# Moderation System for Hate Speech Detection

# AGENDA

- Problem
- Proposed Solution
- Literature Survey
- More about Dataset
- Implementation
- Demo
- Impact
- Future work

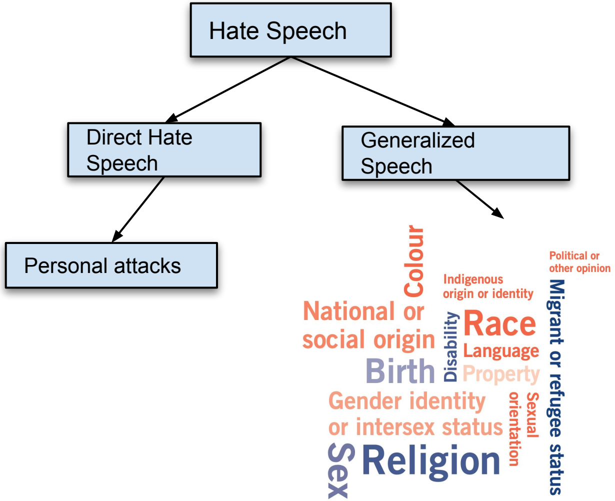
# What is Online Hate Speech?

- Hate speech is a speech that attacks a person or group on the basis of attributes such as race, religion, ethnic origin, national origin, gender, disability, sexual orientation.


Intense and irrational emotion of opprobrium, enmity and detestation towards an individual or group.

Any expression of hate towards an individual or group defined by a protected characteristic.

Any expression imparting opinions or ideas – bringing an internal opinion or idea to an external audience. It can take many forms: written, non-verbal, visual, artistic, etc, and may be disseminated through any media, including internet, print, radio, or television.



# Motivation

- With increasing anonymity and flexibility provided by the Internet, it has made it easy for users to communicate in an aggressive manner

- Hate speech on social media could also lead to harassment, bullying, depression

Fact : The most common type of online bullying is **mean comments 22.5%**.

**Cyberbullying on social media is considered a much bigger threat than in-person bullying**


**Can happen around the clock, 24/7**


**Tends to be more permanent**

**Difficult to pinpoint as typically not in places easily seen**

# Problem Statement

With increase in amount of aggressive content, methods that **Automatically detect hate speech** are very much required

Education on media ethics and awareness about the impact of hate speech could contribute reducing the hate content on social media

# Proposed solution

To develop a ‘Moderation System for Hate Speech Detection’ which can be embedded in the post section of any social media platform

The model alerts users on **Hate Speech Content before posting** and allows them to rethink before **publishing it on social media platforms**

