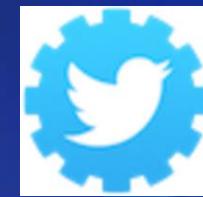




Natural Language Processing with Disaster Tweets



Nicole Hong



September 8, 2022



Natural Language Processing with Disaster Tweets

Predict which Tweets are about real disasters and which ones are not

Why this Kaggle?

- On-going, Knowledge-based Competition
- Same evaluation and ranking system as other competitive Kaggle
- Brush up on NLP, while competing
- NLP on Twitter data

Objective

- Identify disaster related tweets with high accuracy (Classification)
- Determine level of crisis for better response to disasters (Sentiment Analysis)

Why Twitter...?



- Real-time data on experience of those directly affected by a disaster
- Useful sentiment-based data
- Geospatial component, location-based social network
- Developing policy both during and after disasters



Kaggle Workflow

Text Patterns, Counts,
Frequency &
Visualizations

N-Grams (i.e. Text
Sequence), BOW &
TFIDF

Prediction & Model
Evaluation

Deployment

Tokenization, URL/
Contraction Removal &
Numbers /Special Characters
Removal

Supervised Machine Learning,
Deep Learning & Transformer
Models

VADER Sentiment Lexicon
& Sentiment Scores

EDA

Preprocessing

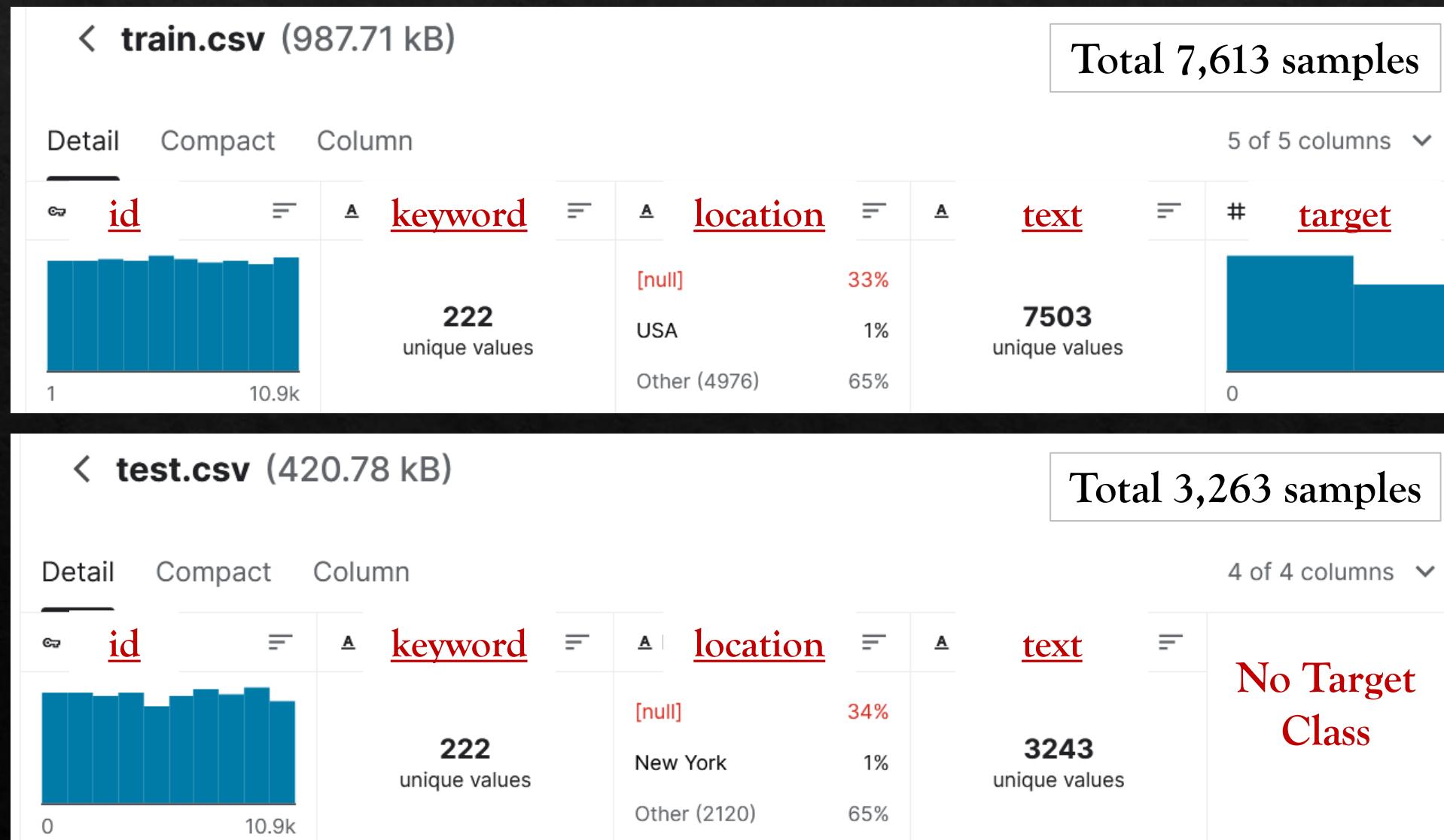
Feature
Extraction

Modelling

Sentiment Analysis

Deployment

Kaggle Data



Data Preparation

Class 0

Non-disaster
related Tweets

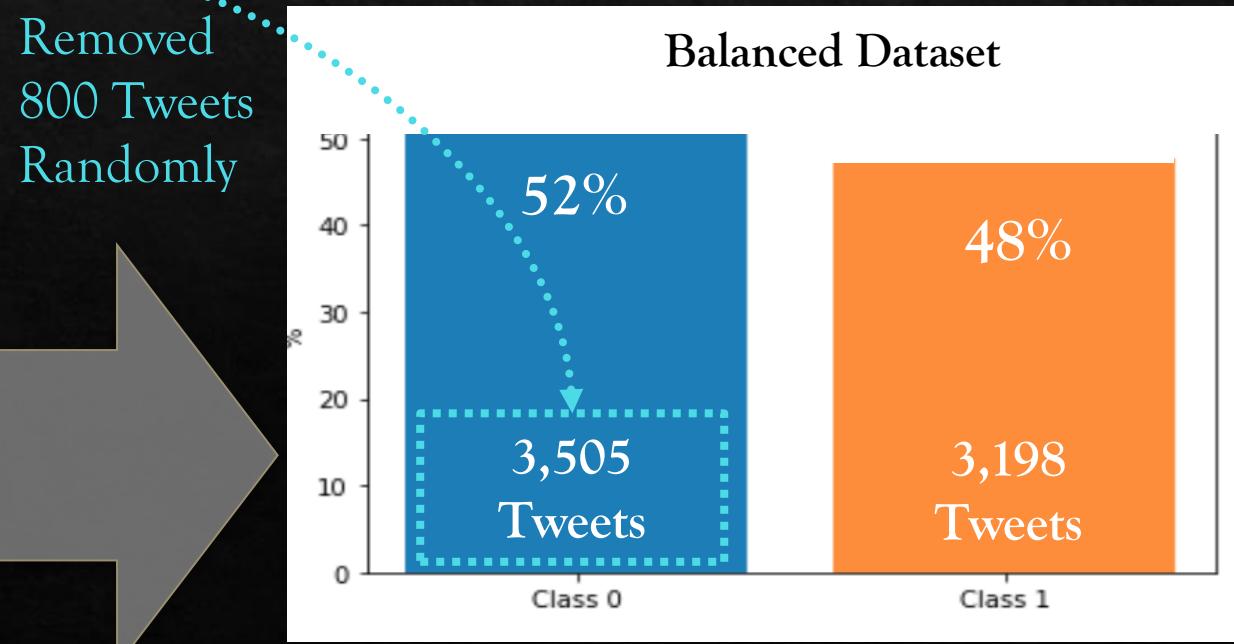
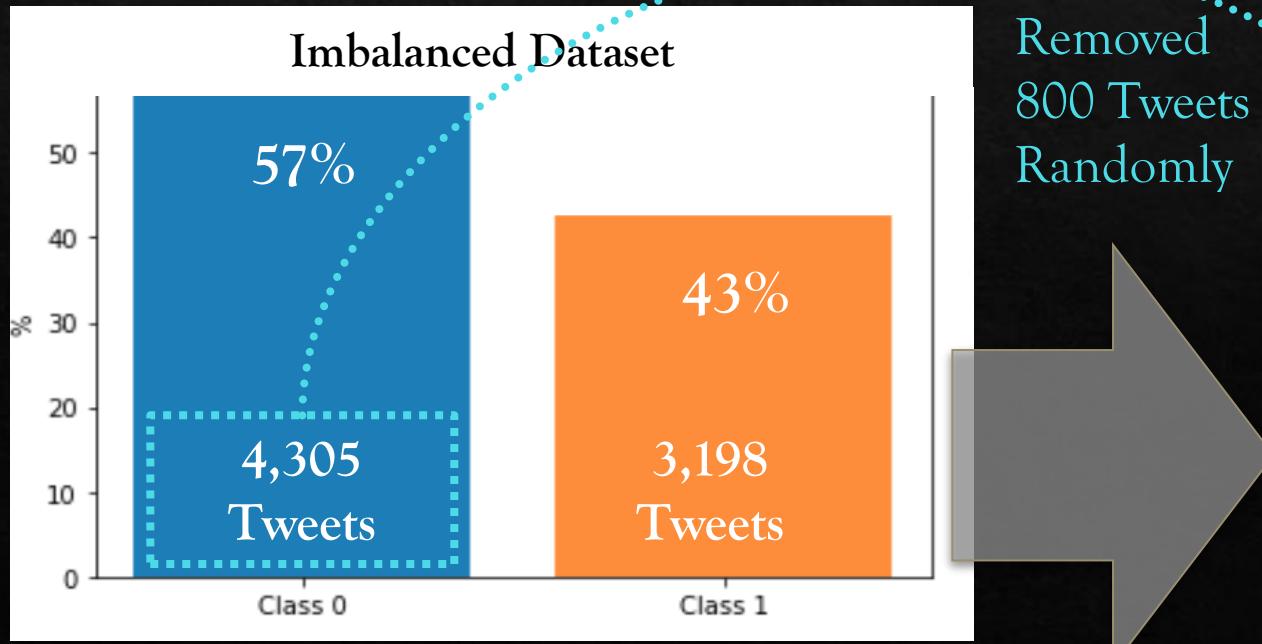
Class 1

Disaster related
Tweets

(1) Remove Duplicate Tweets



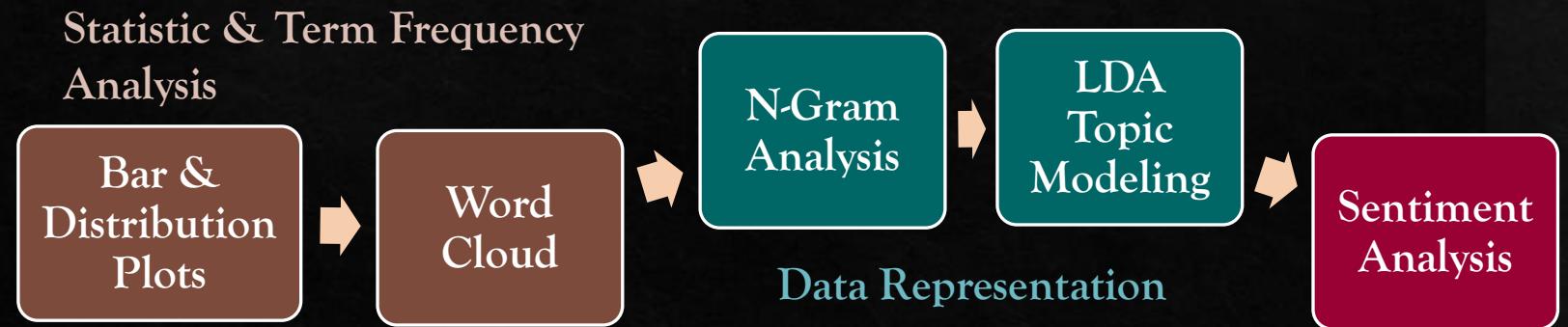
(2) Balance Target Class – Random Re-Sampling Train Data



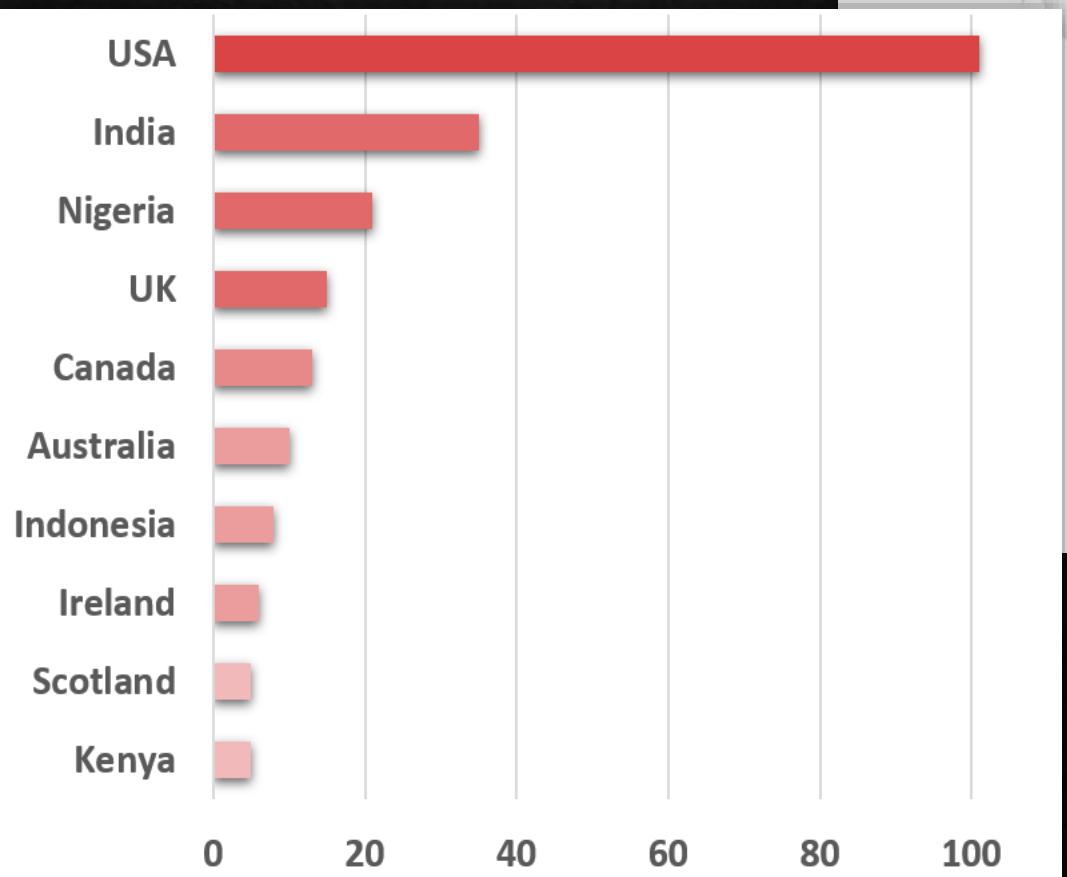
Exploratory Data Analysis (EDA)

EDA for NLP:

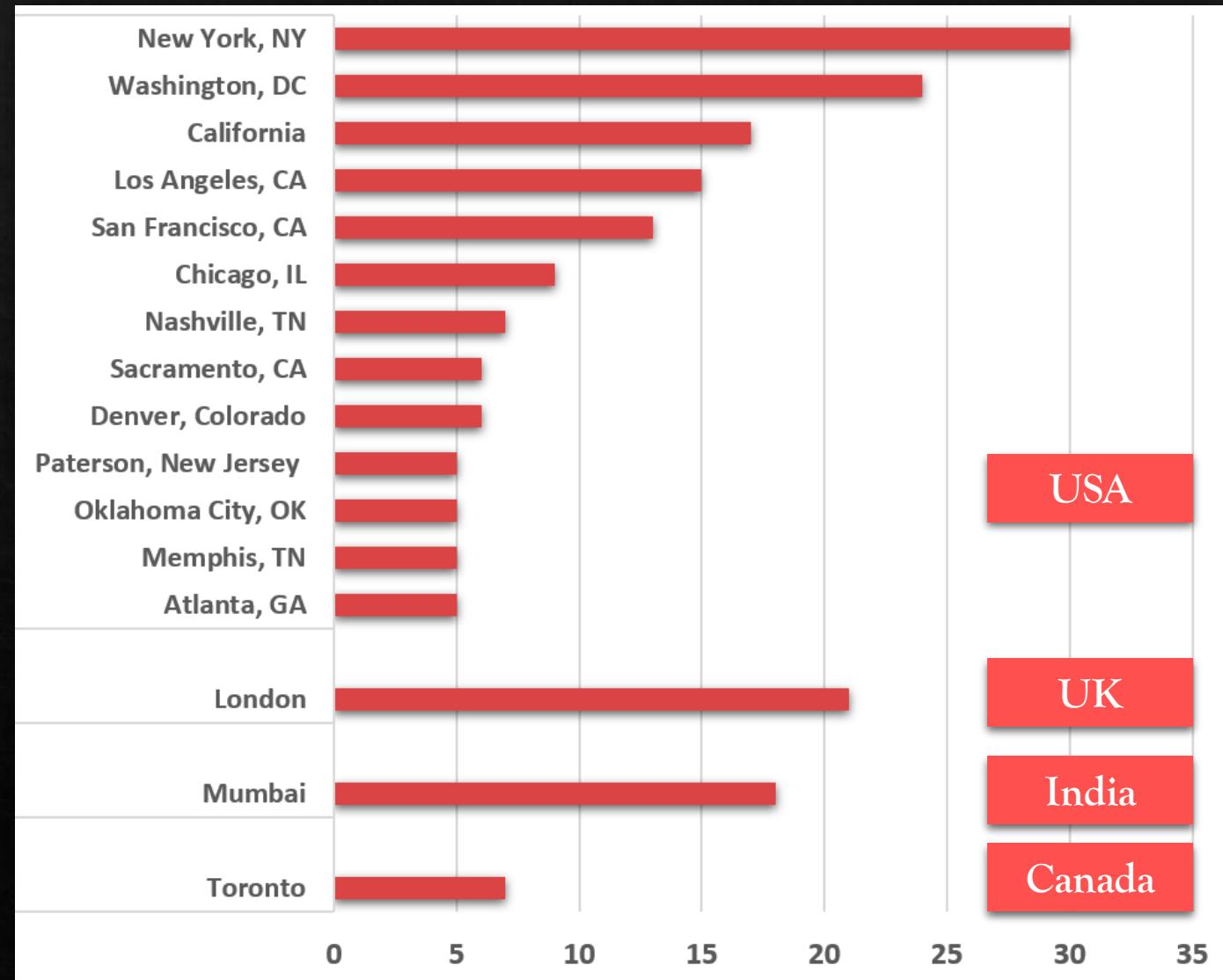
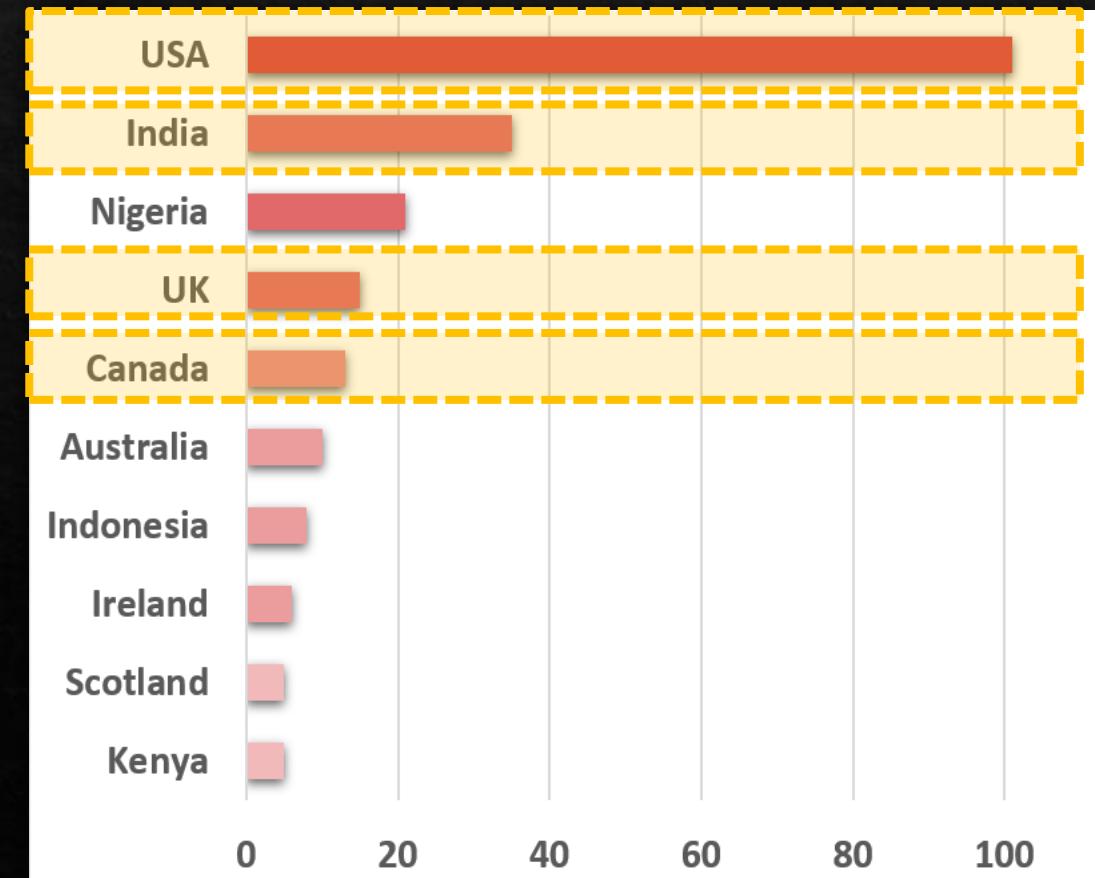
- Descriptive statistics & visualization
- Missing values and value patterns correlated with the target
- Highly correlated, duplicate features or most predictive features for the problem



