Real or Not? NLP with Disaster Tweets

Social media has become one of the most important part of our day-to-day life. We get information about all the topics, starting from news, stock markets, retail related etc. Since these applications also provide facility of trending important topics of the day, what is going on in the worlds is simply now one swap away!

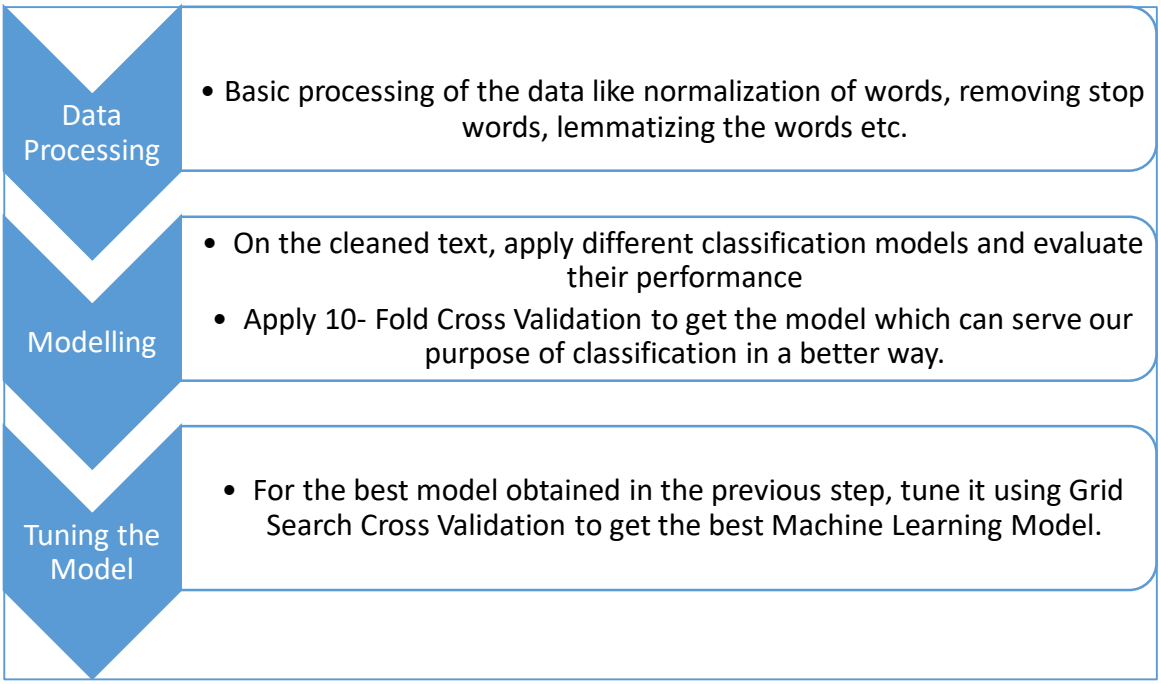
One such application is Twitter where we see the users sharing their opinions on variety of topics like, politics, movies, etc. It has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter (i.e. disaster relief organizations and news agencies).

But, it's not always clear whether a person's words are actually announcing a disaster.

In this project, we will be building a machine learning model that predicts which Tweets are about real disasters and which one's aren't.

