BASELINE TOXIC LANGUAGE

CHANGING THE WORLD ONE MODEL AT A TIME




Toxic Language Plumbers

Joseph Lee Yisang Yoon Paul Durkin Walt Burge

W207 Applied Machine Learning April 2018

TODAY'S AGENDA

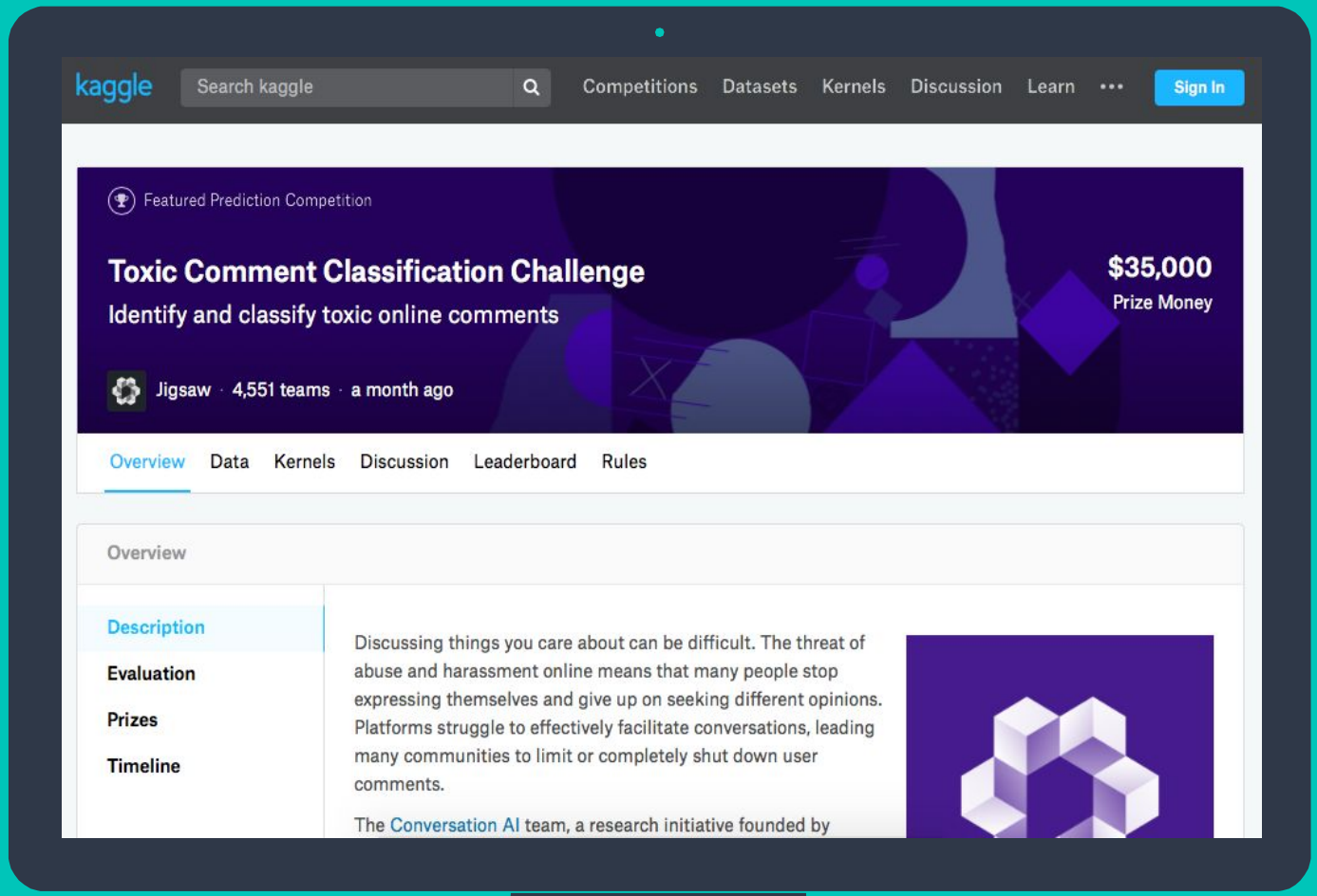
introduction

data science methodology

major takeaways

18+

The dataset for this competition contains text that may be considered profane, vulgar, or offensive.



