

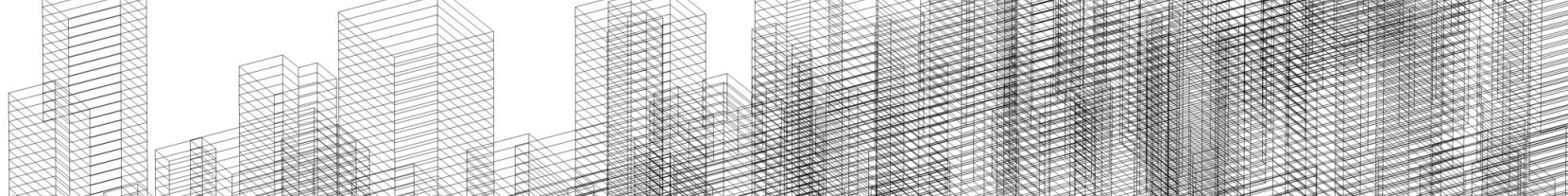
A large fire is burning at night, with bright orange and yellow flames rising from a structure. In the background, a bus and a building are visible, also partially obscured by the fire. The scene is dark, with the fire providing the primary light source.

# Real or Not?

## NLP with Disaster Tweets

Team 8  
Yan Sun  
Chao Ma  
Ruonan Ren





# Fake Disaster Tweets Prediction



## Group 8

Chao Ma  
Yan Sun  
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**.

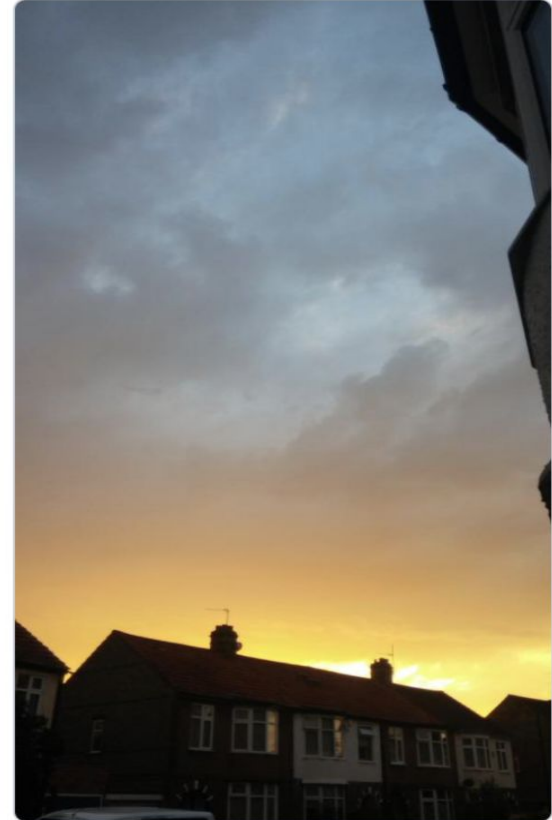
The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programatically monitoring Twitter (i.e. disaster relief organizations and news agencies).