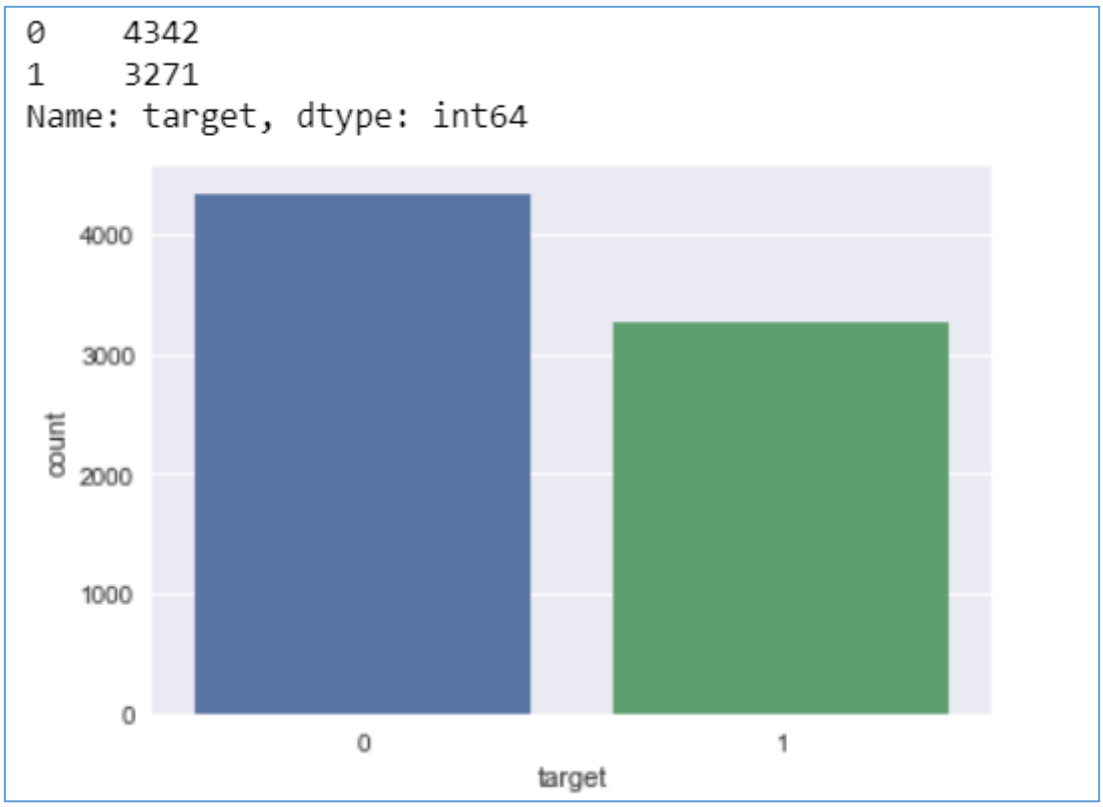
Dataset is taken from Kaggle: https://www.kaggle.com/c/nlp-getting-started/data

Process followed in the study:

