       hamming_loss = 4.10%

severe_toxic:              threat:                    identity_hate:
accuracy= 98.73%           accuracy= 99.44%           accuracy= 98.76%
precision = 32.23%         precision = 4.32%          precision = 26.84%
recall = 38.51%            recall = 3.25%             recall = 15.76%
f1 score = 35.09%          f1 score = 3.71%           f1 score = 19.86%
hamming_loss = 1.27%       hamming_loss = 0.56%       hamming_loss = 1.24%
```

Comparing to the baseline of guessing all clean comments (all 0’s)



```
toxic:                     obscene:                   insult:
accuracy= 90.32%           accuracy= 94.50%           accuracy= 94.93%
precision = nan%           precision = nan%           precision = nan%
recall = 0.00%             recall = 0.00%             recall = 0.00%
f1 score = nan%            f1 score = nan%            f1 score = nan%
hamming_loss = 9.68%       hamming_loss = 5.50%       hamming_loss = 5.07%

severe_toxic:              threat:                    identity_hate:
accuracy= 99.11%           accuracy= 99.67%           accuracy= 99.03%
precision = nan%           precision = nan%           precision = nan%
recall = 0.00%             recall = 0.00%             recall = 0.00%
f1 score = nan%            f1 score = nan%            f1 score = nan%
hamming_loss = 0.89%       hamming_loss = 0.33%       hamming_loss = 0.97%
```

Notice that the MNB model performs better in predicting the categories of “toxic”, “obscene” and “insult” than the baseline. However, the MNB model performs worse in predicting the categories of “severe_toxic”, “threat” and “identity_hate” than the baseline.

Example of an incorrectly predicted comment ---

“*rebirth sales how the hell do you figure its well sourced what the hell source says rebirth sold over 500 000 units check your shit*“

This was labeled as “toxic” but the model did not predict it correctly.

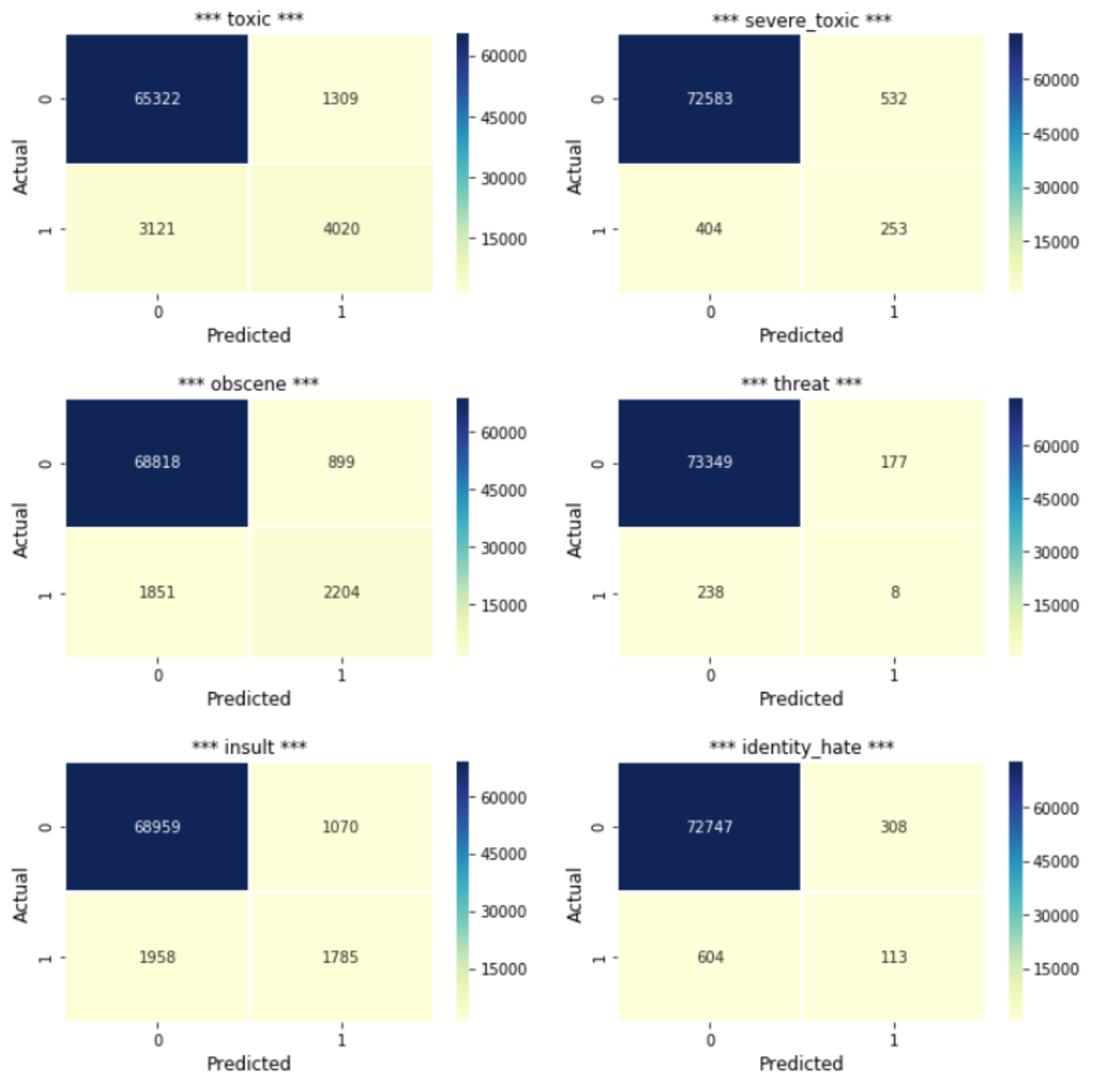
Let’s look at other metrics ---

True Positive (TP) = toxic comment predicted as toxic comment
False Positive (FP) = non-toxic comment predicted as toxic comment
True Negative (TN) = non-toxic comment predicted as non-toxic comment
False Negative (FN) = toxic comment predicted as non-toxic comment

Looking at the precision, TP / (TP + FP), the categories of “toxic”, “obscene” and “insult” can achieve as high as 75%. However, the rest of the categories perform poorly. For recall, TP / (TP + FN), the categories of “toxic”, “obscene” and “insult” can achieve only about 50%. This is not very good. Furthermore, the rest of the categories perform poorly.

The f1-score considers both precision and recall. This can be used as comparison going forward.

The hamming loss represents the fraction of labels that are incorrectly predicted. The lower the number means better model performance.
See below confusion matrix for each label to better visualize the results. Consistent with the recall score is conveying, given a toxic comment, it is only slightly better than a coin toss (50/50 chance) for the categories of “toxic” and “obscene”. The other categories are worse than guessing randomly.



Let’s use feature_count from the MNB’s classifier to see the most frequent words for the ‘toxic’ category and the corresponding proportions relative to the number of toxic comments. This in general agrees with the Word Count in Section 3.

```
token
fuck         0.771818
fucking      0.226708
suck         0.226006
faggot       0.208734
like         0.207821
nigger       0.206768
wikipedia    0.190269
ass          0.189286
shit         0.188514
hate         0.181703
gay          0.180088
die          0.171874
just         0.150460
fat          0.147090
know         0.141754
sucks        0.123359
page         0.120199
bitch        0.117882
moron        0.116548
stupid       0.109317
```

Using TfidVectorizer and hyper-parameter tuning

An effort is made to improve the results by using the TfidVectorizer and varying the parameters of min_df and n-gram range. Upon sensitivity analysis of the parameters, the MNB model slightly improves comparing to the previous in some aspects, but worsened in other aspects.

Below is another variation.