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



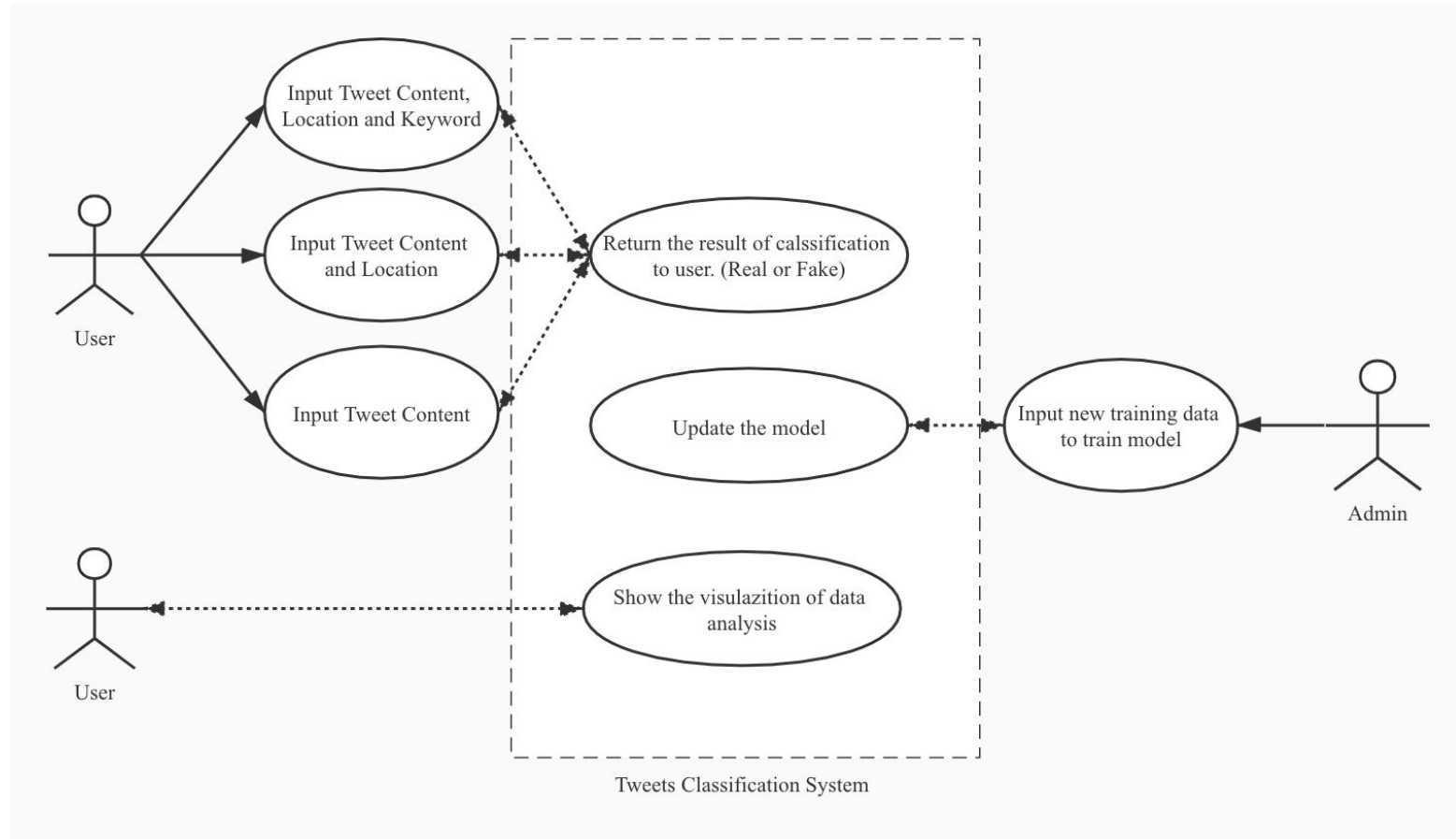
Anna K  
@AnyOtherAnnaK

On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE



12:43 AM · Aug 6, 2015 · [Twitter for Android](#)