Educate users on social media policies on hate speech

# Literature Survey

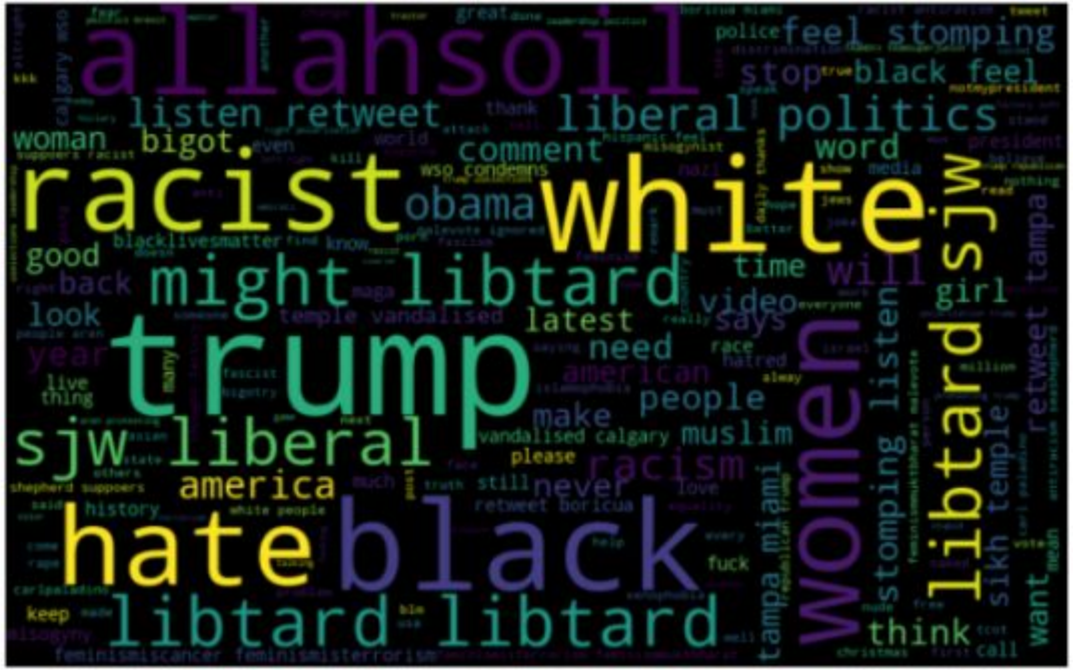
| Reference | Dataset | Technique | Results |
| --- | --- | --- | --- |
| Greevy Edel (2004) | **PRINCIP Corpus**<br>**Size:** 3M words from tweets | **Model**: SVM<br>**Feature Extraction**: BOW, Bi-gram | **BOW:**<br>Precision: 92.5%<br>Recall: 87%<br><br>**Bi-gram**<br>Precision: 92.5%<br>Recall: 87% |
| Waseem and Hovy (2016) | **Total Annotated tweets:** 16,914<br>#Sexist tweets: 3,383<br>#Racist tweets: 1,972<br>#tweets Neither racist nor sexist: 11,559 | **Model**: Char n-grams<br>Word n-grams | **Char n-gram:**<br>Precision: 73.89%<br>Recall: 77.75%<br>F1 score: 72.87%<br><br>**Word n-grams:**<br>Precision: 64.58%<br>Recall: 71.93%<br>F1 score: 64.58% |
| Akshita et al (2016) | Waseem and Hovy, 2016<br>**Size**: 22,142 tweets<br>**Class**: Benevolent, Hostile, others | **Model**: SVM, Seq2Seq (LSTM), FastText Classifier(by Facebook AI research)<br>**Feature Extraction**: TF-IDF, Bag of n-words | Average F1 score among classes: 0.723(SVM), 0.74(Seq2Seq)<br>Overall F1 Score: 0.84(FastText) |

# The Dataset

| | Attributes | Description |
|---|---|---|
| **Train data** | Id | Unique number assigned to each tweet |
| | Label | Contains label's data (1 : Hate , 0 : Not-Hate) |
| | Tweet | Unique Sentences |
| **Test data** | Id | Unique number assigned to each tweet |
| | Tweet | Unique Sentences |

- **Dataset**: Twitter tweets data to do sentiment analysis (https://www.kaggle.com/nitin194/twitter-sentiment-analysis)
- **Number of tweets:** 31,935
- **Classes (%):** Not-Hate Labeled(93%), Hate Labeled(7%)
- **Target Class**: Hate, Offensive, Abusive

# Word Cloud of Hate Speech Tweets



# Implementation

# Techniques used

### Data Cleaning

Lemmatization,Stemming, Tokenization, Removal of stopwords, emoji, URL, orphaned characters and slang words, replace shorthand words

### Word Embedding Techniques and Bag of Words

Word2Vec with genism, TF IDF Vectorizor

### Feature Selection

Chi-Square Test, Lime Text Explainer

### OverSampling and Classification Algorithms:

RandomOverSampler, Best model adoption using Autogluon

### Language Modelling

BERT,  DistilBERT

# Procedure

```
#Lemmitization
lemmatizer = WordNetLemmatizer()
data_frame['clean_tweet'] = data_frame['clean_tweet'].apply(lambda x : ' '.join([lemmatizer.lemmatize(word) for word in x.split()]))
```

```
#Stemming
ps = PorterStemmer()
adwait = data_frame
#adwait.head()
data_frame['clean_tweet'] = data_frame['clean_tweet'].apply(lambda x : ' '.join([ps.stem(word) for word in x.split()]))
```

```
#Tokenization
corpus = []
for i in range(0,21387):
    tweet = data_frame['clean_tweet'][i]
    tweet = tweet.lower()
    tweet = tweet.split()
    tweet = [ps.stem(word) for word in tweet if not word in set(stopwords.words('english'))]
    tweet = ' '.join(tweet)
    corpus.append(tweet)
```