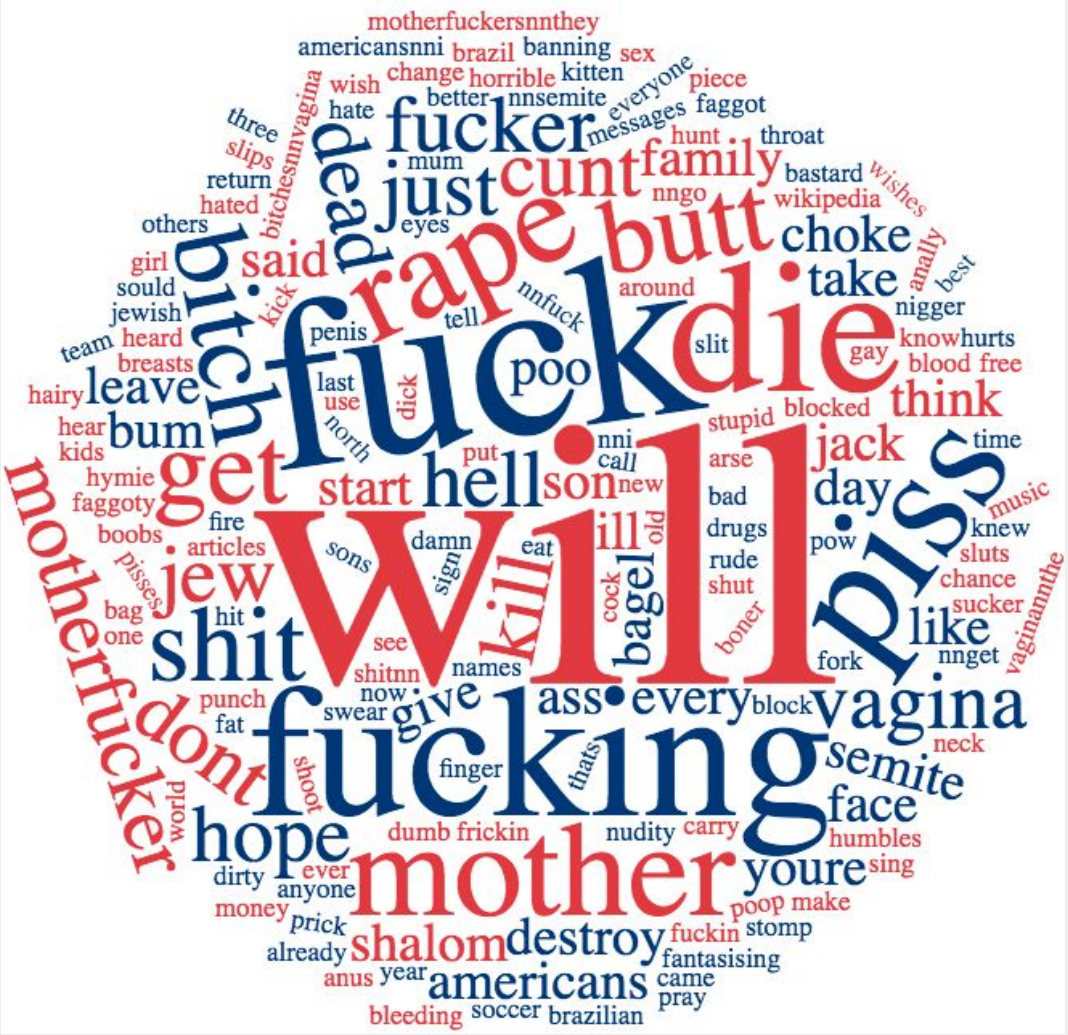
Looking for threats, obscenity, insults, identity-based hate

Determining level of toxicity

CONTEXT & MOTIVATION

- Words are powerful medium of communication
- More so in the IoT context
- Predicting/identifying factors for toxic language can provide important insights
- Toxic language plumbers will test ML applications to gain insight to this problem and to see if an ML solution is feasible

TEXT DATA AT A GLANCE



1.