```
NB_pipeline_t = Pipeline([
                    ('vect', TfidfVectorizer(min_df=20, ngram_range=(1, 2),stop_words='english')),
                    ('clf', OneVsRestClassifier(MultinomialNB(fit_prior=True, class_prior=None)))])

NB_pipeline_t.fit(X_train, train[categories].values)

prediction_NB = NB_pipeline_t.predict(X_test)
```

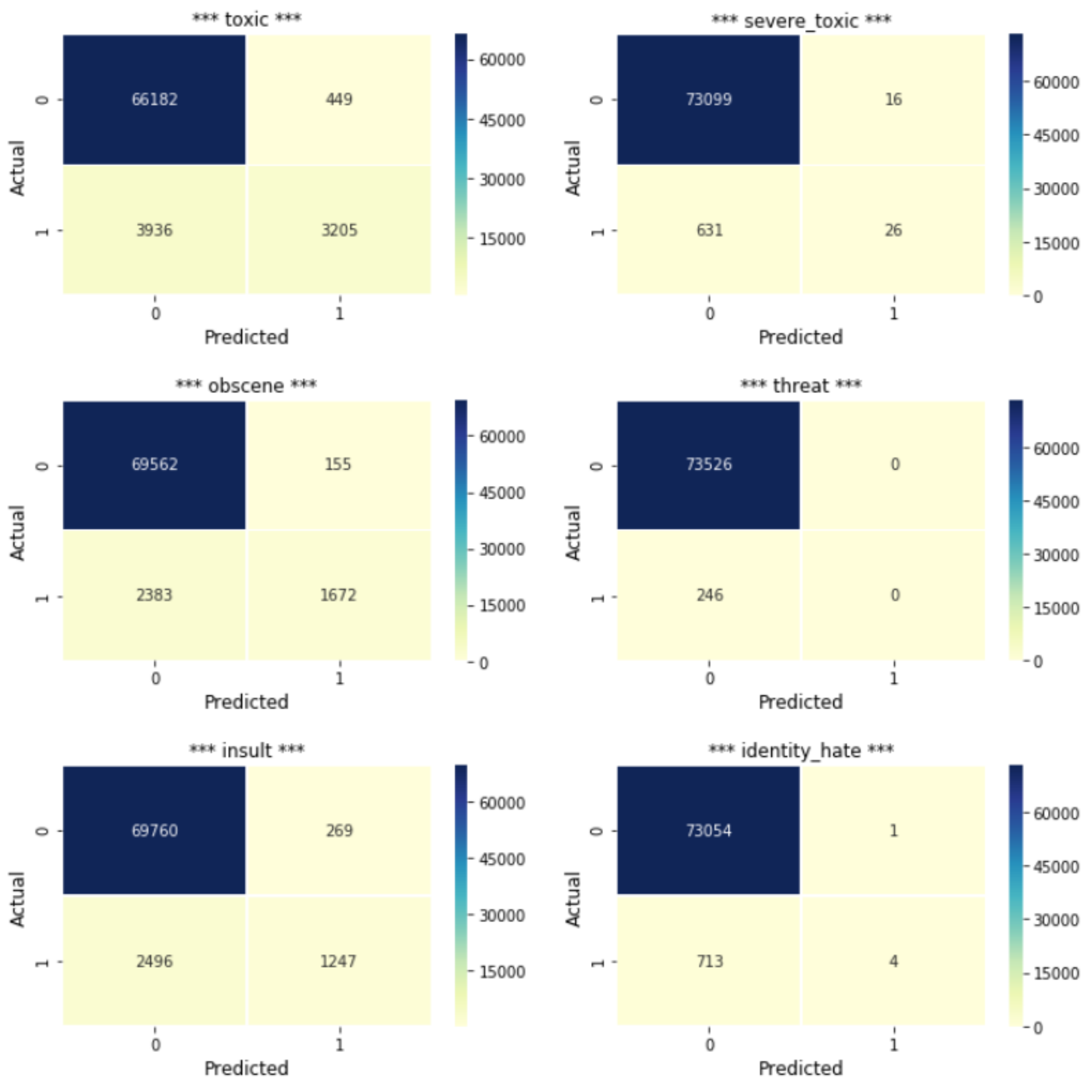
The overall accuracy for all (6) labels together has improved and now better than the baseline.

```
For all (6) labels:
          accuracy = 90.65%
          hamming_loss = 2.55%
```

Some of the results for individual labels have slightly improved and better than the baseline, however, some have slightly worsen.

```
toxic:                  obscene:                insult:
accuracy= 94.06%        accuracy= 96.56%        accuracy= 96.25%
precision = 87.71%      precision = 91.52%      precision = 82.26%
recall = 44.88%         recall = 41.23%         recall = 33.32%
f1 score = 59.38%       f1 score = 56.85%       f1 score = 47.42%
hamming_loss = 5.94%    hamming_loss = 3.44%    hamming_loss = 3.75%

severe_toxic:           threat:                 identity_hate:
accuracy= 99.12%        accuracy= 99.67%        accuracy= 99.03%
precision = 61.90%      precision = nan%        precision = 80.00%
recall = 3.96%          recall = 0.00%          recall = 0.56%
f1 score = 7.44%        f1 score = nan%         f1 score = 1.11%
hamming_loss = 0.88%    hamming_loss = 0.33%    hamming_loss = 0.97%
```

See below confusion matrix for each label to better visualize the results. As one can see, this is not much better, or can be considered worse, than the previous model.

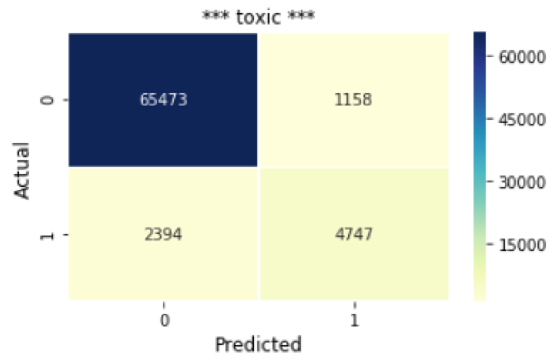


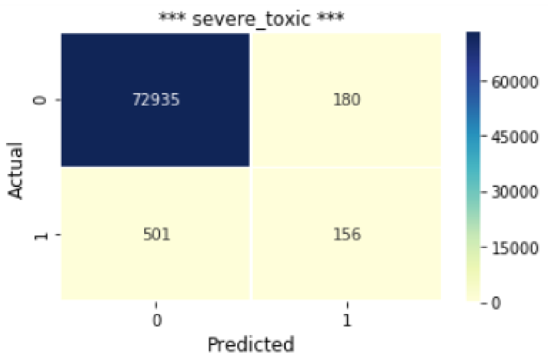
Let's try using other binary techniques and multi-label techniques.

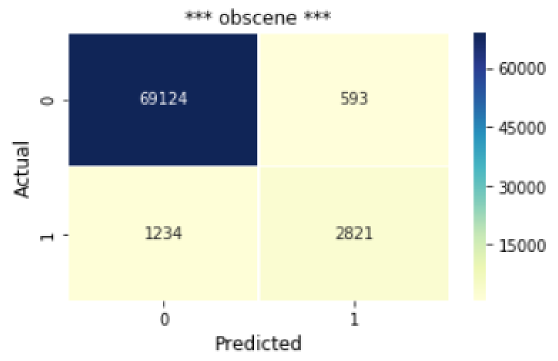
• Linear Support Vector Classifier (LSVC)

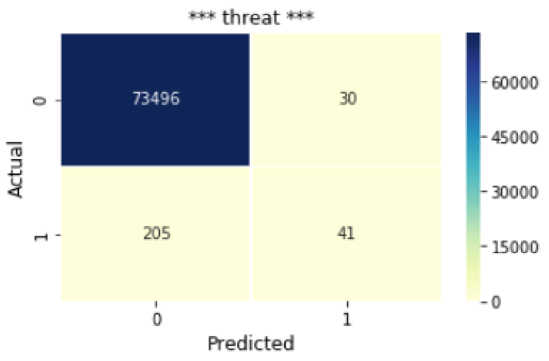
```
lsvc_pipeline = Pipeline([
                ('vect', TfidfVectorizer(min_df=1, ngram_range=(1, 2),stop_words='english')),
                ('clf', OneVsRestClassifier(LinearSVC()))])

lsvc_pipeline.fit(X_train, train[categories].values)

prediction_lsvc = lsvc_pipeline.predict(X_test)
```

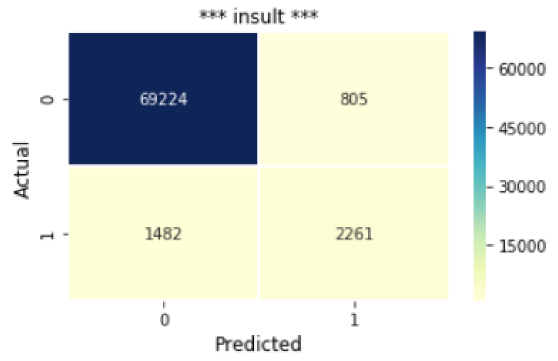
```
For all (6) labels:
          accuracy = 91.22%
          hamming_loss = 2.07%