EDA: Location Analysis

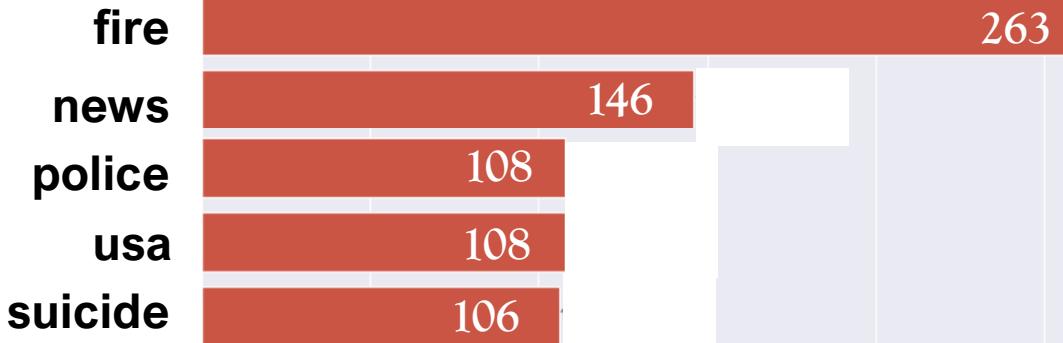
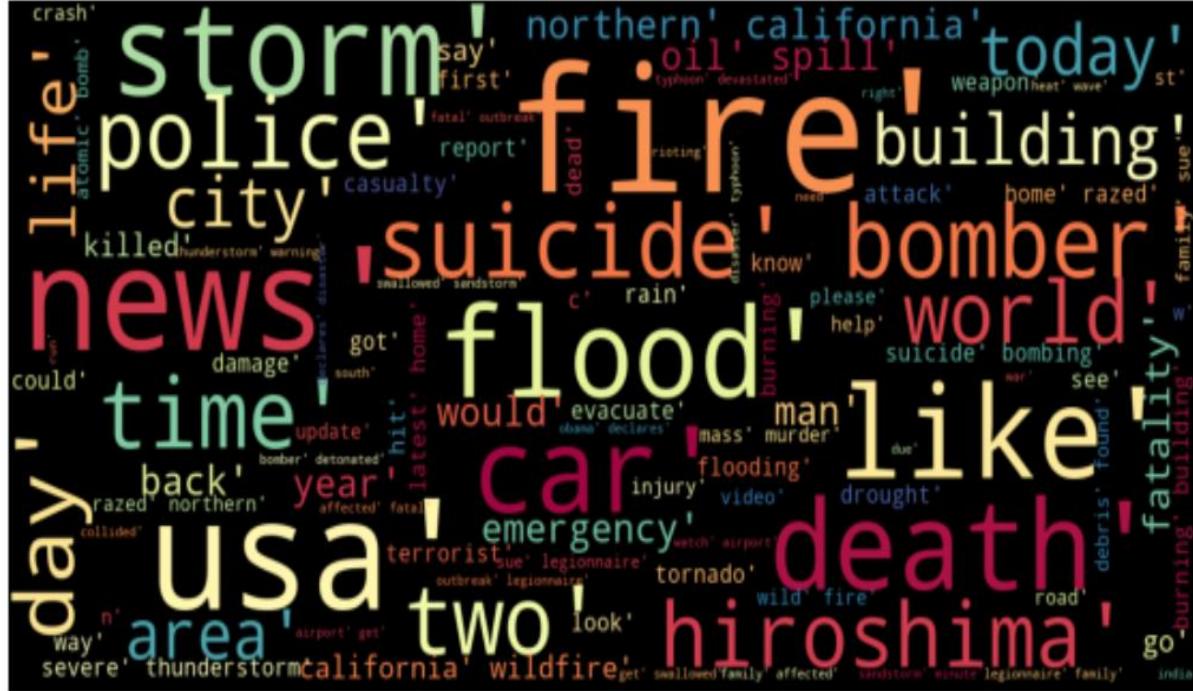


EDA: Location Analysis

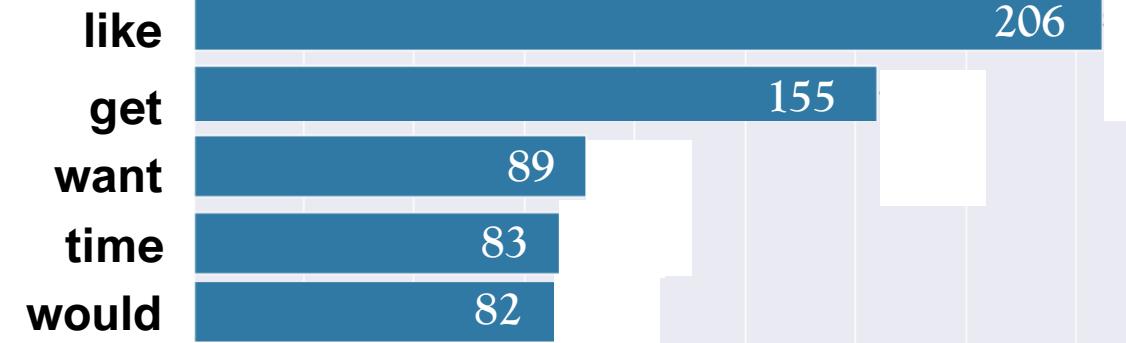
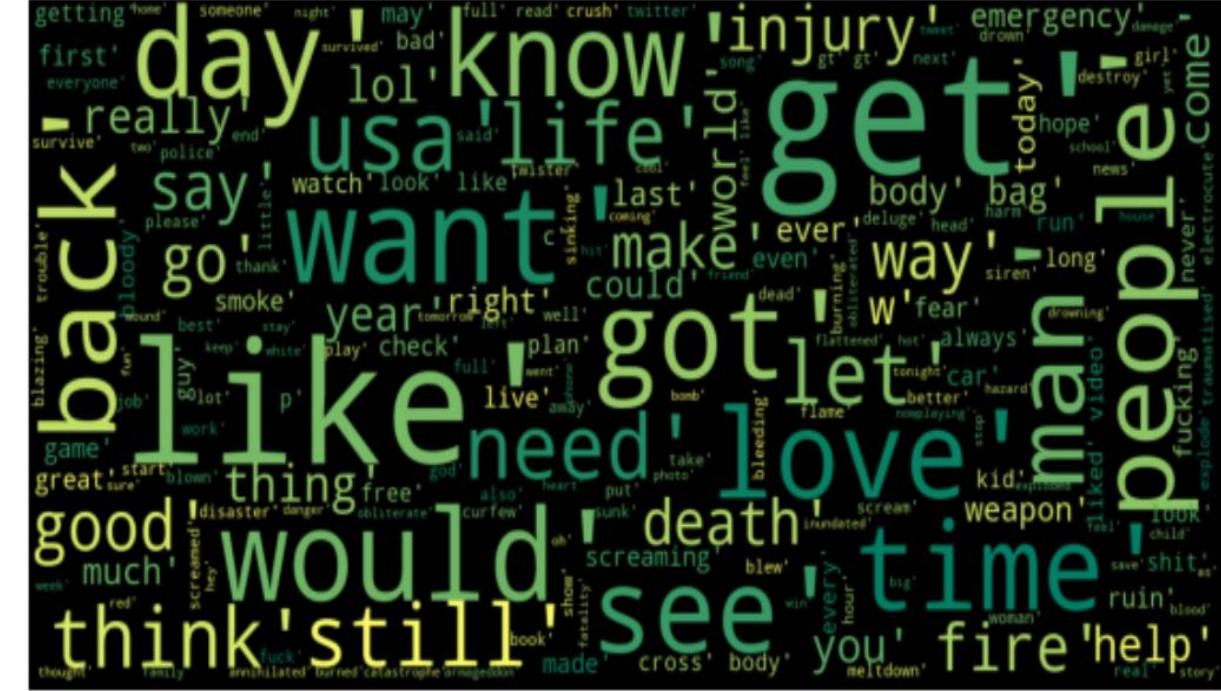


WordCloud: Commonly Used in Tweets

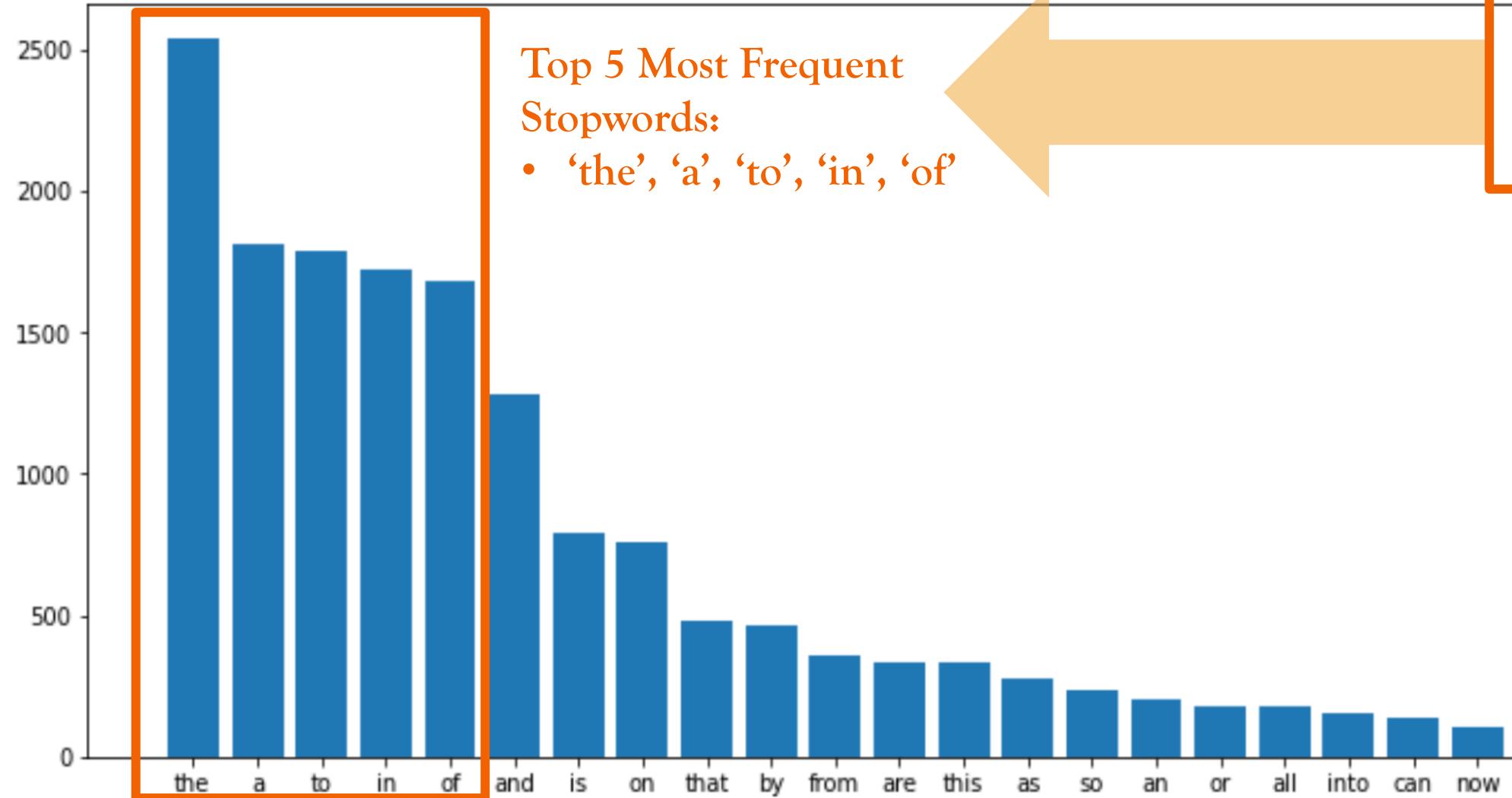
Disaster Tweets | Target = 1



Non-Disaster Tweets | Target = 0

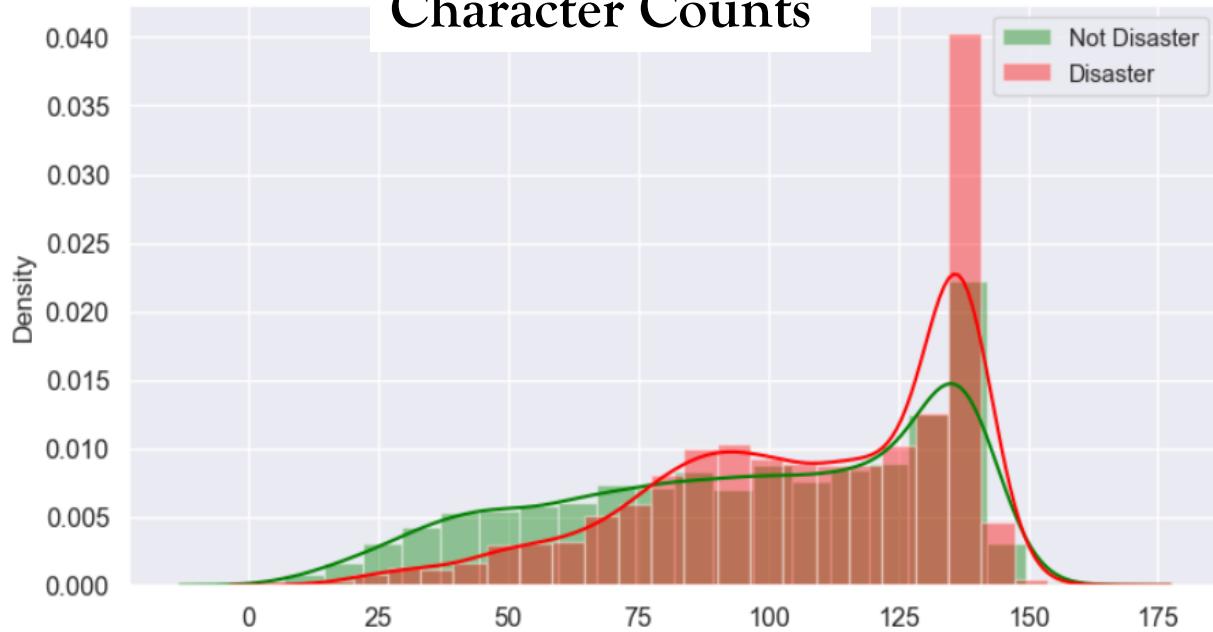


EDA: Stopwords Analysis

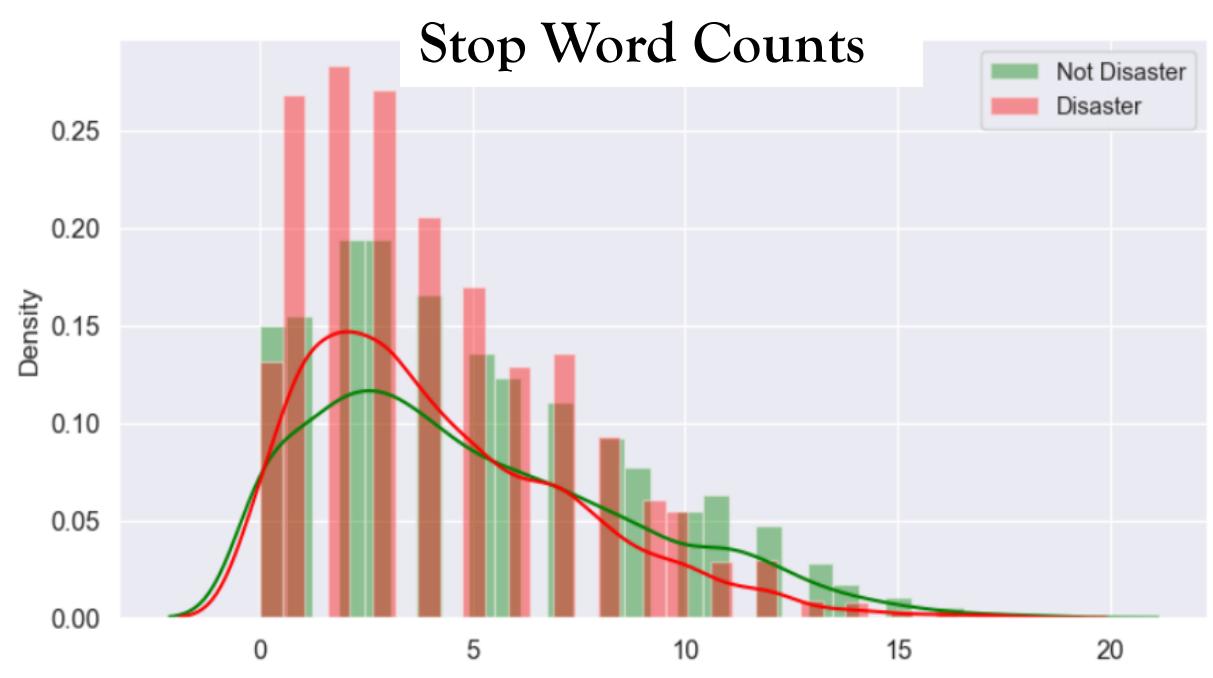


```
{'the': 2536,  
 'a': 1810,  
 'to': 1783,  
 'in': 1724,  
 'of': 1679,  
 'and': 1279,  
 'is': 792,  
 'on': 758,  
 'that': 484,  
 'by': 460,  
 'from': 360,  
 'are': 336,  
 'this': 332,  
 'as': 278,  
 'so': 235,  
 'an': 203,  
 'or': 179,  
 'all': 176,  
 'into': 155,  
 'can': 139,  
 'now': 101,}
```

