Automated News Classification and Verification

Apoorv Anand
College of Computer and Information Science, Northeastern University
anand.ap@husky.neu.edu

Under Guidance of: -

Professor David A. Smith

1. ABSTRACT

"A lie gets halfway around the world before the truth has a chance to get its pants on."
– Winston Churchill

In the times when people used to get news from newspaper/radio/TV, the news used to have natural categories such as Politics, Business, Sports and Entertainment. But, with the advent of the social media and its becoming the primary source of news, a new category of news has emerged – Fake News. Fake news is the news which is not true and deceptive in nature. Fake news can create false perceptions in reader's mind which in turn can result in shocking consequences. To tackle this I used the NLP techniques to classify the news as fake or as Real. I have used four classification techniques for this purpose namely, Multinomial Naïve Bayes, Gaussian Naïve Bayes, Random Forest and Stochastic Gradient Descent. The best results were obtained by Stochastic Gradient Descent.

2. INTRODUCTION

Since the 2016 US Presidential elections various political pundits have claimed that the emergence of the fake news influenced the elections. Motivated by these claims Stanford and New York University did a study which concluded otherwise. Nonetheless, the News which are significantly biased and untrue regularly make their way to billions of user's feeds.

"What is fake news?"

A deliberately misleading story is "fake news" but lately blathering social media discourse, is changing its definition. A satire or sarcastic news is sometimes also considered as Fake News because of poor human ability to distinguish truth from deception.

For this project, I was interested in assessing classic NLP techniques and using it to distinguish fake news from real news. For this purpose, I assembled a dataset of fake and real news and employed four classifiers namely - Multinomial Naïve Bayes, Gaussian Naïve Bayes, Random Forest and Stochastic Gradient Descent, to create a model to classify an article as fake or real based on its lexical features.

3. RELEVANT WORK

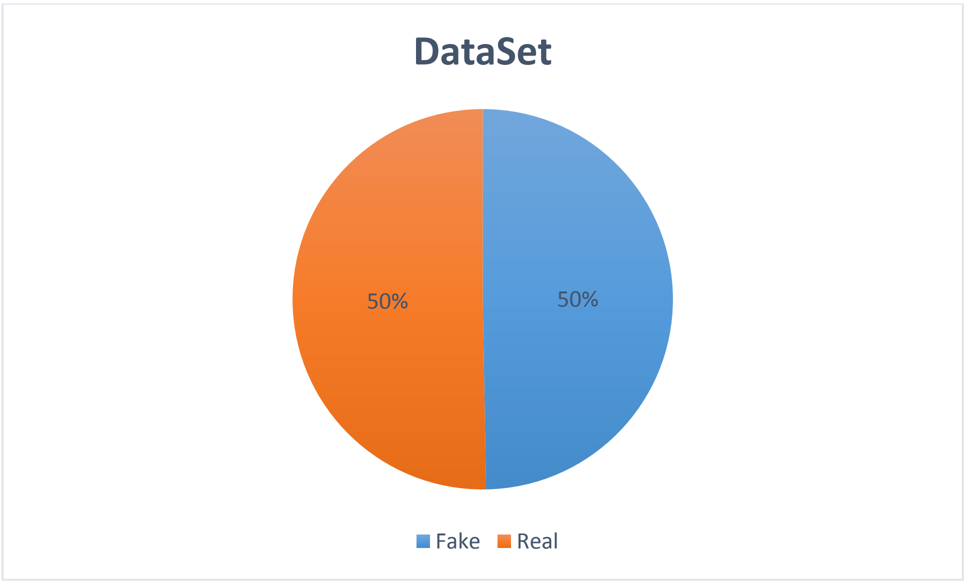
Lot of research is currently going on to tackle the problem of fake news. The Data Science community has responded by building models which can help predict if the news is fake or not. But, there are no assuring results yet. Facebook is most hardly hit by this issue and is currently trying to employ AI to tackle this problem [4]. Few publications have tried to address this issue and I have read a couple of them. The best publication related to my project is ‘Deception Detection Methods for News Discourse’ by Victoria L. Rubin, Niall J. Conroy, and Yimin Chen from University of Western Ontario, Canada. [5]

4. DATASET

To perform the classification and evaluation I used the BBC data set [1] as training data set for Real News. This dataset includes 2225 news articles corresponding to stories from five topical areas from 2004-2005. The five natural classes are (Business, Politics, Sports, Entertainment, Technology). The dataset has news articles from the five natural categories in five different folders. The folder name specifies the category or class the article belongs to. These folder names were used as labels while training the classifiers on the data set.

For the Fake news, I have used Kaggle Fake News dataset[11]. It contains text and metadata scraped from 244 websites tagged as “fake” by the BS detector, a chrome extension [2] that marks unreliable sources as fake. The dataset contains 13,000 articles published during the 2016 election cycle.

