Classifying Disaster Tweets

The problem

Company

disaster relief organizations / news agencies

Context

- Twitter has become an important communication channel in times of emergency.
- The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time.
- agencies are interested in programatically monitoring Twitter.

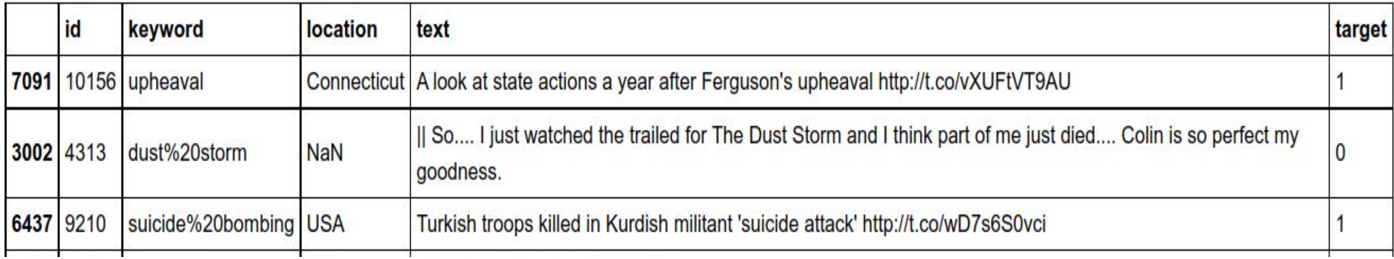
Problem statement

- It's not always clear whether a person's words are actually announcing a disaster.

Data Wrangling and Exploratory Data Analysis (EDA)

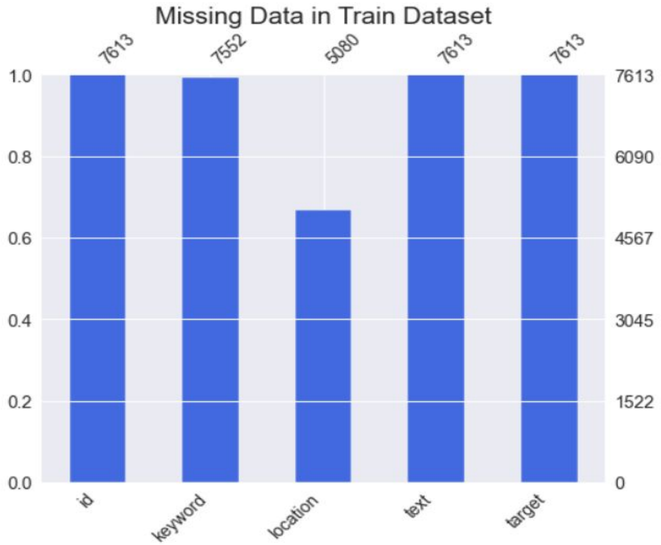
- The data set contains Train and Test sets and taken from Kaggle
  https://www.kaggle.com/c/nlp-getting-started/data

- Train set has 7613 instances, 4 features, binary target variable.

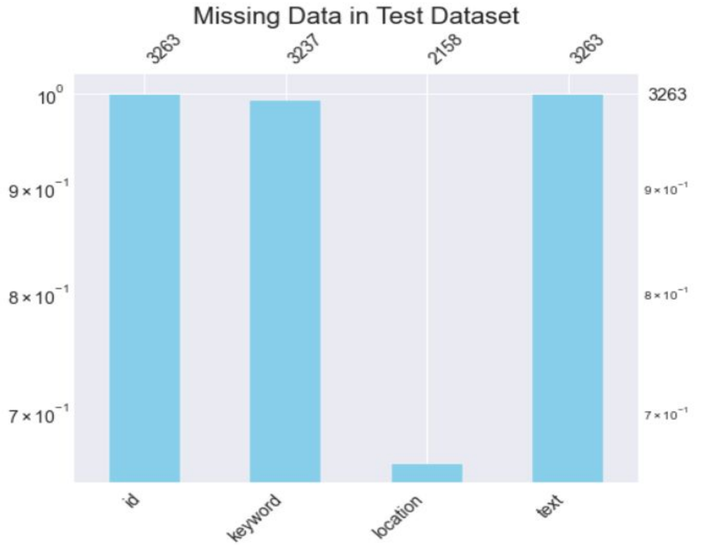


	id	keyword	location	text	target
7091	10156	upheaval	Connecticut	A look at state actions a year after Ferguson's upheaval http://t.co/vXUFtVT9AU	1
3002	4313	dust%20storm	NaN	|| So.... I just watched the trailed for The Dust Storm and I think part of me just died.... Colin is so perfect my goodness.	0
6437	9210	suicide%20bombing	USA	Turkish troops killed in Kurdish militant 'suicide attack' http://t.co/wD7s6S0vci	1

- Test set has 3265, same features but no target variable.
- Missing values in both sets for (location and keyword features)



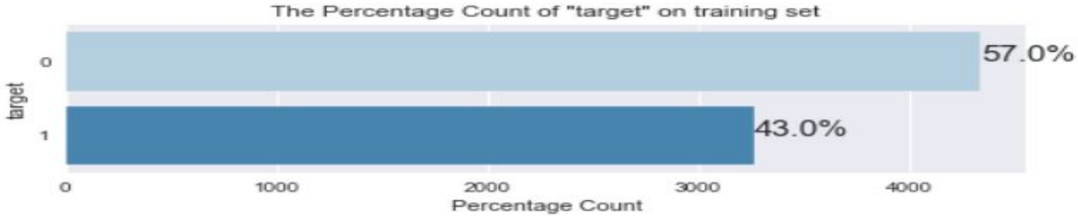
Missing Data in Train Dataset



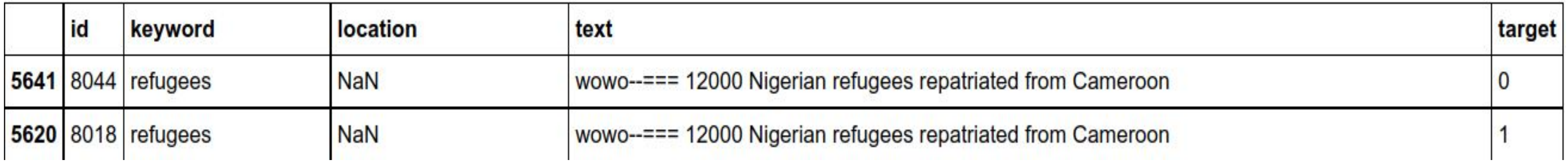
Missing Data in Test Dataset

Inspecting Train Data

- Target Class Distribution

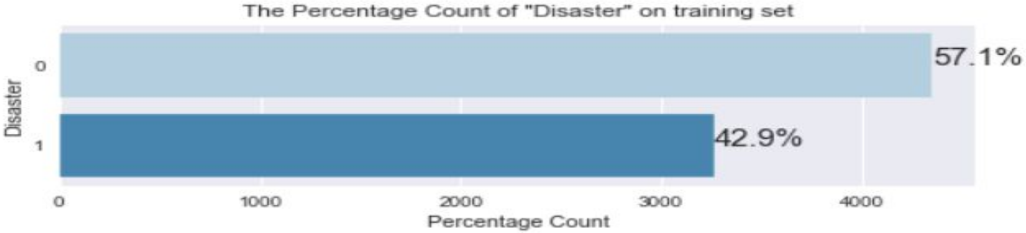


- There are 110 text duplicates and 18 has labeling inconsistency
- Duplicates dropped and inconsistent labels fixed manually



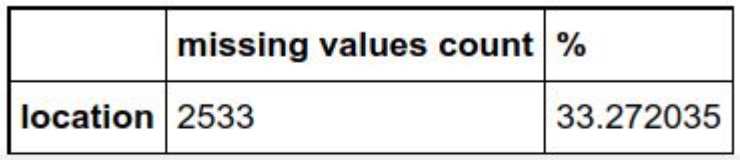
	id	keyword	location	text	target
5641	8044	refugees	NaN	wowo--=== 12000 Nigerian refugees repatriated from Cameroon	0
5620	8018	refugees	NaN	wowo--=== 12000 Nigerian refugees repatriated from Cameroon	1

- Target Class Distribution after cleaning duplicates.

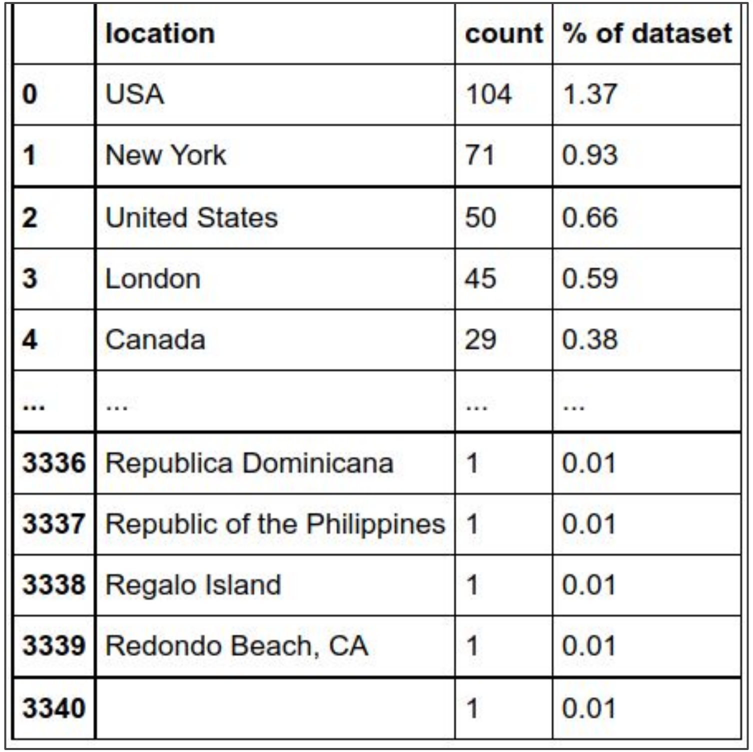


Inspecting the "location" feature

- 33% of the location feature is missing.
- There are 3341 unique locations for the tweets.
- The naming of location in this data set is not consistent. We see some tweets' location as a country and others as a city.
- the USA is the most frequent location for the tweets, followed by New York, then united states representing (1.37%, 0.93%, and 0.66%) of the tweets respectively.
- **With this large number of missing values, it is not feasible to clean the location values. The location in this data set is not good to be predictive feature and will not be considered in the modeling.**