Character Counts

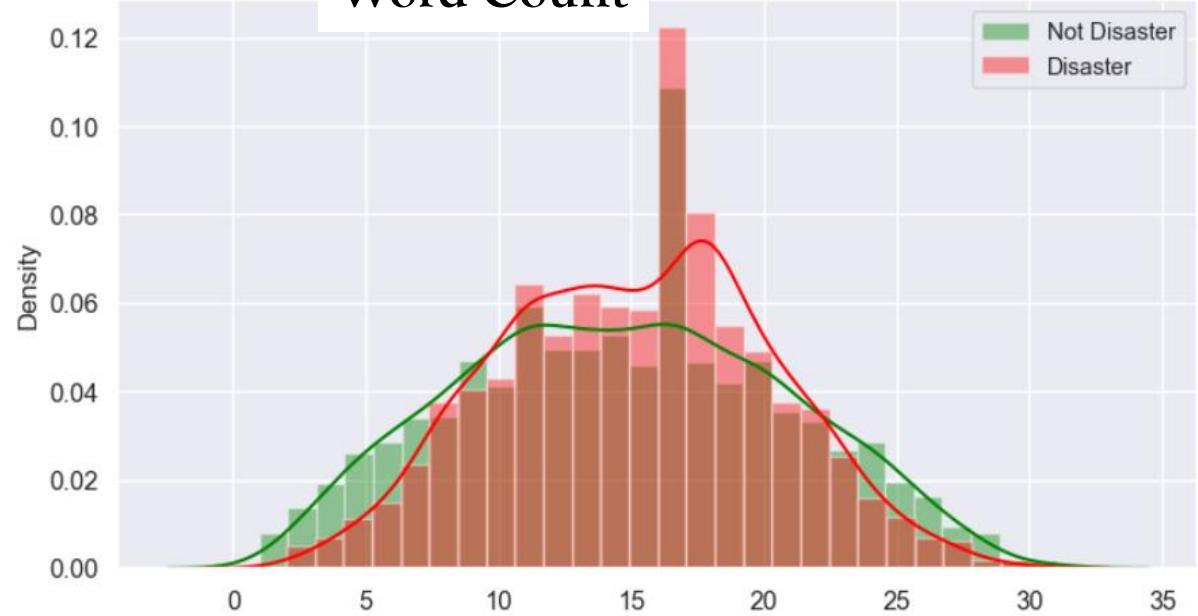


Stop Word Counts



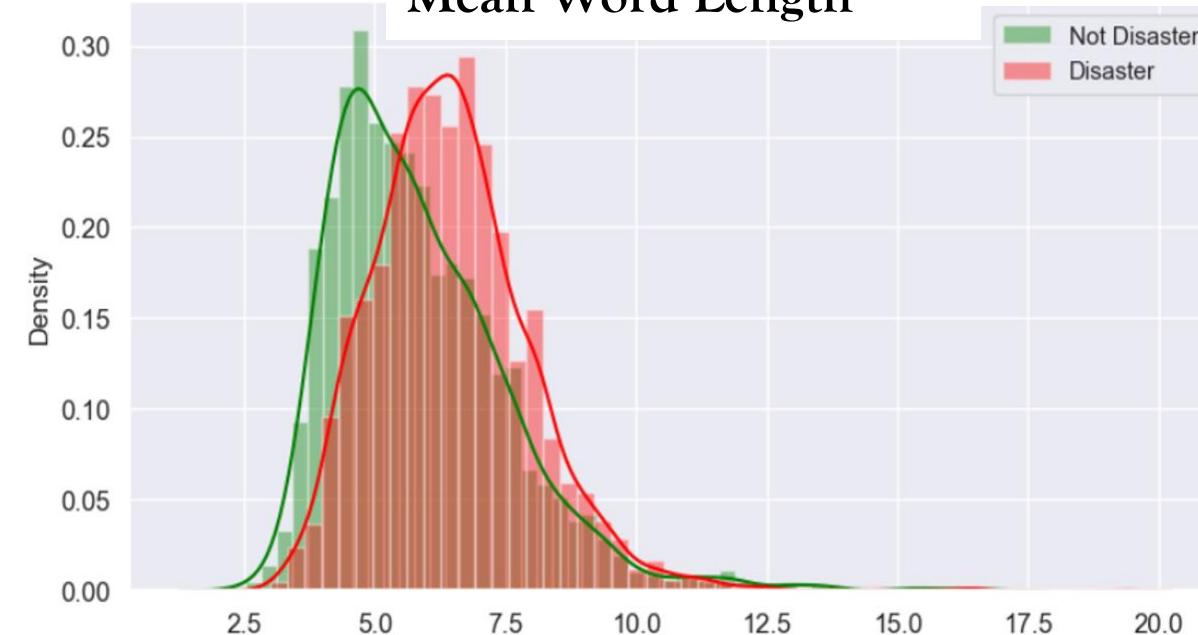
EDA:
Distribution Plots
1

Word Count

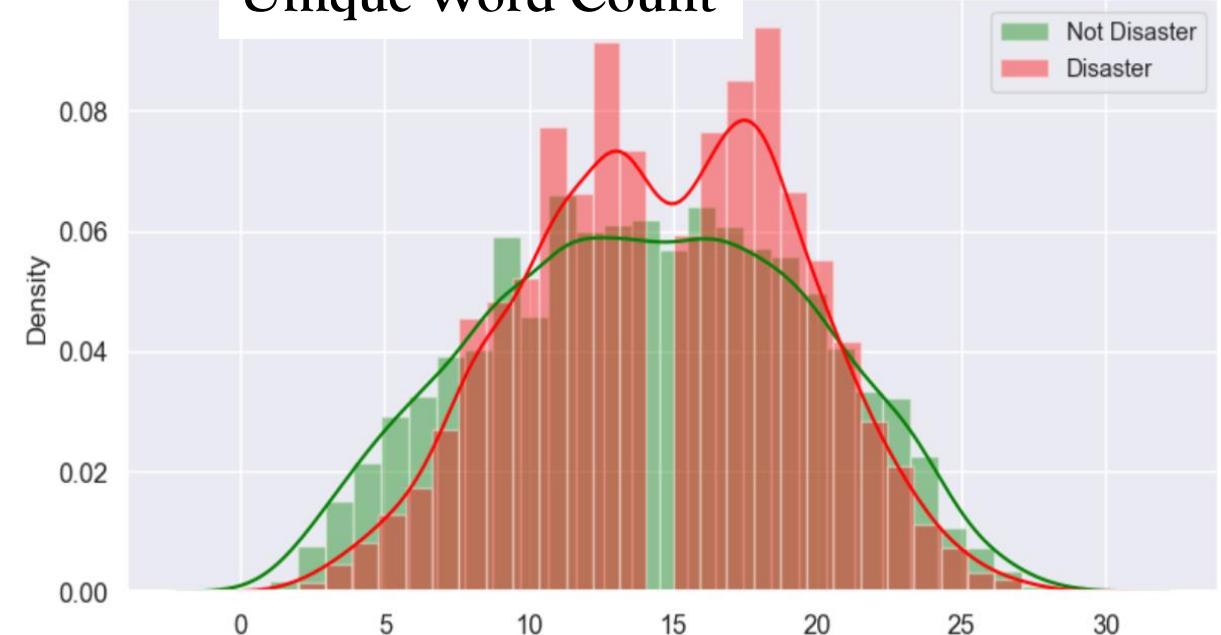


EDA: Distribution Plot2 #2

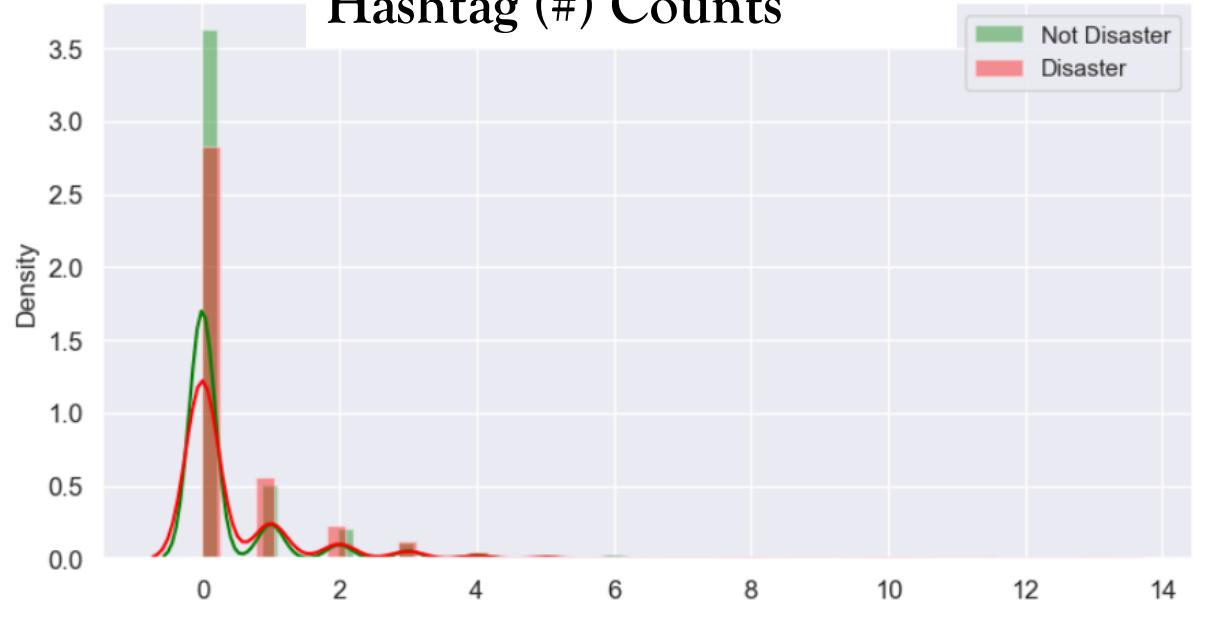
Mean Word Length



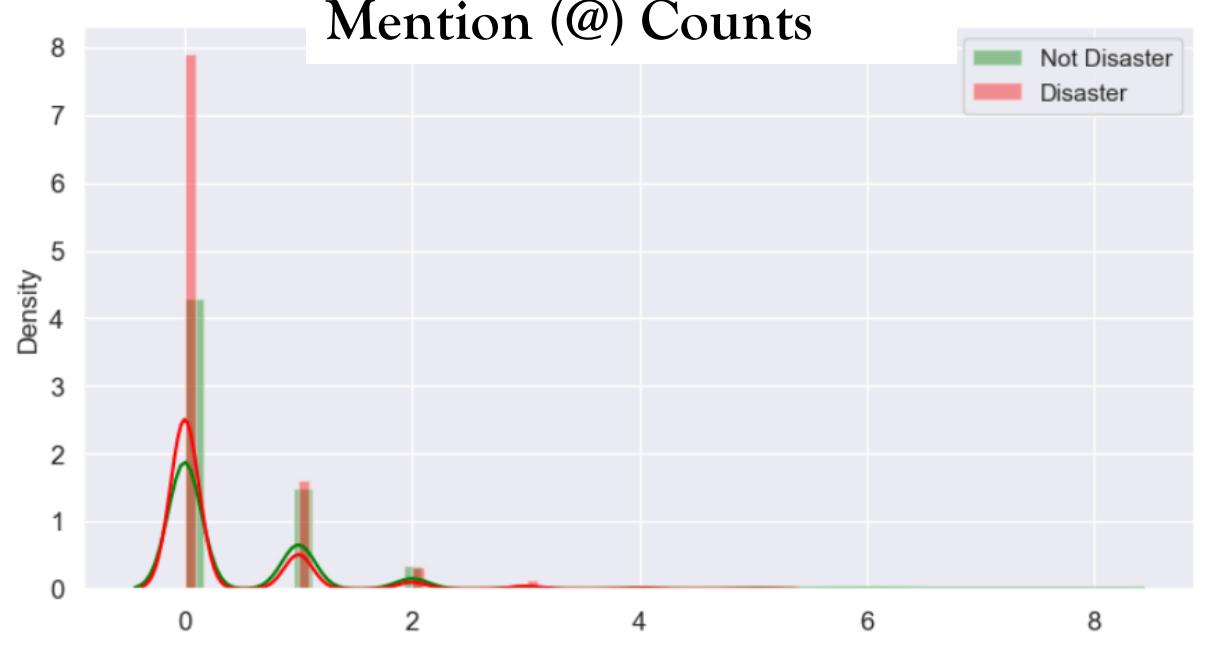
Unique Word Count



Hashtag (#) Counts

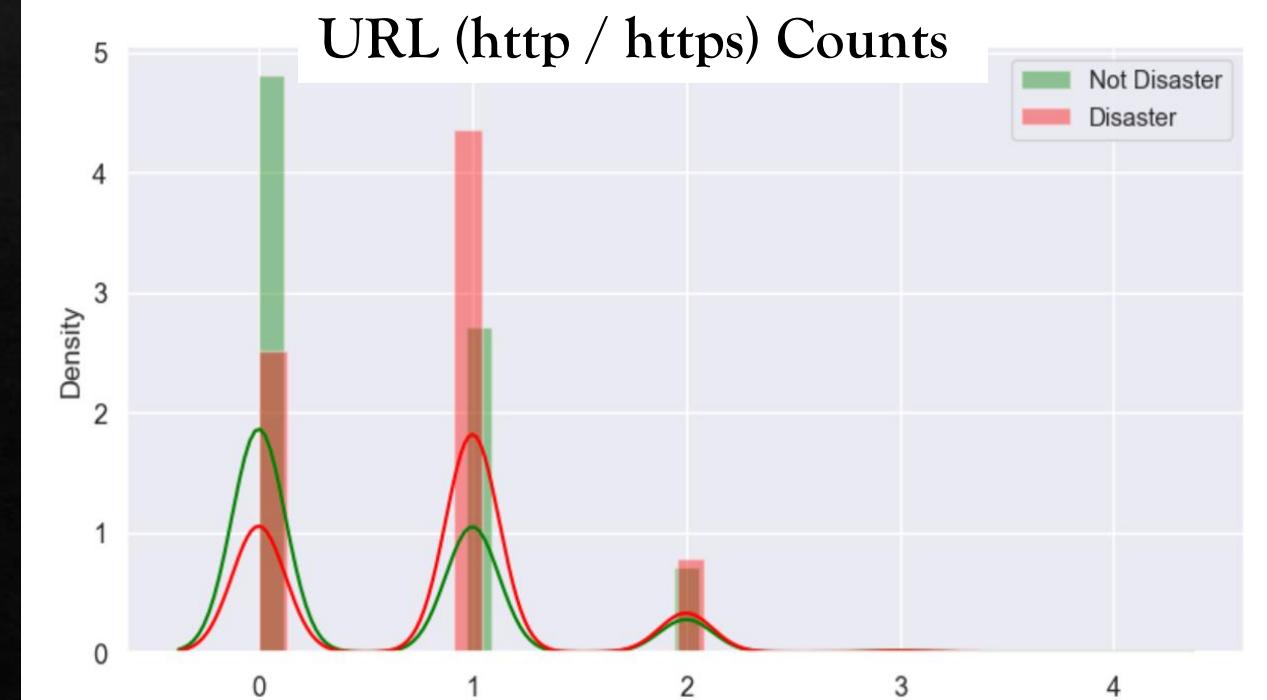
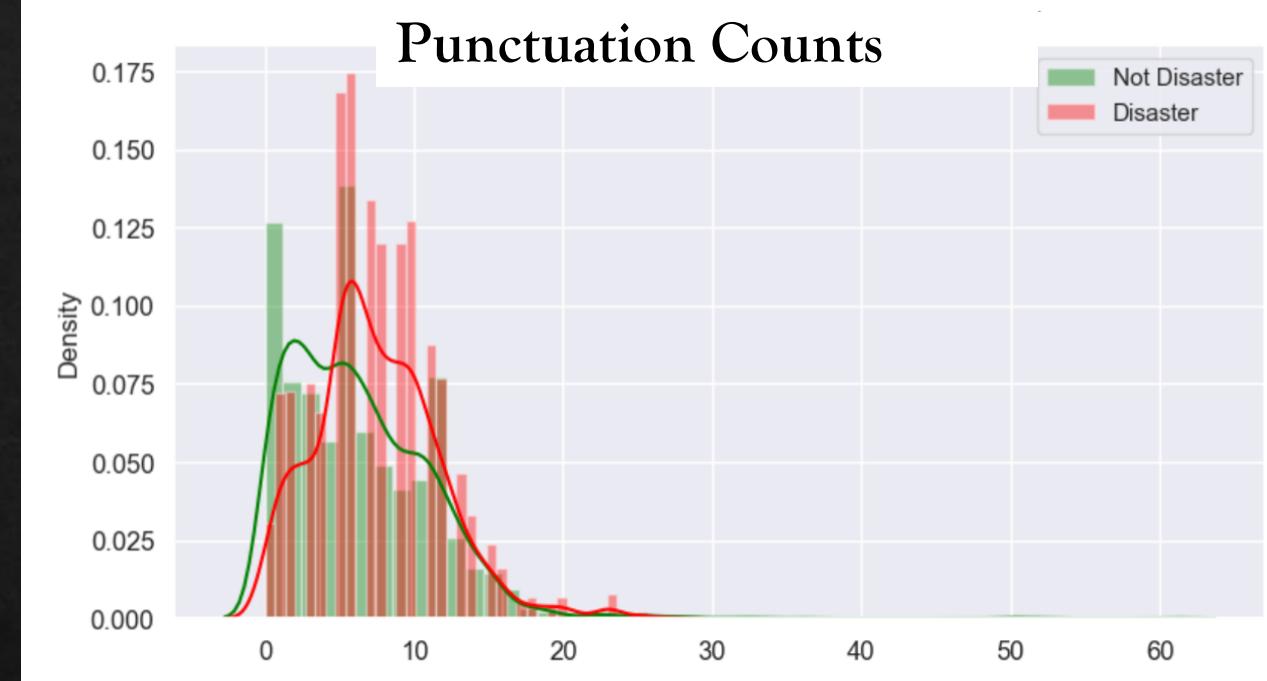


Mention (@) Counts

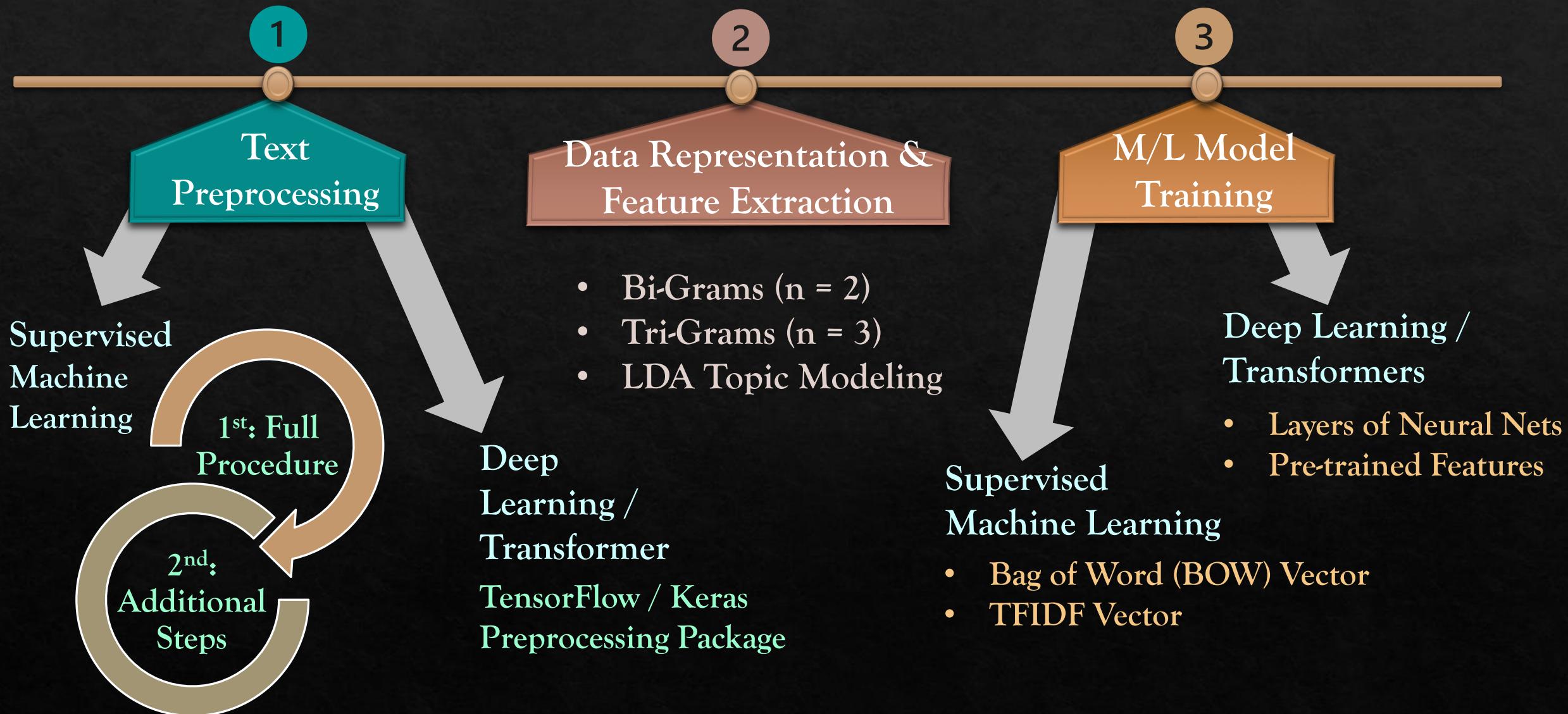


EDA:
Distribution Plots
#3

EDA: Distribution Plots #4



Basic Steps for Text Classification



Text Preprocessing

1st Preprocessing:

