Fake News Detection

Nisarg Antony (800989573)

Dhiksha Ramkumar (800987925)

1. Introduction

With the advent of social media and its ever increasing role as a platform for dissemination of news, it has become incredibly easy for any entity to produce and circulate content for others to consume, regardless of its authenticity. Some of the fundamental tenets of journalism like fact checking and accountability tend to be overlooked or downright ignored when it comes to content publication on social media platforms like Twitter and Facebook. The nature of fake news while seemingly harmless can have a significant impact on real world events as evident in the case of the 2016 US presidential elections. The extensive propagation of fake news can affect the balance of the news ecosystem. Fake news intentionally persuades consumers to accept biased or fake beliefs. It is often spread by propagandists to sway people's beliefs and amass influence. Fake news changes the way people interpret and respond to real news. It is sometimes spread with the intent to trigger peoples distrust and cause confusion thereby impeding their ability to differentiate between what's fake and what's not.

This project explores the application of classification techniques to identify news sources that produce fake news stories. It is a well-known fact that people and groups with potentially malicious agendas initiate fake news in order to influence events and policies around the worlds. A brief overview of related work is provided along with some explanation on the data and the models that run on it. Also included is a brief summary of the feature extraction and pre-processing efforts that have to be undertaken before the models can be generated form our data. We will use a corpus of labeled real and fake news articles to build a classifier that would classify fake news articles for the corpus. We are going to use a text classification approach to the problem, making use of multiple different classification models and comparing their respective results. Some of the models that we plan on implementing include but are Naïve Bayes classifier, Support Vector Model (SVM), Random Forest network and Logistic Regression. The datasets used in the paper was taken from Kaggle. Before training the models, it is necessary to pre-process the dataset and extract the features of interest.

In this project we will undertake this particularly complex task by building classification models that can make decisions on the "fakeness" of articles present in the corpus. Since the data contains articles originating from mixed sources, we will focus on the content of the articles rather than the sources they originate from. This report will include visualizations and metrics to help the reader gain a much better understanding of the models and the differences between them. The final goal is to create a model that would demonstrate a high level of accuracy in the classification of fake news.

The report is divided into numerous sections. Section 1 provides a brief introduction of the project detailing the topic, its application and an overview. Section 2 gives a brief description of the previous work that has been undertaken in the area of fake news detection. Section 3 expands on the project, describing the various aspects of the project like the approach, datasets used, models implemented and results generated. Section 4 summarizes the project, linking the various aspects of the project and providing the bigger picture. Section 5 provides a conclusion and future scope. Section 6 lists all the references cited in this report.

2. Background

Identifying fake news has been somewhat of a hot topic in recent times. A variety of techniques have been proposed in this regard by a number of individuals and organizations. One such undertaking is Stanford University's popular paper on this subject titled "Stance Detection for Fake News identification" [1]. This paper uses stance detection to address the problem of fake news detection. The authors achieve this goal by feeding the headline-article pairs into a bidirectional LSTM which first analyzes the article and then uses the acquired article representation to analyze the headline. On top of the output generated from the bidirectional LSTM model, they concatenate global statistical features extracted from the headline-article pairs.

Another related work is the paper on "Evaluating machine learning algorithms for fake news detection" by Shlok Gilda [2]. This paper explores the application of natural language processing techniques for the detection of fake news which it defines as misleading news sourced from non-reputable sources. The crux of the paper is about comparing the performance of models using three distinct feature sets to find the factors that are most predictive of fake news: TF-IDF using bigram frequency, syntactical structure frequency and a combined feature union. The study uses a dataset from Signal Media and a list of sources from OpenSources.com and applies the techniques of TF-IDF and probabilistic context free grammar (PCFG) detection to a corpus of about 11,000 articles. The authors test their dataset on several classification algorithms such as SVM's, Stochastic Gradient Descent, Gradient Boosting, Bounded Decision Trees and Random forests. It is observed that the TF-IDF model of bigrams fed into a Stochastic Gradient Descent model has the highest probability of identifying fake news with an accuracy of 77.2%. The paper states that the best performing models based on the metrics of ROC and AUC are Stochastic Gradient descent model trained on the TF-IDF feature set.

Another relevant work is text classification. A paper on "Convolutional neural networks for sentence classification" by Y. Kim uses a convolutional neural network to classify sentences. Here convolutional filters are used to extract different n-grams depending on the filter size which are known to be useful for text classification.

Although much effort has been made in the quest to identify fake news, there remain a few unaddressed issues. In the Stanford paper, the bidirectional LSTM used by the authors only classifies around 31% of disagree pairs correctly. This can be attributed to a limited amount of data for the disagree label. Another potential shortcoming can be seen in the paper by Shlok Gilda wherein the models using absolute probability thresholds may not turn out to be the most reliable models considering that the probability scoring is not well-calibrated.

3. Project

3.1 Approach

The proposed approach to deal with the fake news detection problem aims at training a model with a dataset consisting of text data and label of the text as fake or real. The dataset is first pre-processed and then fed to the Doc2Vec model to create vector representations of all the text rows in the dataset. The vectors are then fed to classification models such as Naive Bayes, SVM, Random Forest and Logistic Regression and their metrics are calculated to find the best model to conduct classification of fake news. Testing on the same dataset was not an option as the testing dataset provided, did not contain the label field and so the accuracy

measures could not be calculated. So, we divided the dataset into 80-20 split and tested on the 20% and achieved an accuracy of almost 80%. Testing of our model was also done on an entirely different dataset. This new dataset had only political news whereas our original dataset had a mixture of news topics. This did affect the accuracy of the model to a great extent but it did give an accuracy of approximately 38%.