Dataset consisted of two categories ‘Fake News’ and ‘Real News’.In total, there were 4425 articles in this dataset out which 2200 were fake and other belonged to real news categories. Only 2200 fake articles were taken to make the total null accuracy approximately 50% The task was to identify the fake news from this data set.



5. DATA PRE-PROCESSING

I have 2225 news articles from BBC in separate folders based on the category they belonged to. These articles were in separate text files and in total there were 2225 news articles which were real.

The Fake News dataset comprised of a csv file in which the 'text' column had the article body and there were several other columns which contained the meta data such as source, time, title of these articles. I wrote a script to extract only the article body out of this csv and create a separate text file for each article. In total, I got 13000 news articles which were fake.

Data Pre-processing consisted of three main steps:

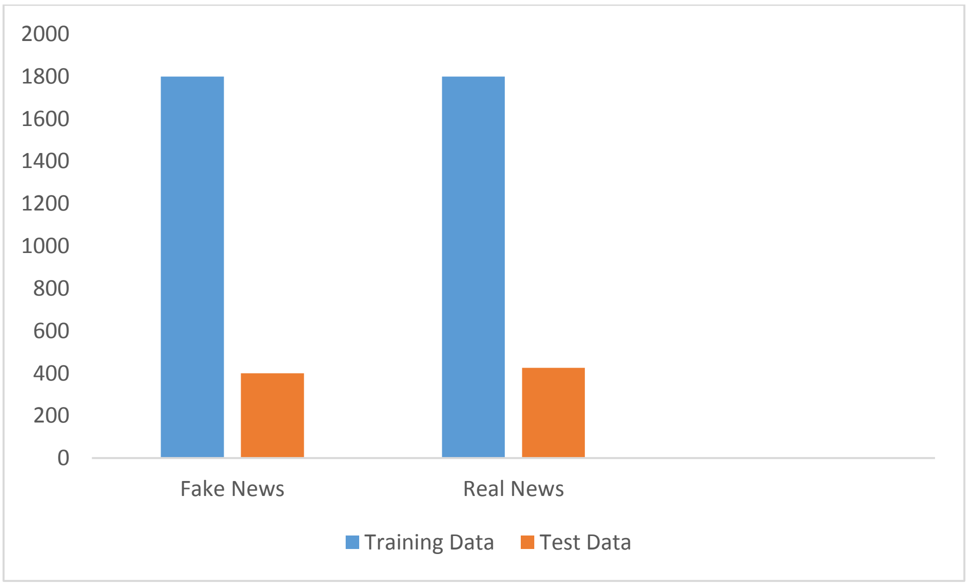
1. Removed all the characters which were not alphabets i.e. included only the characters from a-z and A-Z.
2. Removed all the stop words using NLTK.
3. Stemmed the words using Snowball Stemmer (part of NLTK package).

The above steps helped to reduce the vocabulary space and improve the size of the index and thus resulted in a better model.

6. METHODOLOGY

To test the four classifiers, I divided the two datasets into training and testing. The breakup of division of both the datasets into training and test is as following :-

Dataset



The next step was text transformation i.e. using count vectorizer and tf-idf vectorizer. This gave me two pairs of datasets to work with. One used n number of most common features (count vectorized) and the other used the words that appeared at least n number of times in the corpus(tf-idf vectorizer).

On further observation, I found out that my models worked best with the tf-idf vectorized data. Thus, I dumped the count vectorizer and used tf-idf vectorizer for all the classifiers.

EXTRACTING FEATURES: -

The features were extracted from the raw text document and fed to the classifier as matrix of tf-idf.

Tf means **term-frequency** while tf–idf means term-frequency times **inverse document-frequency**: $\text{tf-idf(t,d)} = \text{tf(t,d)} \times \text{idf(t)}$.

Following tf-idf algorithm was used to convert raw document into a matrix of tf-idf features.

The term frequency, the number of times a term occurs in a given document, is multiplied with idf component, which is computed as

$$\text{idf}(t) = log\frac{1+n_d}{1+\text{df}(d,t)} + 1,$$

where n_d is the total number of documents, and $\text{df}(d,t)$ is the number of documents that contain term t. The resulting tf-idf vectors are then normalized by the Euclidean norm:

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{v_1{}^2 + v_2{}^2 + \dots + v_n{}^2}}.$$

IMPLEMENTING THE CLASSIFIERS

I selected to evaluate Multinomial Naïve Bayes, Gaussian Naïve Bayes, Random Forest and Stochastic Gradient Descent, due to their ability to perform supervised learning on multiclass datasets. In the following section I explain each algorithm and how it was used for goal of News Classification.

i. ***Multinomial Naïve Bayes Classifier***
implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{|T|} N_{yi}$ is the total count of all features for class y.