EXPLORATORY ANALYSIS

95,850 entries

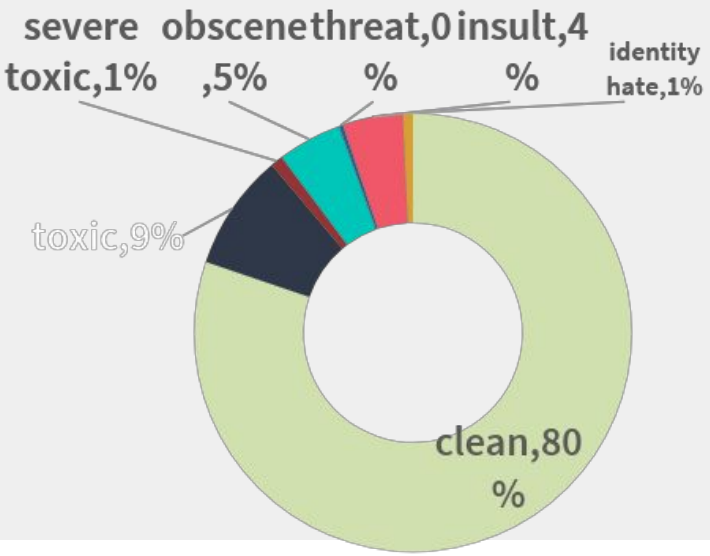
Wikipedia comments labeled by humans

6 classes of toxicity

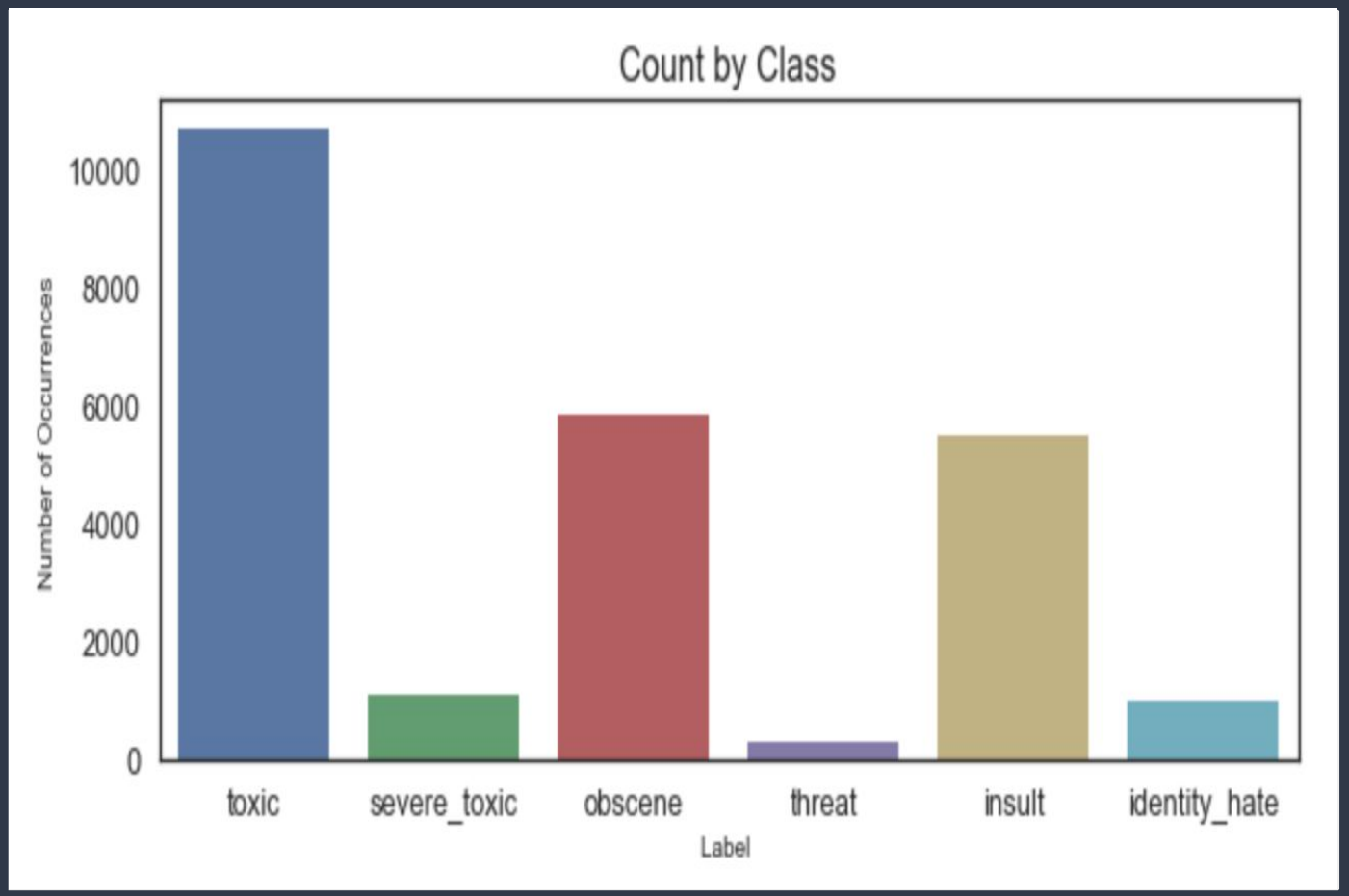
Toxic, severe_toxic, obscene, threat, insult, identity_hate