For individual labels:
toxic:                obscene:              insult:
accuracy= 95.19%      accuracy= 97.52%      accuracy= 96.90%
precision = 80.39%    precision = 82.63%    precision = 73.74%
recall = 66.48%       recall = 69.57%       recall = 60.41%
f1 score = 72.77%     f1 score = 75.54%     f1 score = 66.41%
hamming_loss = 4.81%  hamming_loss = 2.48%  hamming_loss = 3.10%

severe_toxic:         threat:               identity_hate:
accuracy= 99.08%      accuracy= 99.68%      accuracy= 99.20%
precision = 46.43%    precision = 57.75%    precision = 70.57%
recall = 23.74%       recall = 16.67%       recall = 31.10%
f1 score = 31.42%     f1 score = 25.87%     f1 score = 43.18%
hamming_loss = 0.92%  hamming_loss = 0.32%  hamming_loss = 0.80%
```

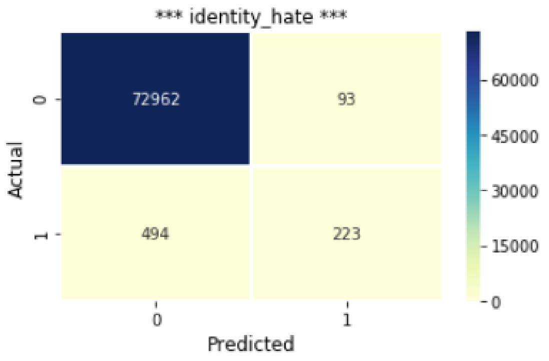










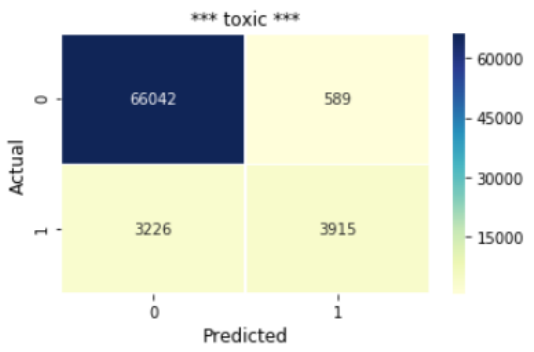


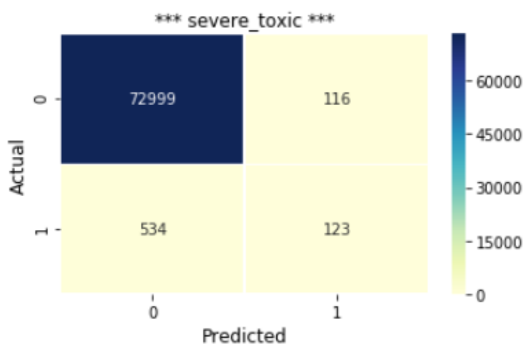
• Logistic Regression (LREG)

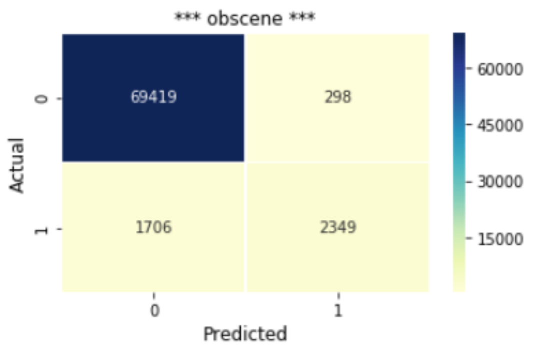
```
lreg_pipeline = Pipeline([
                ('vect', TfidfVectorizer(min_df=10, ngram_range=(1, 2),stop_words='english')),
                ('clf', OneVsRestClassifier(LogisticRegression(solver='sag')))])

lreg_pipeline.fit(X_train, train[categories].values)

prediction_lreg = lreg_pipeline.predict(X_test)
```

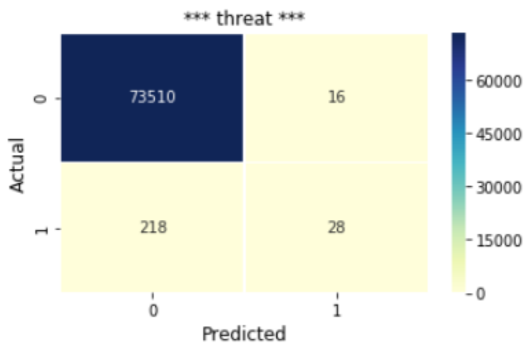
```
For all (6) labels:
        accuracy = 91.24%
        hamming_loss = 2.20%
```

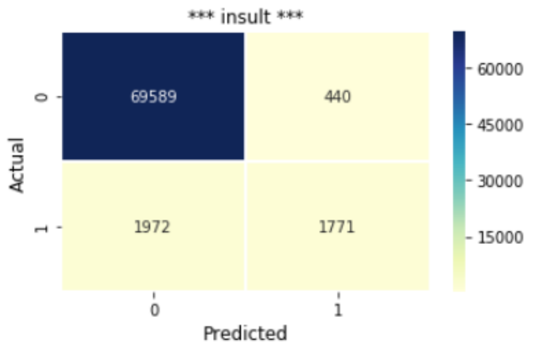
```
For individual labels:
toxic:                  obscene:                insult:
accuracy= 94.83%        accuracy= 97.28%        accuracy= 96.73%
precision = 86.92%      precision = 88.74%      precision = 80.10%
recall = 54.82%         recall = 57.93%         recall = 47.31%
f1 score = 67.24%       f1 score = 70.10%       f1 score = 59.49%
hamming_loss = 5.17%    hamming_loss = 2.72%    hamming_loss = 3.27%