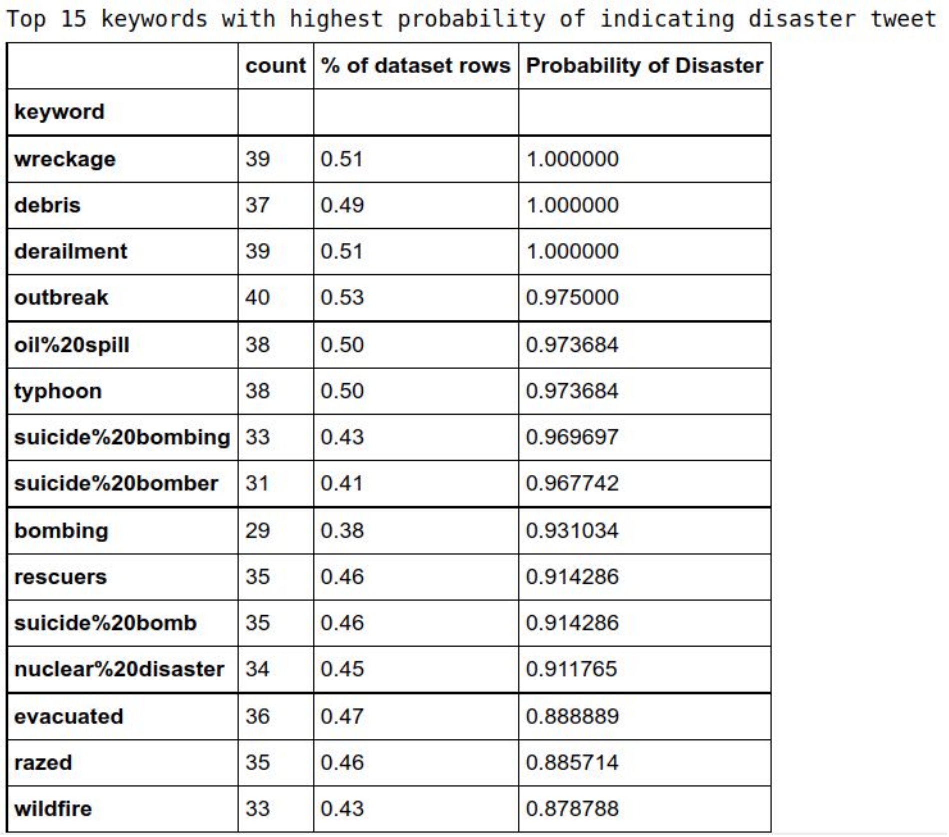
	missing values count	%
location	2533	33.272035



	location	count	% of dataset
0	USA	104	1.37
1	New York	71	0.93
2	United States	50	0.66
3	London	45	0.59
4	Canada	29	0.38
...	...	...	...
3336	Republica Dominicana	1	0.01
3337	Republic of the Philippines	1	0.01
3338	Regalo Island	1	0.01
3339	Redondo Beach, CA	1	0.01
3340		1	0.01

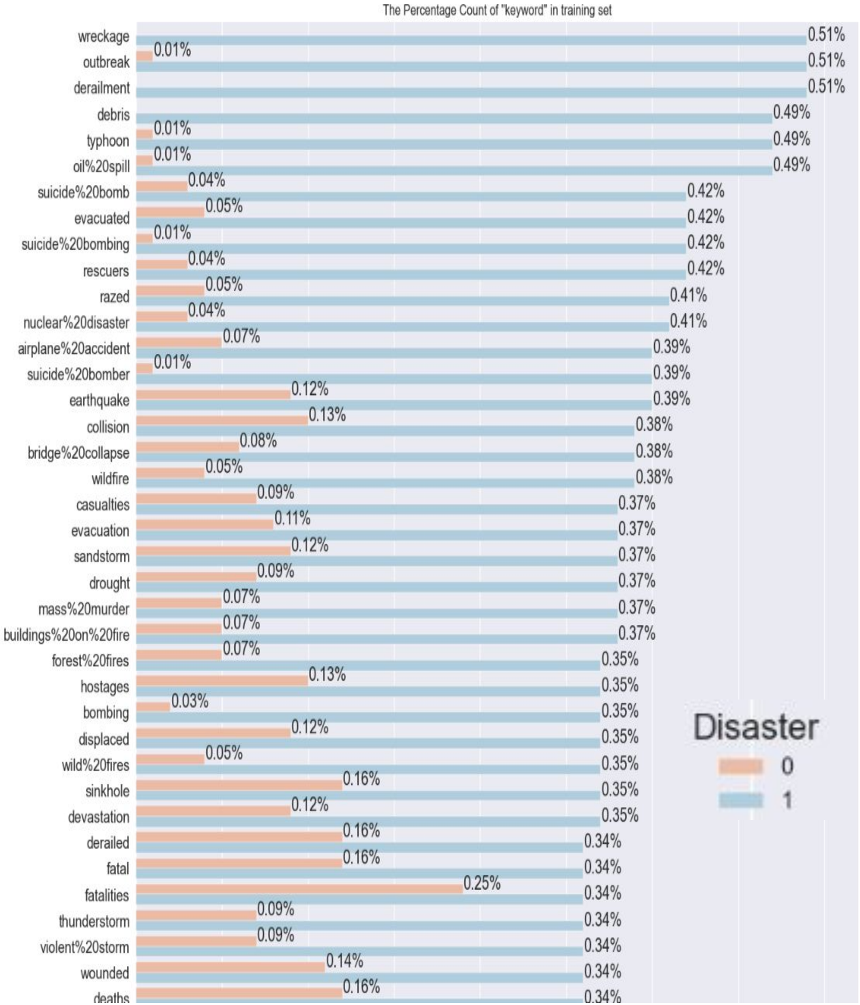
Inspecting the "keyword" feature

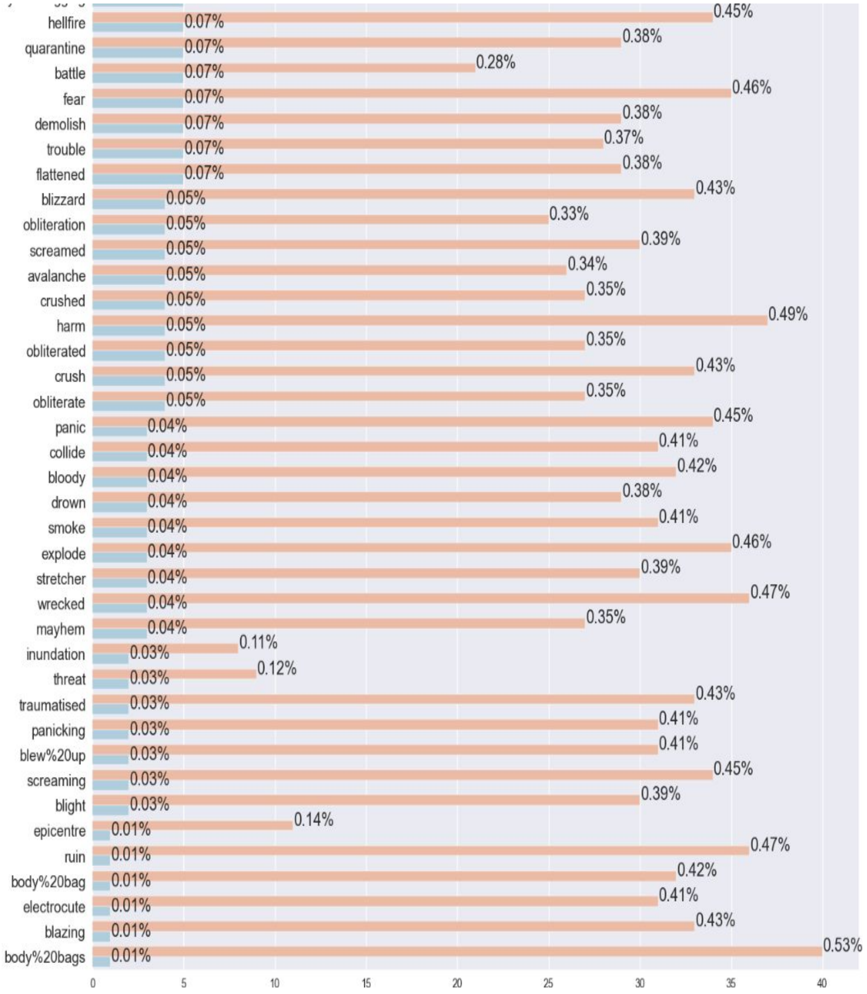
- There are 61 (0.8%) missing keywords
- keywords derailment, wreckage, and debris only appear in the disaster tweets with disaster probability = 1. Also, outbreak, typhoon, oil%20omb, suicide%20bombing, and rescuers have more than 90%
- keywords aftershock, body%20bags, ruin, blazing, body%20bag, electrocute, screaming, traumatised, and blew%20up mostly appear in the nondisaster tweets, near-zero probability of disaster tweets.
- many other keywords that are existed in both classes. The keyword without the context can not be a good predictor of the disaster tweets.