20%

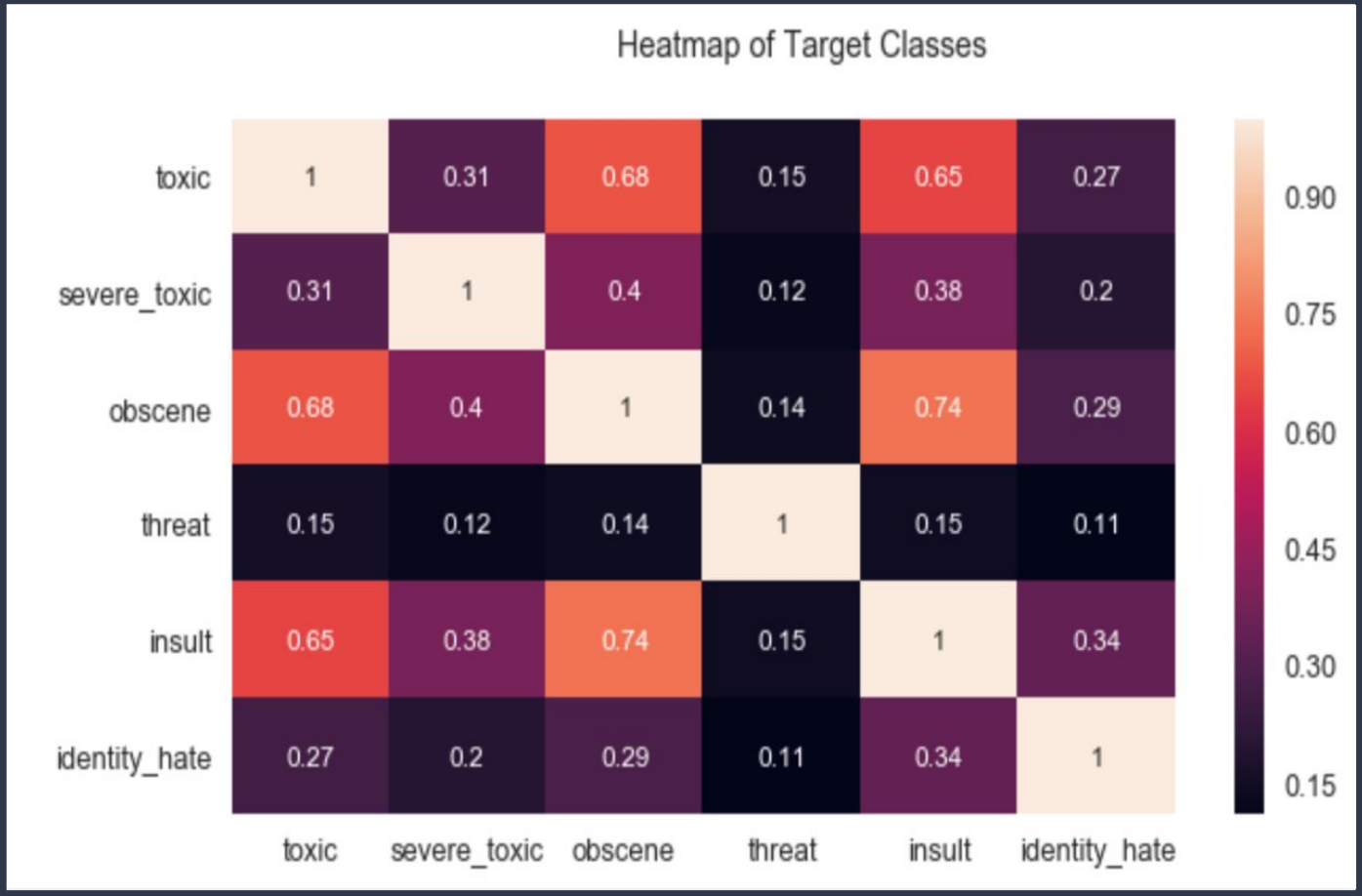
of the comments were toxic



INTERESTING EDA



INTERESTING EDA



CROSS-TAB ANALYSIS

	toxic		severe_toxic		obscene		threat		insult		identity_hate	
toxic	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
toxic												
0	101218	0	101218	0	100865	353	101198	20	100871	347	101142	76
1	0	10688	9573	1115	5159	5529	10385	303	5545	5143	9774	914

Another way of analyzing categorical data is to apply cross-tab analysis, basically assessing correlation through confusion matrices.

In this example, we are looking and comparing the overall “Toxic” label against itself and other target classes.

The major takeaway is that clearly some classes are subsets of another.

1.5

ESTABLISHING ASSUMPTIONS & METRICS

ASSUMPTIONS AND DEFINING THE MODELING PROBLEM

- It is extremely important to clarify whether this is a **multi-label** or **multi-class** problem.

- Given the target labels for this challenge, text observation can be any multiple labels (toxic, severe toxic, threat, insult, etc) at the same time or none at all and is there for a **multi-label problem.**

- Our team will not submit out final prediction to the kaggle leaderboard. Instead, our team will create a new **70/30 train/test** split that all models will be required to use for training and testing. Different notebooks may create internal dev sets, however, the ultimate AUC score is determined by the established test set.

ESTABLISHING METRICS FOR CASE STUDY

- The **primary metric** to evaluate multiple modeling approaches will be the ROC derived **AUC** Score.
  - The Toxic Language Kaggle Competition uses AUC as leaderboard metric
  - Useful metric for evaluating model performance based on the TPR over the FPR.
- Our team will also evaluate **two other secondary metrics** to supplement our analysis.
  - AUC is a good metric but is not perfect. From an AUC score it is difficult to see how well the model deals with False Positive predictions. While it does a better job than accuracy when conveying performance for imbalanced targets, we will consider some additional metrics to gain insight when False Positives and False Negatives come into account.
  - **Precision**
    - (TP)/(TP+FP)
    - Tells us the proportion of text predicted as some kind of toxic label is actually toxic.
  - **Recall**
    - (TP)/(TP+FN)
    - Tells us the proportion of text that were actually toxic had been correctly predicted.

2.