# Use Cases Review

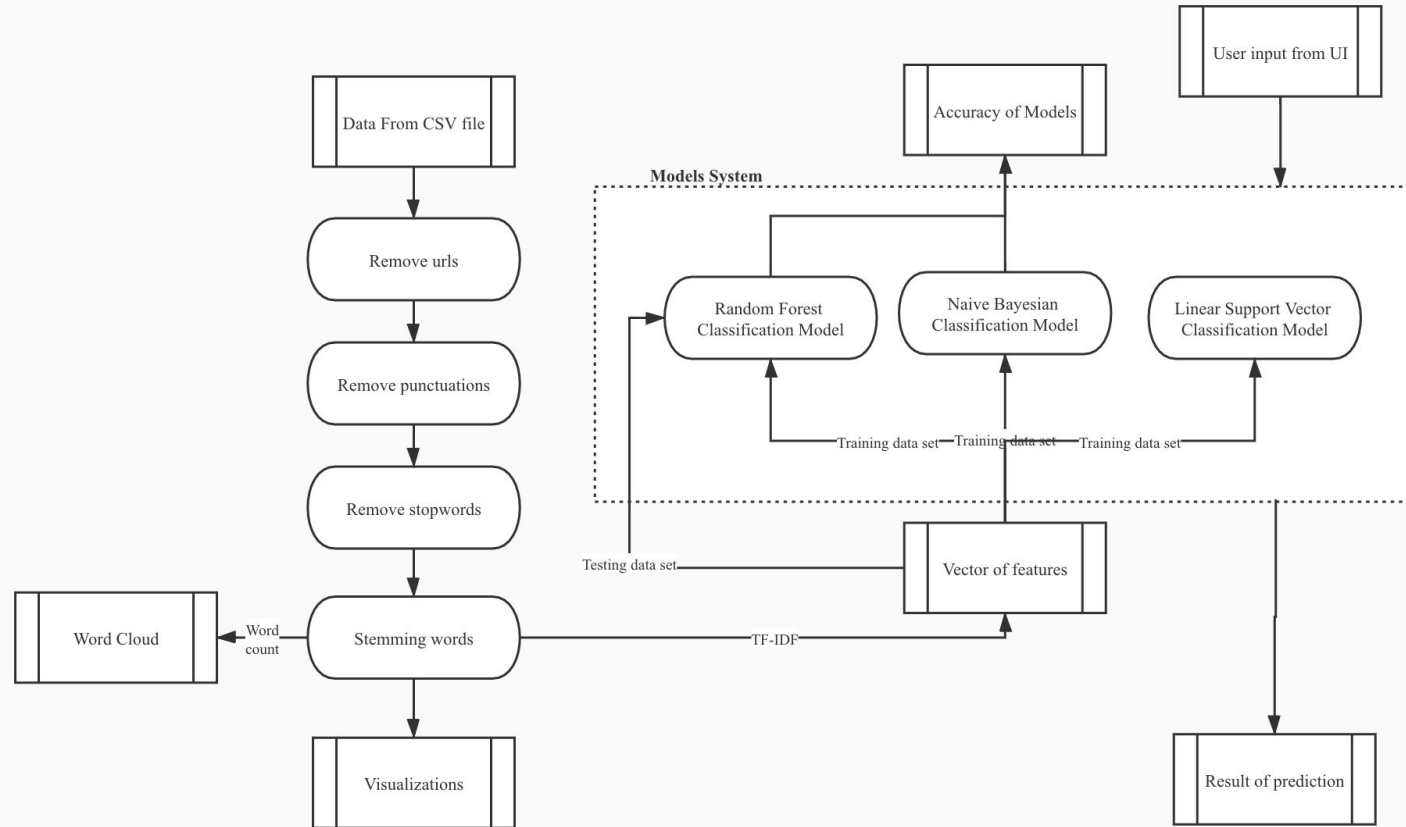


# Goals

- 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



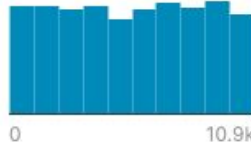


# Methodology

- 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

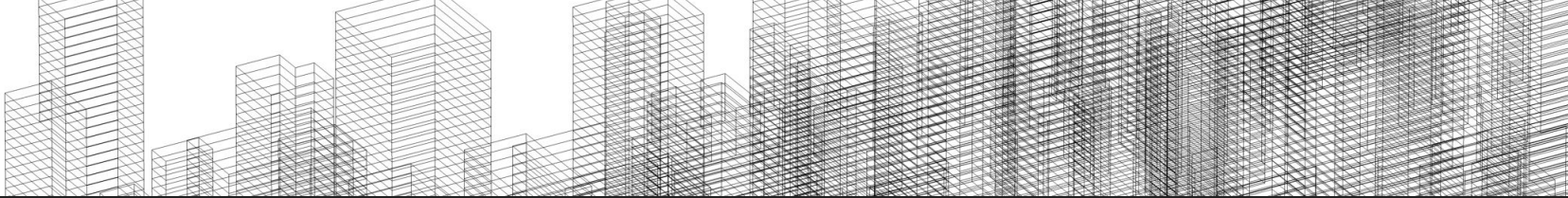
- Data come from Kaggle competition

test.csv (410.92 KB)				4 of 4 columns ▾	
	🔍 id ▾	🔍 keyword ▾	🔍 location ▾	🔍 text ▾	
	 0 10.9k	221 unique values	<div><div>[null] 34%</div><div>New York 1%</div><div>Other (1601) 65%</div></div>	3243 unique values	