Top 15 keywords with highest probability of indicating disaster tweet

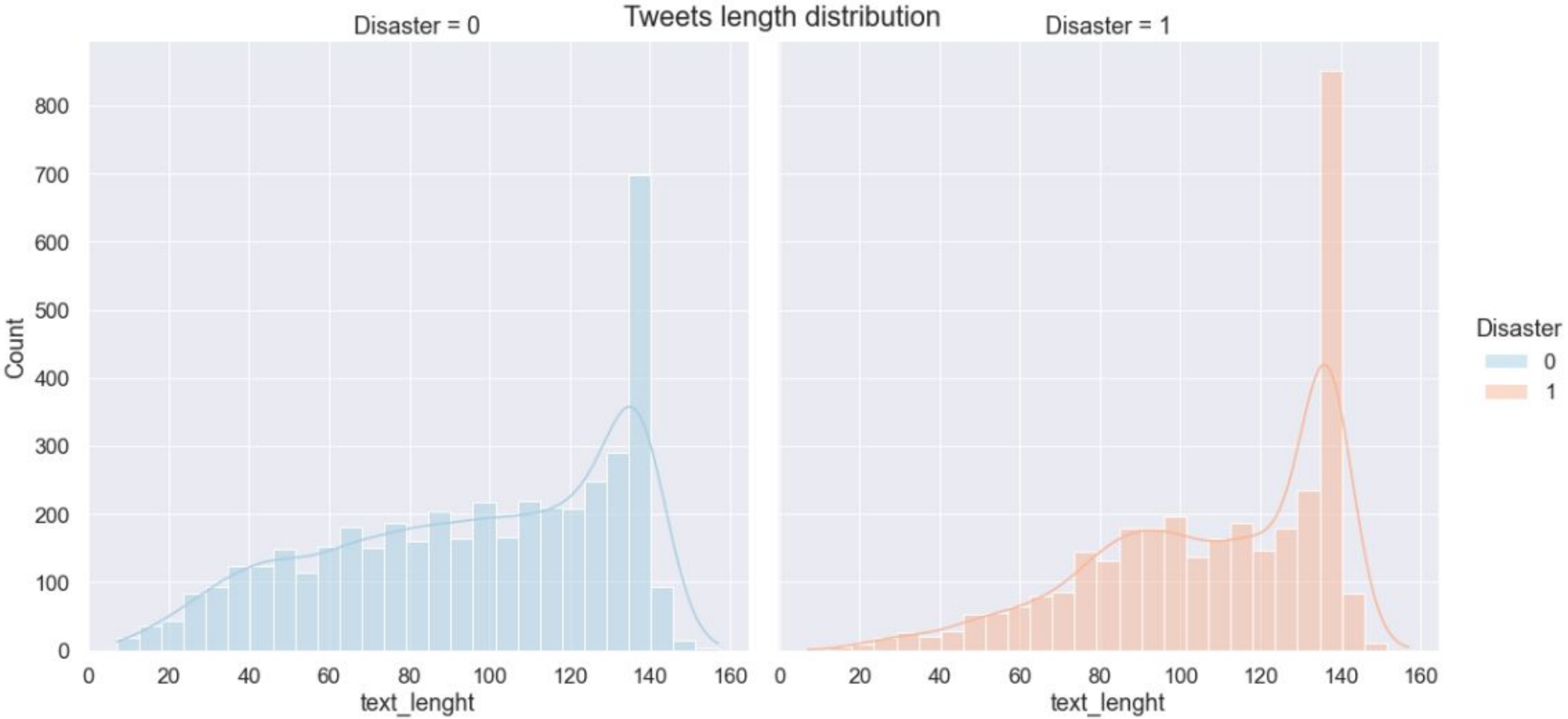
	count	% of dataset rows	Probability of Disaster
keyword			
wreckage	39	0.51	1.000000
debris	37	0.49	1.000000
derailment	39	0.51	1.000000
outbreak	40	0.53	0.975000
oil%20spill	38	0.50	0.973684
typhoon	38	0.50	0.973684
suicide%20bombing	33	0.43	0.969697
suicide%20bomber	31	0.41	0.967742
bombing	29	0.38	0.931034
rescuers	35	0.46	0.914286
suicide%20bomb	35	0.46	0.914286
nuclear%20disaster	34	0.45	0.911765
evacuated	36	0.47	0.888889
razed	35	0.46	0.885714
wildfire	33	0.43	0.878788


The Percentage Count of "keyword" in training set
wreckage 0.51%
outbreak 0.01% 0.51%
derailment 0.51%
debris 0.49%
typhoon 0.01% 0.49%
oil%20spill 0.01% 0.49%
suicide%20bomb 0.04% 0.42%
evacuated 0.05% 0.42%
suicide%20bombing 0.01% 0.42%
rescuers 0.04% 0.42%
razed 0.05% 0.41%
nuclear%20disaster 0.04% 0.41%
airplane%20accident 0.07% 0.39%
suicide%20bomber 0.01% 0.39%
earthquake 0.12% 0.39%
collision 0.13% 0.38%
bridge%20collapse 0.08% 0.38%
wildfire 0.05% 0.38%
casualties 0.09% 0.37%
evacuation 0.11% 0.37%
sandstorm 0.12% 0.37%
drought 0.09% 0.37%
mass%20murder 0.07% 0.37%
buildings%20on%20fire 0.07% 0.37%
forest%20fires 0.07% 0.35%
hostages 0.13% 0.35%
bombing 0.03% 0.35%
displaced 0.12% 0.35%
wild%20fires 0.05% 0.35%
sinkhole 0.16% 0.35%
devastation 0.12% 0.35%
derailed 0.16% 0.34%
fatal 0.16% 0.34%
fatalities 0.25% 0.34%
thunderstorm 0.09% 0.34%
violent%20storm 0.09% 0.34%
wounded 0.14% 0.34%
deaths 0.16% 0.34%
Disaster
0
1


hellfire 0.45% 0.07%
quarantine 0.38% 0.07%
battle 0.28% 0.07%
fear 0.46% 0.07%
demolish 0.38% 0.07%
trouble 0.37% 0.07%
flattened 0.38% 0.07%
blizzard 0.43% 0.05%
obliteration 0.33% 0.05%
screamed 0.39% 0.05%
avalanche 0.34% 0.05%
crushed 0.35% 0.05%
harm 0.49% 0.05%
obliterated 0.35% 0.05%
crush 0.43% 0.05%
obliterate 0.35% 0.05%
panic 0.45% 0.04%
collide 0.41% 0.04%
bloody 0.42% 0.04%
drown 0.38% 0.04%
smoke 0.41% 0.04%
explode 0.46% 0.04%
stretcher 0.39% 0.04%
wrecked 0.47% 0.04%
mayhem 0.35% 0.04%
inundation 0.11% 0.03%
threat 0.12% 0.03%
traumatised 0.43% 0.03%
panicking 0.41% 0.03%
blew%20up 0.41% 0.03%
screaming 0.45% 0.03%
blight 0.39% 0.03%
epicentre 0.14% 0.01%
ruin 0.47% 0.01%
body%20bag 0.42% 0.01%
electrocute 0.41% 0.01%
blazing 0.43% 0.01%
body%20bags 0.53% 0.01%
0
5
10
15
20
25
30
35
40

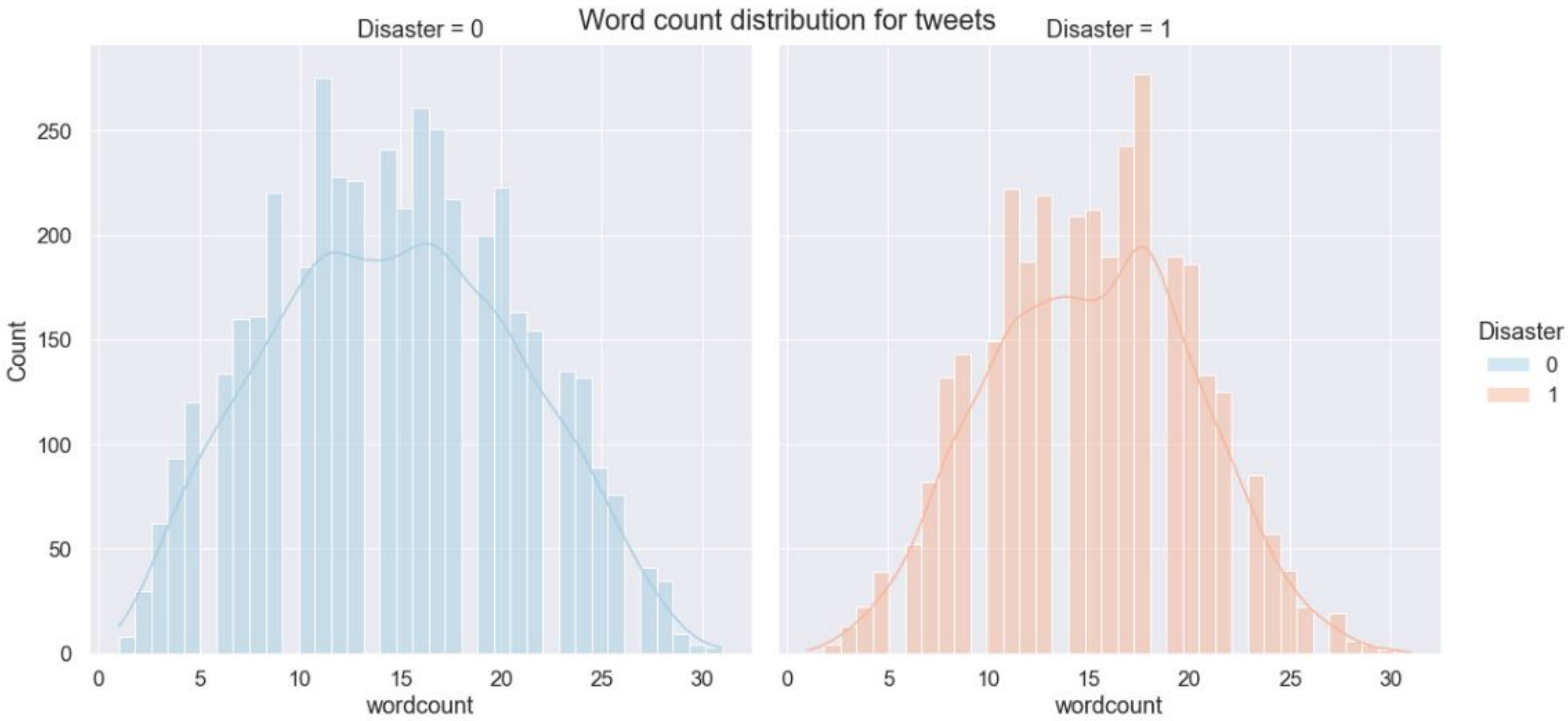
Tweets Metadata Analysis

- Tweets length (number of characters) distribution for Disaster and None Disaster classes