MODELING

Model Assumptions

- All models will use the same new_train.csv and new_test.csv for model training and validation.
- All models will use TF-IDF as main vectorization method
- All models will use the basic pre-processing steps (lemmatization, translation of words, removal of non-ascii)
- Due to different model assumptions, certain models may not require the same preprocessing steps.
- No Explicit Feature Engineering, mostly text processing, selection, and vectorization

MODELS

K-Nearest Neighbors

Naive Bayes

Tree Based

Neural Networks

LOGISTIC REGRESSION (BENCHMARK MODEL)

Performance to Beat

- TFIDF + GLM (Logistic)
- Training Score : 0.97
- Testing Score: 0.57
- OLS Based
  - no regularization applied

Not worth exploring...

Regression is a common industry standard model for interpretable modeling. We will use this model as global benchmark to assess additional modeling and preprocess steps.

Improvements should focus on adding regularization and grid searching for best parameters like alpha.

K-NEAREST NEIGHBORS

Too many neighbors

- Predicting takes too long since distance need to be measured for each of the entries

Not worth exploring further…

TREE-BASED APPROACH (XGBoost)

Best Model Summary

- max_depth: 6
- nrounds : 500
- Learning rate: 0.01
- min_sample : 0.8
- col_sample : 0.6

XGBoost

- Advanced package for tree-boosting and gradient optimization.
- Uses entropy/gain for splits

Average AUC scores

Train: ~0.92

Test: ~0.50

Not worth exploring further...

Scores indicate extreme overfitting or the entropy of each vectorized feature is not adequate for splitting.

Improvements

- Expanding word dictionary and clean and hand process text data.
- Experiment with bagging methods instead of additive boosting

NAIVE BAYES

Best type and parameters

- Gaussian was ignored as it is unsuitable for sparse data
- Bernoulli with a CountVectorizer worked best for both recall and AUC
- Multinomial with Tf-IDF was best for precision getting 100% on 4 of the labels

Interesting Observations

- Feature reduction was key
- Preprocessing was important
- Both precision and recall had a variety of parameters for the best classifiers, roc_auc had one set that worked for all
- Train AUC: ~ 0.90
- Test AUC: ~ 0.89

Improvements

- More preprocessing of the data

NAIVE BAYES RESULTS (Detailed)

label	model	alpha	type	preprocessor	tokenizer	max_features	stop_words	lowercase	strip_accents	score_type	score
toxic	multi	10	tfidf	0	0	None	None	TRUE	None	precision	1
severe_toxic	multi	10	tfidf	0	0	6000	english	FALSE	None	precision	0.8
obscene	multi	2	tfidf	0	0	None	None	TRUE	None	precision	1
threat	multi	0.5	tfidf	0	0	None	None	FALSE	None	precision	1
insult	multi	2	tfidf	0	0	None	None	FALSE	None	precision	1
identity_hate	multi	2	tfidf	0	0	4000	english	TRUE	None	precision	0.882353
Average											**0.947058833**
toxic	bern	1	count	0	0	10000	english	TRUE	unicode	recall	0.886398
severe_toxic	bern	0.5	count	0	0	10000	english	TRUE	None	recall	0.953782
obscene	bern	1	count	1	0	10000	english	FALSE	None	recall	0.897975
threat	bern	0.5	count	0	0	4000	None	TRUE	ascii	recall	0.868421
insult	bern	1	count	0	0	10000	english	TRUE	unicode	recall	0.885738
identity_hate	bern	0.5	count	1	0	4000	None	FALSE	None	recall	0.88785
Average											**0.896694**
toxic	bern	15	count	1	0	4000	english	FALSE	None	roc_auc	0.858937
severe_toxic	bern	2	count	1	0	4000	english	FALSE	None	roc_auc	0.93949
obscene	bern	10	count	1	0	4000	english	FALSE	None	roc_auc	0.888569
threat	bern	0.5	count	1	0	4000	None	FALSE	None	roc_auc	0.899946
insult	bern	10	count	1	0	4000	english	FALSE	None	roc_auc	0.868953
identity_hate	bern	1	count	1	0	4000	english	FALSE	None	roc_auc	0.886601
Average											**0.890416**

NEURAL NETWORKS

Best Model Summary

- hidden Layers (12,6)
- solver: adam
- activation: relu
- tolerance: 1e-13
- alpha : 1
- learning rate: adaptive

Advantage

Neural Nets support many free parameters for learning complex data (e.g. text)

Average AUC scores

Train: (need to re-test)

Test: 96%

Worth exploring further

Tuning preprocessing, testing with large (deep vs wide) hidden layers, Convolutional Neural Networks.