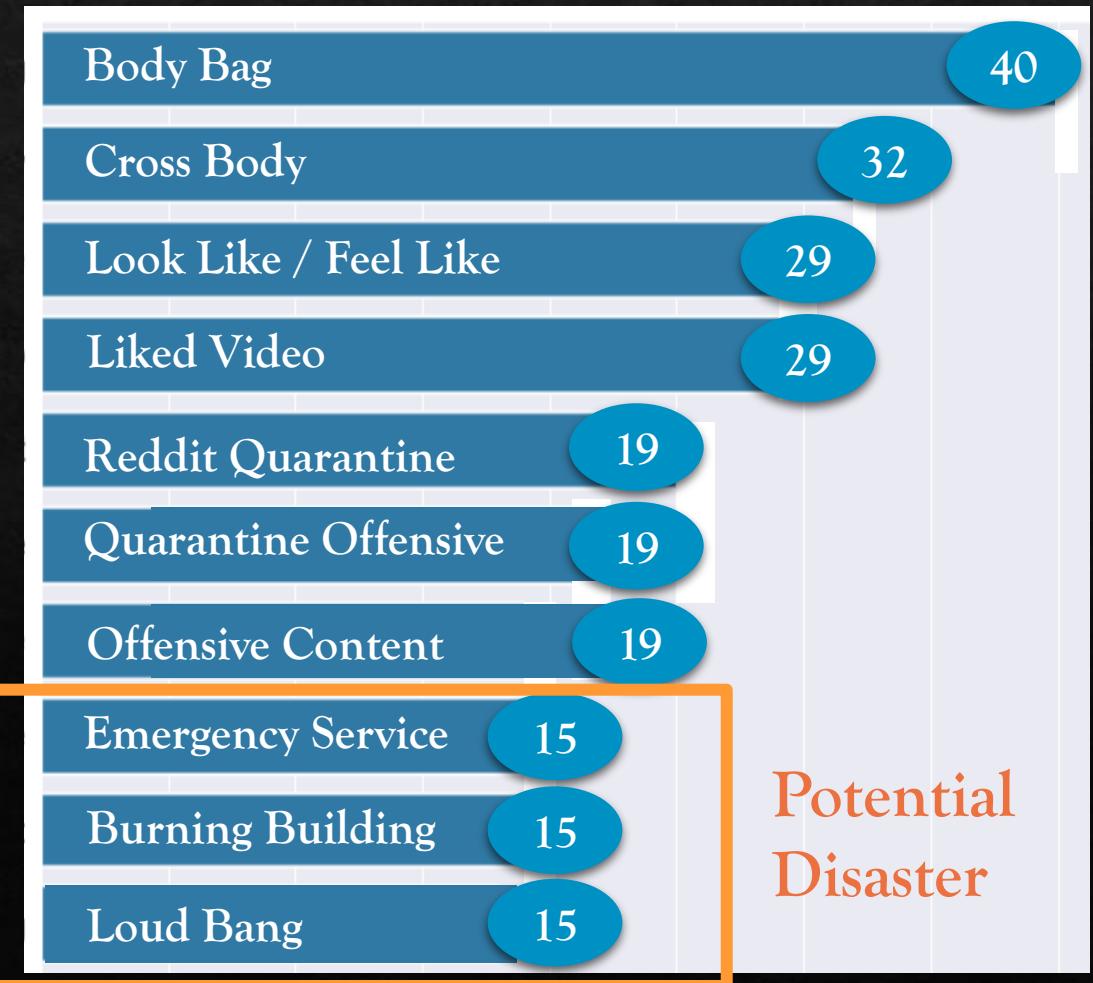
- Replace Abbreviated/ Special Text to Standard Text
- Remove URL
- Remove Contractions
- Tokenize
- Remove Numbers & Unicode
- Remove Punctuation
- Normalize
- Clean Repeated Characters in Text
- Remove Stop Words
- Lemmatize
- Remove Contractions
- Join Tokenized Words to Sentence

Data Representation: Bi-Grams

Disaster Tweets | Target = 1

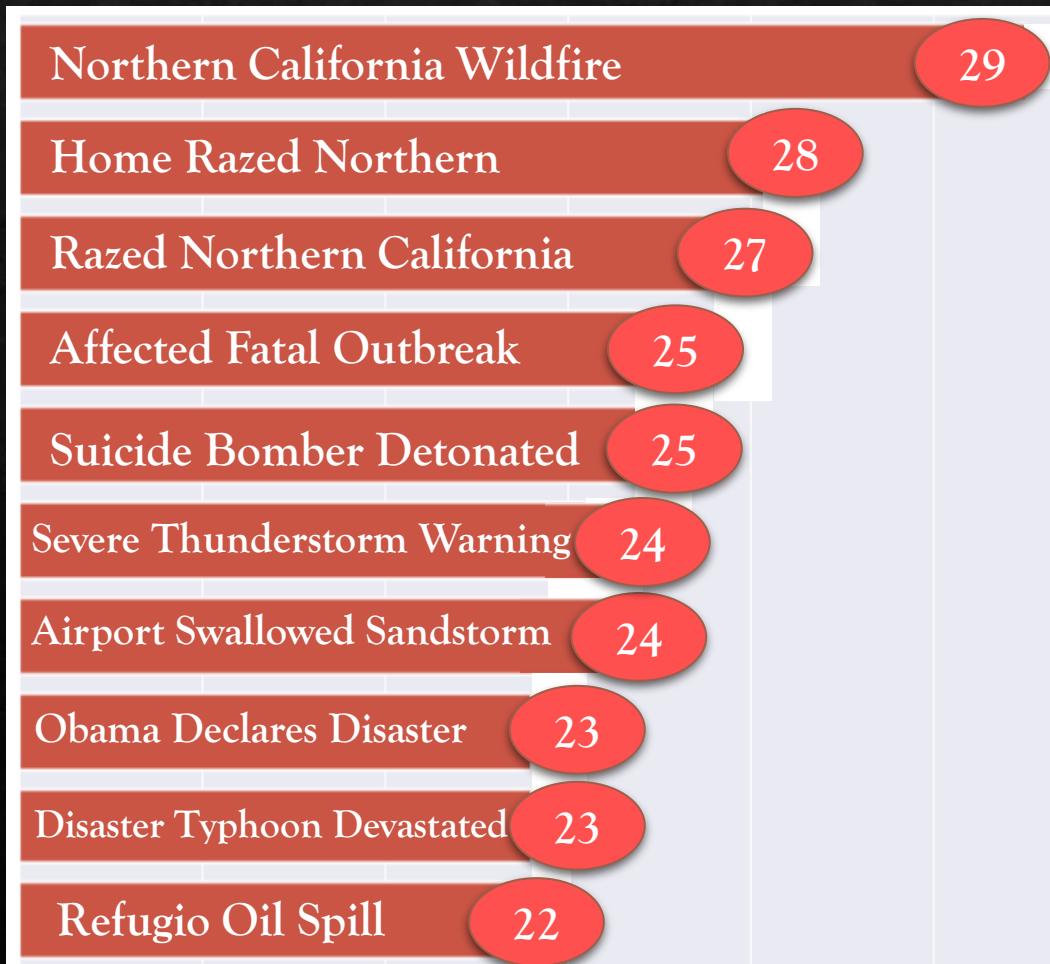


Non-disaster Tweets | Target = 0



Data Representation: Tri-Grams

Disaster Tweets | Target = 1

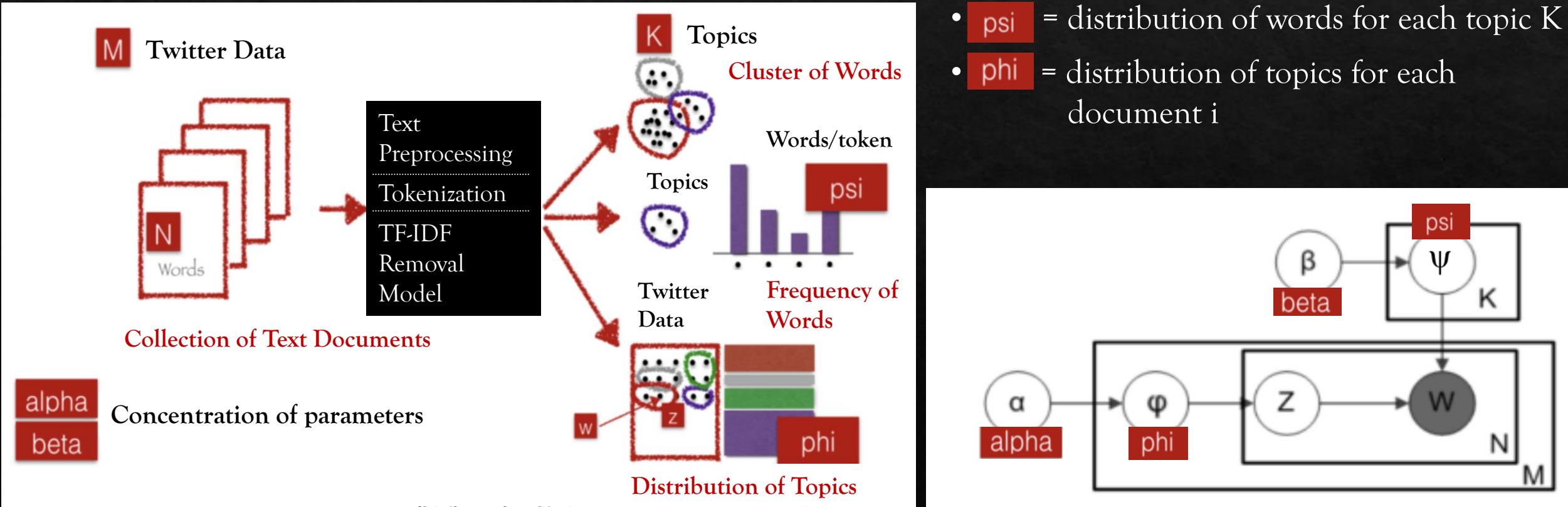


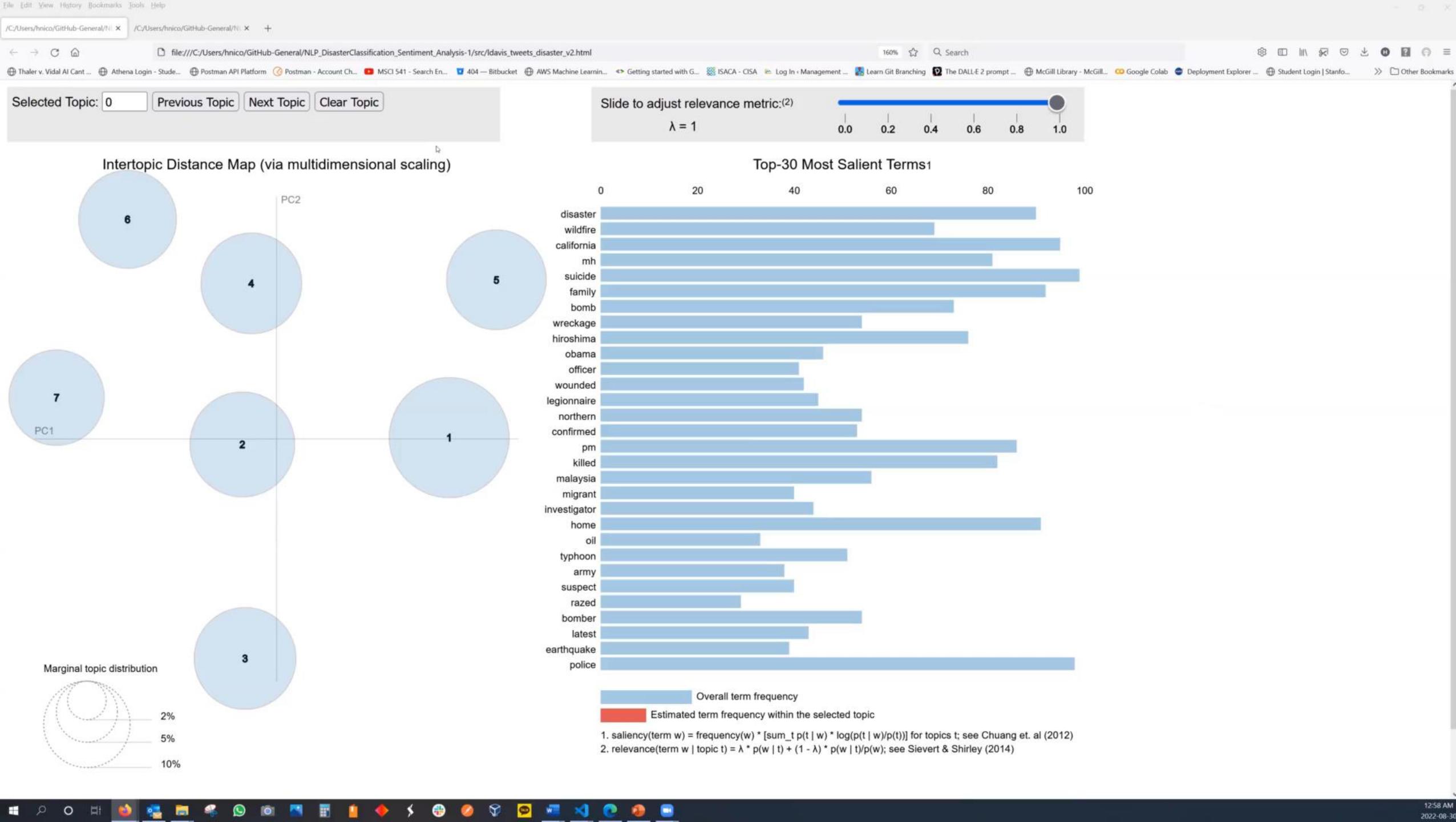
Non-disaster Tweets | Target = 0



LDA Topic Modeling

Latent Dirichlet Allocation (LDA) to determine topics present in Twitter data:





Machine Learning - Training Process

Supervised Machine Learning

Ridge
Logistic Regression
Gaussian Naïve Bayes
SGD Classifier
KNN
Decision Tree
Random Forest
XGBoost
Gradient Boosting
SVC Classifier

Bag of Words (BOW)