Tweets Meta Data Analysis

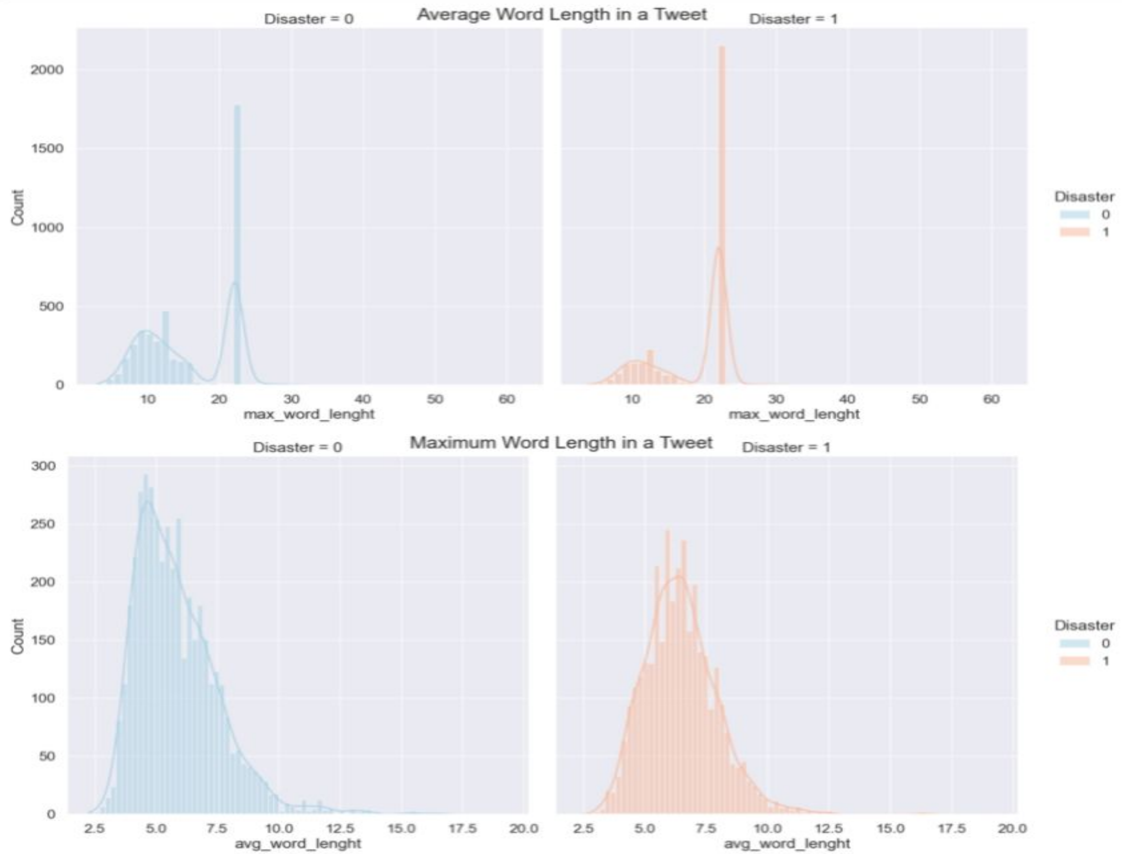
- Word count distribution for tweets



Tweets Metadata Analysis

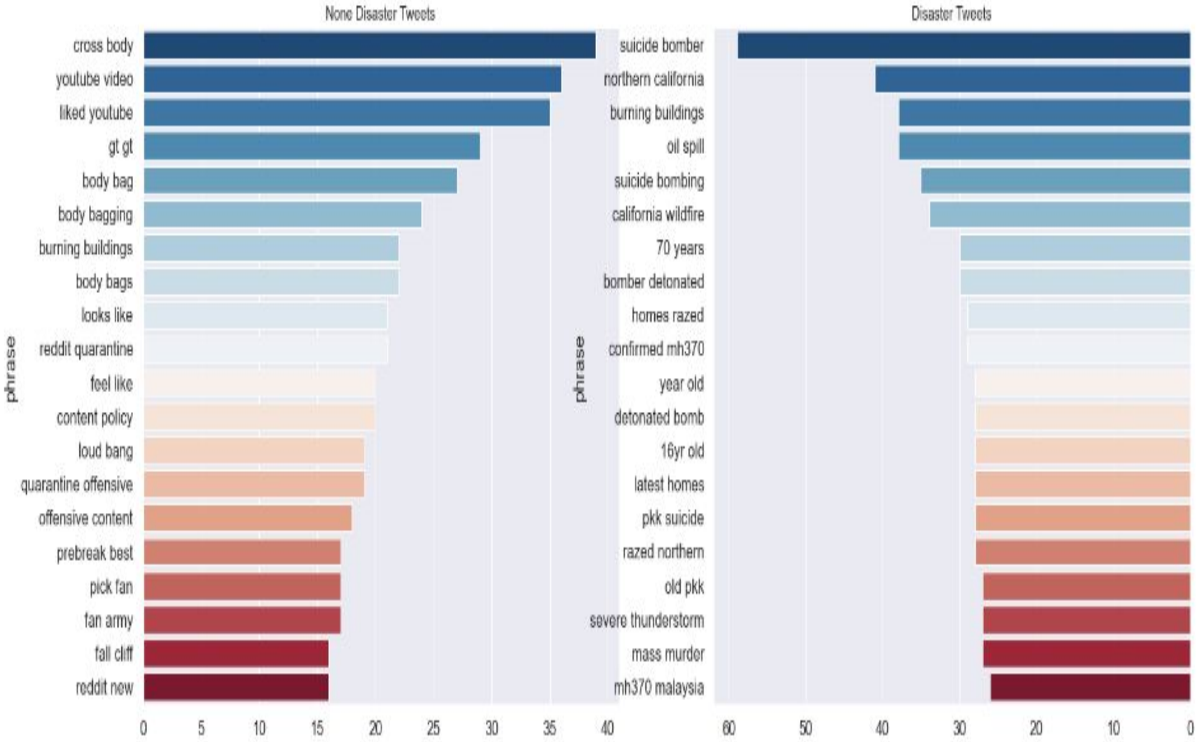
- Average and Maximum word length in a tweet

  - The Meta data features of both classes have a similar distribution
  - None of these Metadata features seems to be a good predictor of the disaster tweets
  - We will further check the predictive power of these features



Exploring Common Words and Phrases

- The distribution of top bigrams after removing stop words



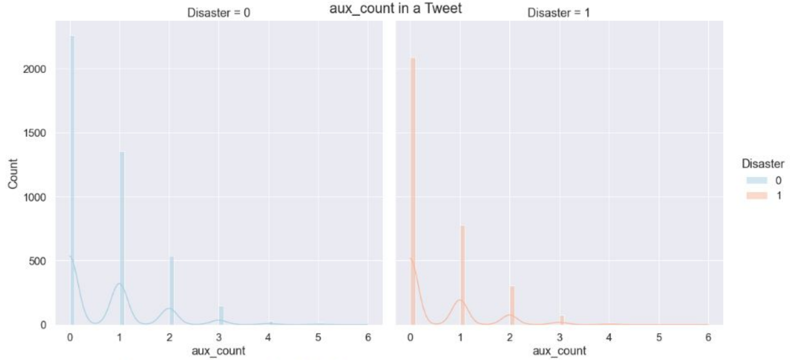
- The data needs to be cleaned from URLs and stop words should be removed to have a meaningful distribution of common phrases in both classes

- There is a noticable difference of N-Gram output for each class

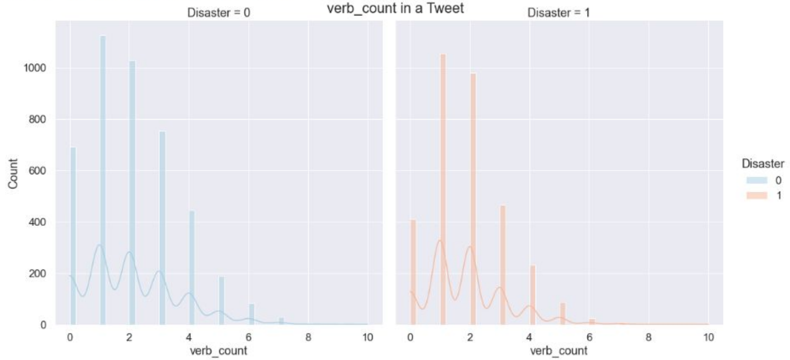
Exploring Linguistic Features

- Part of Speech Analysis using Spacy

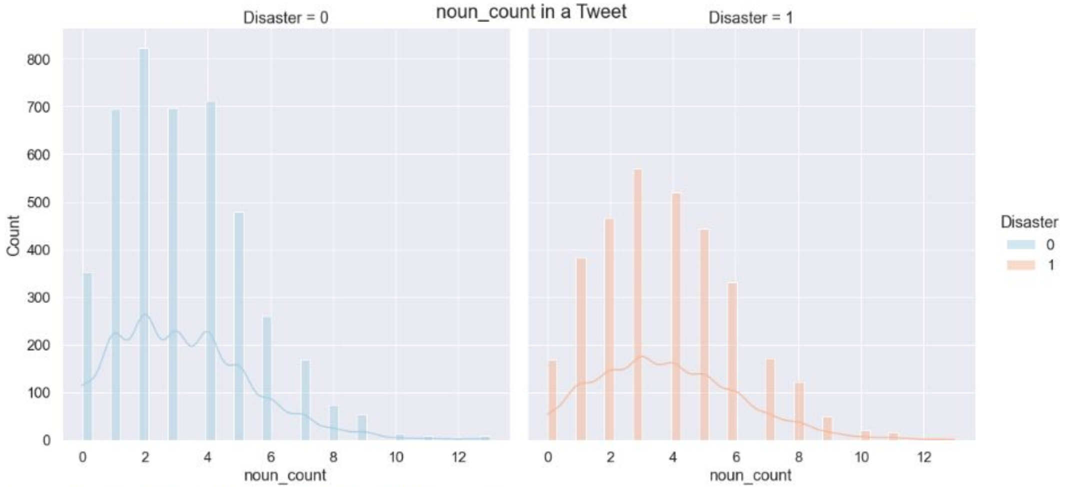
```
Average aux_count for disaster tweet is 0.5186887254901961
Average aux_count for none disaster tweet is 0.7052195907105082
```