NEURAL NETWORKS TEXT CLASSIFICATION PIPELINE

Preprocessing

(150000+ features -> 6000 subset features)

Options:

- NLTK preprocessing (tokenize and lemmatize) + TfidfVectorizer
- TfidfVectorizer: tokenization + vectorization

Dimensionality Reduction

(6000 features -> 3000 component features)

- Latent Semantic Analysis (LSA): TruncatedSVD

Classification

- MLPClassifier (3000:12:6:6) 3000 input features through 2 hidden layers to 6 output labels

3.

MAJOR TAKEAWAYS

COMPARISON OF TOP 3 MODELS (Average across labels)

Test Scores	Average AUC	Average Precision	Average Recall
Logistic Regression (Benchmark, TFIDF)	~0.57	~0.60	~0.55
Neural Network Pipeline (NLTK + TFIDF)	~0.96	~0.43	~0.25
Naive Bayes Pipeline (TFIDF + Bernoulli)	~0.89	~0.94	~0.90
XGBoost Pipeline (TFIDF)	~0.50	~0.43	~0.55

ASSESSING TOP 3 MODELS BY TARGET CLASS

Naive Bayes

Best Target Label:

- **severe_toxic**
  - AUC: 0.93

Worst Target label:

- **toxic**
  - AUC: 0.83

Neural Networks

Best Target Label:

- **severe_toxic**
  - AUC: 0.99

Worst Target label:

- **threat**
  - AUC: 0.94

XGBoost

- Equally bad across the target classes (~0.50 AUC)

MAJOR TAKEAWAYS

- **Naive Bayes** overall does a good job mitigating False Negative and False Positives as indicated by the high AUC and similarity decent Precision and recall scores.
    - However, NB seems to have a tendency to generate more False Positives for "Severe Toxic". Since "Severe Toxic" is likely a subgroup of "Toxic", these two fields are likely highly correlative and not independent from each other.
- **Neural Networks** initially demonstrated high AUC scores (93% on original train data).
    - This ignores the fact that several of the labels were not correctly classified at all, which is almost certainly due to considerable imbalance in the training dataset.
    - Additionally, the complexity and scale of text data may be best handled with deep and/or wide Neural Nets, which we're not currently able to exercise.
- **KNN** or any distance based model would likely suffer due to the high dimensionality.
- **XGBoost** seems to perform well with text that it has seen but generalizes poorly. Trees seem valid to run, but may require further analysis and expansion of the training dictionary. Given the average 50% auc performance across test labels, there may be an issue with the creating splits on vectorized features.
- **Classifying human language** is a complex and multi-faceted problem.

POSSIBLE AREAS OF IMPROVEMENTS

To-Dos for completing the project

- Standardize training and preprocessed data across all models and report scores (e.g. LSA).
- Clean and finalize code
- Demonstrate NN Overfitting
- Demonstrate XGBoost overfitting
- Finalize computational performances of each model (will run every model through the same architecture)

Other possible Improvements

- Simple Ensemble of Top 2 models (basic)
- CNN Modeling
- Create a class structure for easy model importing
- Explore and test out hypothesis to help models differentiate between similar classes (toxic and severe_toxic)

Thoughts beyond the project

- Stacking/Blending
- Expanding Word Dictionary
- Hand-based text preprocessing (removing any expletives that correlate with multiple targets)

Thanks!

Any questions?

- Ideas for additional gains?
- Thoughts on model process?
- Potential gaps that may be missing in our process?