Real or Not?

NLP with Disaster Tweets

Team 8
Yan Sun
Chao Ma
Ruonan Ren

Overview

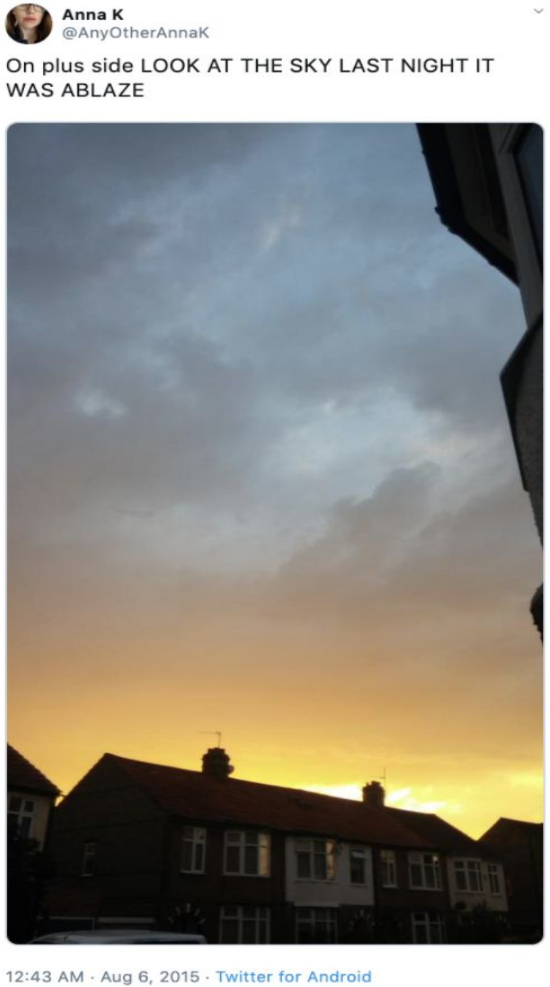
- Use Cases
- Goals
- Methodology
- TF-IDF
- Classification Algorithms
- Project Results
- Acceptance Criteria Review