severe_toxic:           threat:                 identity_hate:
accuracy= 99.12%        accuracy= 99.68%        accuracy= 99.16%
precision = 51.46%      precision = 63.64%      precision = 75.00%
recall = 18.72%         recall = 11.38%         recall = 20.92%
f1 score = 27.46%       f1 score = 19.31%       f1 score = 32.72%
hamming_loss = 0.88%    hamming_loss = 0.32%    hamming_loss = 0.84%
```

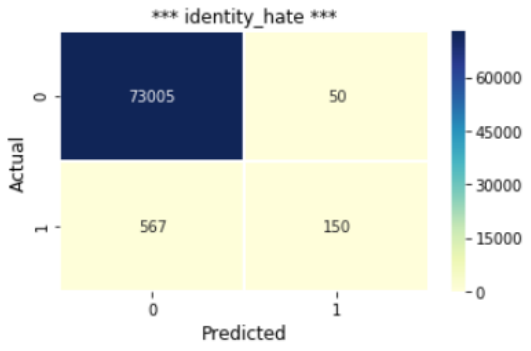






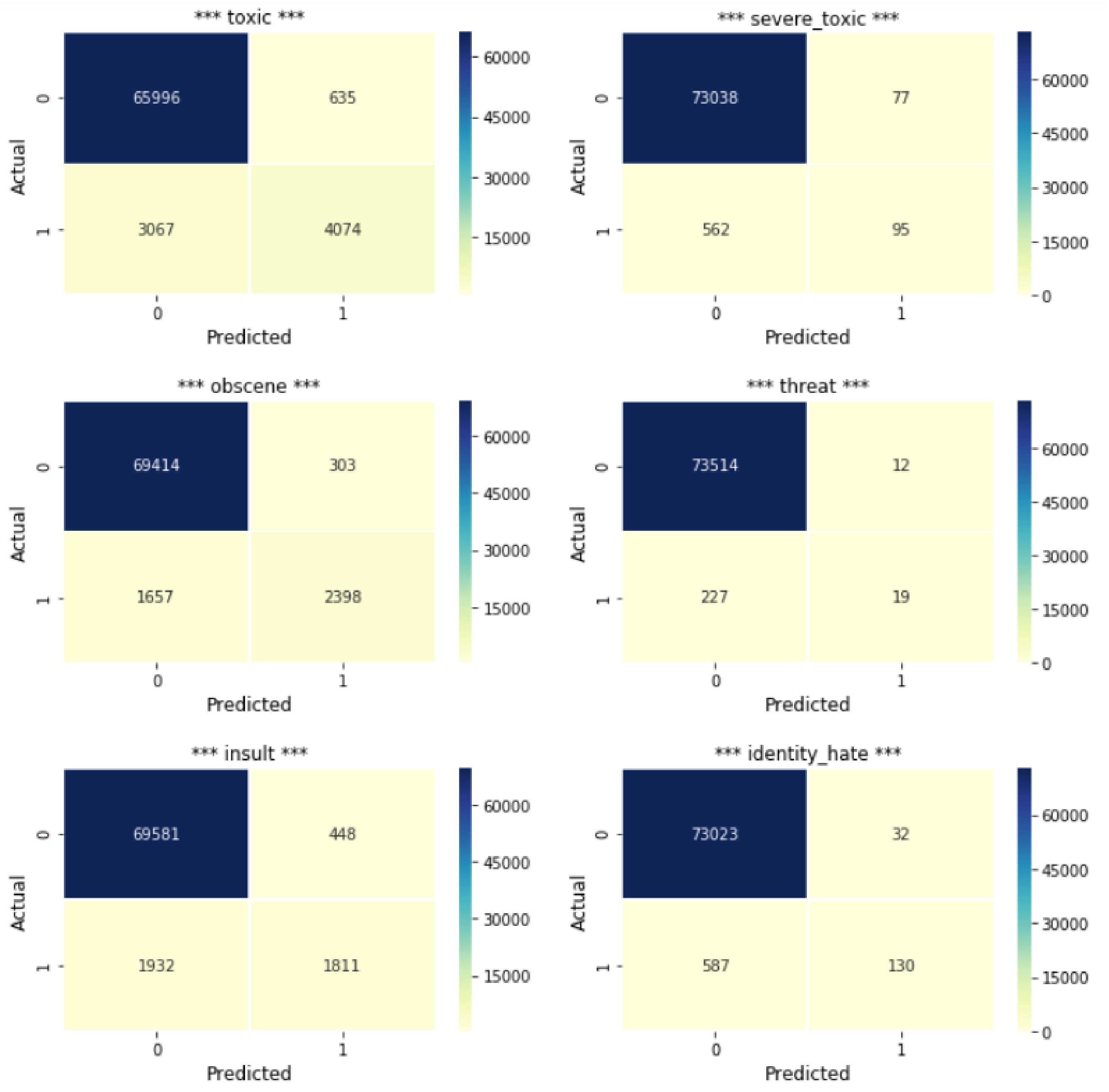






• Ensemble Method (Majority Rule between MNB, LSVC and LREG models)

```
For all (6) labels:
         accuracy = 91.28%
         hamming_loss = 2.16%

For individual labels:
toxic:                 obscene:              insult:
accuracy= 94.98%       accuracy= 97.34%      accuracy= 96.77%
precision = 86.52%     precision = 88.78%    precision = 80.17%
recall = 57.05%        recall = 59.14%       recall = 48.38%
f1 score = 68.76%      f1 score = 70.99%     f1 score = 60.35%
hamming_loss = 5.02%   hamming_loss = 2.66%  hamming_loss = 3.23%

severe_toxic:          threat:               identity_hate:
accuracy= 99.13%       accuracy= 99.68%      accuracy= 99.16%
precision = 55.23%     precision = 61.29%    precision = 80.25%
recall = 14.46%        recall = 7.72%        recall = 18.13%
f1 score = 22.92%      f1 score = 13.72%     f1 score = 29.58%
hamming_loss = 0.87%   hamming_loss = 0.32%  hamming_loss = 0.84%
```



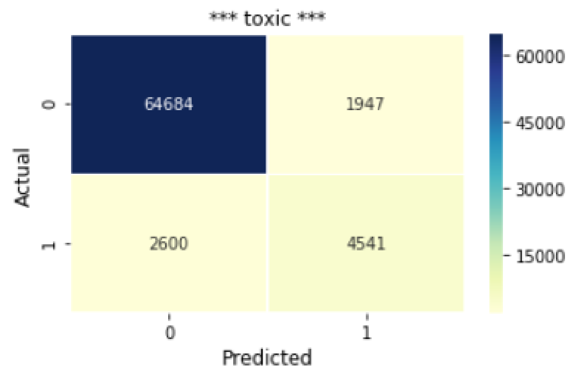
Multi-Label Techniques

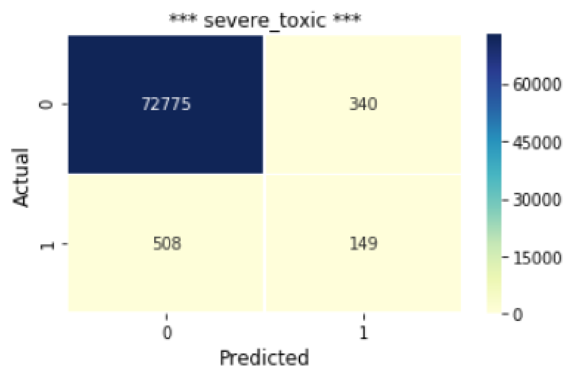
Let’s continue by trying multi-label techniques that consider all the categories and their correlations between categories.

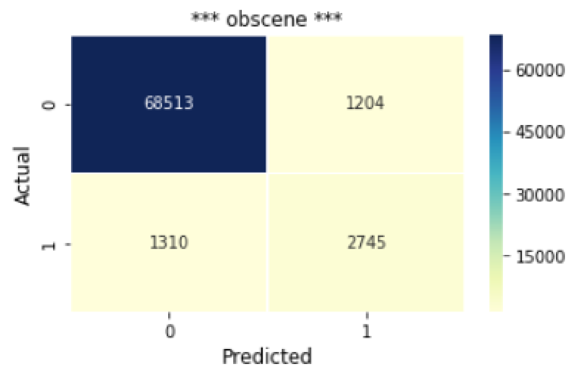
- Classifier Chain (CC) with LSVC

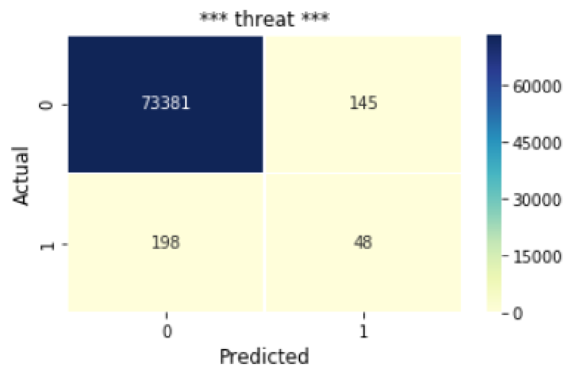
```
cc_pipeline = Pipeline([
                    ('vect', CountVectorizer(min_df=10, ngram_range=(1, 1),stop_words='english')),
                    ('clf', ClassifierChain(LinearSVC()))])

cc_pipeline.fit(X_train, train[categories].values)

prediction_cc = cc_pipeline.predict(X_test)
```

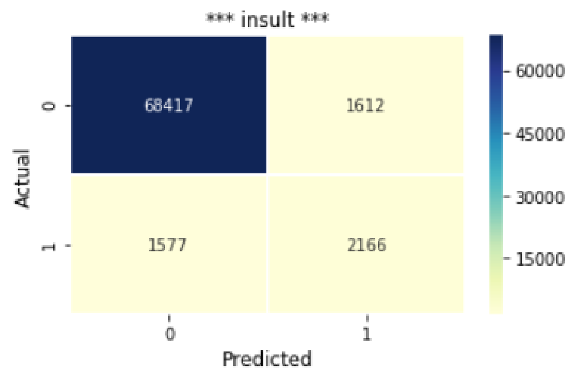
```
For all (6) labels:
         accuracy = 88.84%
         hamming_loss = 2.79%