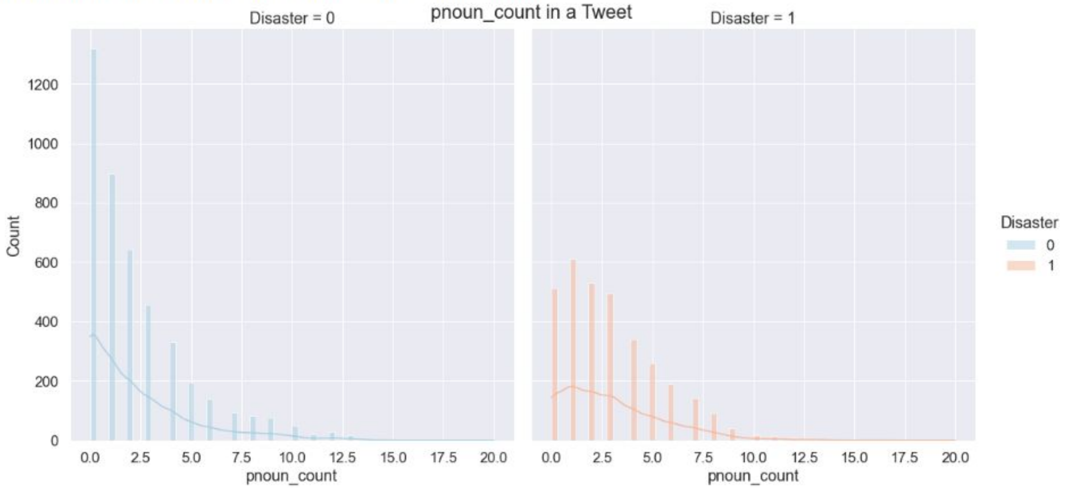
```
Average verb_count for disaster tweet is 1.8385416666666667
Average verb_count for none disaster tweet is 2.0531156587721315
```



```
Average noun_count for disaster tweet is 3.792892156862745
Average noun_count for none disaster tweet is 3.191998160496666
```



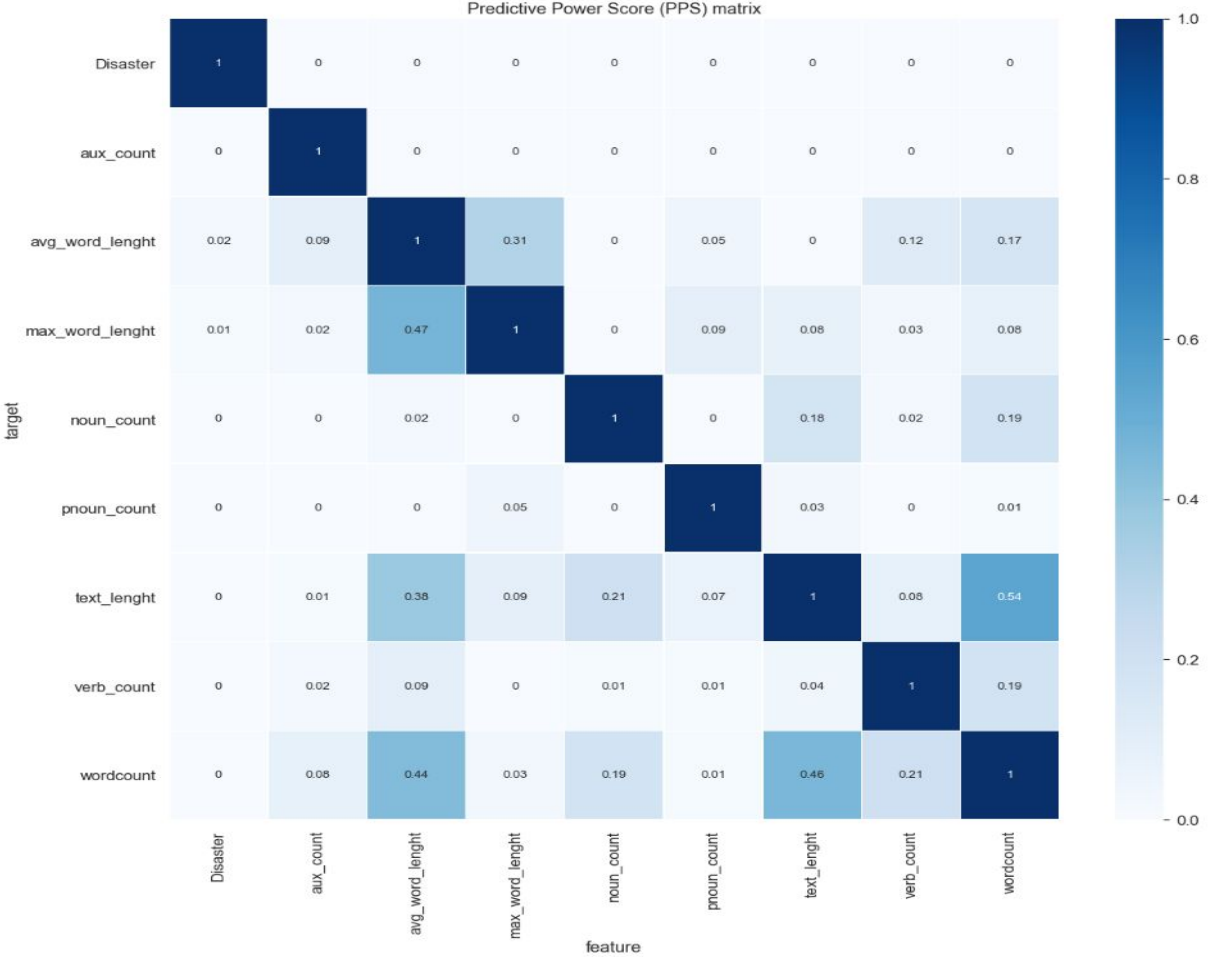
```
Average pnoun_count for disaster tweet is 2.9607843137254903
Average pnoun_count for none disaster tweet is 2.300988733042079
```



1. Predictive Power of the Imputed Features

- As we noticed from the distribution analysis of the metadata features, the PPS confirms that the computed meta features have no predictive power to the target variable (PPS = 0 for all variables), "disaster" in this case.
- Pearson correlation showed some weak correlation of these metadata variables with the maximum word length being the highest (r = 0.25).
- However, Pearson correlation can give misleading results for a binary classification problem (i.e. examining binary target variable), at least, it is not good option. Chick this for more details.

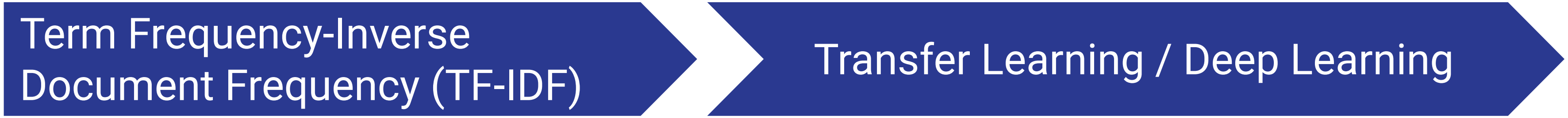
Based on PPS values, these variables should not be used in modeling.



target \ feature	Disaster	aux_count	avg_word_lenght	max_word_lenght	noun_count	pnoun_count	text_lenght	verb_count	wordcount
Disaster	1	0	0	0	0	0	0	0	0
aux_count	0	1	0	0	0	0	0	0	0
avg_word_lenght	0.02	0.09	1	0.31	0	0.05	0	0.12	0.17
max_word_lenght	0.01	0.02	0.47	1	0	0.09	0.08	0.03	0.08
noun_count	0	0	0.02	0	1	0	0.18	0.02	0.19
pnoun_count	0	0	0	0.05	0	1	0.03	0	0.01
text_lenght	0	0.01	0.38	0.09	0.21	0.07	1	0.08	0.54
verb_count	0	0.02	0.09	0	0.01	0.01	0.04	1	0.19
wordcount	0	0.08	0.44	0.03	0.19	0.01	0.46	0.21	1

Data Pre-Processing and Modeling

Modeling Approaches



Term Frequency-Inverse Document Frequency (TF-IDF)

- Multinomial Naive Bayes
- Logistic Regression
- XGboost with Bayesian Hyperparameter Optimization
- Deep Learning:
    - Simple Neural Network
    - Recurrent Neural Network (LSTM)
    - Convolutional Neural Network (CNN)

Transfer Learning / Deep Learning

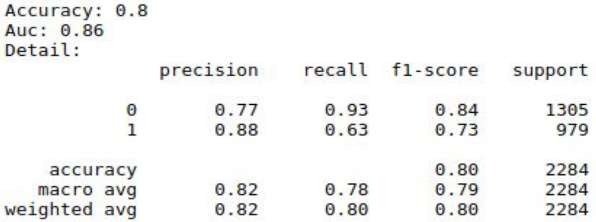
- Bidirectional Encoder Representations from Transformers (BERT):
    - Distilled BERT uncased
    - BERT base uncased

Term Frequency — Inverse Document Frequency

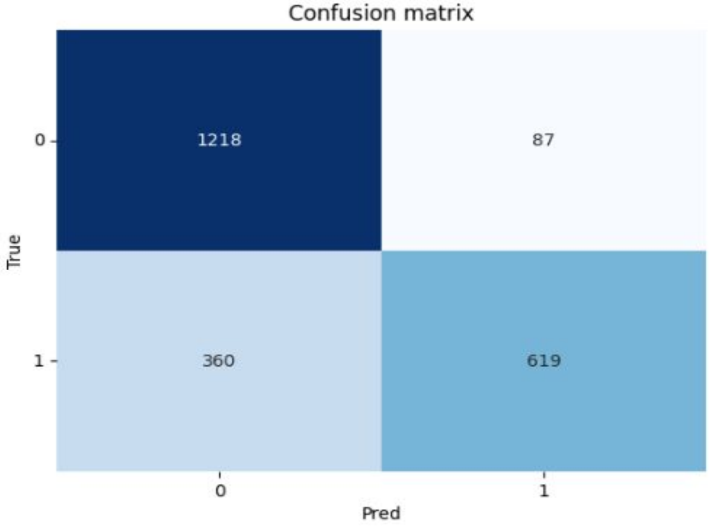
Pre-processing

- Remove stop words
- Remove urls
- Remove digits
- Remove hashtags
- Remove tags
- Lower case text
- Text lemmatization

Multinomial Naive Bayes

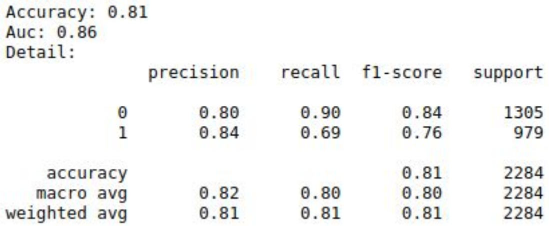


```
Accuracy: 0.8
Auc: 0.86
Detail:
              precision    recall  f1-score   support

           0       0.77      0.93      0.84      1305
           1       0.88      0.63      0.73       979

    accuracy                           0.80      2284
   macro avg       0.82      0.78      0.79      2284
weighted avg       0.82      0.80      0.80      2284
```

