# Offensive Language Classification Model

**Course:** Data Foundations 6713
**Team:** Automat3rs
**Members:** Brenda Parnin, Rudy Martinez, Jose Fernandez, Arsalan Bhojani

# Agenda

- Background
- Our Approach
- Model Selection Procedure
- Error Analysis

# Background

The Connectivity of Social Media

## Social Media

- Embedded in the fabric of our society
- Can be leveraged to share aspects of our lives
- Ensures an unfathomable amount of data is available
- Has the potential to unite or destroy

# Our Approach

**Goal:** Create a model that accurately predicts the classification of offensive language in Twitter tweets.

**Datasets:** Trained and tested models on the train.tsv test.tsv datasets respectively.

**Data Identifiers:** Twitter identification numbers, Text of Twitter tweets, and Labels that correspond to the tweet's classification.

**Data Labels:** Not Offensive (NOT), Targeted Insult (TIN), and Untargeted (UNT)

- **NOT**: Posts that do not contain offense or profanity
- **TIN**: Posts containing insult/threat to an individual, a group, or others
- **UNT:** Posts containing non-targeted profanity and swearing. Posts with general profanity are not targeted, but they contain non-acceptable language.

# Our Approach Continued...

**Process - Models Tested**

- LinearSVC, RandomForest, and RandomForest + Lexicon models

**Parameters and Features**

- A variety of parameters and GridSearchCV values were tested
- A negative word dataset taken from the Data Foundations course was used to create Lexicon Features
- Lexicon Features were incorporated into the Random Forest model to enhance its performance

# LinearSVC

Initialize the classifier LinearSVC, Create the params with the C values

```
svc = LinearSVC()

params = {"C": [0.0001, 0.001, 0.01, 0.1, 1., 10., 100.]}
```

Initialize GridSearchCV and Fit the Model

```
clf = GridSearchCV(svc, params, cv = 10, scoring = 'f1_micro')
clf.fit(X_train, y_train)
```

# Random Forest

Build Parameter Grid

```
n_estimators = [10, 30, 50, 70, 90, 100]

parameters = {'n_estimators': n_estimators}
```

Initialize GridSearchCV and Fit the Model

```
rand_forest = RandomForestClassifier()
clf_rand = GridSearchCV(rand_forest, parameters, cv = 10, verbose = 2)
clf_rand.fit(X_train, y_train)
```

# Lexicon Features

Build Negative Word Lexicon Function

```
class LexiconClassifier():
    def __init__(self):

        self.negative_words = set()
        with open('negative-words.txt', encoding='iso-8859-1') as iFile:
            for row in iFile:
                self.negative_words.add(row.strip())

    def count_neg_words(self, sentence):
        num_neg_words = 0
        for word in sentence.lower().split():
            if word in self.negative_words:
                num_neg_words += 1
        return num_neg_words
```

Load Features

```
lex_class = LexiconClassifier()

X_train_lexicon_features = []
X_test_lexicon_features = []

for string in X_text_test:
    X_test_lexicon_features.append([lex_class.count_neg_words(string)])

for string in X_text_train:
    X_train_lexicon_features.append([lex_class.count_neg_words(string)])
```

Combine RandomForestClassifier with LexiconClassifier

```
vec = CountVectorizer(ngram_range = (1,1))

X_train_w_lex = vec.fit_transform(X_text_train)
X_test_w_lex = vec.transform(X_text_test)

X_train_lexicon_features = np.array(X_train_lexicon_features)
X_test_lexicon_features = np.array(X_test_lexicon_features)

X_train_w_lex = sp.hstack((X_train_lexicon_features, X_train_w_lex))
X_test_w_lex = sp.hstack((X_test_lexicon_features, X_test_w_lex))
```

# Model Selection Procedure

**In our exploration of possible offensive language classification models, we determined:**

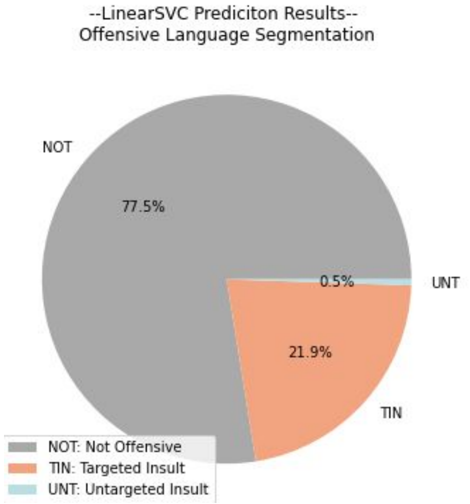
- Random Forest + Lexicon machine learning model yielded the best validation results
- F1 Micro and F1 Macro scores of **0.8758** and **0.3113** respectively

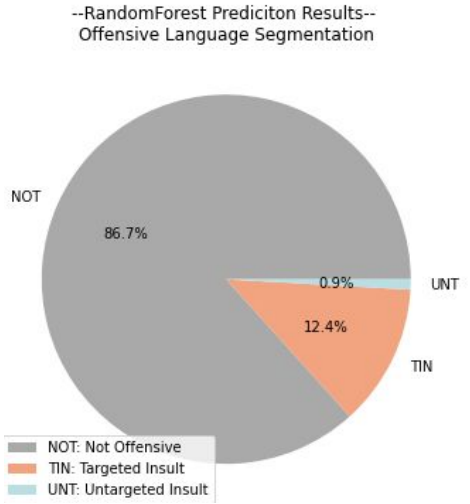
**Final Model Parameters + GridSearchCV Values**

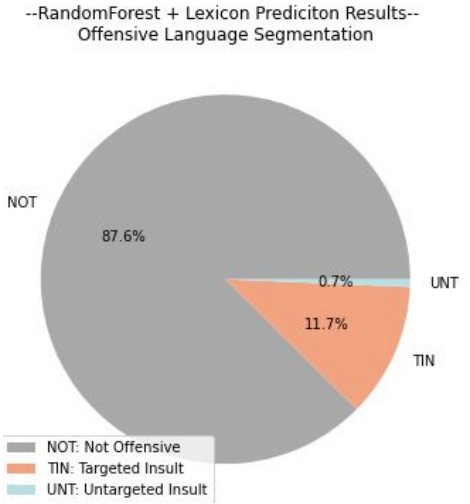
- n_estimators = [10, 30, 50, 70, 90, 100]
- parameters = {'n_estimators': n_estimators}

- rand_forest_lex = RandomForestClassifier()
- clf_rand_lex = GridSearchCV(rand_forest_lex, parameters, cv = 10, verbose = 2)
- clf_rand_lex.fit(X_train_w_lex, y_train)

# Model Results - Classification Segmentation



# Model Results - Scoring

**LinearSVC**

F1 Micro: 0.7753
F1 Macro: 0.2911

**Random Forest**

F1 Micro: 0.8667
F1 Macro: 0.3095

**Random Forest + Lexicon**

F1 Micro: 0.8758
F1 Macro: 0.3113

# Error Analysis - Random Forest + Lexicon

- **Step 1: Random sample of model results**
  - Our sample included 30 observations

- **Step 2: Manual review of 30 observations**
  - The model predicted 80% of the observations correctly (24 out of 30)

- **Step 3: Categorize Errors**
  - 6 observations were classified incorrectly
    - 5 were found to be *false negatives*
    - 1 was found to be a *false positive*

**Rationale:**

- Lack of specific negative words within the sample
- Model's inability to recognize the targeting of Twitter users in Tweets

# Thank you!