For individual labels:

toxic:                  obscene:                insult:
accuracy= 93.84%        accuracy= 96.59%        accuracy= 95.68%
precision = 69.99%      precision = 69.51%      precision = 57.33%
recall = 63.59%         recall = 67.69%         recall = 57.87%
f1 score = 66.64%       f1 score = 68.59%       f1 score = 57.60%
hamming_loss = 6.16%    hamming_loss = 3.41%    hamming_loss = 4.32%

severe_toxic:           threat:                 identity_hate:
accuracy= 98.85%        accuracy= 99.54%        accuracy= 98.78%
precision = 30.47%      precision = 24.87%      precision = 34.81%
recall = 22.68%         recall = 19.51%         recall = 29.57%
f1 score = 26.00%       f1 score = 21.87%       f1 score = 31.98%
hamming_loss = 1.15%    hamming_loss = 0.46%    hamming_loss = 1.22%
```

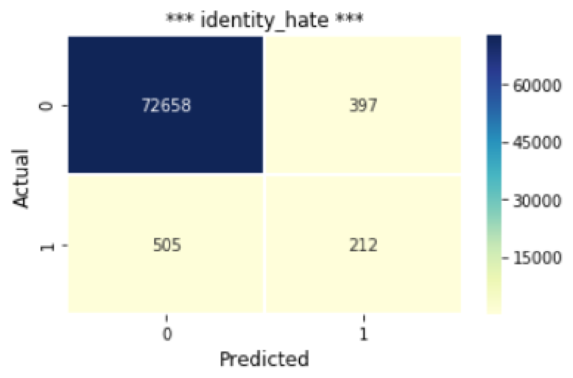








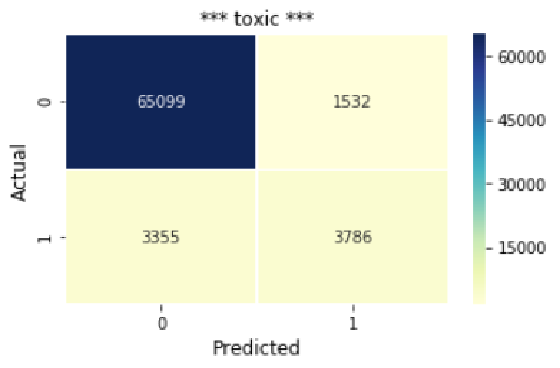


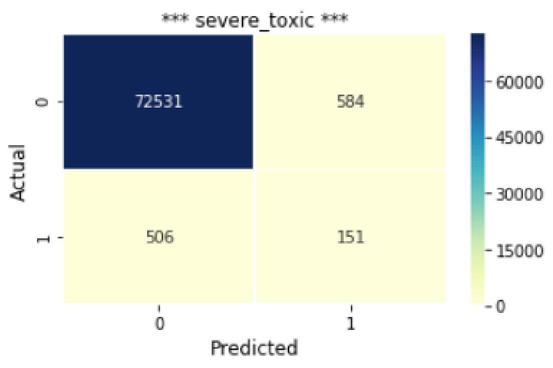


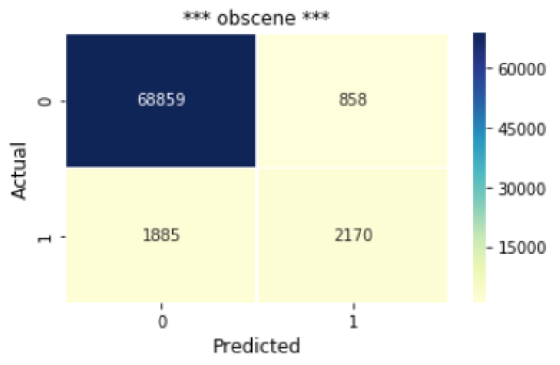
• LabelPowerset (LP) with LSVC

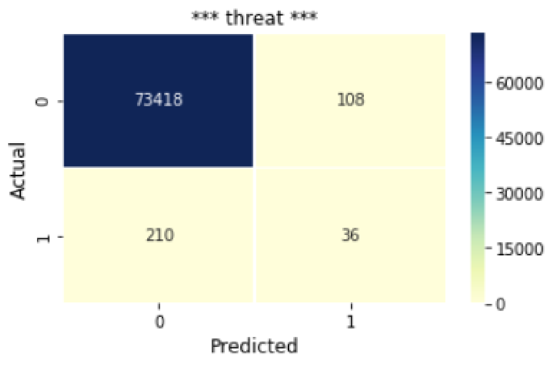
```
lbs_pipeline = Pipeline([
                    ('vect', CountVectorizer(min_df=10, ngram_range=(1, 1),stop_words='english')),
                    ('clf', LabelPowerset(LinearSVC()))])

lbs_pipeline.fit(X_train, train[categories].values)

prediction = lbs_pipeline.predict(X_test)
```

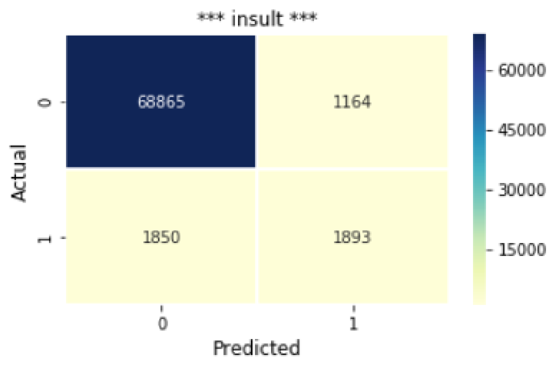
```
For all (6) labels:
        accuracy = 89.52%
        hamming_loss = 2.95%
For individual labels:

toxic:                     obscene:                   insult:
accuracy= 93.38%           accuracy= 96.28%           accuracy= 95.91%
precision = 71.19%         precision = 71.66%         precision = 61.92%
recall = 53.02%            recall = 53.51%            recall = 50.57%
f1 score = 60.78%          f1 score = 61.27%          f1 score = 55.68%
hamming_loss = 6.62%       hamming_loss = 3.72%       hamming_loss = 4.09%