# Procedure (cntd.)

```python
#Techniques to convert the tweets into Bag-of-Words, TF-IDF, and Word Embeddings
#Building various classifiers: -
#TF-IDF approach
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df=2,stop_words='english')
# TF-IDF feature matrix
X1 = tfidf_vectorizer.fit_transform(corpus).toarray()
Y1 = df.loc[:,'label'].values
```

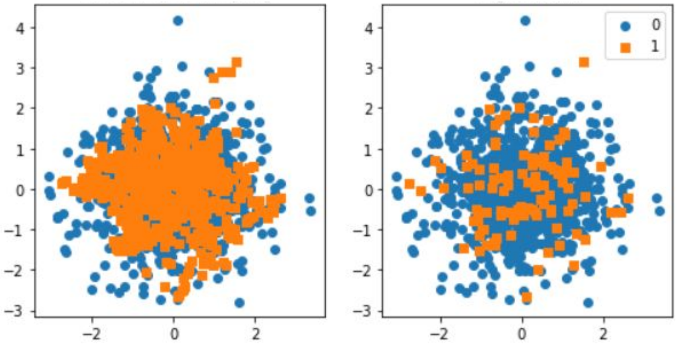
```python
# Skip-gram model (sg = 1)
size = 1000
window = 3
min_count = 1
workers = 3
sg = 1

stemmed_tokens = pd.Series(data_frame['stemmed_tokens']).values
# Train the Word2Vec Model
w2v_model = Word2Vec(stemmed_tokens, min_count = min_count, size = size, workers = workers, window = window, sg = sg)
```

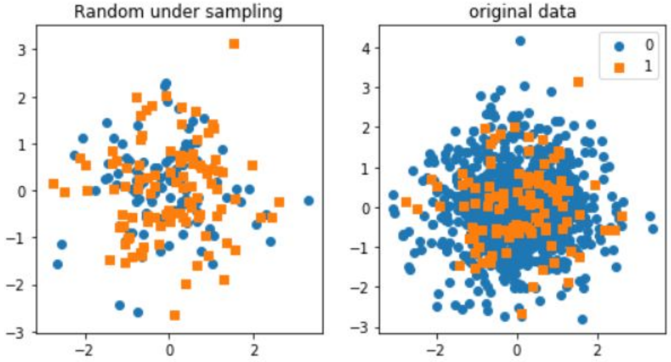
# Random Oversampling and UnderSampling

0 : Not- Hate
1 : Hate

Random Over Sampling

Original Data



# Procedure (cntd.)

```
ros = RandomOverSampler()

X_train, Y_train = ros.fit_sample(X_train, Y_train)
```

```
#PreTraing model
#For DistilBERT:
model_class, tokenizer_class, pretrained_weights = (ppb.DistilBertModel, ppb.DistilBertTokenizer, 'distilbert-base-uncased')

##Want BERT instead of distilBERT? Uncomment the following line:
#model_class, tokenizer_class, pretrained_weights = (ppb.BertModel, ppb.BertTokenizer, 'bert-base-uncased')

#Load pretrained model/tokenizer
tokenizer = tokenizer_class.from_pretrained(pretrained_weights)
model = model_class.from_pretrained(pretrained_weights)
```

# Performance

# Results

| | Model | Class | Precision | Recall | F1 Score | Accuracy |
|---|---|---|---|---|---|---|
| 1 | LightGBM ClassifierCustom with AutoGluon | 0 | 0.95 | 0.99 | 0.95 | 95% |
| | | 1 | 0.84 | 0.44 | 0.58 | |
| 2 | RandomForestClassifier with TfidfVectorizer | 0 | 0.96 | 1.00 | 0.98 | 96% |
| | | 1 | 0.93 | 0.49 | 0.64 | |
| 3 | RandomForestClassifier with Word2Vec | 0 | 0.93 | 1.00 | 0.96 | 93% |
| | | 1 | 0.91 | 0.34 | 0.51 | |
| 4 | distilBERT | 0 | 0.67 | 0.017 | 0.016 | 94% |
| | | 1 | 0.51 | 0.012 | 0.23 | |