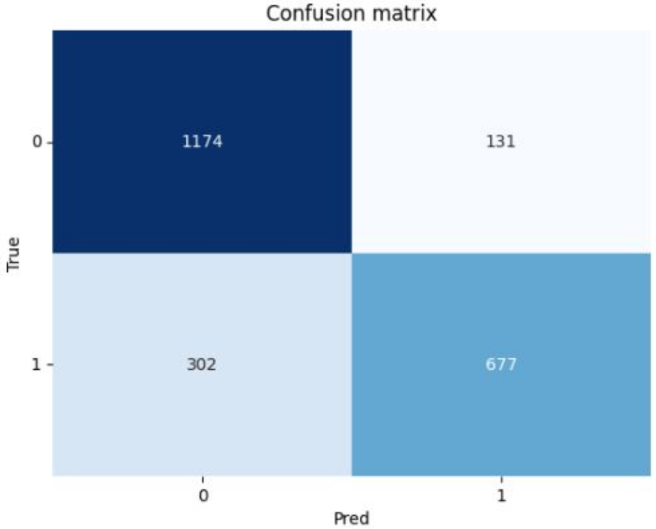


Term Frequency — Inverse Document Frequency

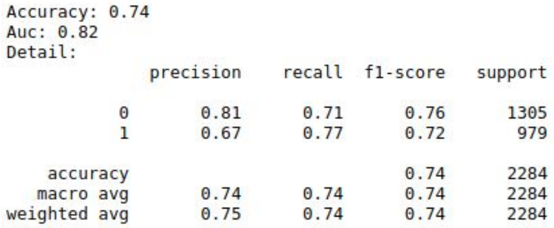
Logistic Regression



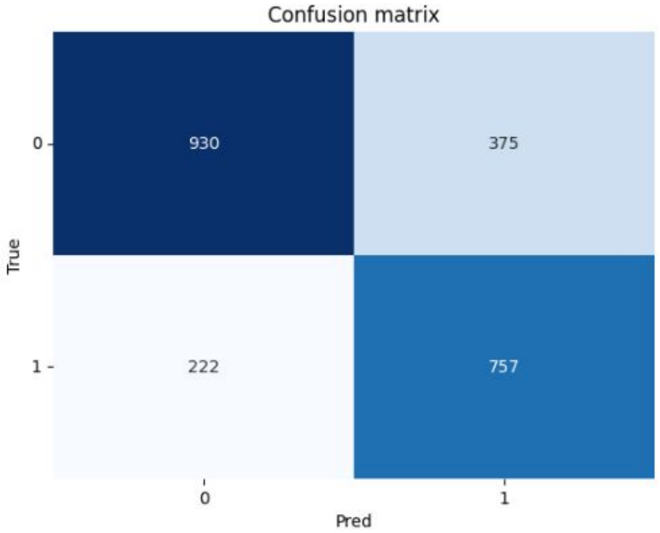
```
Accuracy: 0.81
Auc: 0.86
Detail:
              precision    recall  f1-score   support

           0       0.80      0.90      0.84      1305
           1       0.84      0.69      0.76       979

    accuracy                           0.81      2284
   macro avg       0.82      0.80      0.80      2284
weighted avg       0.81      0.81      0.81      2284
```



XGboost / Bayesian Hyperparameter Optimization

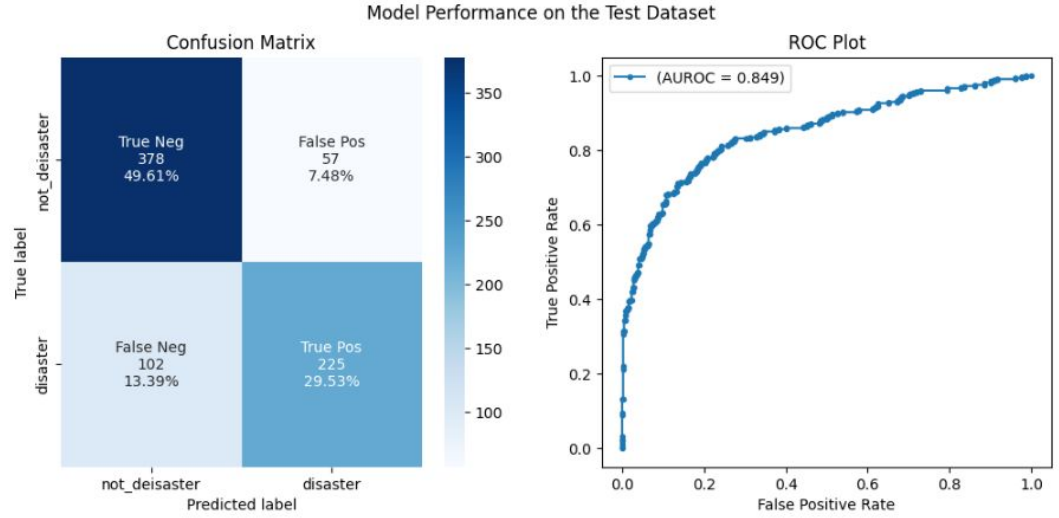


```
Accuracy: 0.74
Auc: 0.82
Detail:
              precision    recall  f1-score   support

           0       0.81      0.71      0.76      1305
           1       0.67      0.77      0.72       979

    accuracy                           0.74      2284
   macro avg       0.74      0.74      0.74      2284
weighted avg       0.75      0.74      0.74      2284
```



Term Frequency — Inverse Document Frequency

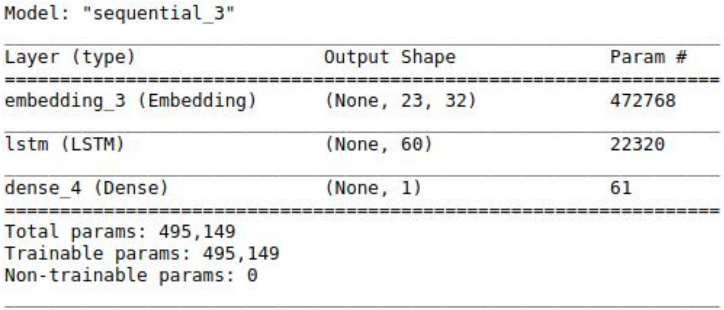
Recurrent Neural Network with LSTM with Dropout



Accuracy=0.791
Precision=0.798
Recall=0.688
F1 Score=0.739

```
Details:
              precision    recall  f1-score   support

           0       0.79      0.87      0.83       435
           1       0.80      0.69      0.74       327

    accuracy                           0.79       762
   macro avg       0.79      0.78      0.78       762
weighted avg       0.79      0.79      0.79       762
```

Model Architecture

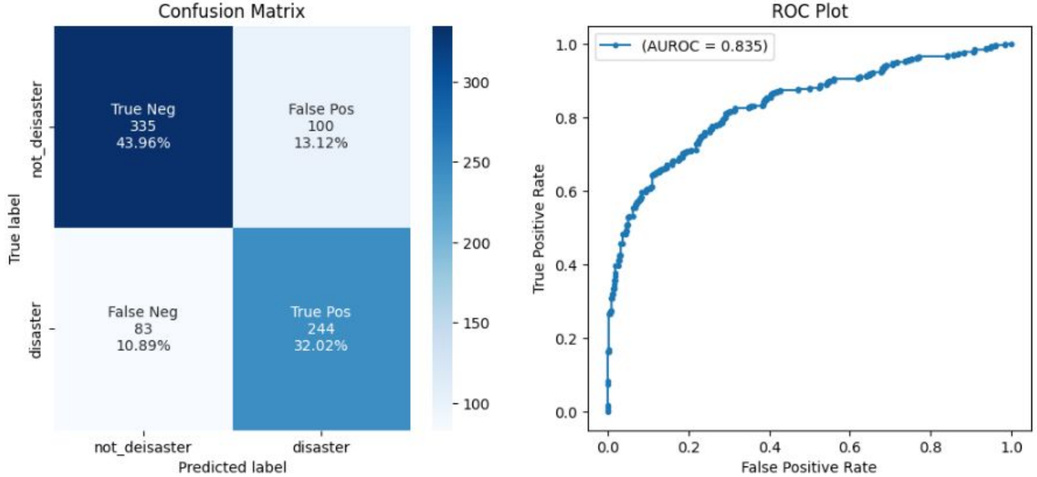


```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 23, 32)            472768
_________________________________________________________________
lstm (LSTM)                  (None, 60)                22320
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 61
=================================================================
Total params: 495,149
Trainable params: 495,149
Non-trainable params: 0
_________________________________________________________________
```

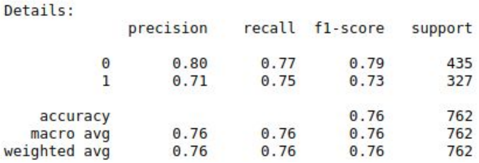
Term Frequency — Inverse Document Frequency

Convolutional Neural Network (CNN)

Model Performance on the Test Dataset

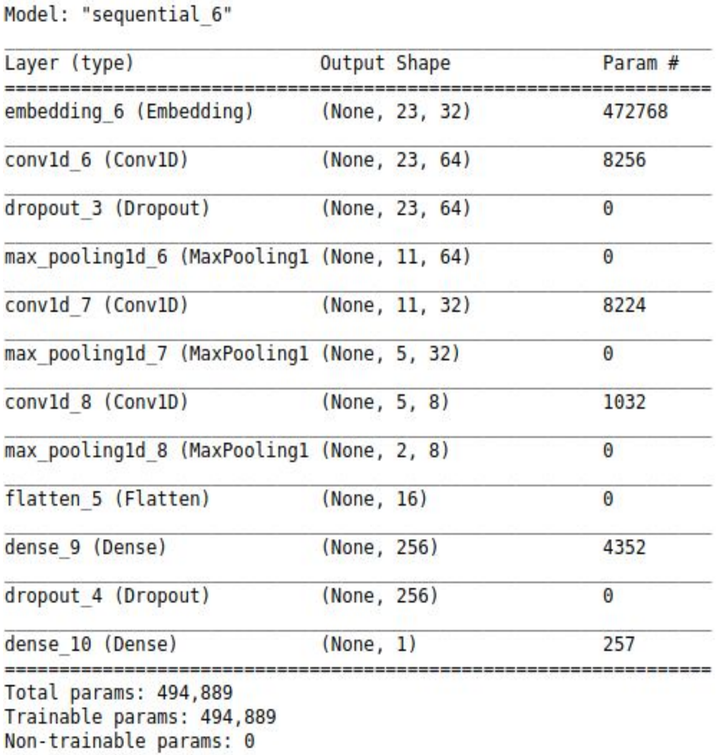


Accuracy=0.760
Precision=0.709
Recall=0.746
F1 Score=0.727



```
Details:
              precision    recall  f1-score   support

           0       0.80      0.77      0.79       435
           1       0.71      0.75      0.73       327

    accuracy                           0.76       762
   macro avg       0.76      0.76      0.76       762
weighted avg       0.76      0.76      0.76       762
```