severe_toxic:              threat:                    identity_hate:
accuracy= 98.52%           accuracy= 99.57%           accuracy= 98.66%
precision = 20.54%         precision = 25.00%         precision = 28.21%
recall = 22.98%            recall = 14.63%            recall = 24.83%
f1 score = 21.70%          f1 score = 18.46%          f1 score = 26.41%
hamming_loss = 1.48%       hamming_loss = 0.43%       hamming_loss = 1.34%
```

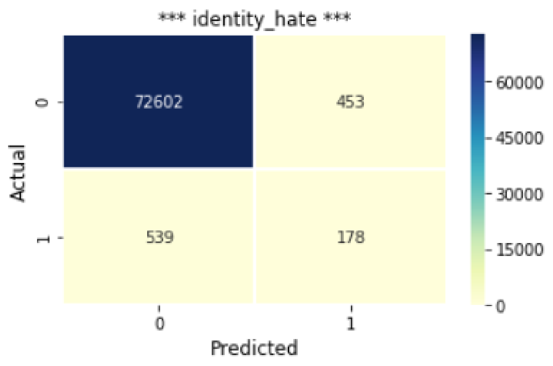




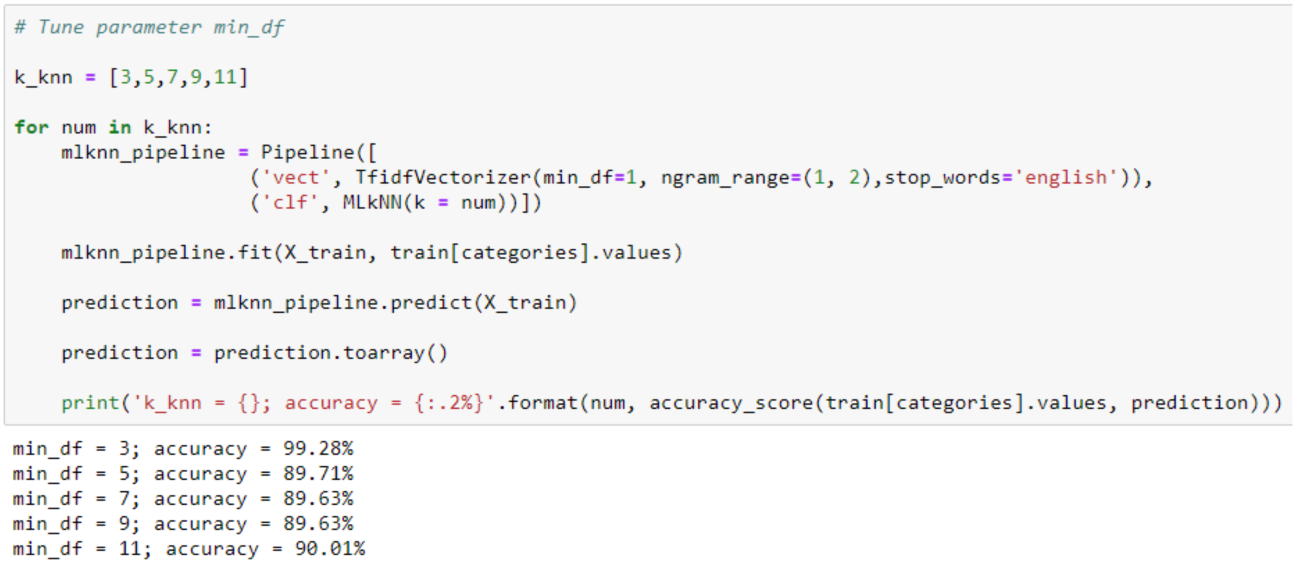






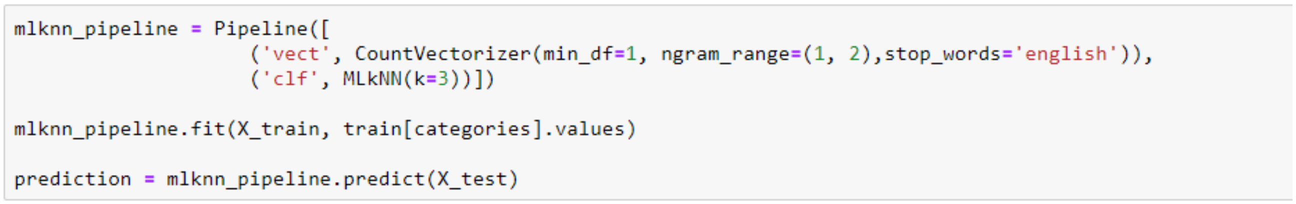


• Multi-Label k-Nearest Neighbor (MLkNN)

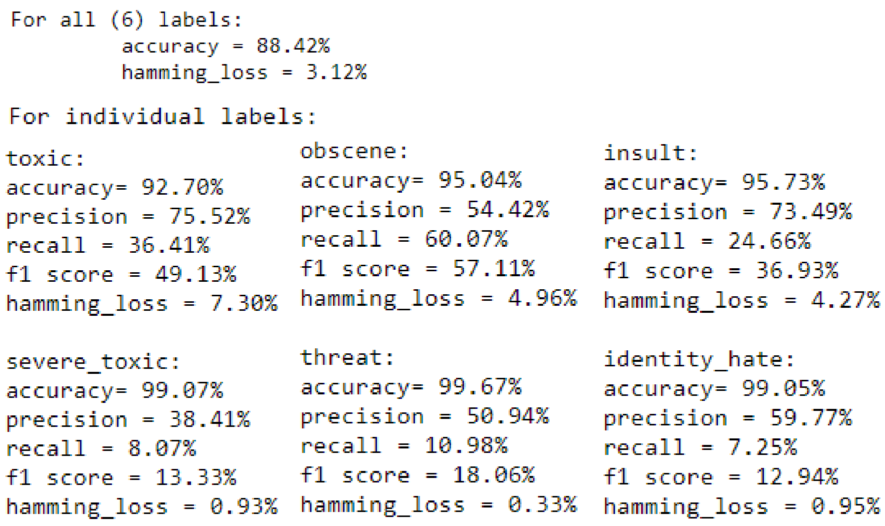


```
# Tune parameter min_df

k_knn = [3,5,7,9,11]

for num in k_knn:
    mlknn_pipeline = Pipeline([
                    ('vect', TfidfVectorizer(min_df=1, ngram_range=(1, 2),stop_words='english')),
                    ('clf', MLkNN(k = num))])

    mlknn_pipeline.fit(X_train, train[categories].values)

    prediction = mlknn_pipeline.predict(X_train)

    prediction = prediction.toarray()