TF-IDF
with or without Scaler / PCA

Select high performing models

Hyperparameter Tuning

Deep Learning

MLP
RNN/LSTM

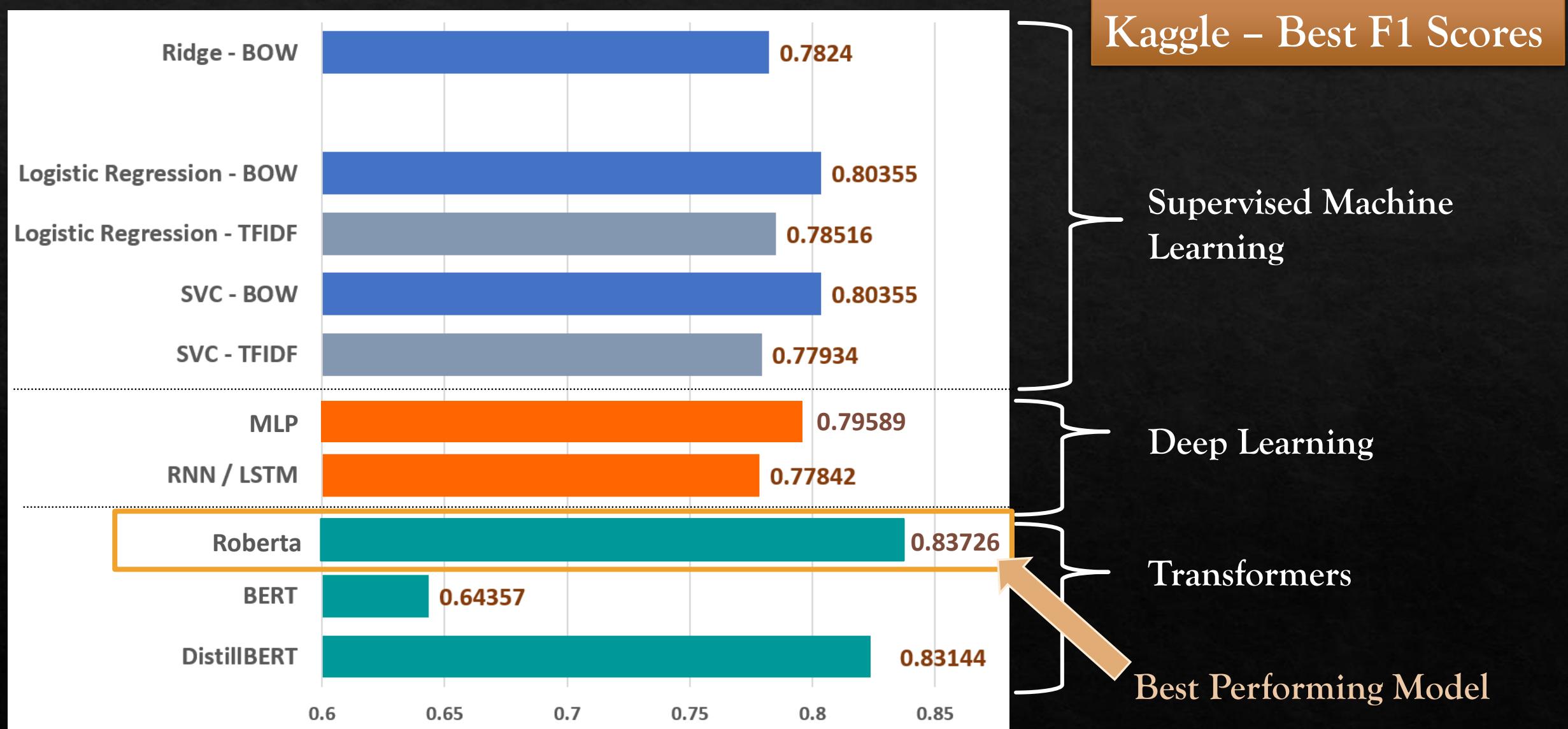
Transformers

BERT
DistilBERT
ROBERTA

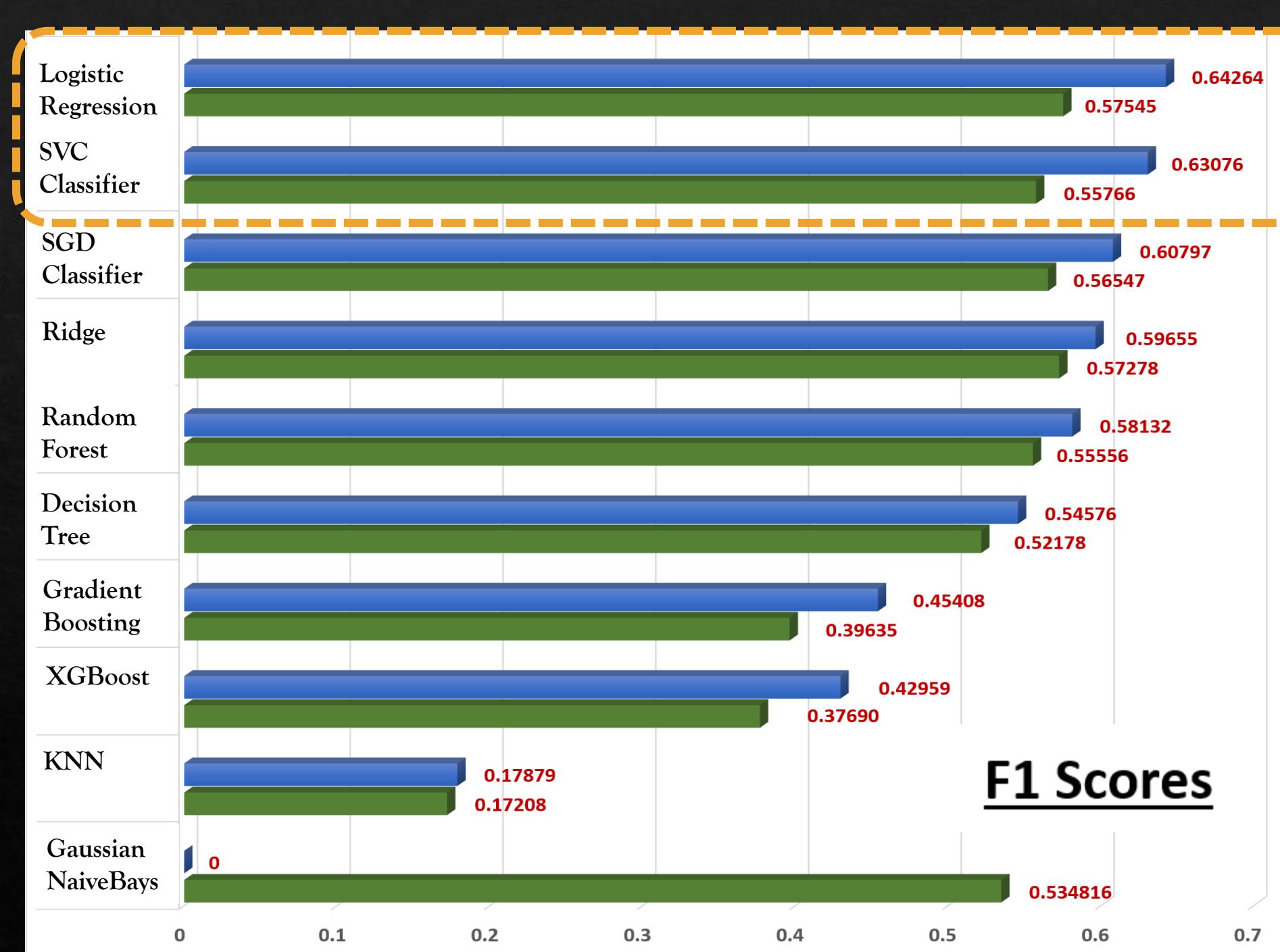
Prediction on Test Dataset

Submission to Kaggle Leaderboard

Machine Learning - Classification Performance



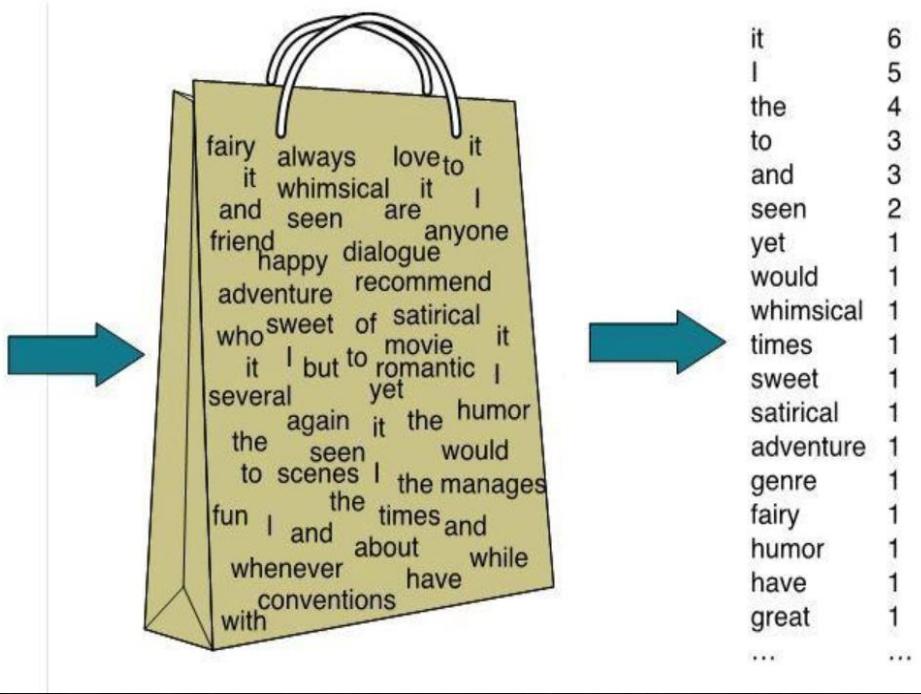
Supervised M/L: Performance Summary



Feature Extraction: Bag of Words (BOW)

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

15



```
from sklearn.feature_extraction.text import CountVectorizer  
  
count_vectorizer = feature_extraction.text.CountVectorizer()  
  
train_vectors = count_vectorizer.fit_transform(train_df['text_preprocessed_2nd'])
```

- Extraction of text features for use in machine learning algorithm, where the representation of text describes the occurrence of words within a document
- Extraction involves: vocabulary of known words & measure of presence of known words
- Converts variable-length texts into a fixed-length vector of numbers

Feature Extraction: TF-IDF

Term Frequency (TF)



Frequency of a word
within the document

Inverse Document Frequency (IDF)



Frequency of a word
across the documents

$$w_{x,y} = tf_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y
 df_x = number of documents containing x
 N = total number of documents

- Numerical metric reflecting the importance of a word in a collection of corpus
- Higher TF-IDF score for frequent words in a document but not across documents

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vect = TfidfVectorizer(ngram_range=(1, 2), max_features=1200)
X = tfidf_vect.fit_transform(train_df['text_preprocessed_2nd'])
target = pd.DataFrame(X.toarray(), columns=tfidf_vect.get_feature_names())
```

$$TF_{x,y} = \frac{f_{x,y}}{n_y}$$

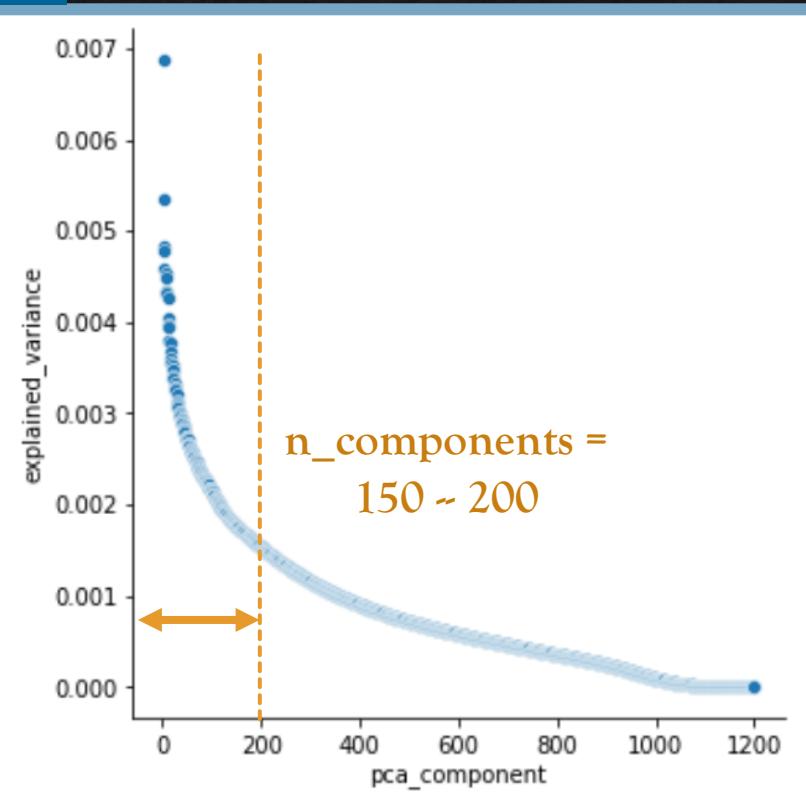
$f_{x,y}$ = frequency of term x in document y
 n_y = total number of words in document y

$$IDF_x = 1 + \log \left(\frac{N}{df_x} \right)$$

Dimensionality Reduction & Data Normalization

Dimensionality Reduction (PCA)

	explained_variance	pca_component	pca_100%
0	6.874733e-03	1	0.6874733
1	5.339239e-03	2	0.5339239
2	4.840314e-03	3	0.4840314
3	4.774374e-03	4	0.4774374
4	4.596527e-03	5	0.4596527
...
1195	6.827159e-36	1196	6.827159e-34
1196	3.230210e-36	1197	3.230210e-34
1197	1.398170e-36	1198	1.398170e-34
1198	1.039025e-36	1199	1.039025e-34
1199	4.694233e-38	1200	4.694233e-36

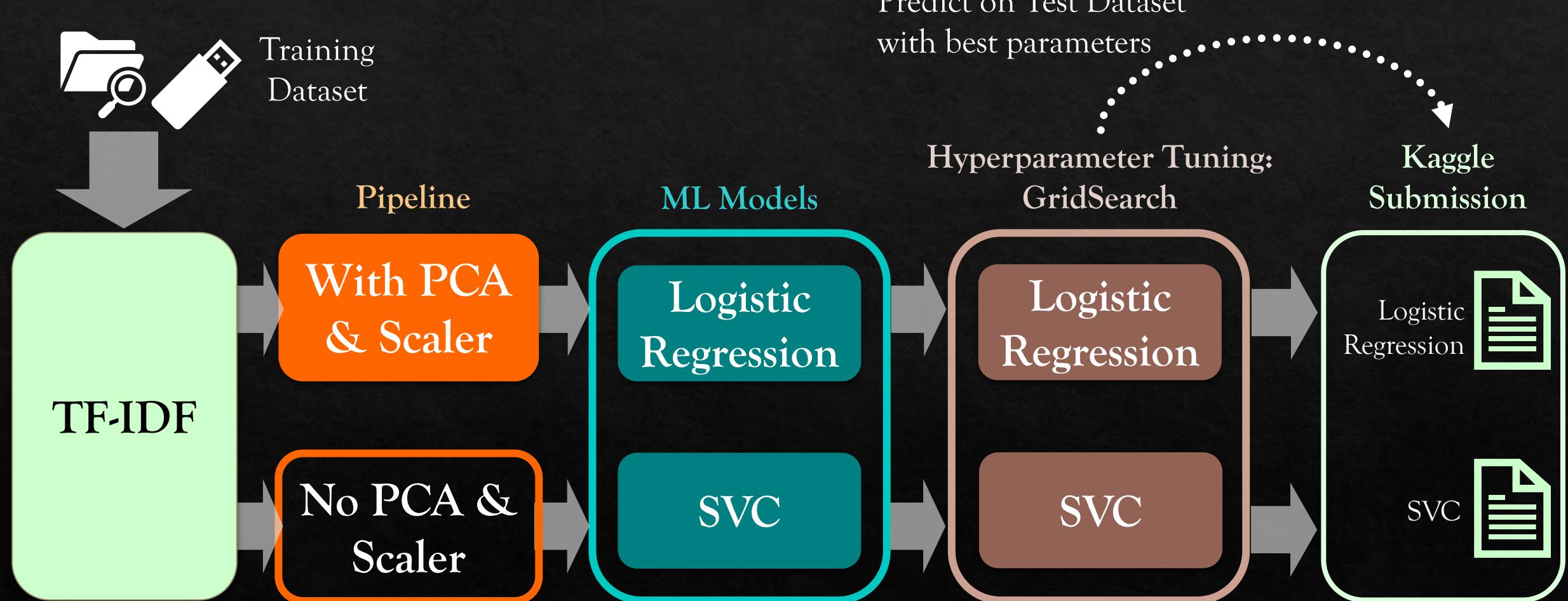


Data Normalization: MinMaxScaler

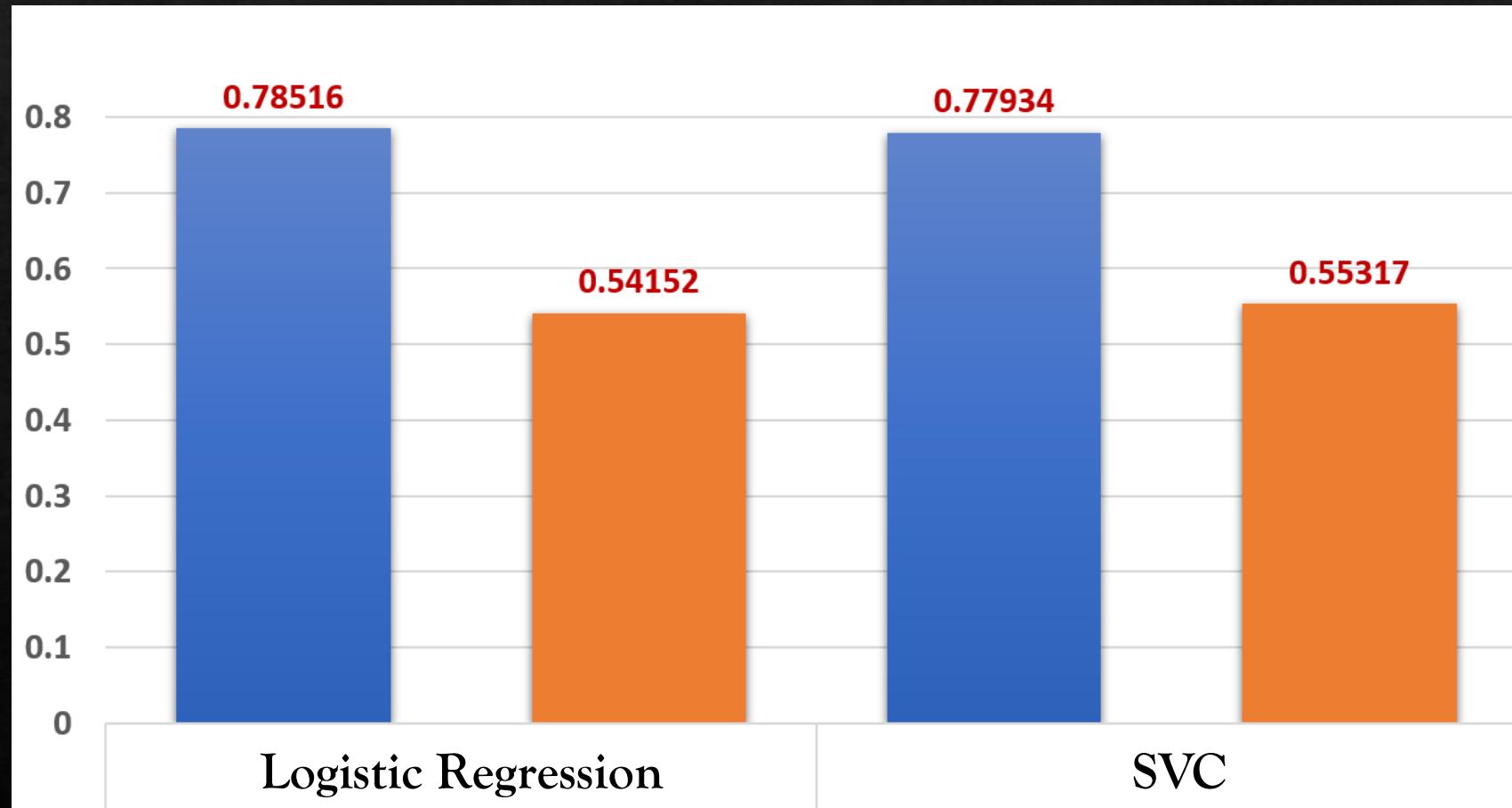
```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

- Normalized Twitter datasets, using the scikit-learn object MinMaxScaler
- Default scale for Normalization: to rescale variables into the range [0,1]

TF-IDF: Scaler and PCA



TF-IDF: Scaler and PCA - Kaggle F1 Scores



Hyperparameter Tuning: Logistic Regression

With PCA & Scaler

2.31 minutes

```
scaler = MinMaxScaler()  
pca = PCA(n_components = 200)
```

```
param_grid = {  
    'solver': [ 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga' ],  
    'penalty': [ 'l2' ],  
    'C': [ 1e-05, 0.001, 0.1, 1.0, 10.0, 100.0 ],  
    'random_state': [ 10, 25, 35, 42, 53 ] }
```

```
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3)
```

```
grid_lg = GridSearchCV(clf_lg, param_grid, n_jobs = -1,  
                      scoring='f1', cv=cv, verbose = 3)
```

```
clf_lg2 = LogisticRegression(C=100, penalty='l2',  
                            random_state=35, solver='saga')
```

Kaggle F1 Score

0.54152

No PCA & Scaler

8.42 minutes

```
scaler → None  
pca → None
```

```
param_grid = {  
    'solver': [ 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga' ],  
    'penalty': [ 'l2' ],  
    'C': [ 1e-05, 0.001, 0.1, 1.0, 10.0, 100.0 ],  
    'random_state': [ 10, 25, 35, 42, 53 ] }
```

```
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3)
```

```
grid_lg = GridSearchCV(clf_lg, param_grid, n_jobs = -1,  
                      scoring='f1', cv=cv, verbose = 3)
```

```
clf_lg3 = LogisticRegression(C=10.0, penalty='l2',  
                            random_state=42, solver='sag')
```

Kaggle F1 Score

0.78516

Hyperparameter Tuning: SVC

With PCA & Scaler

2.33 hours

```
scaler = MinMaxScaler()  
pca = PCA(n_components = 200)
```

```
param_grid = {  
    'C': [ 1e-05, 0.001, 0.1, 1.0, 10.0, 100.0 ],  
    'gamma': [ 1, 0.1, 0.01, 0.001, 0.0001 ],  
    'kernel': [ 'linear', 'rbf' ],  
    'max_iter': [ -1 ],  
    'random_state': [ 10, 25, 35, 42, 53 ] }
```

```
grid_svc = GridSearchCV( clf_svc, param_grid,  
                        refit = True, verbose = 3)
```

```
clf_svc = SVC( C=100, gamma = 0.01, kernel = 'rbf',  
               max_iter = -1, random_state=10 )
```

Kaggle F1 Score

0.55317

No PCA & Scaler

15.78 minutes

```
scaler → None  
pca → None
```

```
param_grid = {  
    'C': [ 1e-05, 0.001, 0.1, 1.0, 10.0, 100.0 ],  
    'gamma': [ 1, 0.1, 0.01, 0.001, 0.0001 ],  
    'kernel': [ 'linear', 'rbf' ],  
    'max_iter': [ -1 ],  
    'random_state': [ 10, 25, 35, 42, 53 ] }
```

```
grid_svc2 = HalvingGridSearchCV( clf_svc, param_grid,  
                                 refit = True, verbose = 3)
```

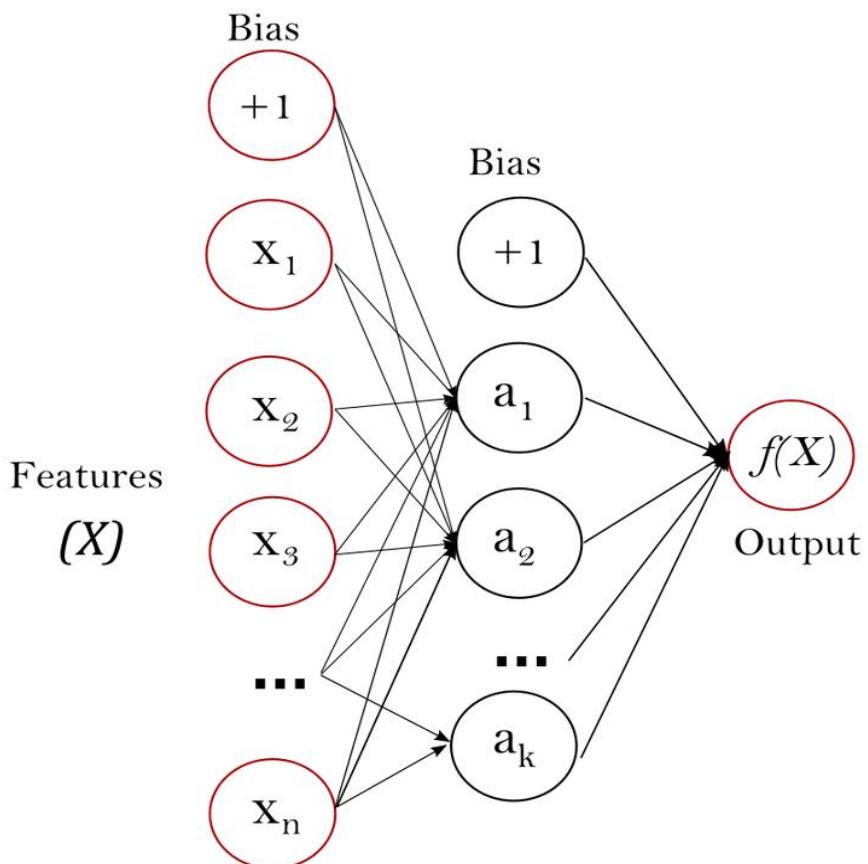
```
clf_svc2 = SVC( C=0.1, gamma = 0.1, kernel = 'linear',  
                max_iter = -1, random_state=10 )
```