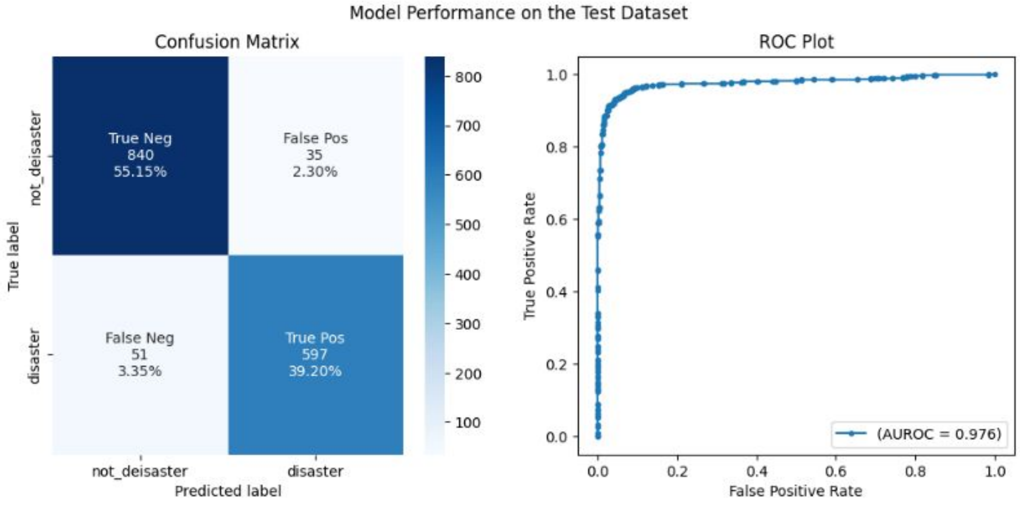
Model Architecture



```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
embedding_6 (Embedding)     (None, 23, 32)            472768
_________________________________________________________________
conv1d_6 (Conv1D)           (None, 23, 64)            8256
_________________________________________________________________
dropout_3 (Dropout)         (None, 23, 64)            0
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 11, 64)           0
_________________________________________________________________
conv1d_7 (Conv1D)           (None, 11, 32)            8224
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 5, 32)            0
_________________________________________________________________
conv1d_8 (Conv1D)           (None, 5, 8)              1032
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 2, 8)             0
_________________________________________________________________
flatten_5 (Flatten)         (None, 16)                0
_________________________________________________________________
dense_9 (Dense)             (None, 256)               4352
_________________________________________________________________
dropout_4 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_10 (Dense)            (None, 1)                 257
=================================================================
Total params: 494,889
Trainable params: 494,889
Non-trainable params: 0
```

Transfer Learning / Deep Learning

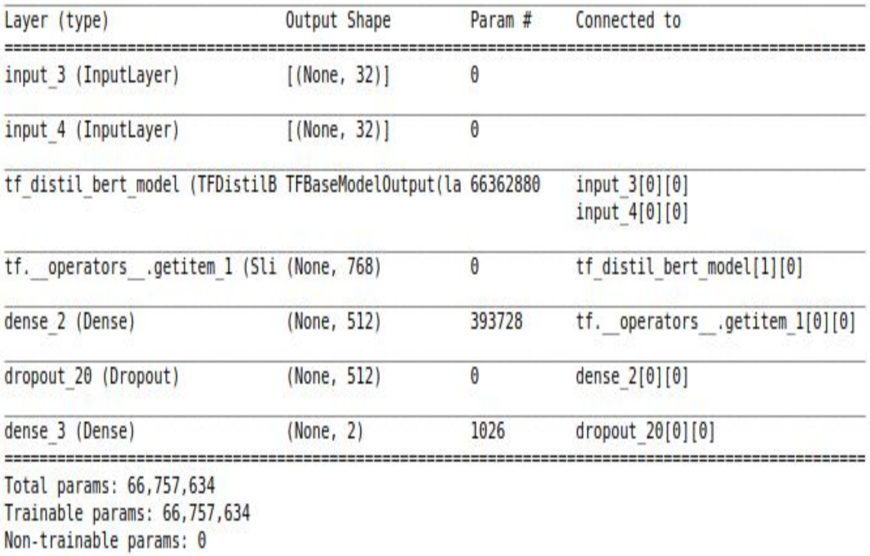
Bidirectional Encoder Representations from Transformers (BERT)



Accuracy=0.944
Precision=0.945
Recall=0.921
F1 Score=0.933

```
Details:
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       875
           1       0.94      0.92      0.93       648

    accuracy                           0.94      1523
   macro avg       0.94      0.94      0.94      1523
weighted avg       0.94      0.94      0.94      1523
```

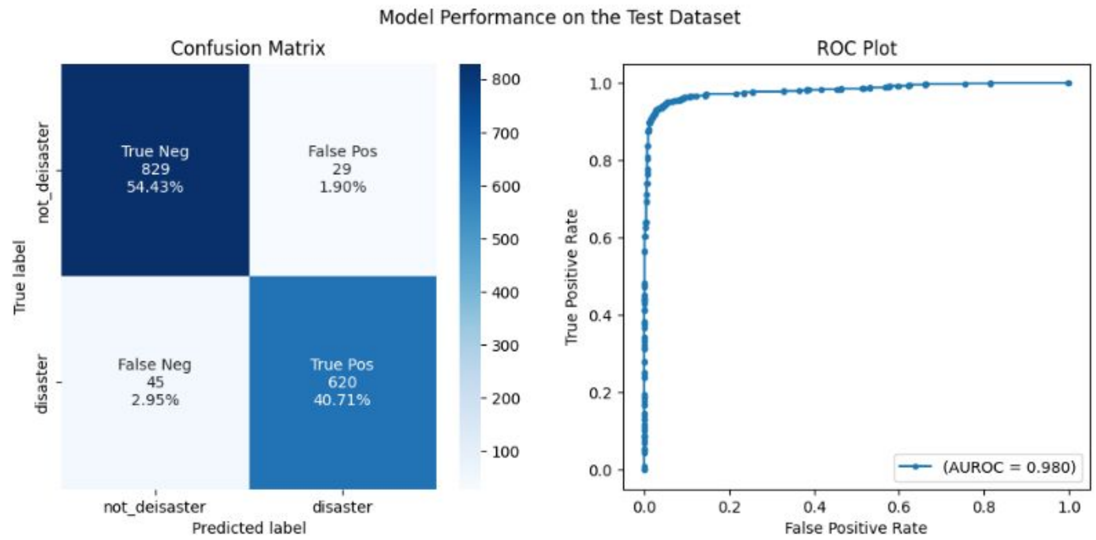
Model Architecture - DistilBERT



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 32)]         0

input_4 (InputLayer)            [(None, 32)]         0

tf_distil_bert_model (TFDistilB TFBaseModelOutput(la 66362880    input_3[0][0]
                                                                 input_4[0][0]

tf.__operators__.getitem_1 (Sli (None, 768)          0           tf_distil_bert_model[1][0]

dense_2 (Dense)                 (None, 512)          393728      tf.__operators__.getitem_1[0][0]

dropout_20 (Dropout)            (None, 512)          0           dense_2[0][0]

dense_3 (Dense)                 (None, 2)            1026        dropout_20[0][0]
==================================================================================================
Total params: 66,757,634
Trainable params: 66,757,634
Non-trainable params: 0
```

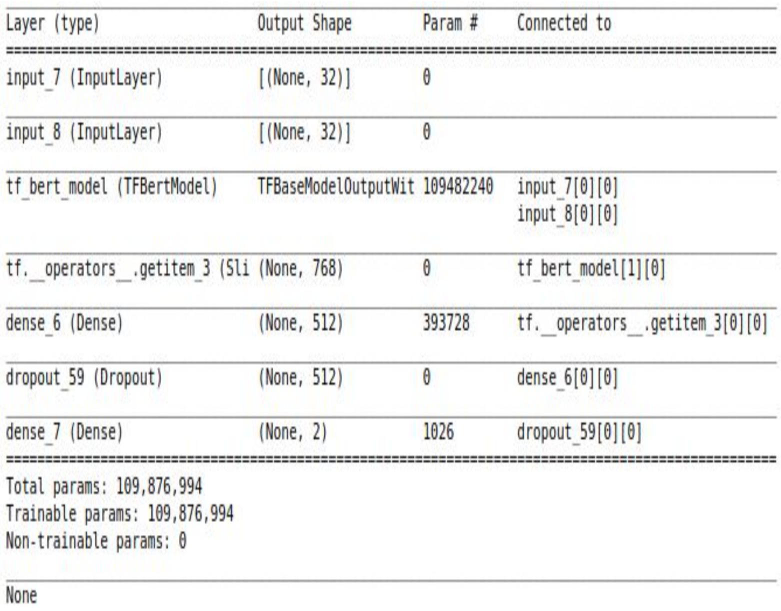
DistilBERT retains 97% performance of the BERT with 40% fewer parameters.

Bidirectional Encoder Representations from Transformers (BERT)