    print('k_knn = {}; accuracy = {:.2%}'.format(num, accuracy_score(train[categories].values, prediction)))
```

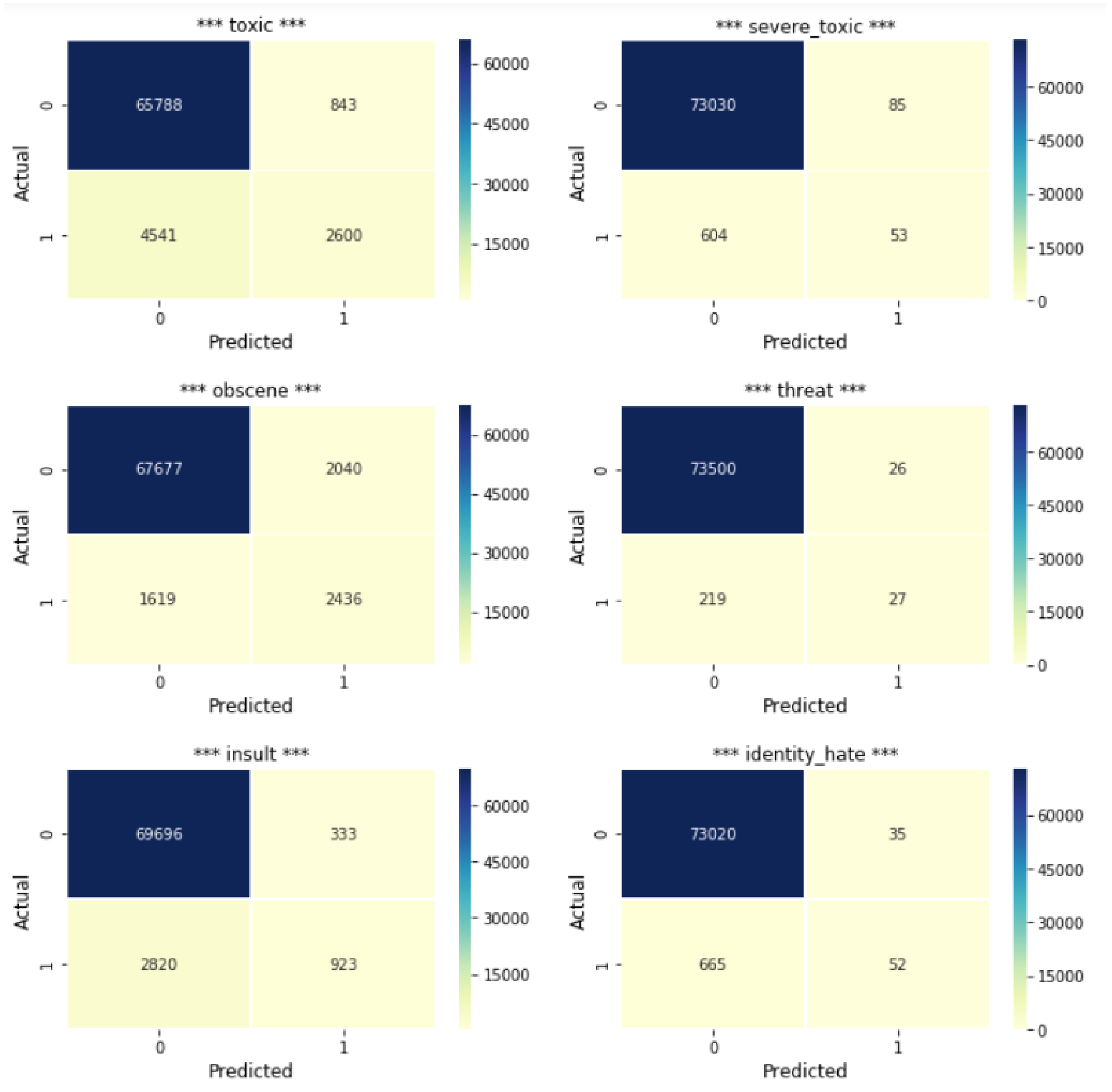
```
min_df = 3; accuracy = 99.28%
min_df = 5; accuracy = 89.71%
min_df = 7; accuracy = 89.63%
min_df = 9; accuracy = 89.63%
min_df = 11; accuracy = 90.01%
```



```
mlknn_pipeline = Pipeline([
                    ('vect', CountVectorizer(min_df=1, ngram_range=(1, 2),stop_words='english')),
                    ('clf', MLkNN(k=3))])

mlknn_pipeline.fit(X_train, train[categories].values)

prediction = mlknn_pipeline.predict(X_test)
```



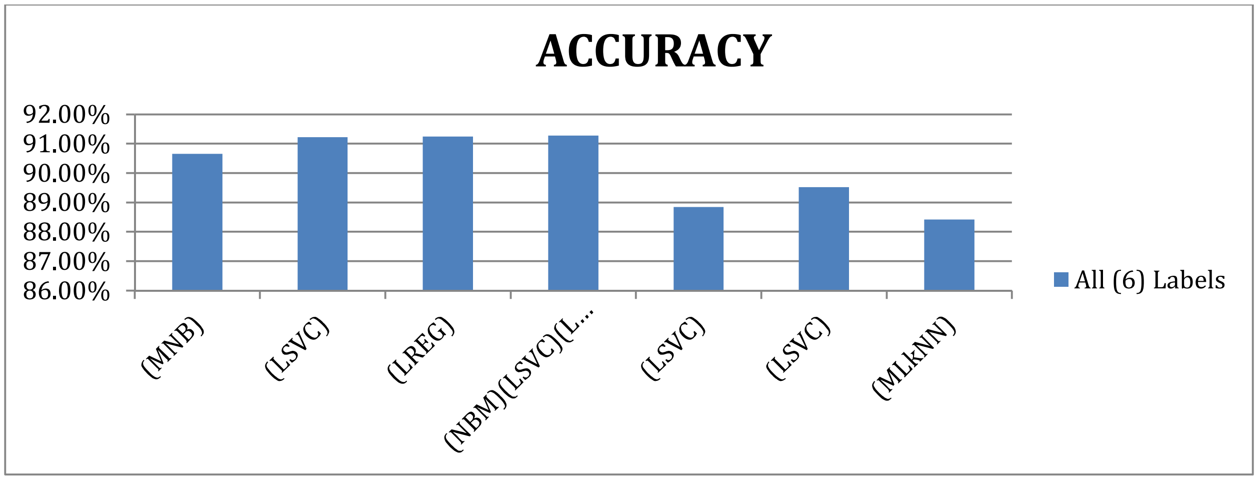
```
For all (6) labels:
         accuracy = 88.42%
         hamming_loss = 3.12%

For individual labels:
toxic:                    obscene:                  insult:
accuracy= 92.70%          accuracy= 95.04%          accuracy= 95.73%
precision = 75.52%        precision = 54.42%        precision = 73.49%
recall = 36.41%           recall = 60.07%           recall = 24.66%
f1 score = 49.13%         f1 score = 57.11%         f1 score = 36.93%
hamming_loss = 7.30%      hamming_loss = 4.96%      hamming_loss = 4.27%