Background

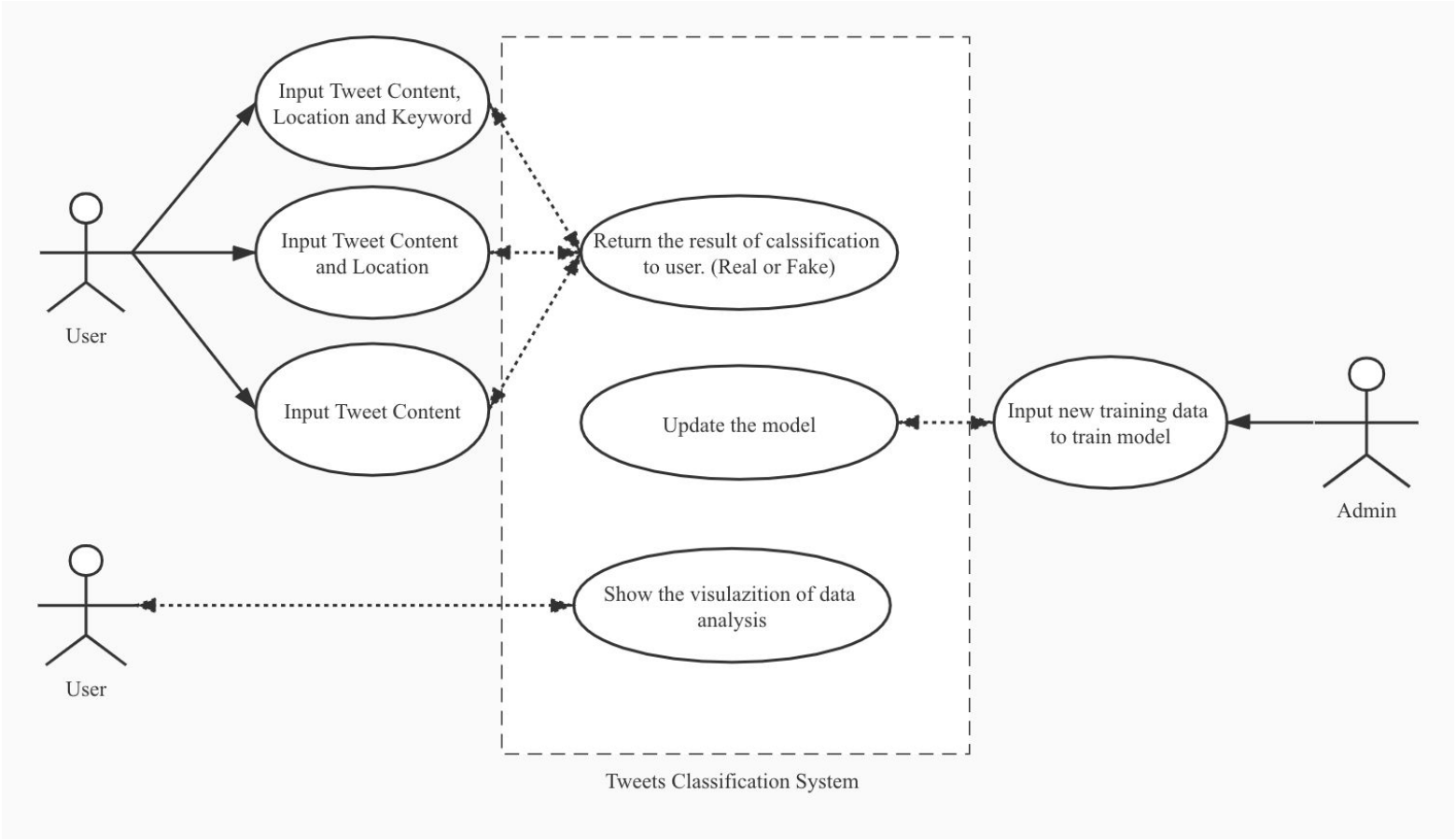
Twitter has become an important communication channel in times of **emergency**.

The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programatically 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.