# Impact

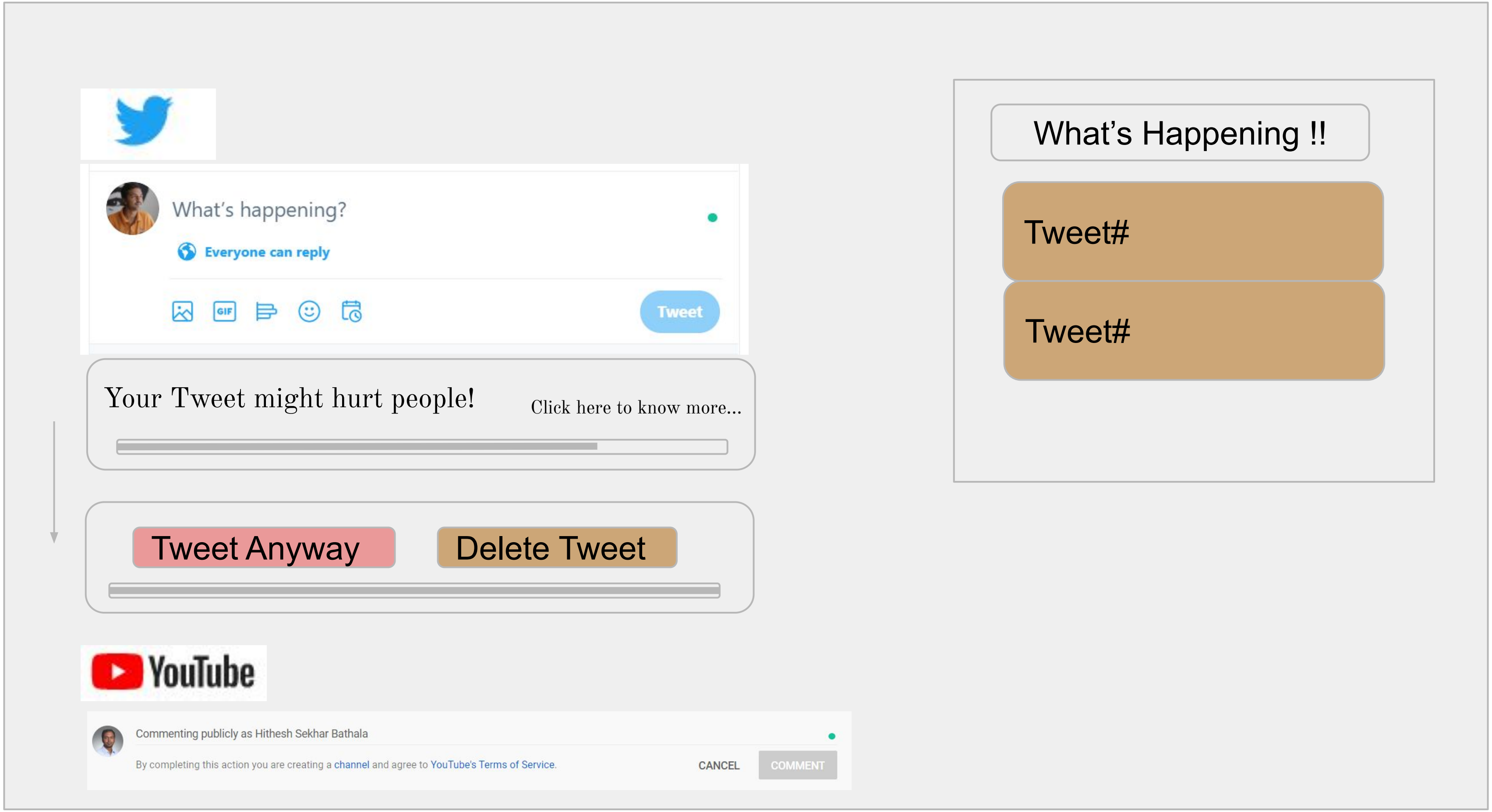
Restricting spread of hate messages

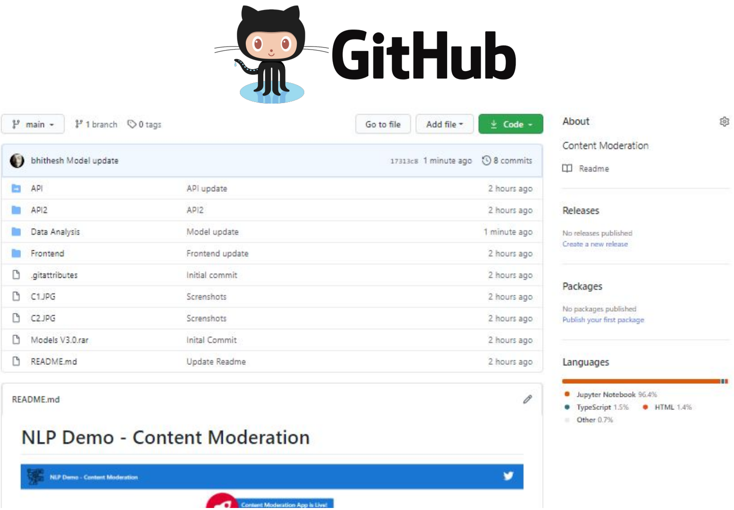
Reduction in cyber bullying and harassment.

Building a peaceful community

Giving users second chance

Digital media Literacy


What's happening?
Everyone can reply
Tweet
Your Tweet might hurt people!
Click here to know more...
Tweet Anyway
Delete Tweet
YouTube
Commenting publicly as Hithesh Sekhar Bathala
By completing this action you are creating a channel and agree to YouTube's Terms of Service.
CANCEL
COMMENT
What's Happening !!
Tweet#
Tweet#




## Contribute to our project Pull Today !

https://github.com/bhithesh/NLP-Demo

# Future Work

- Further fine tuning of the hyperparameters to improve accuracy on the dataset.
- Add more features to the dataset: Number of followers, location, age, etc.
- Use Multi-class classification to categorize the sentiment of the tweets.
- Include tweets in other languages: French, Hindi, etc.

# References


- ML Class Notes: https://srdas.github.io/MLBook2/
- https://scikit-learn.org/stable/
- https://huggingface.co/transformers/model_doc/distilbert.html
- https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/
- https://towardsdatascience.com/end-to-end-deployment-of-a-machine-learning-model-using-flask-dc456abcc6da
- https://medium.com/@tenzin_ngodup/simple-text-classification-using-random-forest-fe230be1e857