```
Accuracy=0.951
Precision=0.955
Recall=0.932
F1 Score=0.944

Details:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       858
           1       0.96      0.93      0.94       665

    accuracy                           0.95      1523
   macro avg       0.95      0.95      0.95      1523
weighted avg       0.95      0.95      0.95      1523
```

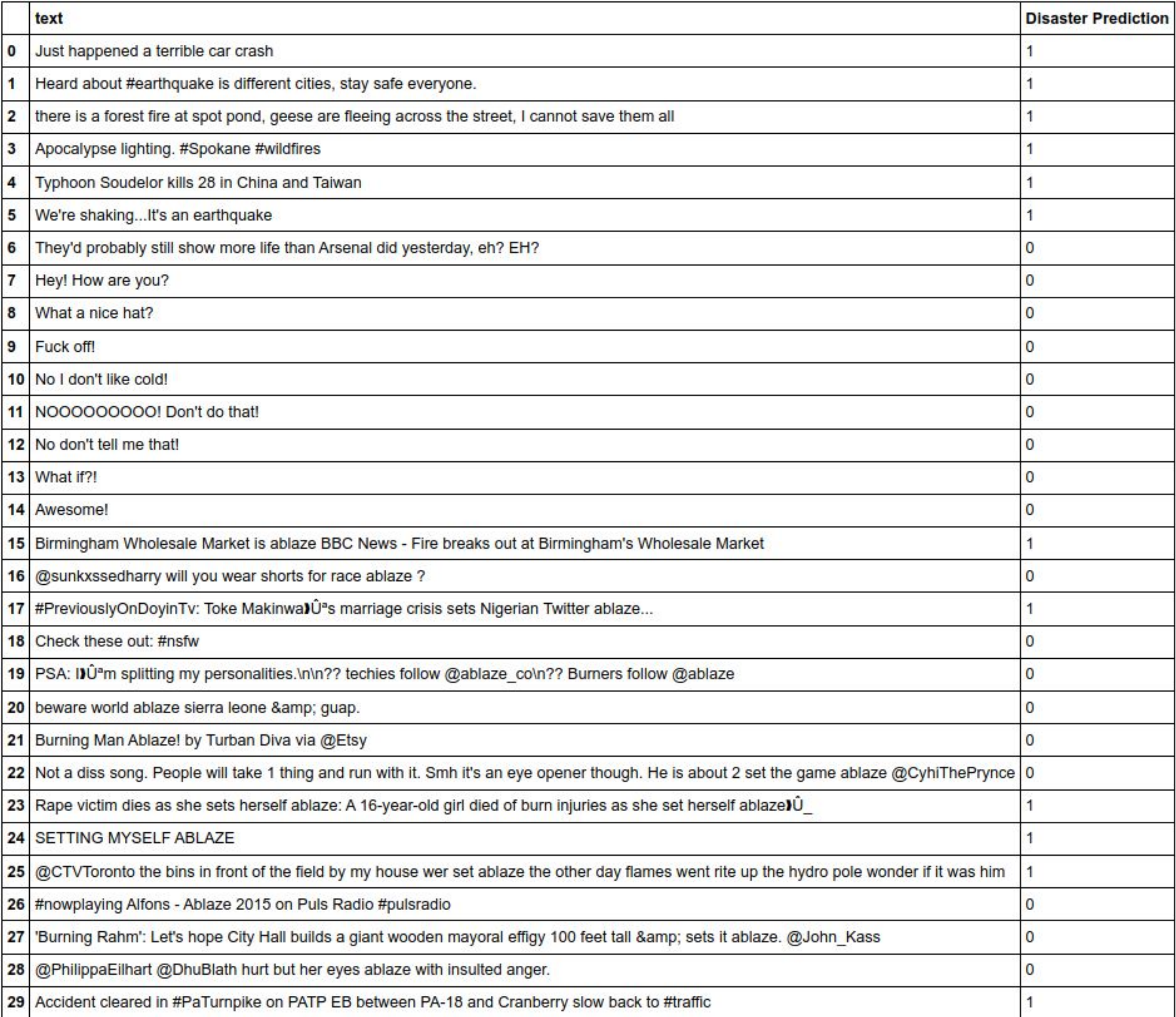
Model Architecture - base uncased



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_7 (InputLayer)            [(None, 32)]         0

input_8 (InputLayer)            [(None, 32)]         0

tf_bert_model (TFBertModel)     TFBaseModelOutputWit 109482240   input_7[0][0]
                                                                 input_8[0][0]

tf.__operators__.getitem_3 (Sli (None, 768)          0           tf_bert_model[1][0]

dense_6 (Dense)                 (None, 512)          393728      tf.__operators__.getitem_3[0][0]

dropout_59 (Dropout)            (None, 512)          0           dense_6[0][0]

dense_7 (Dense)                 (None, 2)            1026        dropout_59[0][0]
==================================================================================================
Total params: 109,876,994
Trainable params: 109,876,994
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

Best performance Model Acc = 0.95

Evaluating the Winning Model on Unlabeled Test Set



	text	Disaster Prediction
0	Just happened a terrible car crash	1
1	Heard about #earthquake is different cities, stay safe everyone.	1
2	there is a forest fire at spot pond, geese are fleeing across the street, I cannot save them all	1
3	Apocalypse lighting. #Spokane #wildfires	1
4	Typhoon Soudelor kills 28 in China and Taiwan	1
5	We're shaking...It's an earthquake	1
6	They'd probably still show more life than Arsenal did yesterday, eh? EH?	0
7	Hey! How are you?	0
8	What a nice hat?	0
9	Fuck off!	0
10	No I don't like cold!	0
11	NOOOOOOOOO! Don't do that!	0
12	No don't tell me that!	0
13	What if?!	0
14	Awesome!	0
15	Birmingham Wholesale Market is ablaze BBC News - Fire breaks out at Birmingham's Wholesale Market	1
16	@sunkxssedharry will you wear shorts for race ablaze ?	0
17	#PreviouslyOnDoyinTv: Toke Makinwa◗Ûªs marriage crisis sets Nigerian Twitter ablaze...	1
18	Check these out: #nsfw	0
19	PSA: I◗Ûªm splitting my personalities.\n\n?? techies follow @ablaze_co\n?? Burners follow @ablaze	0
20	beware world ablaze sierra leone &amp; guap.	0
21	Burning Man Ablaze! by Turban Diva via @Etsy	0
22	Not a diss song. People will take 1 thing and run with it. Smh it's an eye opener though. He is about 2 set the game ablaze @CyhiThePrynce	0
23	Rape victim dies as she sets herself ablaze: A 16-year-old girl died of burn injuries as she set herself ablaze◗Û_	1
24	SETTING MYSELF ABLAZE	1
25	@CTVToronto the bins in front of the field by my house wer set ablaze the other day flames went rite up the hydro pole wonder if it was him	1
26	#nowplaying Alfons - Ablaze 2015 on Puls Radio #pulsradio	0
27	'Burning Rahm': Let's hope City Hall builds a giant wooden mayoral effigy 100 feet tall &amp; sets it ablaze. @John_Kass	0
28	@PhilippaEilhart @DhuBlath hurt but her eyes ablaze with insulted anger.	0
29	Accident cleared in #PaTurnpike on PATP EB between PA-18 and Cranberry slow back to #traffic	1

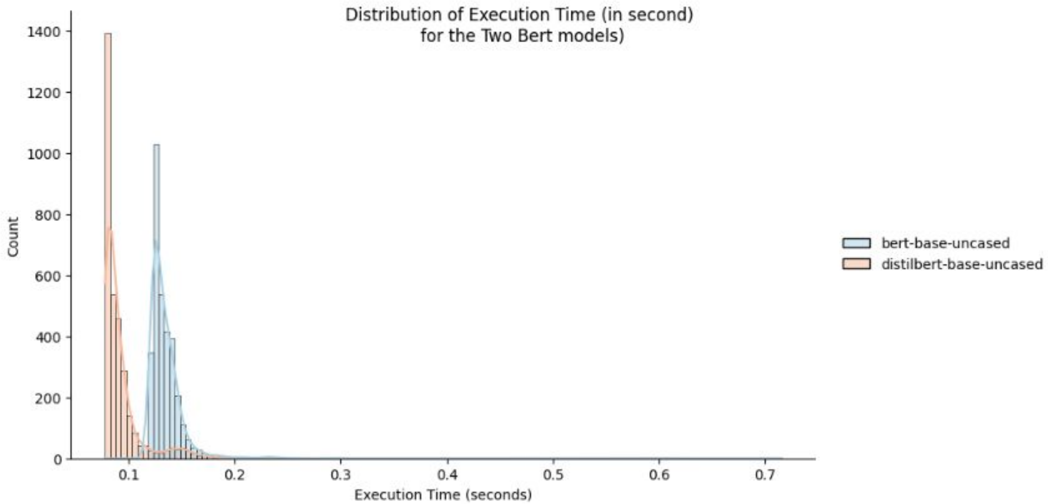
Conclusion and Performance Evaluation

- Both BertDistill and Bert_large_uncased have produced the best performance.

- Bert_large_uncased is the winning model with slightly better performance.

- However, training Bert_large_uncased with 109,876,994 trainable parameters for 30 epochs required more than 9 hours on 16GB Ram / 4 cores CPU laptop.

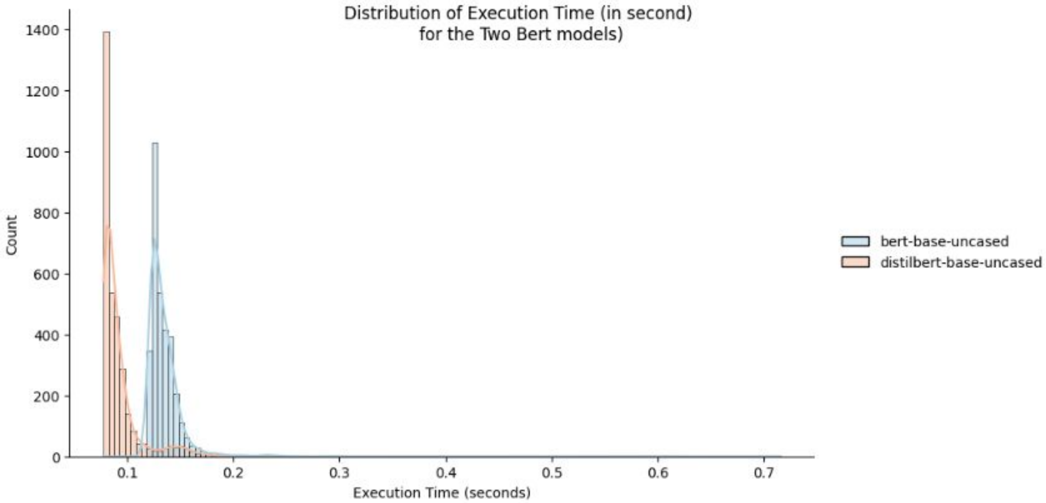
- Although Distilled Bert provided slightly less prediction performance, it is faster considerably faster than Bert base model.
- For predicting a streaming tweets in realtime, scalability and latency are a big deal. Depending on the production environment, a trade-off need to be considered between slightly more performance and faster execution time.
-



Recommendations

- Test the model in the production environment and determine which provides more scalability.

- The model is trained on a small labeled data set. The performance should be monitored and improved by using more representative data.


Distribution of Execution Time (in second)
for the Two Bert models)
Count
Execution Time (seconds)
bert-base-uncased
distilbert-base-uncased