Project: Hate Speech Detection using Transformers

Name: Seoyoung Kim
Email: idellio007@gmail.com
Country: USA
College: University of Michigan
Specialization: NLP
https://github.com/syoungk7/Hate_Speech_Detection

1. Problem description

With the advent of social media, people can express their opinions at any time, regardless of time and space. However, this freedom of expression has made it possible to use derogatory or discriminatory language against individuals or groups. Racial and social discrimination continues online, too. Detecting and mitigating such negative and hate-related speech will help create a safe and inclusive online environment.

2. Project Plan

1. ~~**Understand the problem**: We understand the nature of hate speech, its impact, and the importance of addressing it on online platforms. We define the problem as a sentiment classification task based on labeled Twitter data.~~
2. ~~**Data cleaning and normalization**: Preprocess the dataset by cleaning and normalizing the text data. This includes handling noise, removing irrelevant information, and ensuring consistency in the representation of tweets.~~
3. ~~**Expression learning**: We leverage Transformer-based architecture for representation learning. Transformers have achieved significant success in natural language processing tasks and are well-suited to capturing contextual information in text data.~~

4. **Model building and training**: We develop a deep learning model for hate speech detection using preprocessed data. The model is trained on the training data set to optimize accuracy and minimize false positives and false negatives.
5. **Performance evaluation and reporting**: Evaluate model performance on the test dataset using relevant metrics such as roc curve, precision, recall, and F1 score. Provides a comprehensive report on the strengths and limitations of the model.
6. **Model Deployment**: Deploy the trained model into a production environment, ready to integrate with online platforms to actively detect and filter hate speech.
7. **Model inference**: By implementing the model's inference mechanism, we can classify new tweets in real time to identify and flag potential instances of hate speech.

3. Data Understanding

- What type of data you have got for analysis
    a. The provided dataset for hate speech detection from Twitter consists of two files: test_tweets_anuFYb8.csv and train_E6oV3lV.csv.
    b. The data attributes include a label (0 or 1) and text_format, which contains the original tweets with noise.
- What are the problems in the data (number of NA values, outliers , skewed etc.)
    a. Missing Values (NA): no
    b. Outliers in Text Length: no
    c. Class Imbalance: no
    d. Noise in Text: *need to handle*
- What approaches you are trying to apply on your data set to overcome problems like NA value, outlier etc. and why?
    a. **Text Cleaning and Normalization**: Implement text cleaning techniques to remove noise, special characters, and irrelevant information. Normalize the text by converting to lowercase, handling contractions, and addressing variations in language.

b. **Tokenization and Padding**: Utilize tokenization to convert text into smaller units suitable for input to Transformer models. Apply padding to ensure uniform input lengths, which is crucial for efficient model training.

c. **Exploratory Data Analysis (EDA)**: Conduct a comprehensive exploratory data analysis to understand the distribution of key variables, detect anomalies, and make informed decisions on data preprocessing strategies.

d. **Feature Engineering**: Consider additional feature engineering if necessary, such as extracting information from tweet timestamps, user mentions, or hashtags, to enhance the model's ability to capture relevant patterns.

e. **Validation and Cross-Validation**: Use validation sets and cross-validation techniques to assess the model's performance robustly, ensuring it generalizes well to unseen data.

4. Data Cleansing and Transformation

Cleaning noisy text data is crucial for Hate Speech Detection using Natural Language Processing:

a. Text Cleaning

i. Removing URLs and User Mentions using regex
Usage: URL – ‘http\S+’, Mentions - '@\w+'

ii. Expending Contractions using python module - contractions
(reference: https://www.geeksforgeeks.org/nlp-expand-contractions-in-text-processing/)
Usage: contractions.fix()

iii. Removing Special Characters and Punctuation using regex
Usage: '[^a-zA-Z0-9\s]'

b. Tokenization

i. Tokenization - from nltk.tokenize import word_tokenize
Usage: word_tokenize(new_text)

ii. Lowercasing – core function
Usage: token.lower()

c. **Removing Stopwords** - from nltk.corpus import stopwords

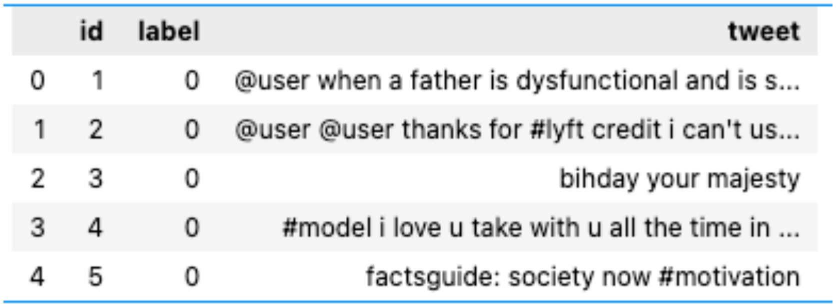
Usage: stop_words = set(stopwords.words('english'))

d. **Lemmatization** - from nltk.stem import WordNetLemmatizer

Usage: lemmatizer = WordNetLemmatizer()