- https://www.kaggle.com/shahules/tackling-class-imbalance
- https://stackabuse.com/text-classification-with-python-and-scikit-learn/
- https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/
- https://towardsdatascience.com/another-twitter-sentiment-analysis-bb5b01ebad90
- https://auto.gluon.ai/stable/tutorials/tabular_prediction/tabular-quickstart.html
- https://www.kaggle.com/c/detecting-insults-in-social-commentary/data
- https://marcotcr.github.io/lime/tutorials/Lime%20-%20basic%20usage%2C%20two%20class%20case.html
- https://rstudio-pubs-static.s3.amazonaws.com/343661_dc127bbf141845b083b2dfa2cc75c9d2.html
- https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data
- https://www.researchgate.net/publication/29651698_Classifying_racist_texts_using_a_support_vector_machine

# Thank you !

1. Extract the model
2. Create a API - Input - Text, Output 0/1
3. Expose the API - URL (Local -> Online)
4. UI Shows - Youtube/ Twitter / Facebook

----------

TF-IDF Model
-- Classification Algo
Random forest

--Performance
Confusion Matrix,
ROC
Precision, Accuracy, Recall, F1, Matth.. coeff.

-> Autoglon -> --Performance

Word2Vec Model
-- Classification Algo

--Performance
Confusion Matrix,
ROC
Precision, Accuracy, Recall, F1, Matth.. coeff.

NLP Techniques used - Creating, Problems solved
How many models ?
Each model --Performance?