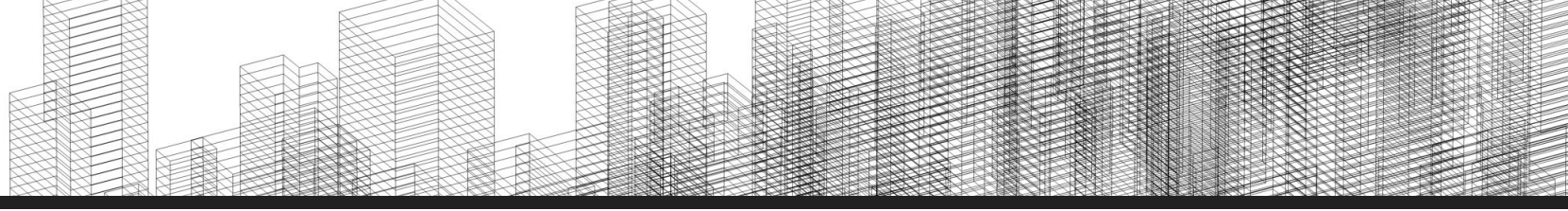
Use Cases Review



UI Design & Use Case

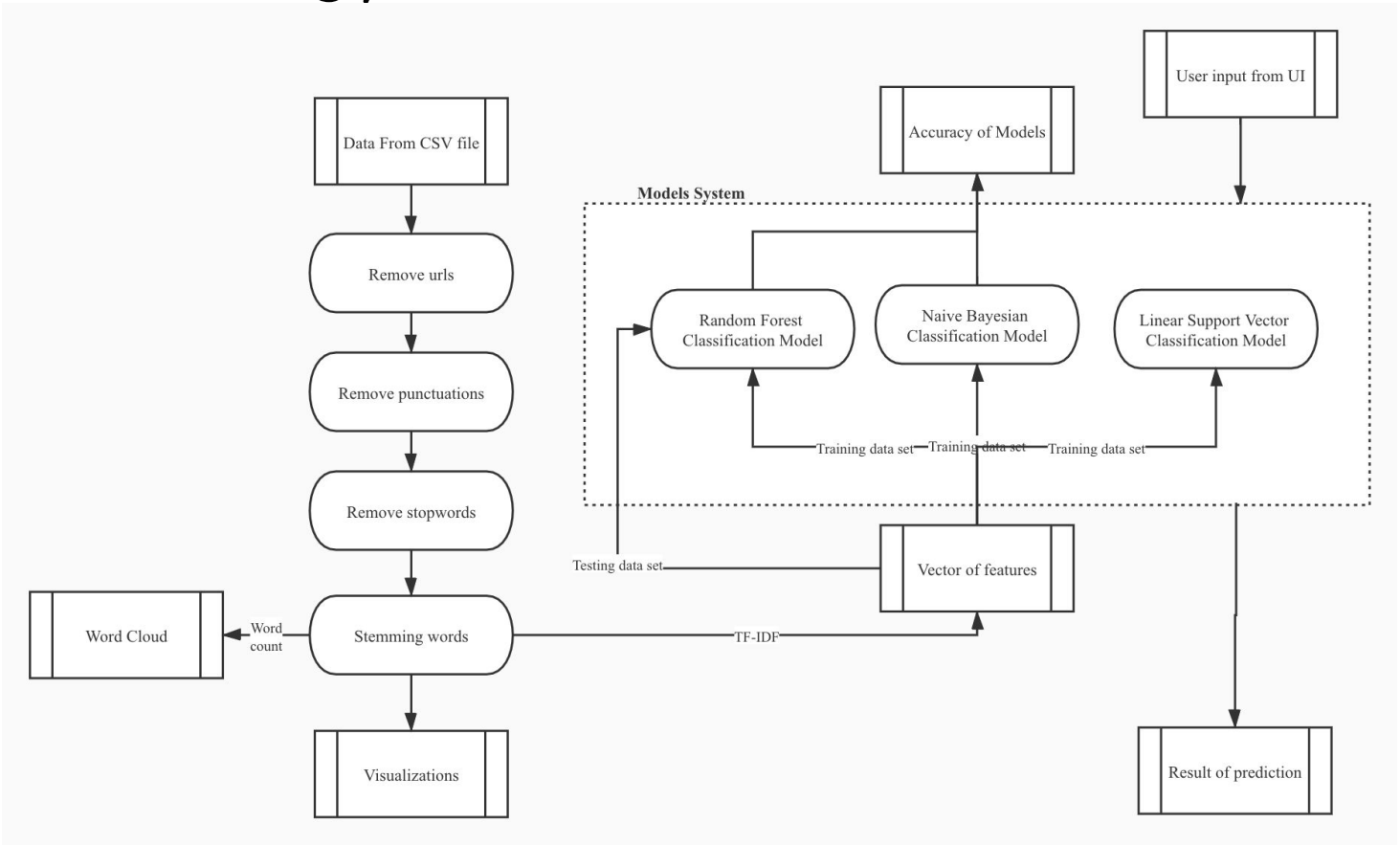
-

Goals



- Create a reactive page to detect fake news on twitter.
- Create a reactive page to analyze the characteristics of fake tweets.
- Get a well trained model for fake tweets prediction.