- Data magnitude is more than 10,000 rows

train.csv (964.56 KB)				5 of 5 columns ▾	
	🔍 id ▴	🔍 keyword ▴	🔍 location ▴	🔍 text ▴	
	 1 10.9k	221 unique values	[null] 33% USA 1% Other (3340) 65%	7503 unique values	

# 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

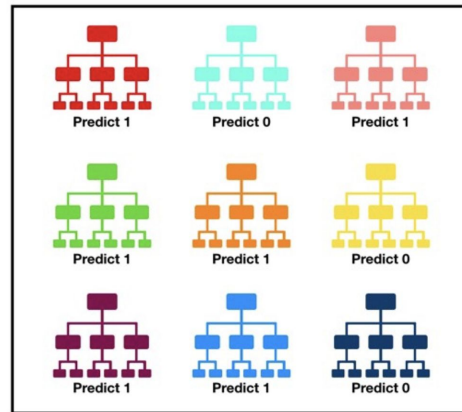
- In order to classify the tweets according to if it is a real disaster tweets, we implemented three different classification algorithms:
  - Random Forest Classifier
  - Naive Bayesian Classifier
  - Linear Support Vector Classifier

# Random Forest Classifier

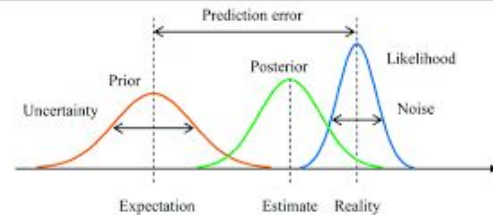
- A large number of relatively **uncorrelated** models (trees) operating as a committee will outperform any of the individual constituent models.
- Given a training set  $X = x_1, \dots, x_n$  with responses  $Y = y_1, \dots, y_n$ , bagging repeatedly ( $B$  times) selects a random sample with replacement of the training set and fits trees to these samples
- For  $b = 1, \dots, B$ :
  1. Sample, with replacement,  $n$  training examples from  $X, Y$ ; call these  $X_b, Y_b$ .
  2. Train a classification or regression tree  $f_b$  on  $X_b, Y_b$ .

$$\hat{f} = \frac{1}{B} \sum_{b=1}^B f_b(x')$$

$$\sigma = \sqrt{\frac{\sum_{b=1}^B (f_b(x') - \hat{f})^2}{B - 1}}.$$



# 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

$$X = (x_1, x_2, x_3, \dots, x_n)$$

- According to Bayesian Theory

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

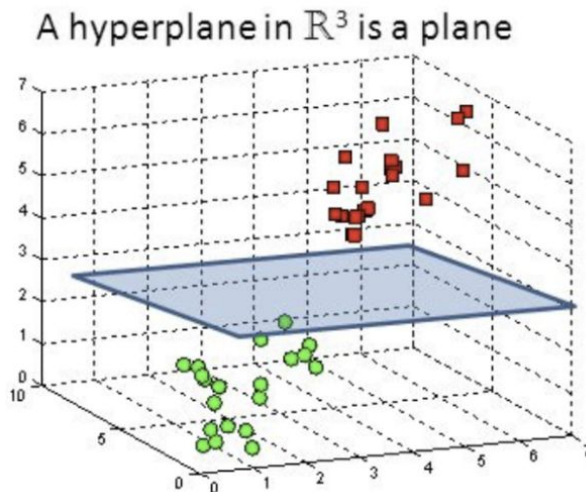
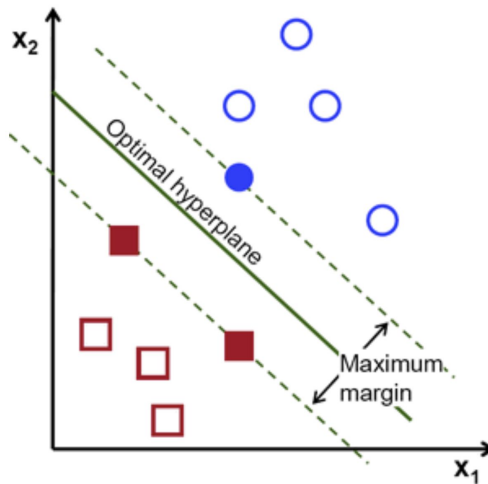
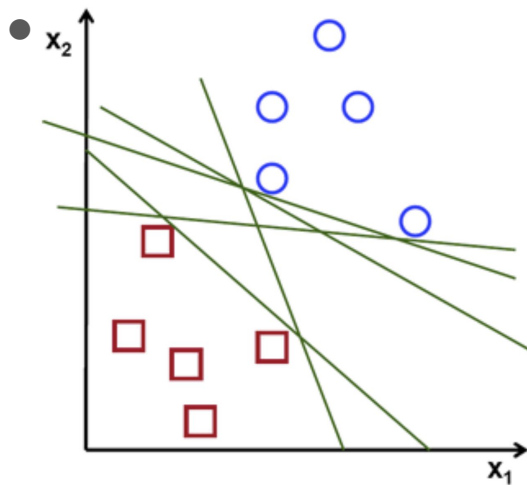
$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y)$$