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called *Lidstone* smoothing.

The Scikit [3] provides the MultinomialNB class which I used for implementing this classifier.

ii. ***Gaussian Naïve Bayes Classifier***
implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.
The Scikit [3] provides the GaussianNB class which I used for implementing this classifier

iii. ***Random Forest Classifier***
In Random Forest Classifier each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. Because of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

I have used the Scikit [3] implementation class of RandomeForestCassifier which combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class. The number of estimators or decision trees considered is 100 and have been specified as parameters.

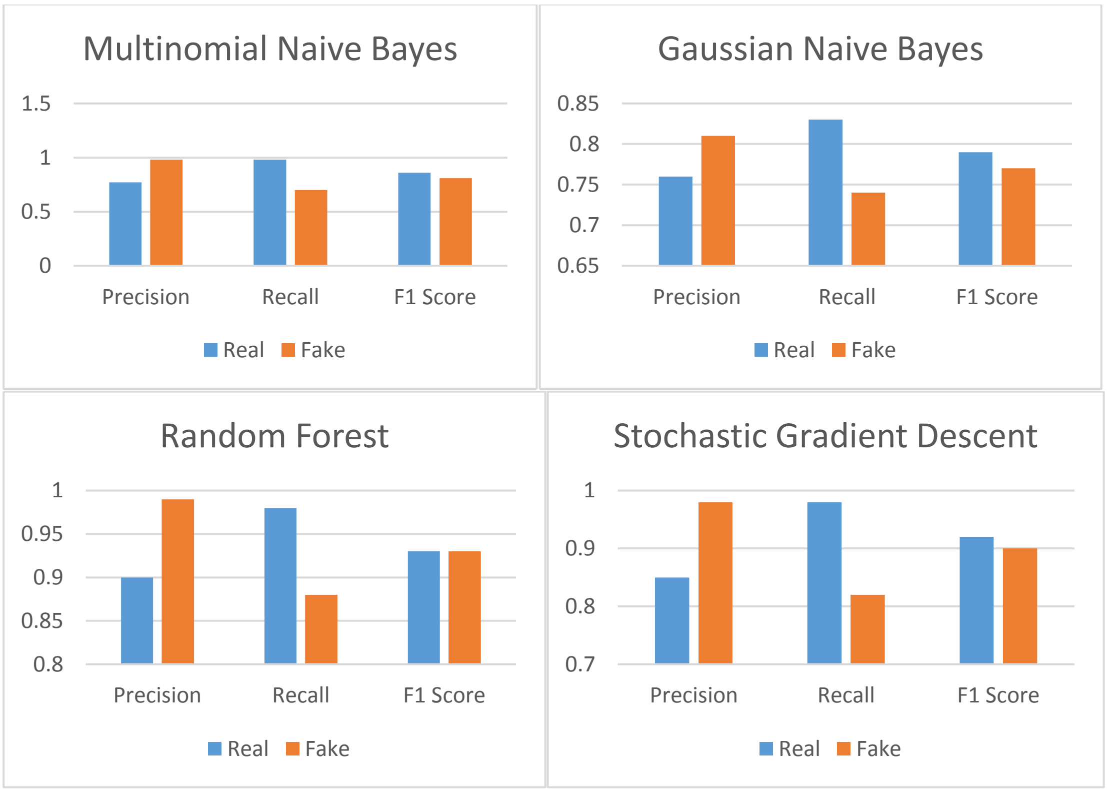
iv. ***Stochastic Gradient Descent Classifier***
Stochastic gradient descent (often shortened in **SGD**), also known as **incremental** gradient descent, is a stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions. In other words, SGD tries to find minima or maxima by iteration.
I have used the Scikit [3] SGDCCclassfier class. The SGDCCclassfier class implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification. As other classifiers, SGD has to be fitted with two arrays: an array X of size [n_samples, n_features] holding the training

samples, and an array Y of size [n_samples] holding the target values (class labels) for the training samples. After being fitted, the model can then be used to predict new values.
SGDClassifier supports multi-class classification by combining multiple binary classifiers in a “one versus all” (OVA) scheme. For each of the K classes, a binary classifier is learned that discriminates between that and all other K-1 classes. At testing time, we compute the confidence score (i.e. the signed distances to the hyperplane) for each classifier and choose the class with the highest confidence.