Methodology

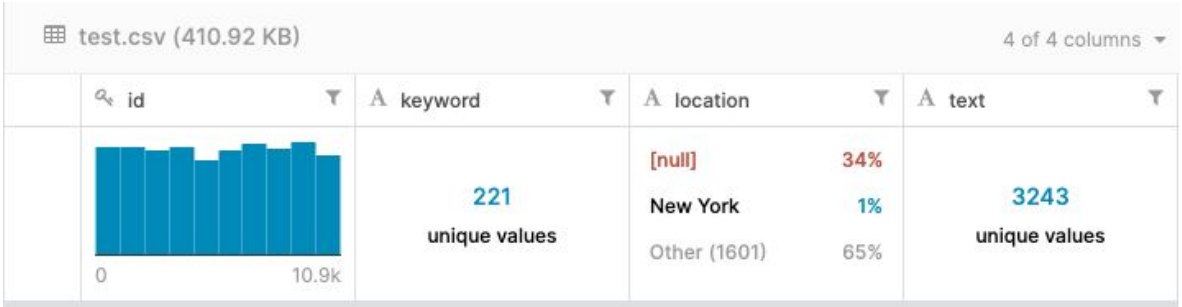


Methodology

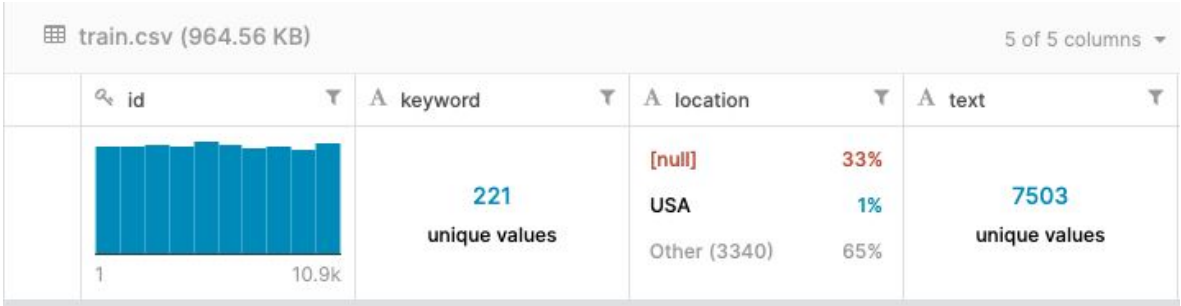
- Used **Zeppelin** for running code line by line
- Extracted features of tweets (nature language) by **TF-IDF**
- Implemented **3 classification algorithms**:
  - Random Forest Classifier
  - Naive Bayesian Classifier
  - Linear Support Vector Classifier
- Applied functions from Spark **MLlib**, Spark **SQL**, Spark **RDD**
- Designed and implemented UI by Scala **Swing**
- Visualized data by **Vegas**

Data Sources

- Data come from Kaggle competition



- Data magnitude is more than 10,000 rows



TF-IDF

Used to extract features and find keywords

- TF(t,d) is the number of times that term t appears in document d
- DF(t,D) is the number of documents that contains term t
- |D| is the total number of documents in the corpus

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D).$$

NLP

- Use Tokenizer to tokenize the tweets
- Use StopWordsRemover to remove the stop words

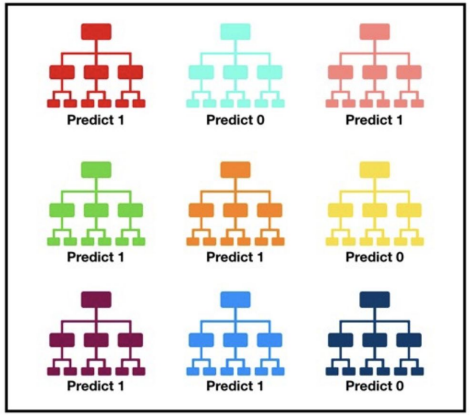
  Such as “ I, am, what, have, is, are..... ”
- Use HashingTF and IDF to vectorize the words
- Then we got featured data from nature language