Kaggle F1 Score

0.77943

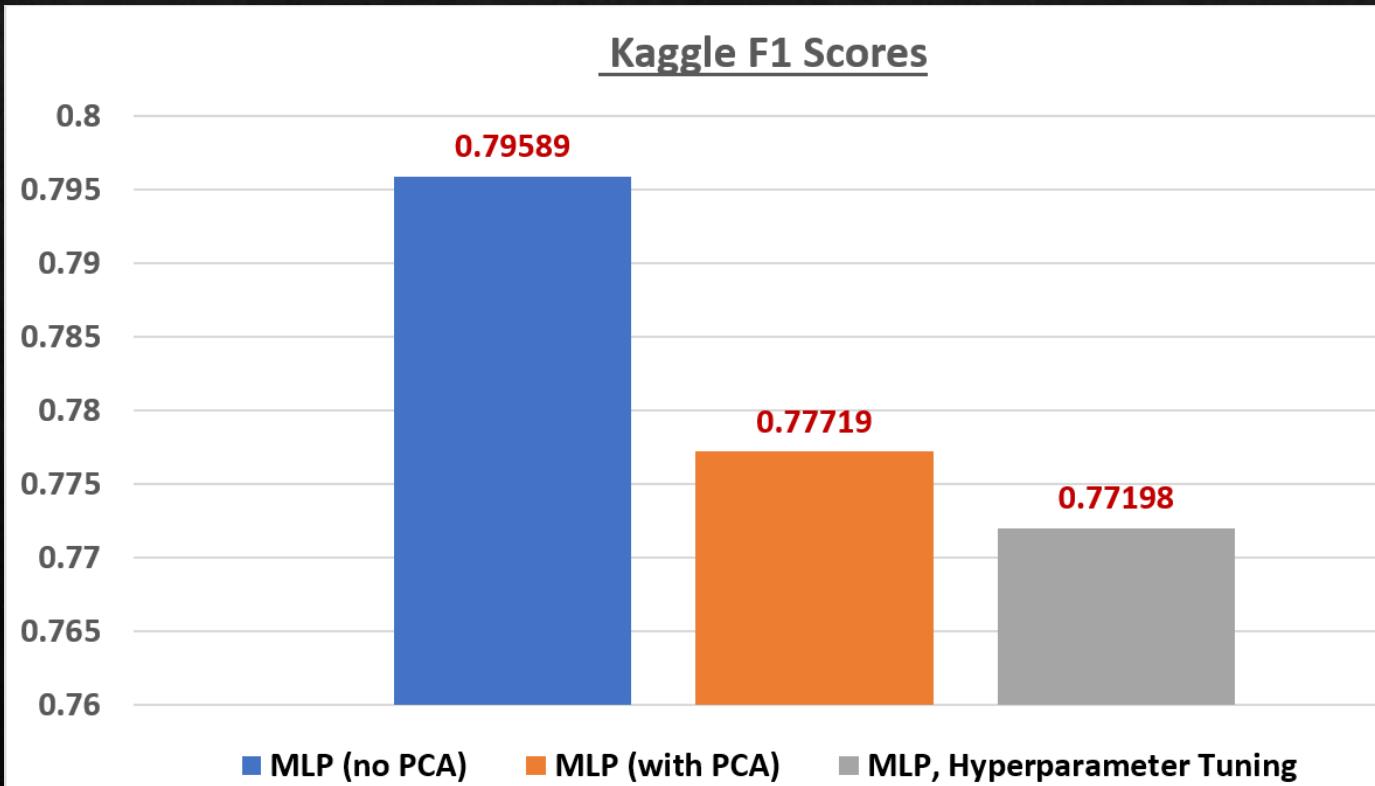
Deep Learning: MLP

```
from sklearn.neural_network import MLPClassifier  
from sklearn.datasets import make_classification  
  
mlp2=MLPClassifier(random_state=0, early_stopping=True, max_iter=100, verbose=2)
```

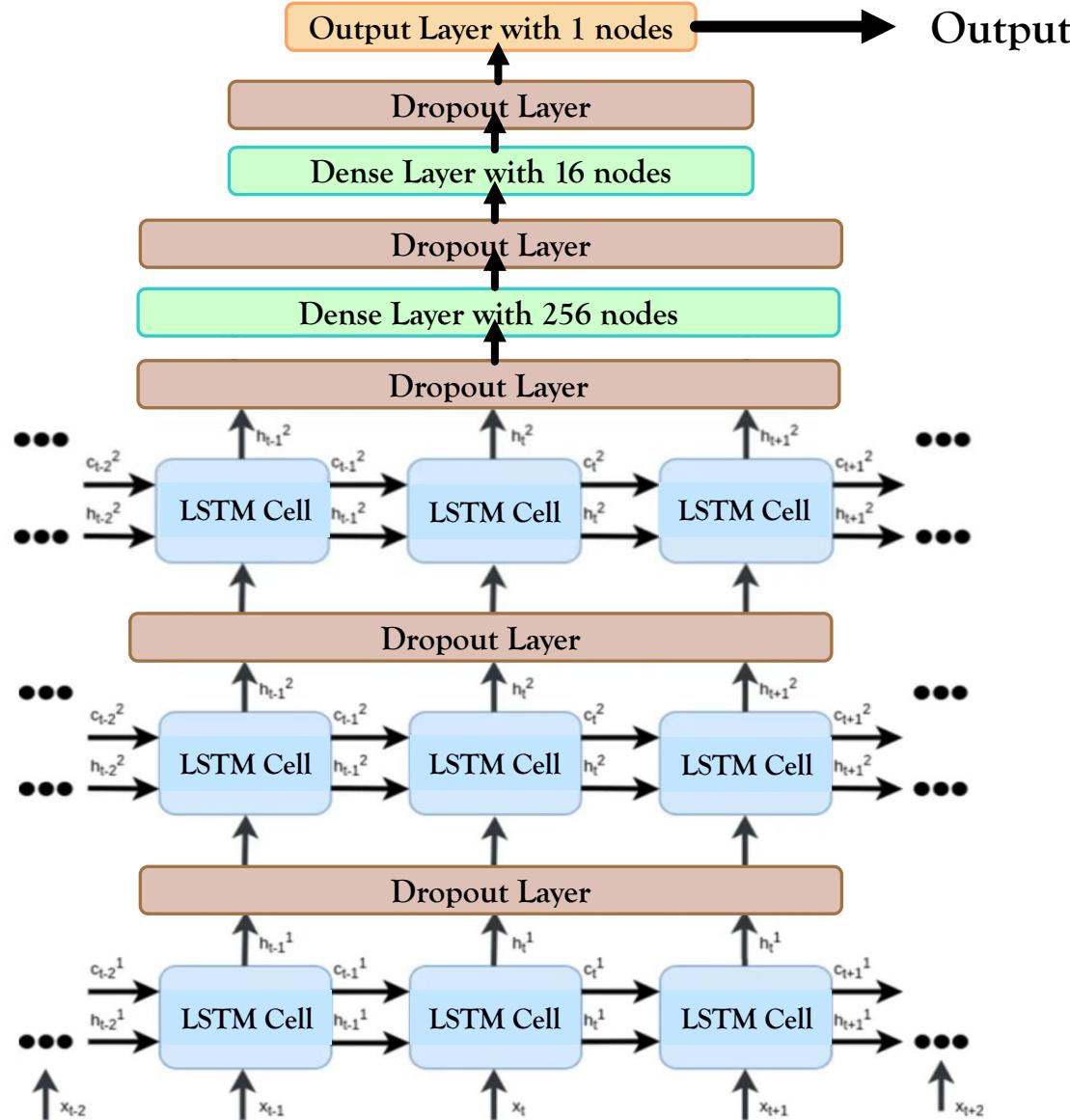


Multi-layer Perceptron (MLP): supervised learning algorithm that learns a function, $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where:

- m = the number of dimensions for input
- o = the number of dimensions for output



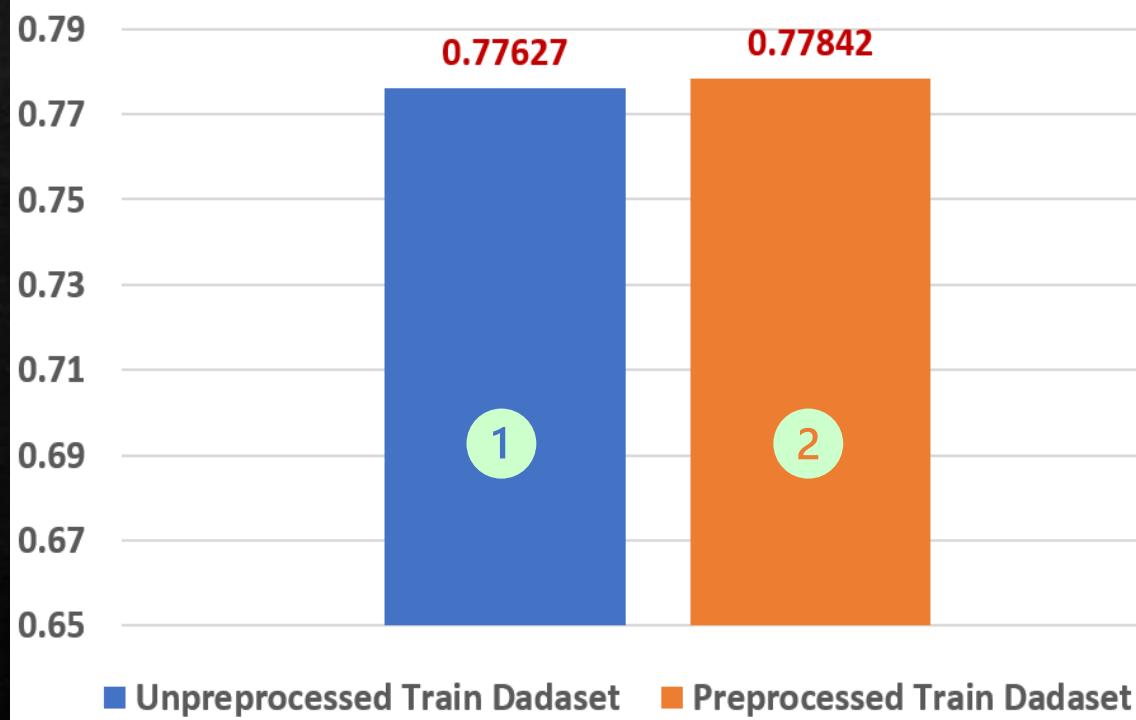
Deep Learning: Bidirectional LSTM based RNN



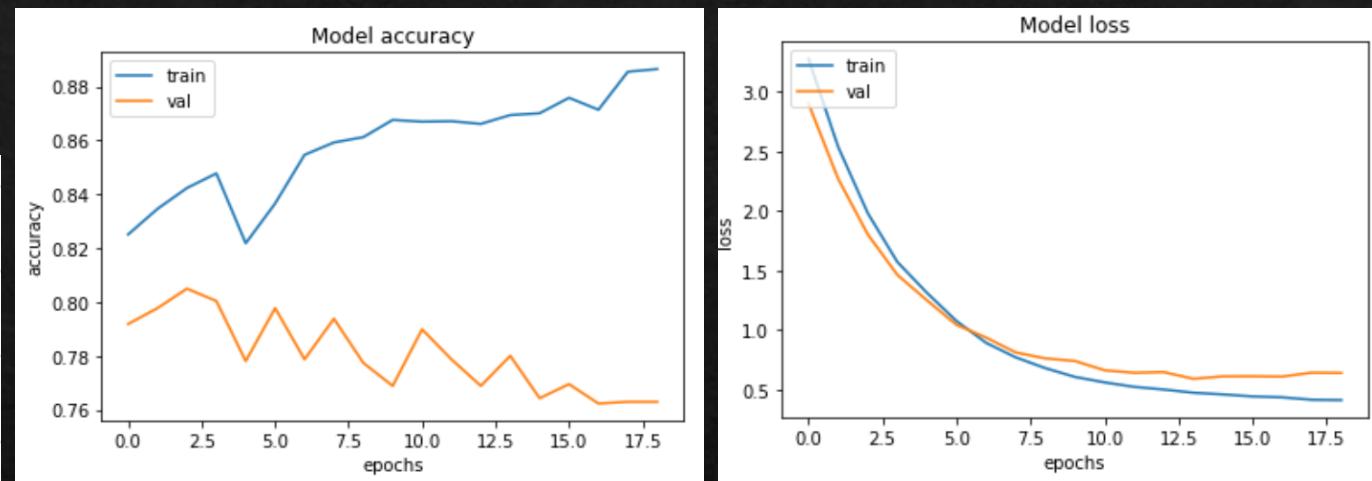
CPU times: user 14 µs, sys: 0 ns, total: 14 µs Wall time: 28.6 µs Model: "Bidirectional_RNN"		
Layer (type)	Output Shape	Param #
embedding_7 (Embedding)	(None, 256, 256)	256000
bidirectional (Bidirectional) 1)	(None, 256, 256)	394240
dropout (Dropout)	(None, 256, 256)	0
bidirectional_1 (Bidirectional) 1)	(None, 256, 128)	164352
dropout_1 (Dropout)	(None, 256, 128)	0
bidirectional_2 (Bidirectional) 1)	(None, 64)	41216
dropout_2 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 256)	16640
dropout_3 (Dropout)	(None, 256)	0
dense_8 (Dense)	(None, 16)	4112
dropout_4 (Dropout)	(None, 16)	0
dense_9 (Dense)	(None, 1)	17
<hr/>		
Total params:	876,577	
Trainable params:	876,577	
Non-trainable params:	0	

Deep Learning: Bidirectional LSTM based RNN

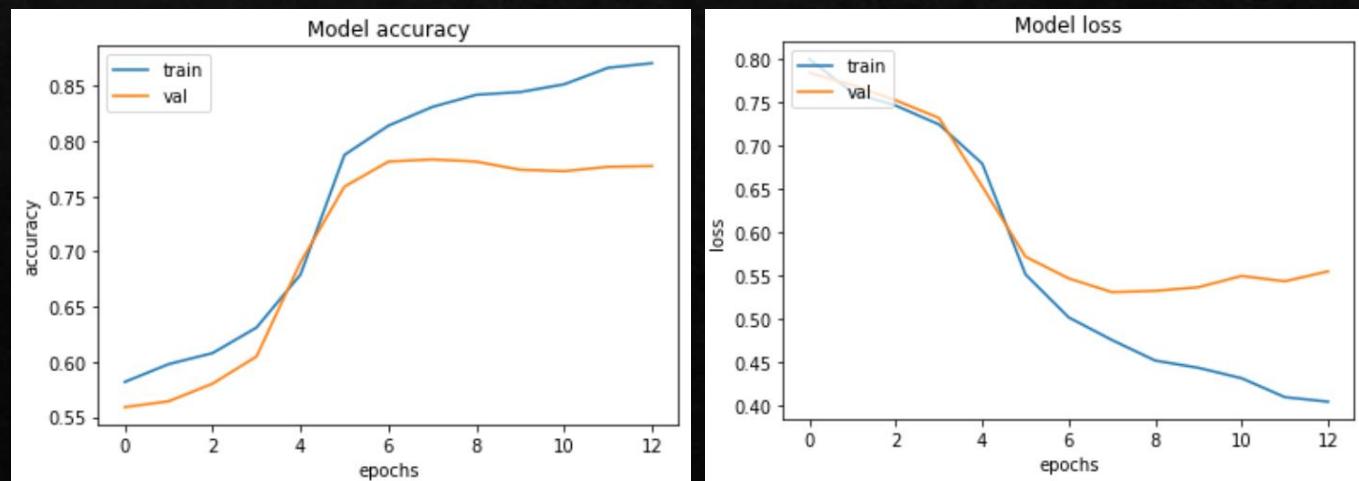
Kaggle F1 Scores



1 Trained on Unpreprocessed Train Dataset



2 Trained on Preprocessed Train Dataset



Transformer-Based Models

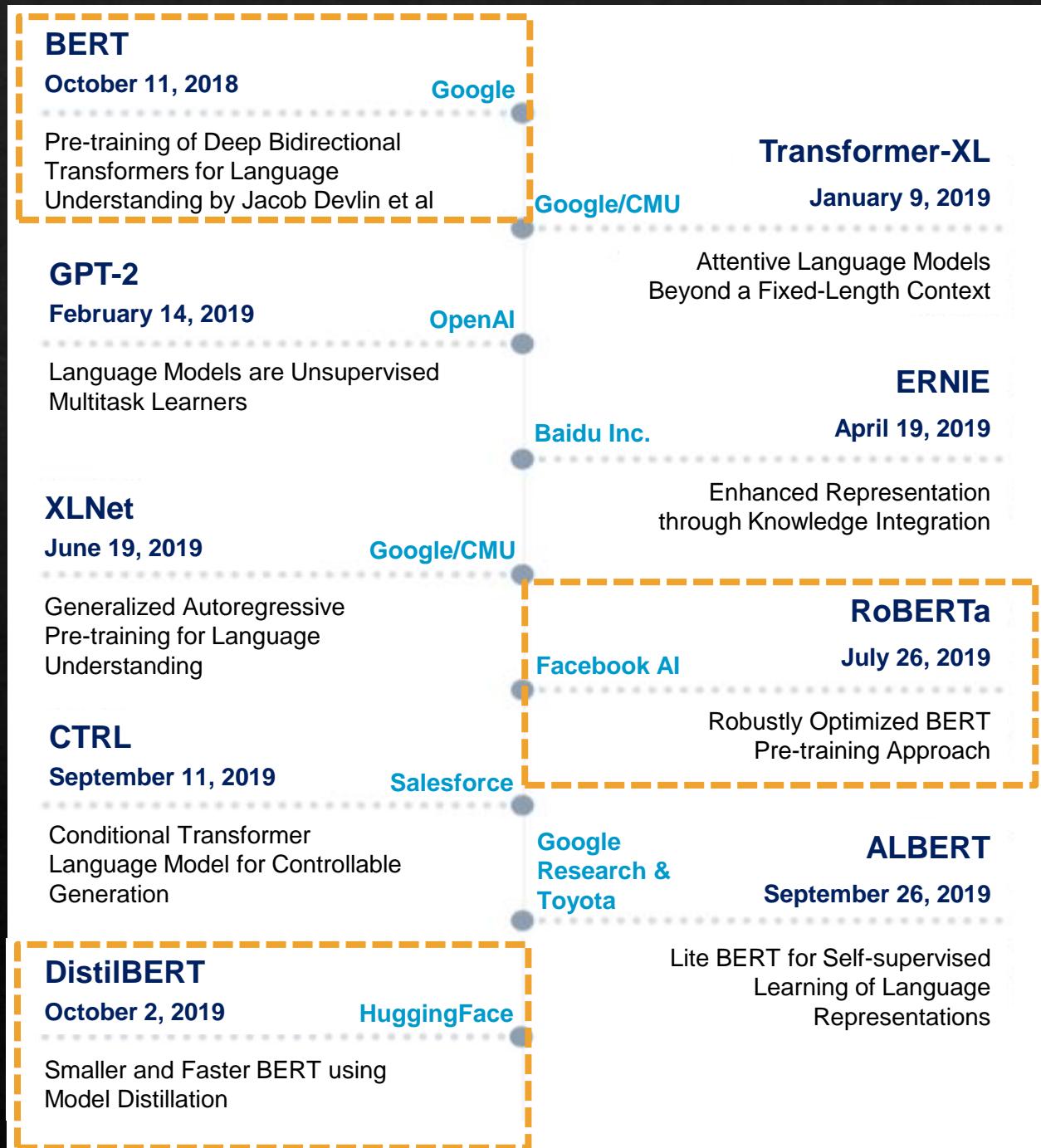
Bert and Ernie Differ on Relationship Status and Twitter Goes Wild



