Twitter as a Disaster Sensor

Challenges

- Contextuality: "Steph Curry on Fire" vs. "Building on Fire"
- Rich media: plain text + special characters + emoji + url + ~~image~~ + ~~video~~
  - *Disaster Tweets*: [''M1.94 [01:04 UTC]?5km S of Volcano Hawaii. http://t.co/zDtoyd8EbJ', 'The Latest: More Homes Razed by Northern California Wildfire - ABC News http://t.co/YmY4rSkQ3d']
  - *Non Disaster Tweets*: ['These boxes are ready to explode! Exploding Kittens finally arrived! gameofkittens explodingkittens89Û_ https://t.co/TFGrAyuDC5', 'Sirens everywhere!', 'I just heard a really loud bang and everyone is asleep great']

Dataset

- This particular capstone uses a Kaggle competition dataset. (Link)
- Number of instances in training data: 7613
- Number of attributes: three strings
- Attribute Information: keyword, location, text
- Missing Attribute Values: (Link)
  - keyword: 61 out of 7613 (0.8%)
  - location: 2533 out of 7613 (33.3%)
- Target: (discrete)
  - 0: ['non-disaster', count=4342]
  - 1: ['disaster', count=3271]

Exploratory Data Analysis

- Is the dataset balanced? (Link) YES
- Is tweet character length an importance feature? (Link) NO
- Is tweet word count an important feature? (Link) NO
- Does tweet text contain url and how does it impact disaster prediction? (Link) YES (urls and url counts are put in two different columns)
- Does tweet text contain emoji and how does it impact disaster prediction? (Link) : YES (only emoji count is retained for the model)

Missing Data Treatment

1. Assess if the missing points have any effect on target
    1.1 Ran *Mann Whitney U test* and determine the distribution difference between targets for missing data and non-missing data is NOT statistically significant.

    1.2 Filled all missing values with "uns_{attribute}" string

Feature Engineering

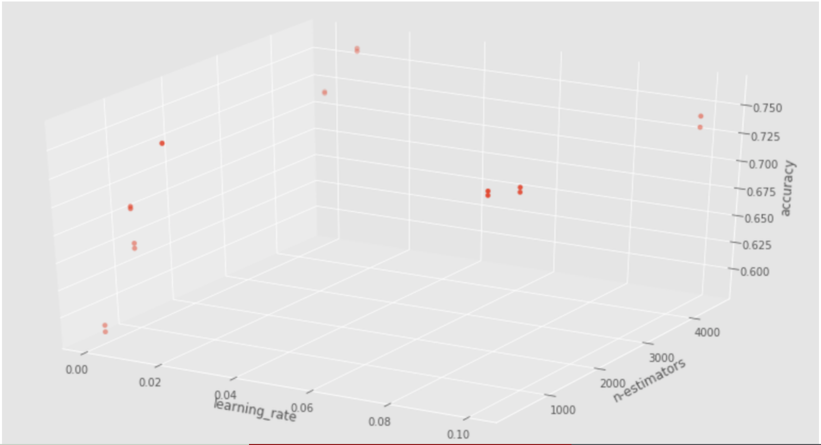
1. Parse rich text data to extract urls and emojis
2. Set urls as string, url count, and emoji count as three new attributes
3. Tfidf vectorize the text atrribute with max_length=500
4. Count vectorize the location, keyword, and url for max_length=100

Baseline Models

Model	Accuracy	Precision	Recall	F1-Score
Bernoulli NB	0.76	0.73	0.65	0.69
Random Forest Classifier	0.77	0.78	0.62	0.69
Gradient Boosting Classifier	0.7	0.84	0.33	0.47
XG Boost Classifier	0.76	0.8	0.57	0.67

Grid Search on Gradient Boosting

- learning_rate: [0.001, 0.01, 0.1], subsample:[0.5, 0.75], n_estimators:[300, 900, 4500]
- accuracy=0.76, precision=0.92, recall=0.66
- *Need More Exhaustive Grid Search*



Experiments with Deep Learning

1. Model 1: Embedding $\rightarrow$ garbage (Link)
2. Model 2: Embedding-LSTM $\rightarrow$ 59% accuracy (Link)
3. Model 3: Embedding-LSTM + Dense Output $\rightarrow$ 62% accuracy (Link)
4. Model 4: Embedding-LSTM + Dense Output + More Features $\rightarrow$ 65% accuracy (Link)
5. Model 5: Embedding-LSTM + More Features Model + Deep Output Layer $\rightarrow$ under progress (Link)

Lessons Learned

- Start with a simple model
- Minimize notebook usage
- Get familiar with TensorFlow end-to-end
- Perform EDA over the weekend
- Minimize unknown risks

Next Steps

- Tune existing models
- Use BERT and other transfer learning frameworks