Classification Algorithm

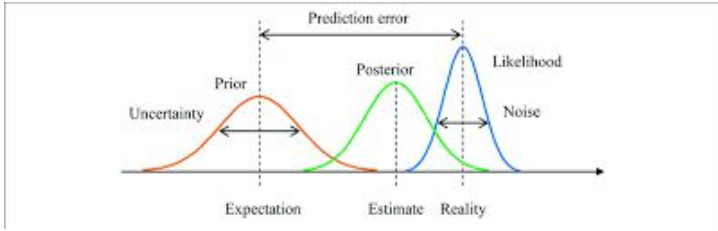
- In order to classify the tweets according to if it is a real disaster tweets, we implemented three different classification algorithms:
  - **Random Forest Classifier**
  - **Naive Bayesian Classifier**
  - **Linear Support Vector Classifier**

Random Forest Classifier

- A large number of relatively **uncorrelated** models (trees) operating as a committee will outperform any of the individual constituent models.
- Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples
- For b = 1, ..., B:
  1. Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
  2. Train a classification or regression tree fb on Xb, Yb.
- $$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$
- $$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$



Naive Bayesian Classifier



- Another assumption made here is that all the predictors have an equal effect on the outcome.
- X represents the features , y represents the label

$$X = (x_1, x_2, x_3, ..., x_n)$$

- According to Bayesian Theory

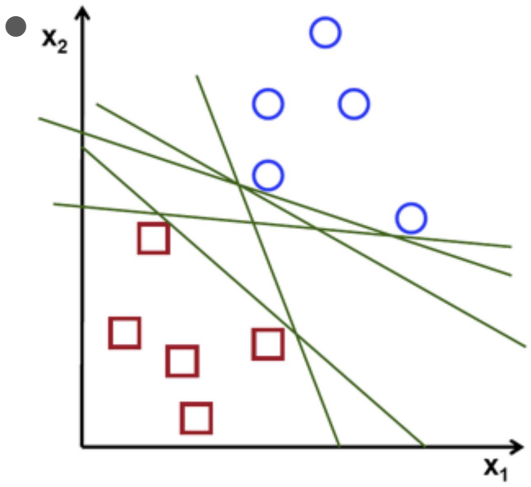
$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

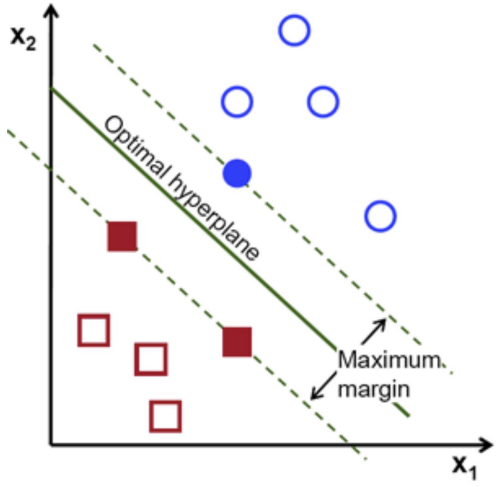
$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

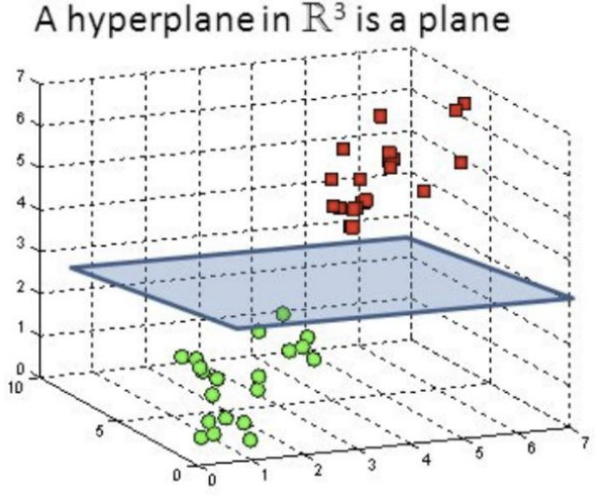
$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Linear Support Vector Classifier

- The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.