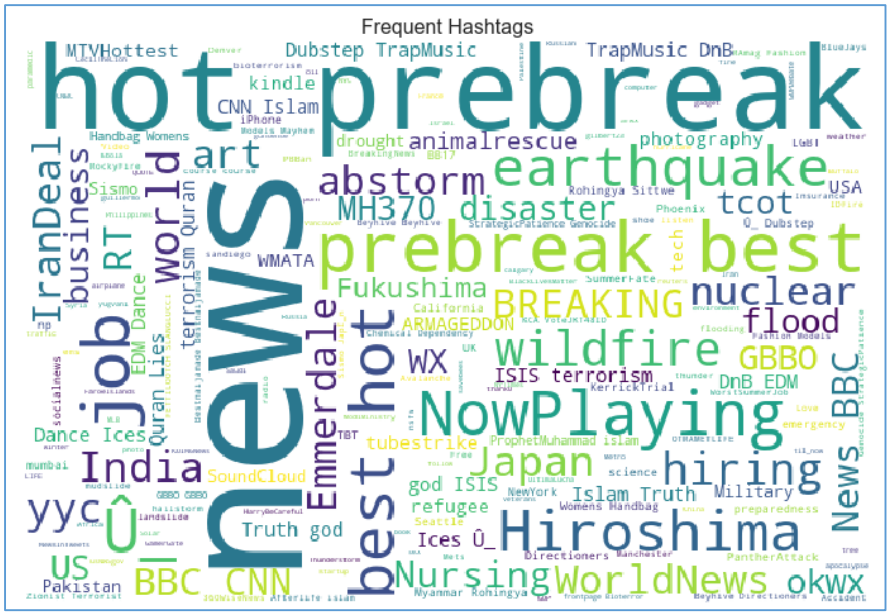


Exploratory Data Analysis on the dataset

1. Target Label Count in the training dataset :

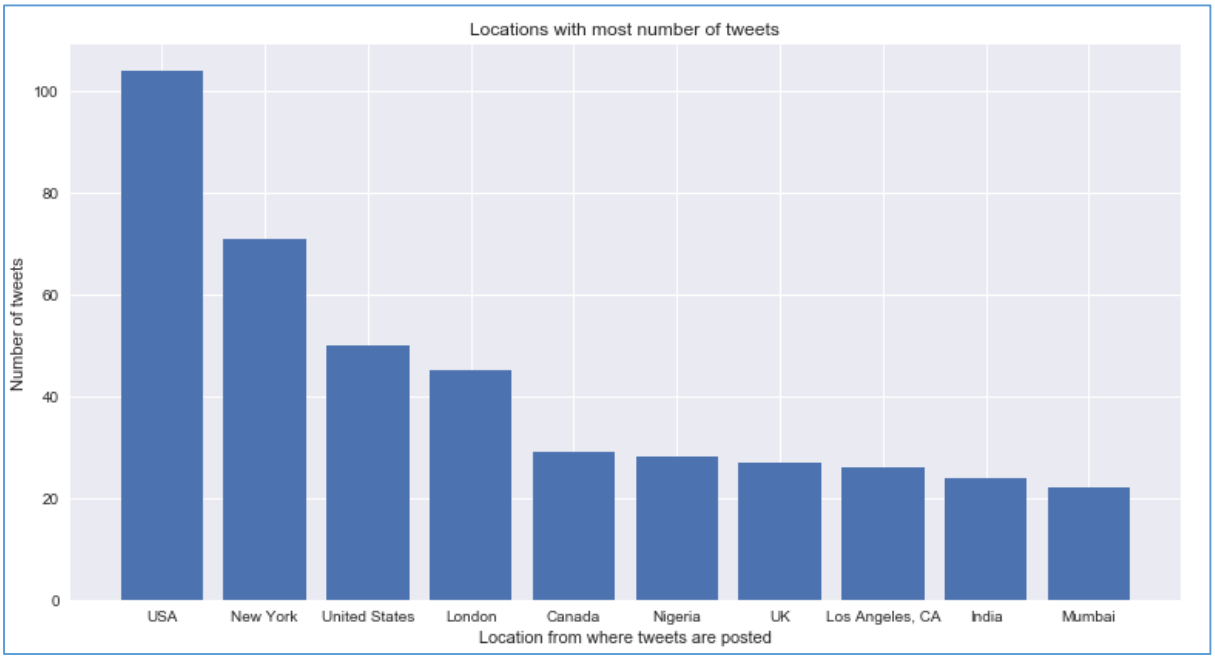


2. Most Frequent HashTags in the dataset :

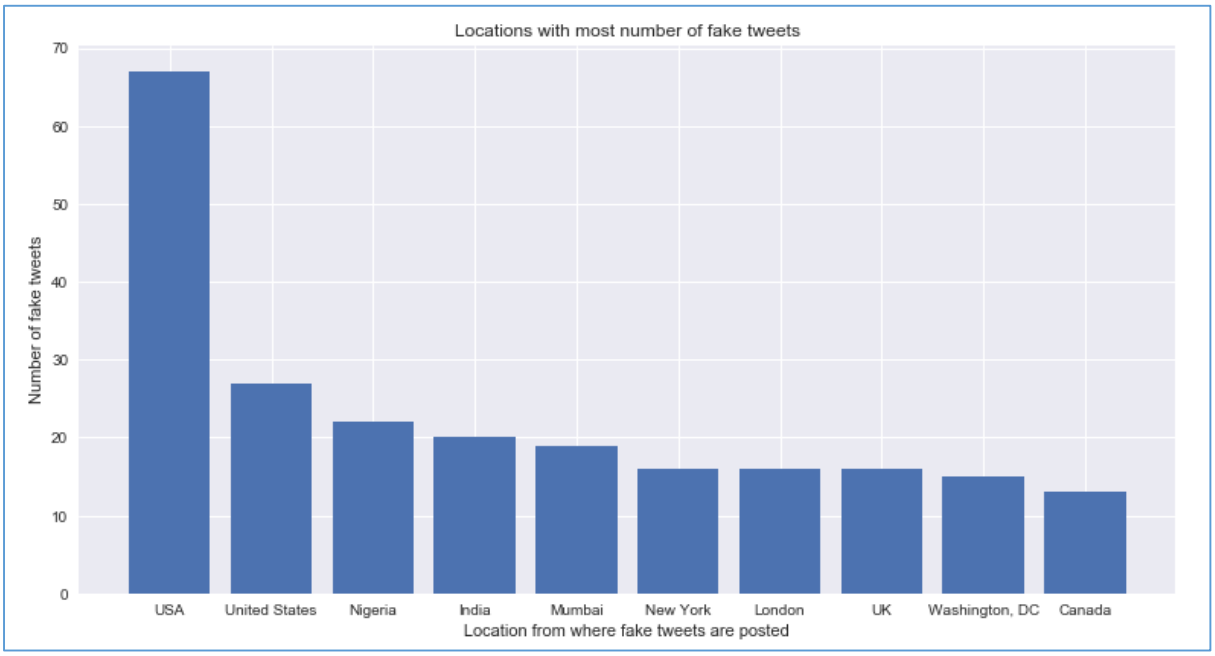


Hmm, so we have quite a lot of hashtags flowing in our dataset. Most Frequent Ones are: news, prebreak, best, nowplaying etc.

3. Location with most number of tweets :



4. Location with most number of fake tweets :



We can infer from the above two plots that the most number of tweets are from USA and most number of fake tweets are also from USA

Data Processing

Since, here we are dealing with text data, hence there is a need to process these tweets in order to keep only relevant information which is best suited for modelling. In this data preparation process, we will:

1. Normalize the tweets to the lower format.
2. Remove tags like #, @ from the words in the text.
3. Remove punctuations.
4. Remove stop words like is, the, on etc.
5. Remove words which are of length less than 2.
6. Remove alpha-numeric string.
7. Reduce the words to their root forms using Lemmatization.

Sample execution from the method written for getting the cleaned text:

```
# Let's test the get_cleaned_text method
print(f'Original Text : {training_dataset.text[0]}')
print(f'Cleaned Text : {get_cleaned_text(training_dataset.text[0])}')
```

```
Original Text : Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all