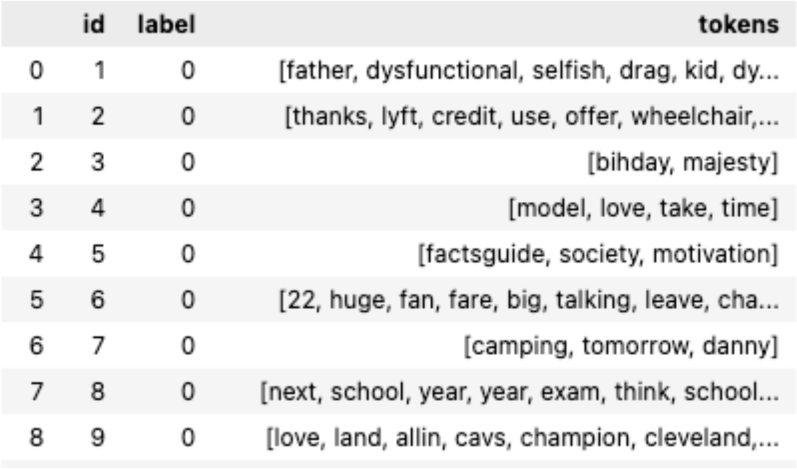
Result

- Raw data



	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

- Cleaned data



	id	label	tokens
0	1	0	[father, dysfunctional, selfish, drag, kid, dy...
1	2	0	[thanks, lyft, credit, use, offer, wheelchair,...
2	3	0	[bihday, majesty]
3	4	0	[model, love, take, time]
4	5	0	[factsguide, society, motivation]
5	6	0	[22, huge, fan, fare, big, talking, leave, cha...
6	7	0	[camping, tomorrow, danny]
7	8	0	[next, school, year, year, exam, think, school...
8	9	0	[love, land, allin, cavs, champion, cleveland,...

5. Model Selection

- **Model Selection**: Multinomial Naive Bayes (MNB), Random Forest (RF), Decision Tree (DT), Logistic Regression (LR), Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Stochastic Gradient Descent (SGD), XGBoost (XGB) Classifiers
- **Feature Extraction:** TF-IDF

6. Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score, roc_curve, classification_report,
PrecisionRecallDisplay
from sklearn.pipeline import make_pipeline


# oversampling
label_0 = train_data[train_data['label'] == 0] # label counting
label_1 = train_data[train_data['label'] == 1]
oversample = label_1.sample(len(label_0), replace=True, random_state=123)
train_oversampled = pd.concat([oversample, label_0], axis=0)
train_oversampled = train_oversampled.sample(frac = 1)
print(train_oversampled['label'].value_counts())
train_oversampled


# Splitting the data into training and testing sets
X_train, X_val, y_train, y_val = train_test_split(train_oversampled['features'],
train_oversampled['label'], test_size=0.2, random_state=123)


# set up models and test them
test_model = [MultinomialNB(), RandomForestClassifier(), DecisionTreeClassifier(),
              LogisticRegression(solver='saga'), SVC(), KNeighborsClassifier(),
             SGDClassifier(), XGBClassifier()]
```

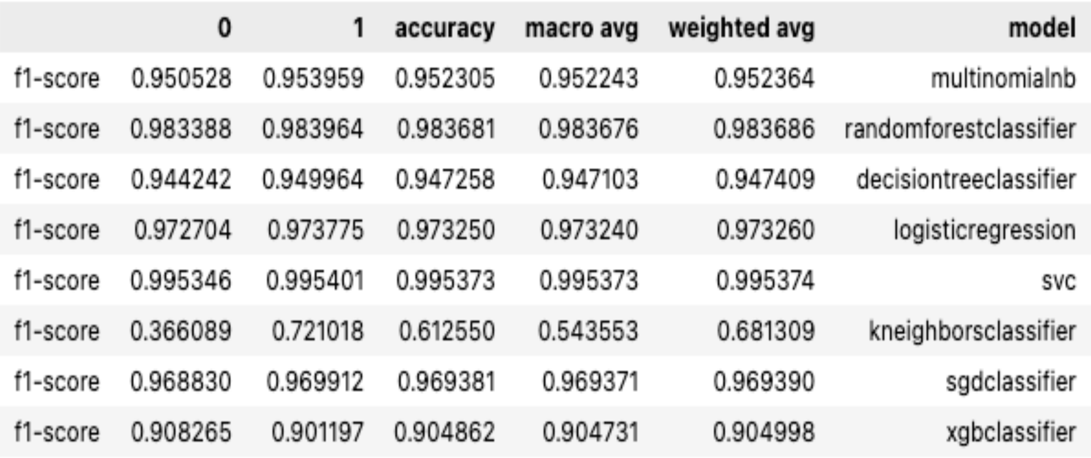
```
ALL_Cases_results = pd.DataFrame()
file = pd.DataFrame()

for val in test_model:
    # Feature extraction using TF-IDF
    # Choose a model from list
    pipe = make_pipeline(TfidfVectorizer(), val)
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_val)
    model = list(pipe.named_steps)
    result = classification_report(y_pred, y_val, output_dict=True)
    df_results = pd.DataFrame(result)
    df_results['model'] = model[1]
    file = pd.concat([file, df_results], axis=0)
    total_result6 = file