Linear Support Vector Classifier

- In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

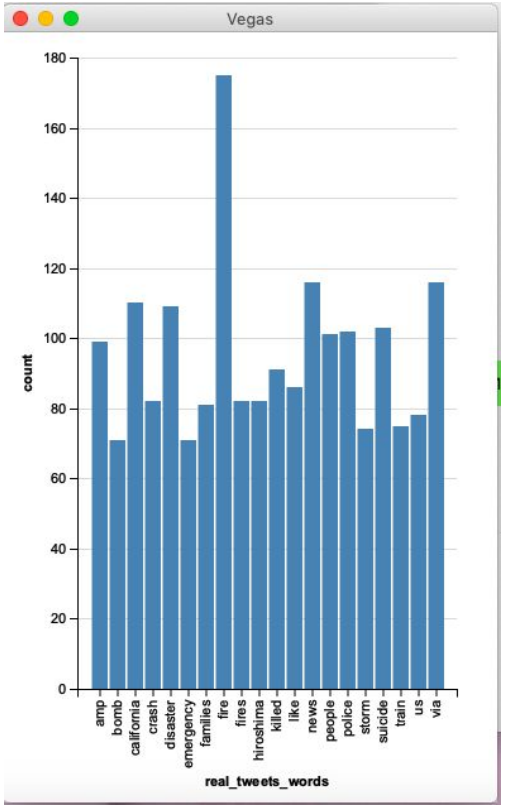
$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

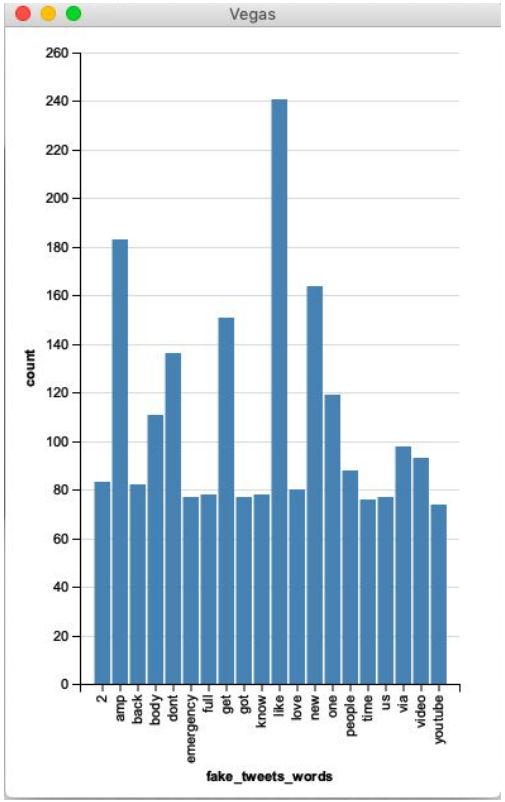
- Take partial derivatives with respect to the weights to find the gradients

$$\frac{\delta}{\delta w_k} \left(1 - y_i \langle x_i, w \rangle\right)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

- Update gradient (no misclassification) $w = w - \alpha \cdot (2\lambda w)$
- Update gradient (with misclassification) $w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$

Visualization of Keyword Extraction





Real Disaster Tweets

Fake Disaster Tweets

Evaluation of models

Model	Accuracy
Random Forest Classifier	0.7456170505328291
Naive Bayesian Classifier	0.8380480905233381
Linear Support Vector Classifier	0.8553220806062694

Acceptance Criteria Review

As a user, I am able to input Disaster Tweet content, location and keyword to get the prediction if the tweet is fake:

- The prediction accuracy for complete input data should be over 70%
- The time to respond should be under 5 seconds

As a user, I am able to show the visualization of data analysis:

- The time to respond should be under 5 seconds

Thank you!