Timeline for Projects After BERT

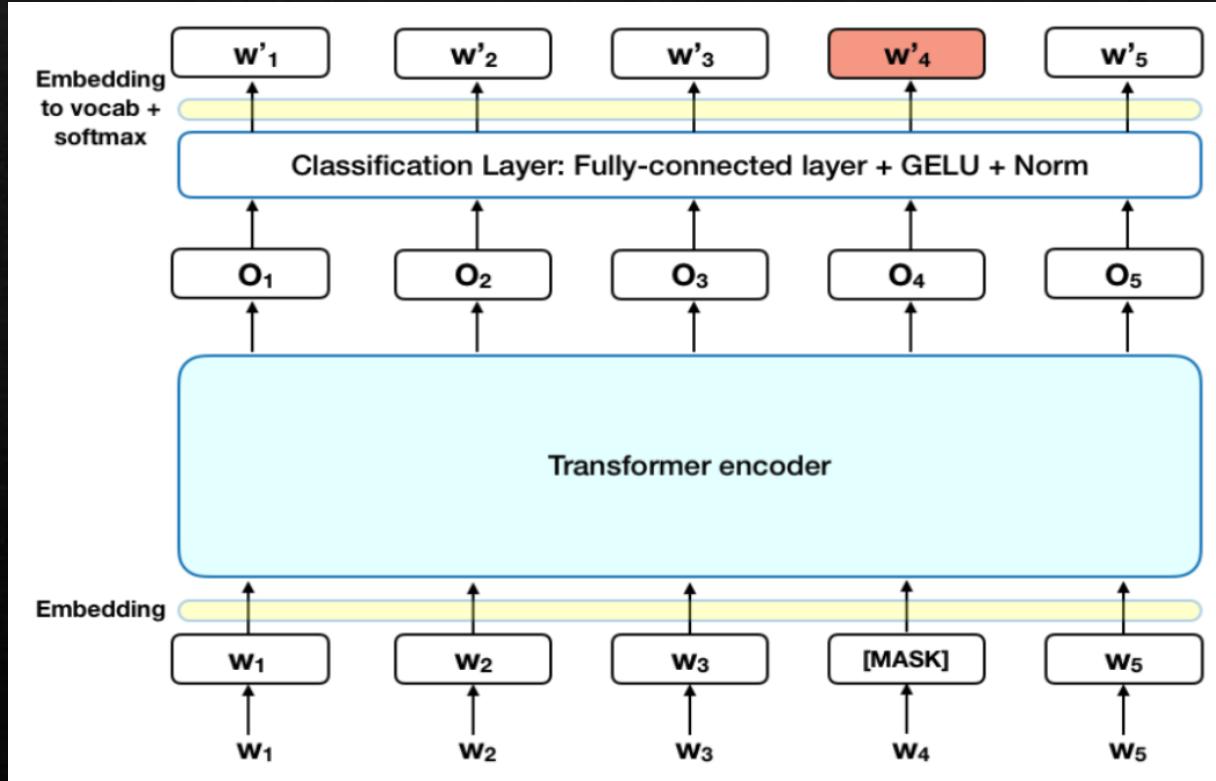
List of Transformers:

<https://github.com/huggingface/transformers>



Transformer: BERT

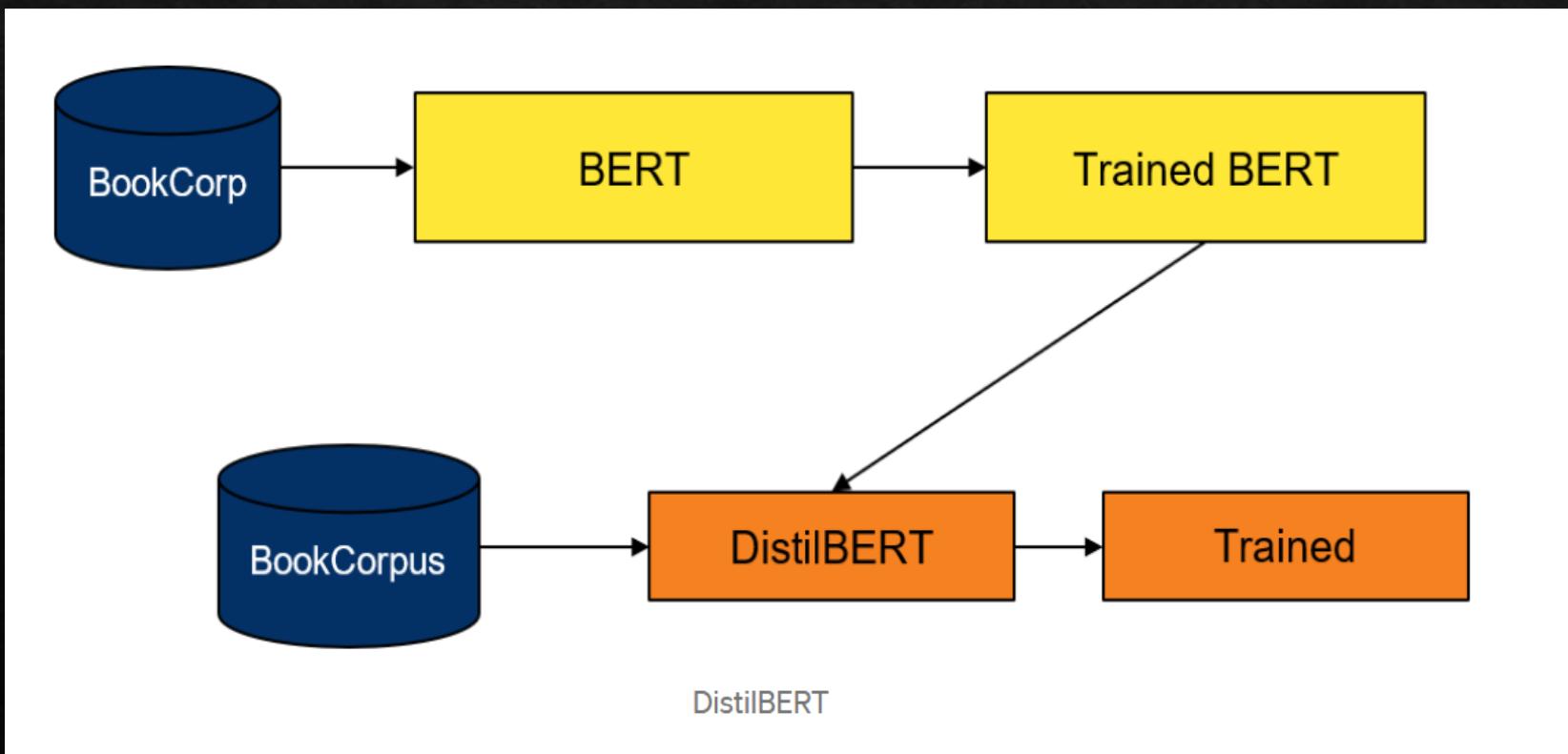
Bidirectional Encoder Representations from Transformers (BERT): transformer-based architecture, ‘designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers’



- Fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks
- Two different architectures: BERT_Base & BERT_Large
- GLUE Score (at GLUE Benchmark) 80.5%

Transformer: DistilBERT

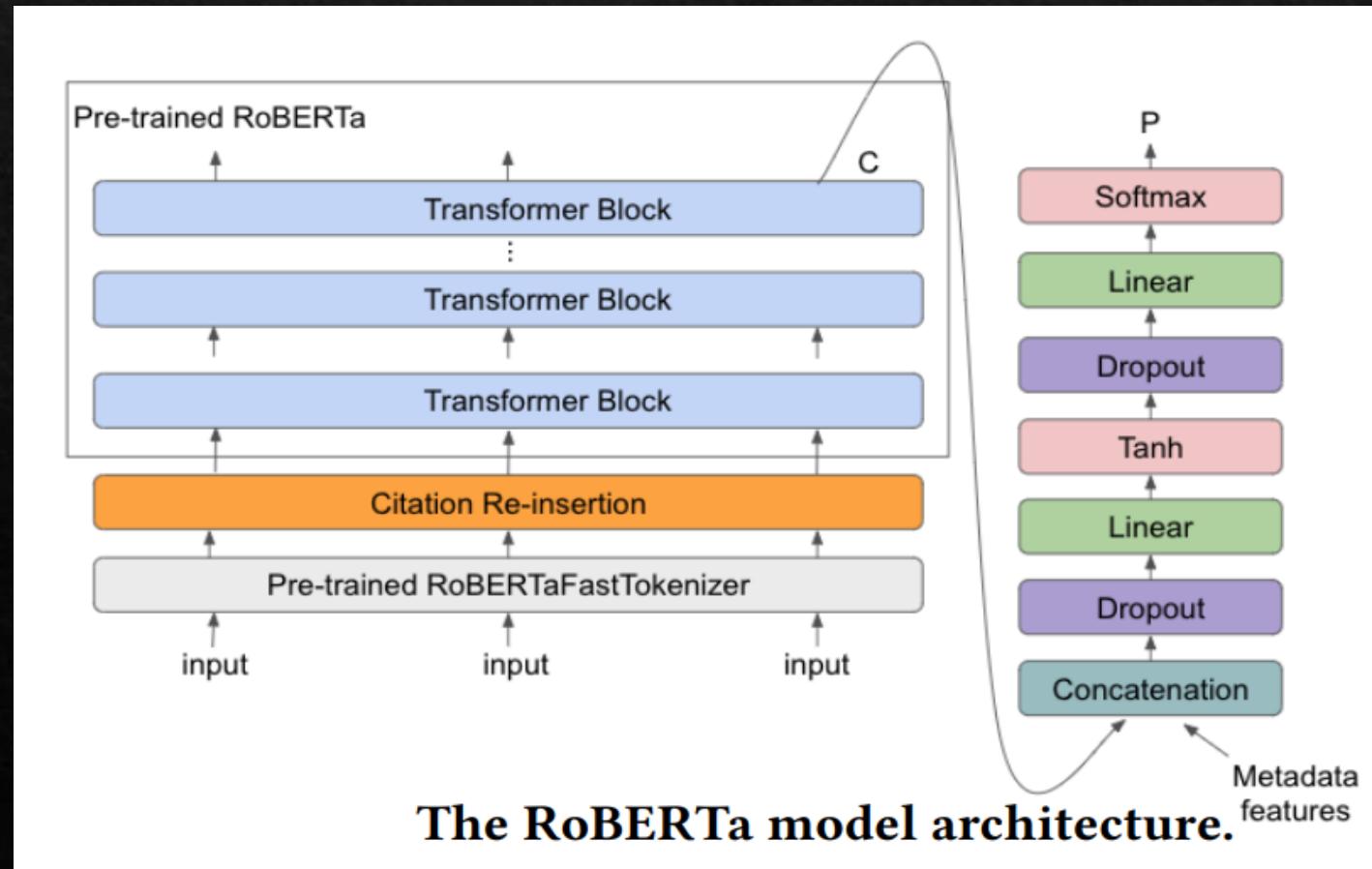
BERT Distillation: smaller version of BERT developed and open-sourced by the team at HuggingFace



- 40% less parameters than BERT-base-uncased, runs 60% faster, preserving over 95% of BERT's performances
- Implementation on TensorFlow/Keras, Trained DistilBERT used to generate sentence embedding
- Basic NN Architecture (with Dense and Dropout layers)

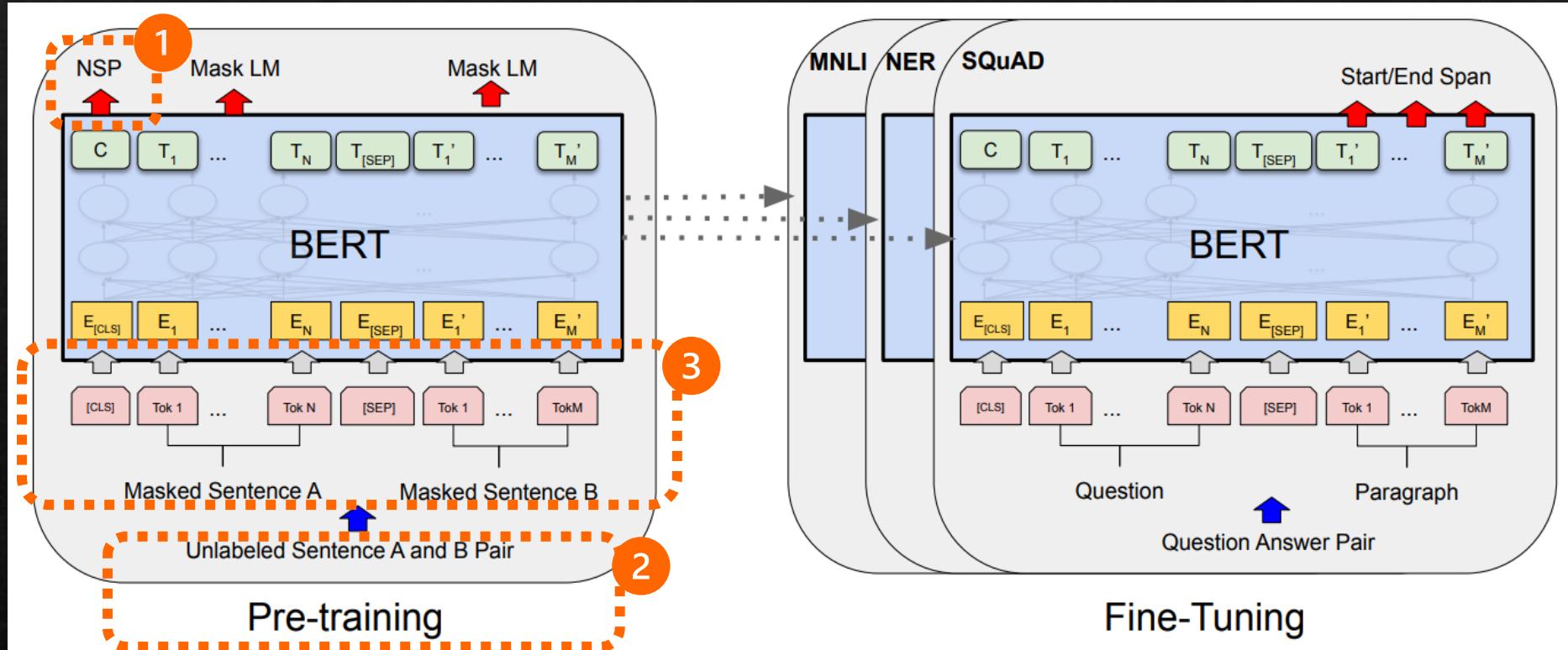
Transformer: RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa): developed by Facebook AI & Washington University. Improves on the BERT model by modifying key hyperparameters and pretraining on a larger corpus



- Modifications to BERT with the following changes:
 - ➔ Removing the Next Sentence Prediction (NSP) objective
 - ➔ Training with bigger batch sizes & longer sequences
 - ➔ Dynamically changing the masking pattern
- Datasets used to train:
 - ➔ BOOK CORPUS and English Wikipedia dataset (16 GB)
 - ➔ CC-NEWS (76 GB)
 - ➔ OPENWEBTEXT (38 GB)
 - ➔ STORIES (31 GB)

BERT Transformer Architecture



BERT vs RoBERTa

1 Removing the Next Sentence Prediction (NSP) objective from BERT

2 Training with bigger batch sizes & longer sequences:

- Batch Size from 256 to 2,048
- Using Gradient Accumulation for low memory resources

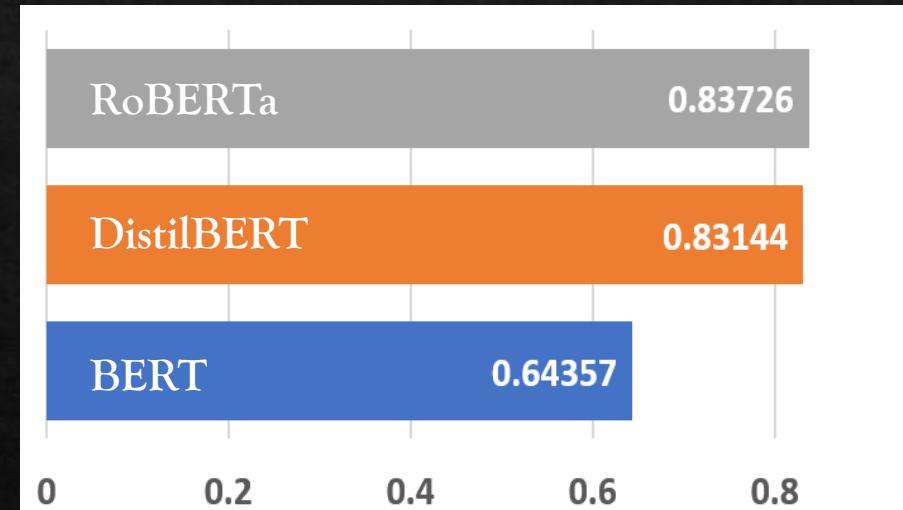
3 Dynamically changing the masking pattern

Model	Parameters (in millions)	GLUE average	Key points
BERT	108	30k Vocab	MLM, NSP
RoBERTa	123	50k Vocab	MLM, larger batch size, larger vocabulary

Transformers: Comparison

	BERT	RoBERTa	DistilBERT
Size (millions)	Base: 110 Large: 340	Base: 110 Large: 340	Base: 66
Training Time	Base: 8 x V100 x 12 days* Large: 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	Large: 1024 x V100 x 1 day; 4-5 times more than BERT.	Base: 8 x V100 x 3.5 days; 4 times less than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation

Kaggle F1 Scores



Easy deployment of the model for a variety of NLP tasks with a few lines of code through the transformers library by Hugging Face (<https://huggingface.co/docs/transformers/index>)

#	Team	Members	Score	Entries	Last	Code	Join
1	Andrej Marinchenko		1.00000	270	17h		
2	Moshi Wei		1.00000	1	2mo		
3	Jewel Liu		1.00000	2	2mo		
4	DinoKing		1.00000	1	2mo		
5	Wonhyeong Seo		1.00000	1	1mo		
6	💥Tesla, Inc.💥		1.00000	1	1mo		

52	nicsli		0.83726	9	1mo		
----	--------	---	---------	---	-----	--	--

53	Nicole Hong		Total 718 Teams	0.83726	2	23d	
----	-------------	---	-----------------	---------	---	-----	--



Your Best Entry!

Your most recent submission scored 0.83726, which is an improvement of your previous score of 0.83144. Great job!