The quality of the ideation (this is usually the hardest part of the project).
> Prove we have an important problem to solve ? Impact may be | Why there should be centralized moderation system

Problem - 2,3,4
Soluiton - 5

The quality of the execution of the project (how well you achieved the goals set out in part 1 through implementation of NLP).
> Website Demo
> NLP - Models tests and --Performance
> NLP - Techniques used to improve the model

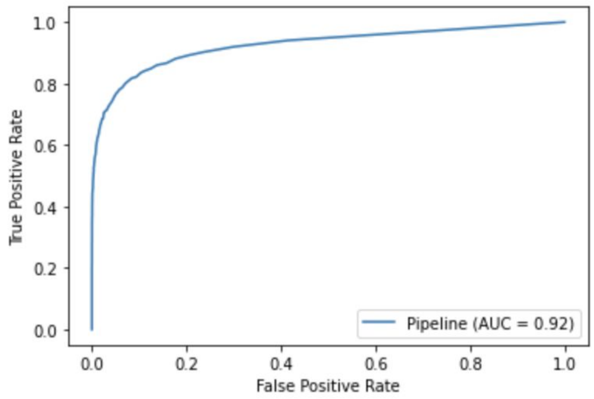
Literature Survey - 8 -> Benchmark ? What people have done in the past
Dataset - 10
Implementation - 12, 13, 14
Results - 16
Impact - 18
Demo - Website

The quality of your final presentation (assessed as to how well it was explained and made accessible to others in the class).

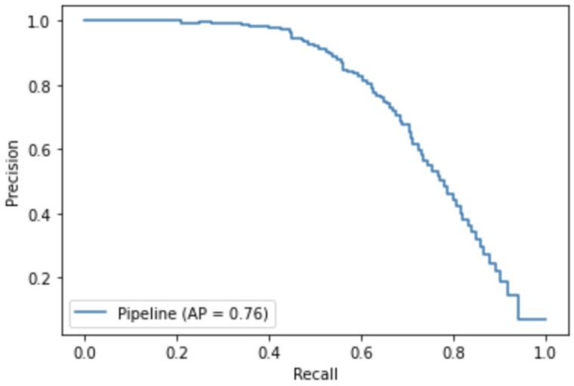
> Post the project and code on line with comments - Github
> How others can replicate

| | model | score_test | score_val | pred_time_test | pred_time_val | fit_time | pred_time_test_marginal | pred_time_val_marginal | fit_time_marginal | stack_level | can_infer | fit_order |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LightGBMClassifierCustom | 0.954844 | 0.962913 | 0.300831 | 0.068296 | 4.130069 | 0.300831 | 0.068296 | 4.130069 | 0 | True | 9 |
| 1 | weighted_ensemble_k0_l1 | 0.954844 | 0.962913 | 0.305958 | 0.072440 | 4.948227 | 0.005127 | 0.004144 | 0.818158 | 1 | True | 10 |
| 2 | LightGBMClassifierXT | 0.952967 | 0.959339 | 0.207682 | 0.054327 | 1.970605 | 0.207682 | 0.054327 | 1.97060 | 0 | True | 6 |
| 3 | ExtraTreesClassifierEntr | 0.952758 | 0.959786 | 0.737171 | 0.221398 | 31.030114 | 0.737171 | 0.221398 | 31.03011 | 0 | True | 4 |
| 4 | ExtraTreesClassifierGini | 0.952654 | 0.960679 | 0.700324 | 0.220746 | 33.622900 | 0.700324 | 0.220746 | 33.62290 | 0 | True | 3 |
| 5 | RandomForestClassifierGini | 0.950568 | 0.958445 | 0.412708 | 0.224484 | 19.729767 | 0.412708 | 0.224484 | 19.729767 | 0 | True | 1 |
| 6 | CatboostClassifier | 0.950464 | 0.962913 | 0.100255 | 0.089109 | 6.425719 | 0.100255 | 0.089109 | 6.425719 | 0 | True | 7 |
| 7 | RandomForestClassifierEntr | 0.949525 | 0.958892 | 0.392173 | 0.123334 | 16.806752 | 0.392173 | 0.123334 | 16.806752 | 0 | True | 2 |
| 8 | LightGBMClassifier | 0.948900 | 0.956658 | 0.158465 | 0.042018 | 1.514441 | 0.158465 | 0.042018 | 1.514441 | 0 | True | 5 |
| 9 | NeuralNetClassifier | 0.933048 | 0.936997 | 0.212609 | 0.092670 | 38.852349 | 0.212609 | 0.092670 | 38.852349 | 0 | True | 8 |