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



# 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

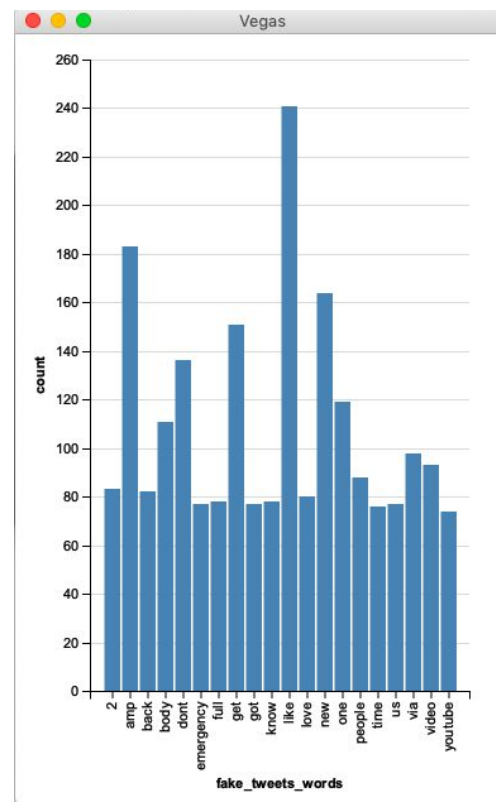
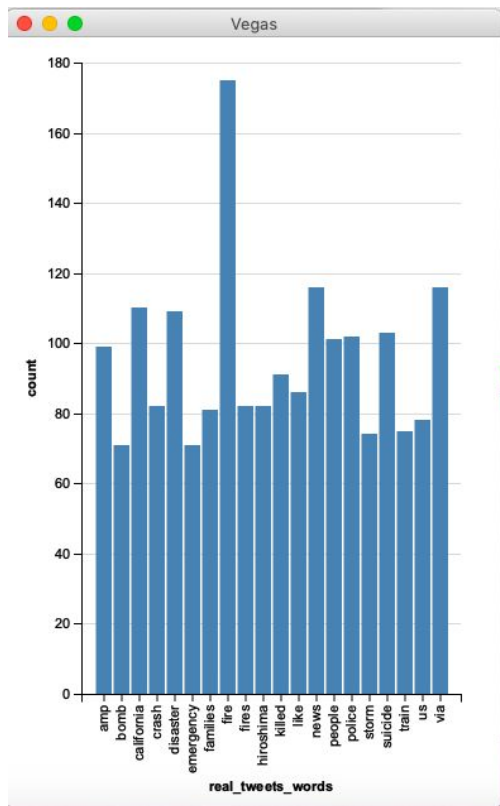
- In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

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

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

- Take partial derivatives with respect to the weights to find the gradients  $\frac{\delta}{\delta w_k} (1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$
- 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







# UI Design & Use Case

-

# 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!