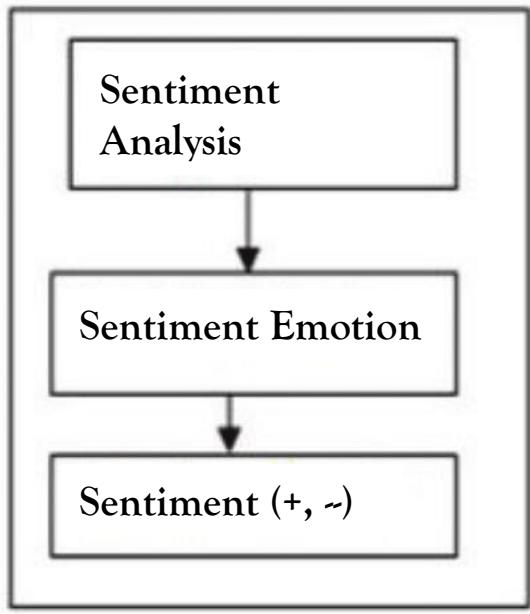
[Tweet this](#)

54	rookie #2		0.83665	5	7d		
----	-----------	---	---------	---	----	--	--

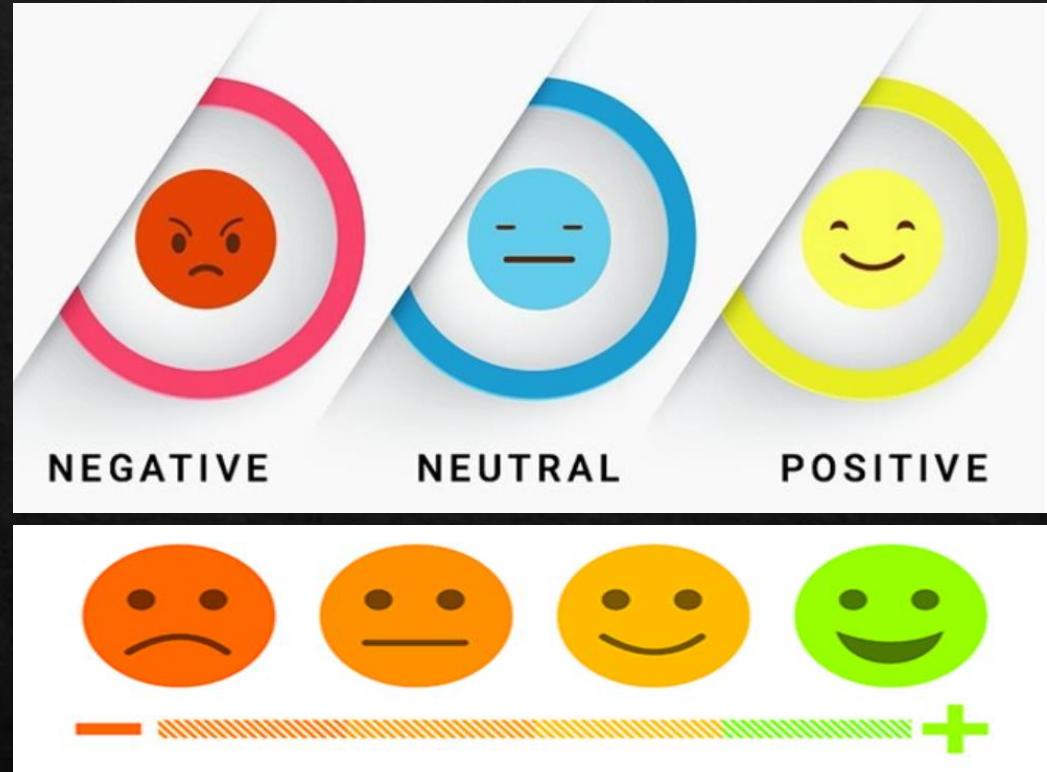
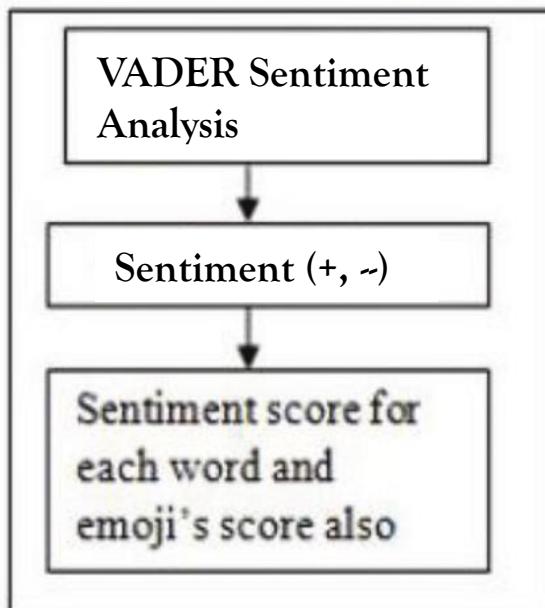
VADER Sentiment Analysis

Valence Aware Dictionary and sEntiment Reasoner (VADER): lexicon and rule-based sentiment analysis tool, specifically attuned to sentiments expressed in social media

Sentiment Analysis

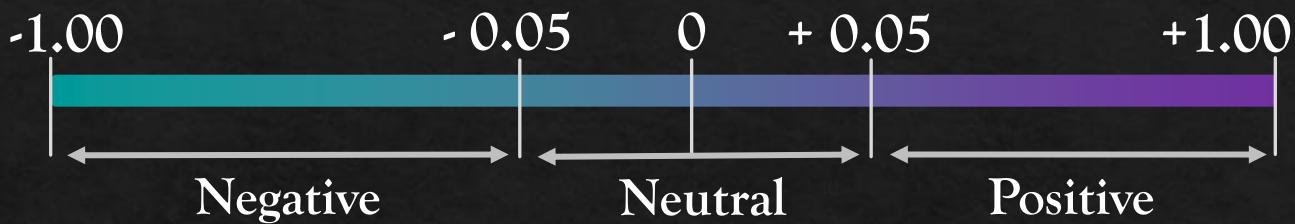


VADER Sentiment Analysis

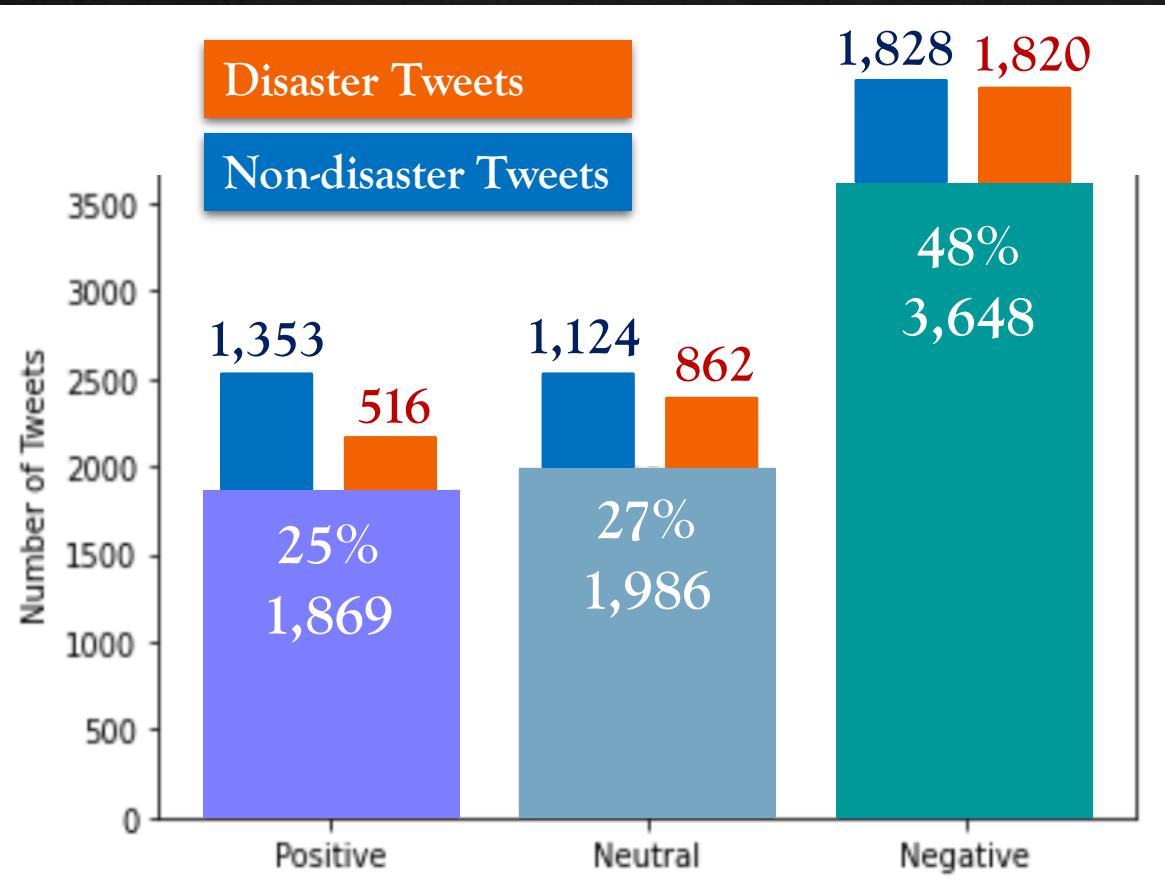


- Uses a combination of A sentiment lexicon in a list of lexical features
- Provides Positive, Negative, Neutral Scores of the sentence

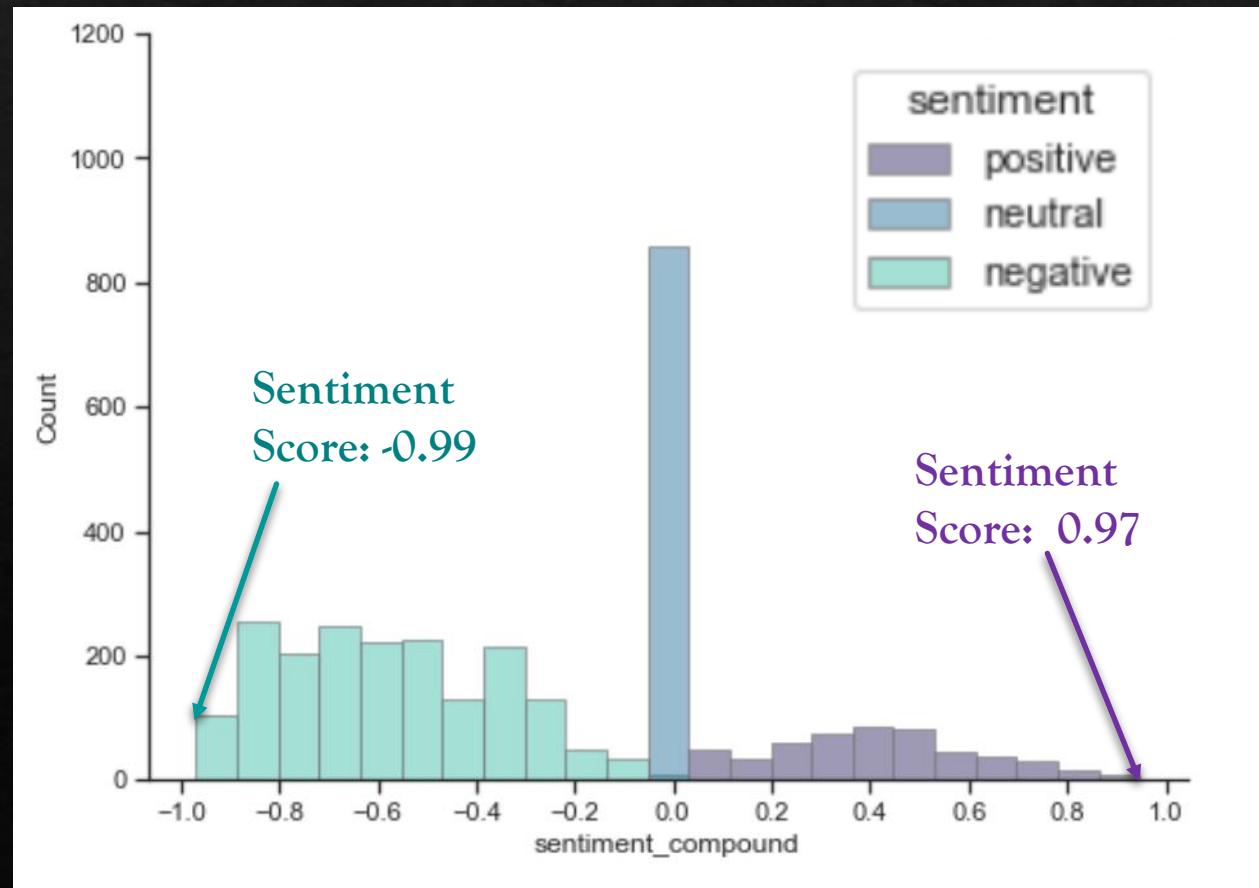
Sentiment Analysis: Result Summary



Counts of Sentiment Classes in All Tweets

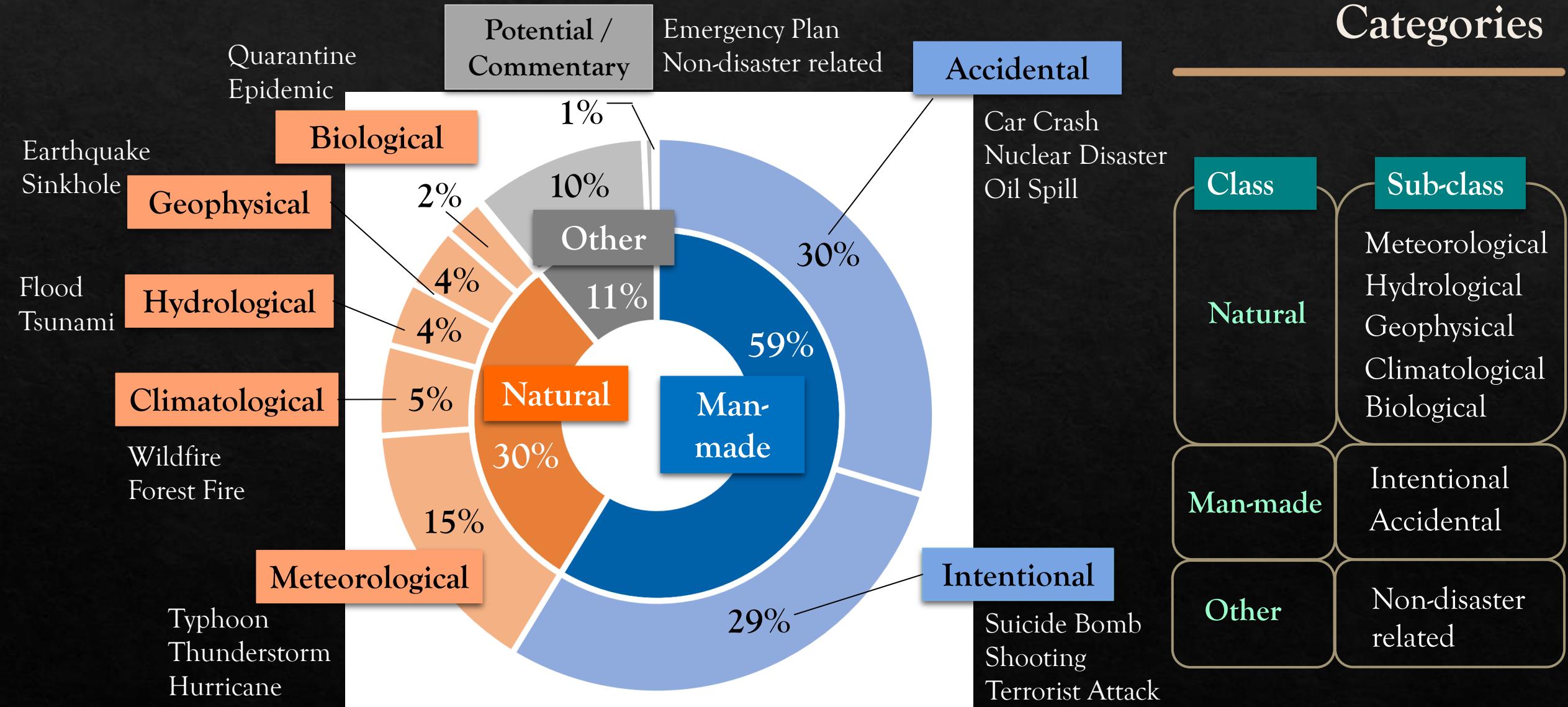


Sentiment Score Distribution – Disaster Tweets

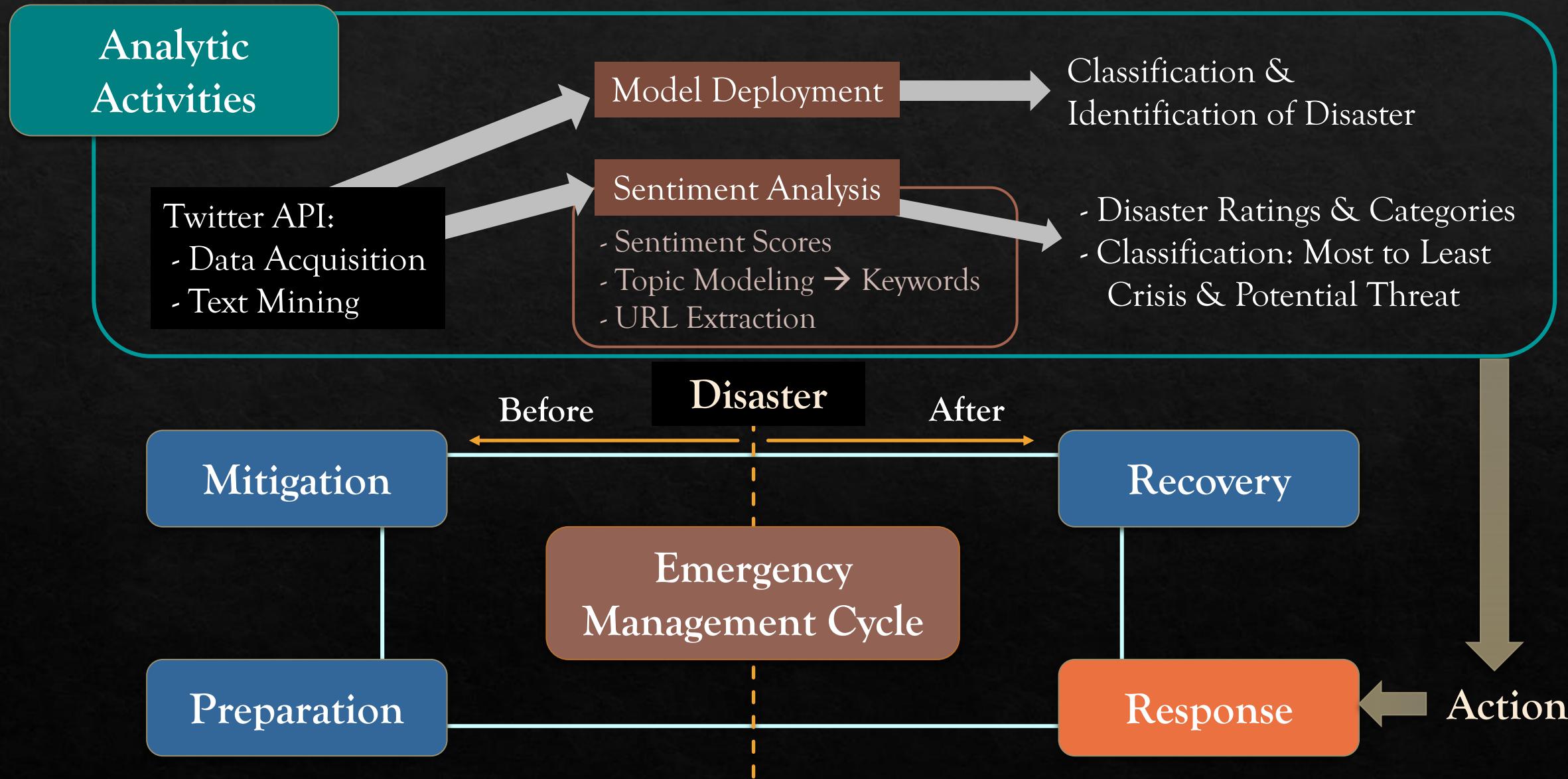


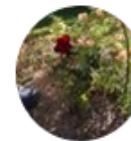
Disaster Tweets

Disaster Class & Categories



Emergency Response Management





Rebecca · Jul 30, 2021

@PolarStarRose · [Follow](#)

❤️ [@elonmusk](#) ❤️ Thank you for the follow. I'm delighted and honored! 😊



Elon Musk ✅

@elonmusk · [Follow](#)

Sorry, accidental tap!

3:47 PM · Jul 30, 2021



201K



Reply



Copy link

✉ n.hong@queensu.ca

🔗 www.linkedin.com/in/nicolehhong

🔗 <https://github.com/Nicole-Hong>

Thank You!