## ROC Curve




## Precision-Recall Curve




Insights:

Insights:

Word2Vec with RandomForestClassifier

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 1.00 | 0.96 | 8899 |
| 1 | 0.00 | 0.00 | 0.00 | 690 |
| accuracy | | | 0.93 | 9589 |
| macro avg | 0.46 | 0.50 | 0.48 | 9589 |
| weighted avg | 0.86 | 0.93 | 0.89 | 9589 |

# Agenda

Problem

Proposed Solution

Literature Survey

More about Dataset

Implementation

Results - Model Perf.

Demo

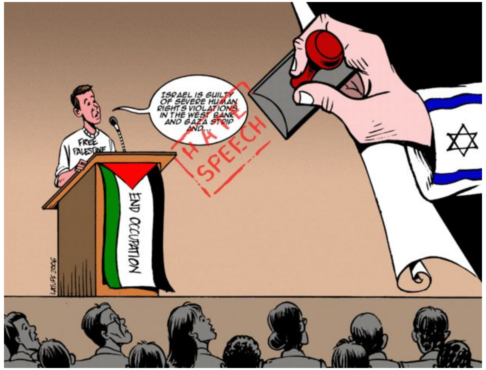
Impact of the project

Future work

Appendix.

# Motivation

With increasing anonymity and flexibility provided by the Internet, it has made it easy for users to communicate in an aggressive manner.


As amount of aggressive content increases, methods that automatically detect hate speech are very much required.

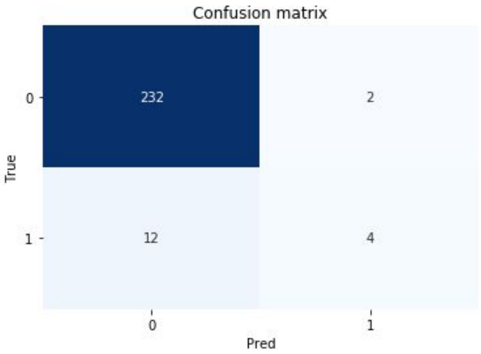
39]:
```
metrics.accuracy_score(test_labels, predicted)
```

39]: 0.944

40]:
```
classes = np.unique(test_labels)
```

41]:
```
## Plot confusion matrix
cm = metrics.confusion_matrix(test_labels, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues, cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes, yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)
```

41]: (array([0.5, 1.5]), <a list of 2 Text yticklabel objects>)




[ ]:

Autogluon

LightGBMClassifierCustom

```
Evaluation: accuracy on test data: 0.9548440921889665
Evaluations on test data:
{
    "accuracy": 0.9548440921889665,
    "accuracy_score": 0.9548440921889665,
    "balanced_accuracy_score": 0.7205960578031265,
    "matthews_corrcoef": 0.596575428138691,
    "f1_score": 0.9548440921889665
}
Detailed (per-class) classification report:
{
    "0": {
        "precision": 0.959037711313394,
        "recall": 0.9938236945536215,
        "f1-score": 0.9761208845750841,
        "support": 8905
    },
    "1": {
        "precision": 0.8476454293628809,
        "recall": 0.4473684210526316,
        "f1-score": 0.5856459330143541,
        "support": 684
    },
    "accuracy": 0.9548440921889665,
    "macro avg": {
        "precision": 0.9033415703381374,
        "recall": 0.7205960578031265,
        "f1-score": 0.7808834087947191,
        "support": 9589
    },
    "weighted avg": {
        "precision": 0.9510919066565839,
        "recall": 0.9548440921889665,
        "f1-score": 0.9482676290878028,
        "support": 9589
    }
}
```