ALL_Cases_results = pd.concat([ALL_Cases_results, file[file.index == 'f1-score']], 
axis=0)
```

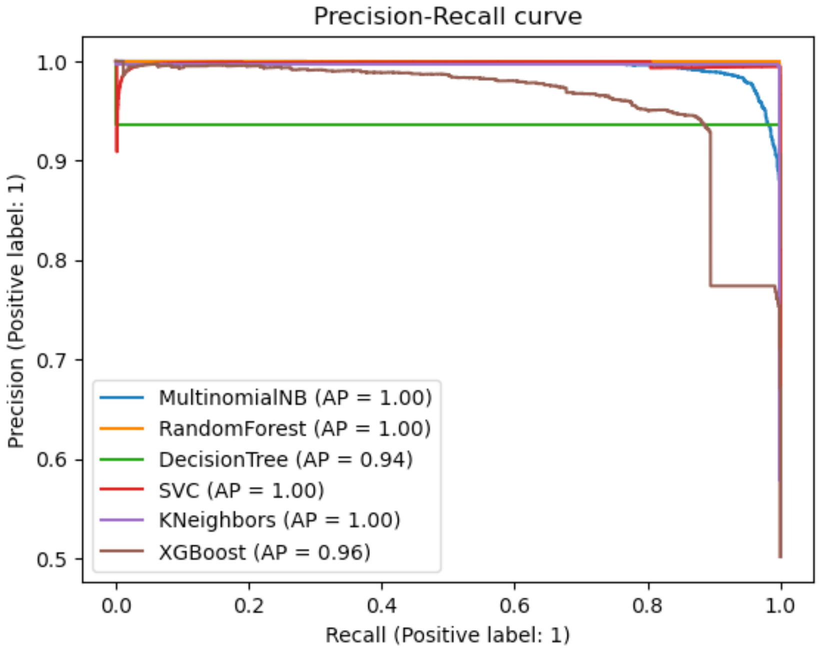
7. Model Result

```
# f1-scores for 8 models
ALL_Cases_results
```

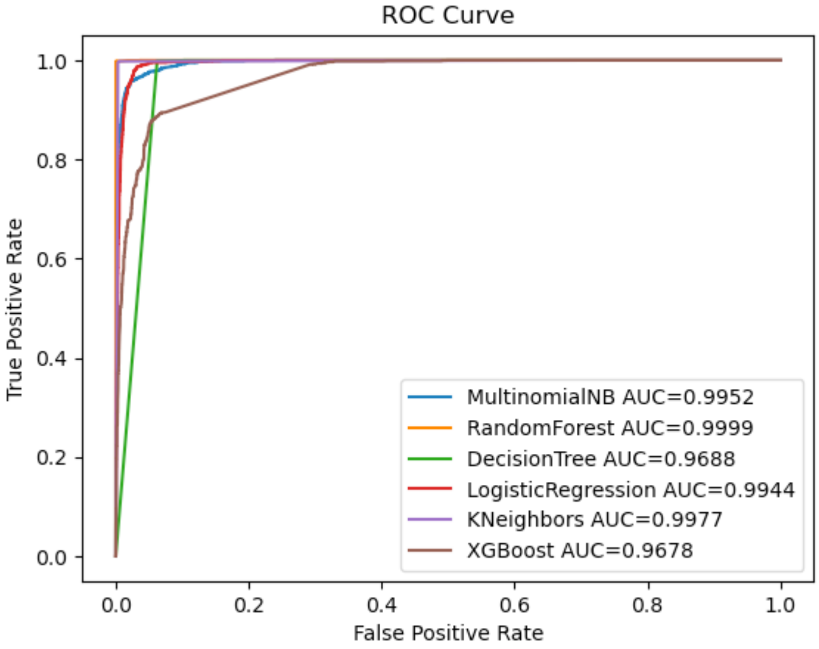


		0	1	accuracy	macro avg	weighted avg	model
'MNB'	f1-score	0.950528	0.953959	0.952305	0.952243	0.952364	multinomialnb
'RF'	f1-score	0.983388	0.983964	0.983681	0.983676	0.983686	randomforestclassifier
‘DT'	f1-score	0.944242	0.949964	0.947258	0.947103	0.947409	decisiontreeclassifier
'LR'	f1-score	0.972704	0.973775	0.973250	0.973240	0.973260	logisticregression
'SVM'	f1-score	0.995346	0.995401	0.995373	0.995373	0.995374	svc
'KNN'	f1-score	0.366089	0.721018	0.612550	0.543553	0.681309	kneighborsclassifier
'SGD'	f1-score	0.968830	0.969912	0.969381	0.969371	0.969390	sgdclassifier
'XGB'	f1-score	0.908265	0.901197	0.904862	0.904731	0.904998	xgbclassifier

```
# precision-recall curves for 6 models
```



```
# ROC scores and curves for 6 models
```



6. GitHub Repo link

https://github.com/syoungk7/Hate_Speech_Detection/data_preprocessing/