severe_toxic:             threat:                   identity_hate:
accuracy= 99.07%          accuracy= 99.67%          accuracy= 99.05%
precision = 38.41%        precision = 50.94%        precision = 59.77%
recall = 8.07%            recall = 10.98%           recall = 7.25%
f1 score = 13.33%         f1 score = 18.06%         f1 score = 12.94%
hamming_loss = 0.93%      hamming_loss = 0.33%      hamming_loss = 0.95%
```

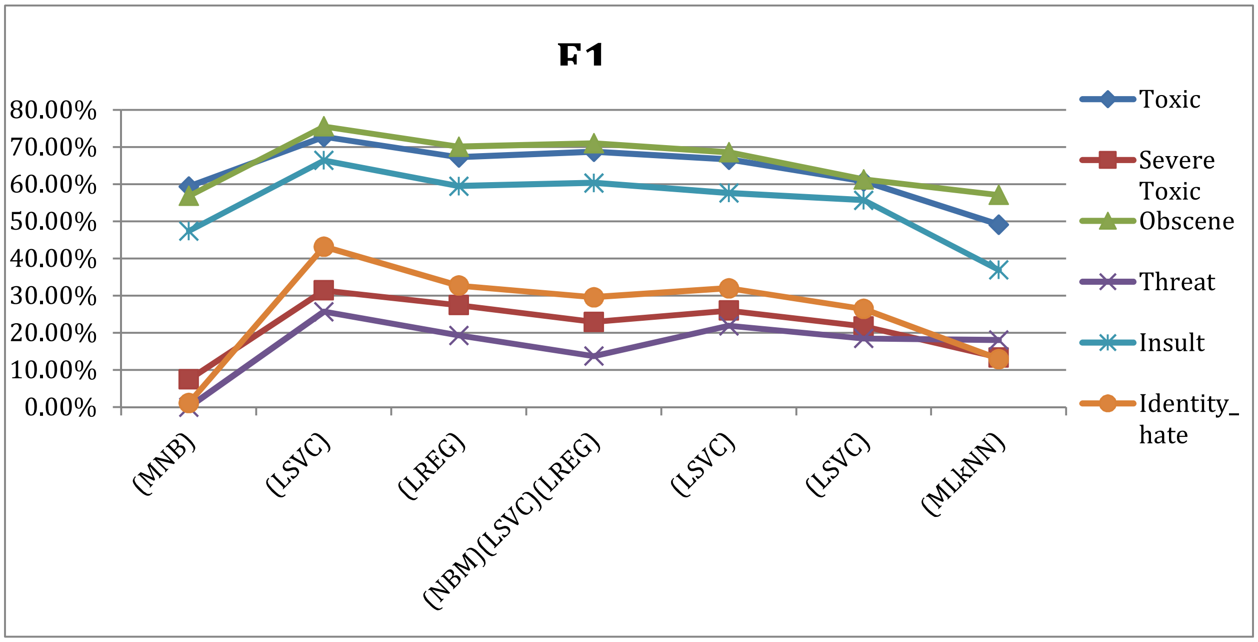

*** toxic ***
Actual
Predicted
65788
843
4541
2600
*** severe_toxic ***
73030
85
604
53
*** obscene ***
67677
2040
1619
2436
*** threat ***
73500
26
219
27
*** insult ***
69696
333
2820
923
*** identity_hate ***
73020
35
665
52
60000
45000
30000
15000
0

5. RESULT SUMMARY

Below table shows the results between the 6 models and 1 ensemble method.

			Multinomial	Linear	Logistic	Ensemble	Classifier	Label	
			Naive Bayes	SVC	Regression	Majority Rule	Chain	Powerset	MLkNN
			(MNB)	(LSVC)	(LREG)	(NBM)(LSVC)(LREG)	(LSVC)	(LSVC)	(MLkNN)
All (6) Labels									
		Accuracy	90.65%	91.22%	91.24%	91.28%	88.84%	89.52%	88.42%
		Precision							
		Recall							
		F1 Score							
		Hamming Loss	2.55%	2.07%	2.20%	2.16%	2.79%	2.95%	3.12%
Individual Labels									
	Toxic	Accuracy	94.00%	95.19%	94.83%	94.98%	93.84%	93.38%	92.70%
		Precision	87.71%	80.39%	86.92%	86.52%	69.99%	71.19%	75.52%
		Recall	44.88%	66.48%	54.82%	57.05%	63.59%	53.02%	36.41%
		F1 Score	59.38%	72.77%	67.24%	68.76%	66.64%	60.78%	49.13%
		Hamming Loss	59.40%	4.81%	5.17%	5.02%	6.16%	6.62%	7.30%
	Severe Toxic	Accuracy	99.12%	99.08%	99.12%	99.13%	98.85%	98.52%	99.07%
		Precision	61.90%	46.43%	51.46%	55.23%	30.47%	20.54%	38.41%
		Recall	3.96%	23.74%	18.72%	14.46%	22.68%	22.98%	8.07%
		F1 Score	7.44%	31.42%	27.46%	22.92%	26.00%	21.70%	13.33%
		Hamming Loss	88.00%	0.92%	0.88%	0.87%	1.15%	1.48%	0.93%
	Obscene	Accuracy	96.56%	97.52%	97.28%	97.34%	96.59%	96.28%	95.04%
		Precision	91.52%	82.63%	88.74%	88.78%	69.51%	71.66%	54.42%
		Recall	41.23%	69.57%	57.93%	59.14%	67.69%	53.51%	60.07%
		F1 Score	56.85%	75.54%	70.10%	70.99%	68.59%	61.27%	57.11%
		Hamming Loss	3.44%	2.48%	2.72%	2.66%	3.41%	3.72%	4.96%
	Threat	Accuracy	99.67%	99.68%	99.68%	99.68%	99.54%	99.57%	99.67%
		Precision	0.00%	57.75%	63.64%	61.29%	24.87%	25.00%	50.94%
		Recall	0.00%	16.67%	11.38%	7.72%	19.51%	14.63%	10.98%
		F1 Score	0.00%	25.70%	19.31%	13.72%	21.87%	18.46%	18.06%
		Hamming Loss	0.33%	32.00%	0.32%	0.32%	0.46%	0.43%	0.33%
	Insult	Accuracy	96.25%	96.90%	96.73%	96.77%	95.68%	95.91%	95.73%
		Precision	82.26%	73.74%	80.10%	80.17%	57.33%	61.92%	73.49%
		Recall	33.32%	60.41%	47.31%	48.38%	57.87%	50.57%	24.66%
		F1 Score	47.42%	66.41%	59.49%	60.35%	57.60%	55.68%	36.93%
		Hamming Loss	3.75%	3.10%	3.27%	3.23%	4.32%	4.09%	4.27%
	Identity_hate	Accuracy	99.03%	99.20%	99.16%	99.16%	98.78%	98.66%	99.05%
		Precision	80.00%	70.57%	75.00%	80.25%	34.81%	28.21%	59.77%
		Recall	0.56%	31.10%	20.92%	18.13%	29.57%	24.83%	7.25%
		F1 Score	1.11%	43.18%	32.72%	29.58%	31.98%	26.41%	12.94%
		Hamming Loss	0.97%	0.80%	0.84%	0.84%	1.22%	1.34%	0.95%




F1
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
(MNB)
(LSVC)
(LREG)
(NBM)(LSVC)(LREG)
(LSVC)
(LSVC)
(MLkNN)
Toxic
Severe Toxic
Obscene
Threat
Insult
Identity_hate

6. DISCUSSION AND CONCLUSION

Discussion and Observations

• As shown in Section 6, in general the binary techniques perform better than the multi-label techniques. The best technique to predict all (6) labels together is the ensemble method using majority rule between the MNB, LSVC and LREG. The accuracy is 91.28%, better than the baseline where one guesses all clean comments (all 0s). The corresponding hamming loss is also second lowest.

• If individual labels are considered, the Linear SVC (LSVC) model performs the best out of all the models. Comparing the f1-scores, which is a good indicator of the performance in this case, the LSVC model has the highest percentages. The categories of “toxic”, “obscene” and “insult” have f1-scores ranging from 66% to 75%. The results are better than the baseline.

• Furthermore, it is apparent that the models provide better predictions for categories of “toxic”, “obscene” and “insult” than “severe_toxic”, “obscene” and “identity_hate”. This is mainly due to the lack of data for the categories – the number of comments for each of these categories is only 1% of the total number of comments.

Other aspects that can be investigated to improve the models

• Using other techniques (e.g neural network, etc.)

• Cross-validation

• More refined hyper-parameter tuning

• Use stemming and lemmatization

• More balanced dataset (e.g. bootstrap the toxic comments to increase the number of toxic comments and make it less unbalanced, collect more toxic comments to improve the dataset, etc.)

• Better comment labeling and eliminate the comments that are in the “gray zone”. These comments might have skewed the data features and the results.

For example, one would argue whether the below marked comments are toxic comments as they are only expressing their views, quite respectfully.

> *“So threatening to try to have me banned for editing is fine, but when I suggest having your account deleted for constantly deleting entire paragraphs, completely ignoring Wikipedia's policies with regards to that, instead of tagging or editing, suddenly it's bullying? Nonsense and you know it, you're just trying to make me look bad by resorting to the ""you're a bully"" nonsense. You have also not answered my question on which part is not neutral, despite my statement that I would edit it - you just deleted it again. Get over yourself. You do not own Wikipedia, the tags you've placed on your user page don't change that fact. You delete entire paragraphs for the sake of one word? I have occasionally made the mistake of putting a subjective word in an edit before, and someone has noticed - you know what they did? THEY EDITED IT! You have just admitted in writing that instead of editing paragraphs that you believe are not neutral, or marking them as such, you are deleting entire paragraphs against the policy of Wikipedia.”*

Or

“'barek my post on barek talk page was not an attack it was a virtual slap upside the head do not threaten me with your empty promises i am not afraid of you i am a bagel baker '”

Or simply...

“ please read provided source instead of talking crap “

Conclusion

• For predicting all (6) labels together, the ensemble method using majority rule between the MNB, LSVC and LREG provided decent results, slightly better than guessing all clean comments. (OK)

• For individual category, the Linear SVC technique performed well for predicting whether a comment is "toxic" and/or "obscene". It performed decently for predicting comments in the "insult" category. (GOOD)

• Due to the lack of data in the "severe_toxic", "threat" and "identity_hate" categories, the models did not perform well on these categories. (NOT GOOD)

“You can block and block and suck my cock but nothing can stop freedom of speech. In ***the end****, America wins. We always do. Ask Saddam. My sockpuppet army is on the march. 02:13, 30 Mar 2005 (UTC)”*