7. **RESULTS AND DISCUSSSION**

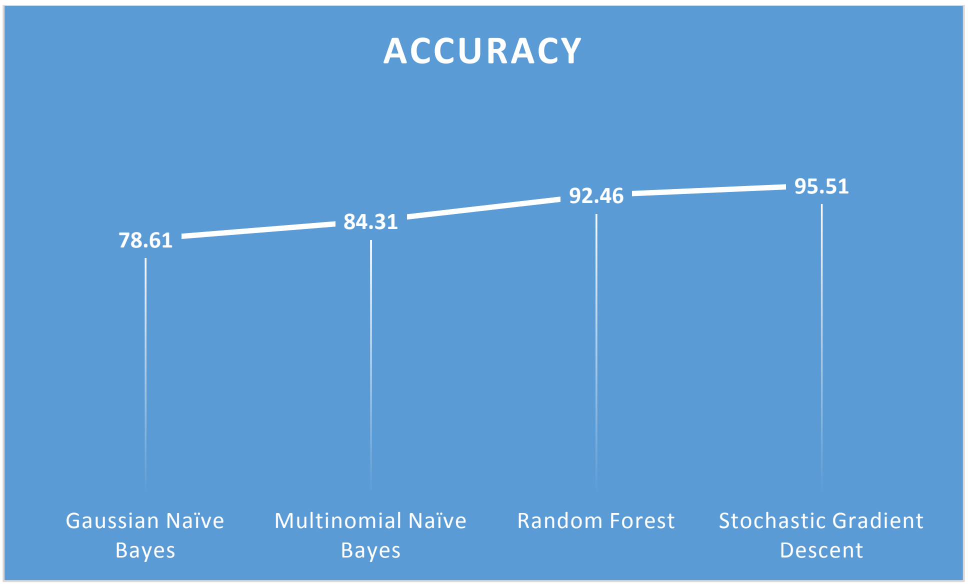
The Below graph gives the Precision, Recall and F1 scores for all the four models. Precision is the class agreement of the data labels with the positive labels given by the classifier, while Recall is the effectiveness of a classifier to identify positive labels [7]. The F1-score is the harmonic mean of precision and recall.
The best precision and recall was obtained in the Random Forest and Stochastic Gradient Descent model. Naïve Bayes performed significantly well despite its dependency on the conditionality.

Figure shows that the Precision for fake news is highest in case of these two models



Best accuracy was achieved by the Stochastic Gradient Model. Accuracy has been calculated as what percentage of predictions made by the model were true.

Although, this does not states the ability to identify fake news (which was the primary goal of this project) as precisely as it can be found out via Precision and Recall, nonetheless it gives the estimate of the scenario when the model was able to distinguish between the fake and the real news.



8. FUTURE W0RK AND CHALLENGES

The future work includes developing a model which can self-train itself. This model can be used in predicting the articles that are published on twitter channels and if the prediction is correct, which can initially be determined by the source of the news article, then the model will self-train itself. The challenge in this task is finding a channel/source which continuously posts fake news.
There exists a Wikipedia link which lists all the fake news websites [6]. The news posted on these sites can also be used for self-training purpose and in ideal scenario one will be able to find a real news article corresponding to a fake news and this will be most accurate test of the model.
The future work also includes combining meta data along with NLP techniques to achieve higher accuracy. The meta data will include the source of the article and based on the flagging technique used by the BSDetector[4] one can also use the meta data to better understand the origin of the article and its probability of being fake.

9. CONCLUSION

In the paper [5] the author has very aptly said about this issue and I would like to quote -
'Though this work is technologically and methodologically challenging, it is timely and carries potential benefits to news consumers. In principle news verification system can

improve credibility assessment of digital news sources. The mere awareness of potential deception can increase new media literacy and prevent undesirable costs of mistaking fake news for authentic reports.'

Towards News Verification: Deception Detection Methods for News Discourse. Available from: [accessed Apr 24, 2017].

10.REFERENCES

1. BBC Dataset from D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006.
2. BS Detector by Daniel Sieradski http://bsdetector.tech/ https://chrome.google.com/webstore/detail/bs-detector/dlcgkekjiopopabcifhebmphmfmdbjod?hl=en
3. http://scikit-learn.org
4. https://techcrunch.com/2016/11/14/facebook-fake-news/
5. https://www.researchgate.net/publication/270571080_Towards_News_Verification_Deception_Detection_Methods_for_News_Discourse
6. https://en.wikipedia.org/wiki/List_of_fake_news_websites
7. M. Sokolova and G Lapalme, "A systematic analysis of performance measures for classification tasks," in Information Processing and Management, vol. 45, no. 4, pp. 427-437, 2009
8. https://web.stanford.edu/~gentzkow/research/fakenews.pdf
9. https://opendatascience.com/blog/how-to-build-a-fake-news-classification-model/
10. http://www.fakenewschallenge.org/
11. https://www.kaggle.com/mrisdal/fake-news
12. http://thehill.com/homenews/media/317646-fake-news-did-not-change-result-of-2016-election-study