Cleaned Text : deed reason earthquake may allah forgive
```

We can clearly see that:

1. Text is transformed into lower case.
2. # is removed from earthquake.
3. Stop words are removed.
4. Lemmatization of words : Reasons --> reason

Applying Different Machine Learning Algorithms

Algorithm	Accuracy without Cross Validation	Accuracy with Cross Validation
Decision Tree	55	51
Logistic Regression	56	63
Support Vector Machine	63	57
Naive Bayes	52	60
K- Nearest Neighbor	62	58

We can see that Logistic Regressor can be a go-to solution to our problem and we can tune it to make it more accurate.

Tune the Logistic Regression Model

We will be using GridSearchCV to fine tune our Logistic Regression Model as below:

Parameters for tuning:

```
# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter space
C = np.logspace(0, 4, 100)

# Create hyperparameter options for Logistic Regressor
lr_grid_params = dict(C=C, penalty=penalty)
```

Fitting the model with parameters for tuning:

```
lr_clf = GridSearchCV(lr_model, lr_grid_params, refit=True, verbose=2)
```

```
lr_clf.fit(x_train, y_train)
```

Score obtained by best fitted model:

```
lr_clf.best_score_
```

```
0.6923683173518981
```

Hence, with model tuning, we can increase the performance of the existing model, as in this case, after tuning, the accuracy was increase from **56% to 69%.**

Conclusion

1. Logistic Regression Model can be the go-to algorithm to solve our problem.

2. On tuning the model, we are able to increase the accuracy from 56% to 69%.