3.2 Data

The 2 datasets used in the report are taken from Kaggle. Pre-processing is done on the dataset in order to train the models. The first dataset has a training size of approximately 21,000 rows and a testing size of approximately 6000 rows. The dataset has the following attributes

1. Id: unique id for new article
2. Title: title of the news article.
3. Author: author of the news article.
4. Text : the content of the news article
5. Label: label indicating whether the news article is genuine or fake.

The second dataset is similar to the first dataset with the only difference being the lack of an author attribute. It has a total of 6335 rows and 4 columns.

3.3 Pre-Processing

The dataset picked included 558 empty title cells, 1957 empty author cells and 39 empty text cells. The title attribute was not of any use for the classification and so all those rows were removed and having no text would not mean anything for the prediction as the model was focused on making text the driver variable, so those rows were removed as well. The number of empty author cells were comparatively higher and the "author" attribute was initially used for the classification, so naive bayes was used to predict the authors, making an assumption that the authors could only possibly be the ones existing in the data set. However this does not impact the classification performance because it was later dropped as a potential candidate feature because the second dataset used to test the model did not have the "author" field. Then the text column went through pre-processing which included removing stop words, using lemmatizers, converting to lowercase, etc for a more even text to be fed to the Doc2Vec model.
Gensim's doc2vec was used to create vectors of each of text rows that could further be used to classify labels. A function for getting tagged documents were created as the inbuilt function could only deal with multiple files and not rows in a dataframe. This was then used to create Doc2Vec model and the model was trained on that vocab. The model then produced a 300x1 embedding vector for each article and this was then used in classification algorithms.

3.4 Models

The models implemented in the project are detailed below:

3.4.1 Naïve Bayes

Naïve Bayes methods are a set of supervised learning algorithms based on applying Bayes'

theorem with the "naïve" assumption of conditional independence between every pair of features, given the value of the class variable. Here we have used the scikit-learn implementation of Gaussian Naïve Bayes. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\, P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Equation 1: Bayes Theorem

The different Naïve Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.In spite of their apparently over-simplified assumptions, naïve Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. Naïve Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution [5]. This in turn helps to alleviate problems stemming from the curse of dimensionality. On the flip side, although Naive Bayes is known as a decent classifier, it is known to be a bad estimator.

3.4.2 Support Vector Machine

Support vector machines are a set of supervised learning methods used for classification, regression and outlier detection. They are effective in high dimensional spaces and in cases where the number of dimensions is greater than the number of samples. The algorithm uses a subset of training points in the decision function, so it is also memory efficient. Another advantage of using SVM is that different kernel functions can be specified for the decision function. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

$$\operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho\right)$$

Figure 1: Decision function for SVC

The decision function is fully specified by a (usually very small) subset of training samples, the support vectors. Here we have used the Support Vector Classifier (SVC) function of the scikit-learn module.

3.4.3 Logistic Regression

Logistic regression is a method for fitting a regression curve, $y = f(x)$, when y is a categorical variable. The typical use of this model is predicting y given a set of predictors x. The predictors can be continuous, categorical or a mix of both. The categorical variable y, in general, can assume different values. In the simplest case scenario y is binary meaning that it can assume either the value 1 or 0. A classic example used in machine learning is email

classification: given a set of attributes for each email such as number of words, links and pictures, the algorithm should decide whether the email is spam (1) or not (0).

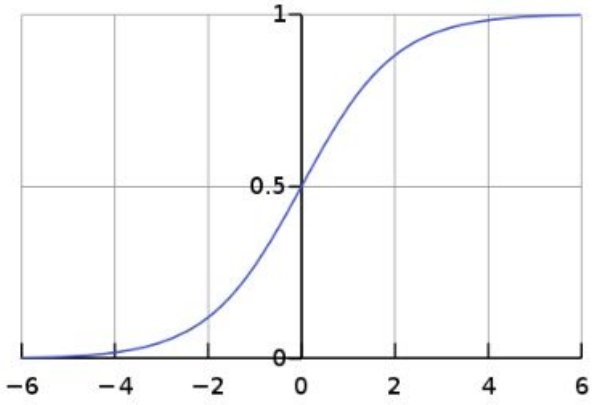


Figure 2: Logistic Function

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

3.4.4 Random Forest

The Random forest algorithm works by training multiple weak classification trees using a fixed number of randomly selected features, then taking the mode of each class to create a strong classifier. Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data. However, as this method selects a limited number of features in each iteration, the performance of random trees is faster. With a few exceptions, a random-forest classifier has all the hyperparameters of a decision tree classifier and all the hyperparameters of a bagging classifier, to control the ensemble itself. The random-trees method introduces an element of randomness into the model. Instead of looking

for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process generally results in a better model.

3.5 Distribution of work

The distribution of work done in the project is as follows:

Project Work Contribution		
Index	Team Member	Contribution
1	Nisarg Antony	Preprocessing (including removal of punctuation, converting all text to lower-case, removing stop words, lemmatizing the words predicting missing values and creating bag of words), comparing models.
2	Dhiksha Ramkumar	Feature extraction using Doc2Vec for generating word embeddings, creating machine learning models

3.6 Results

The following results can be extrapolated from the models:

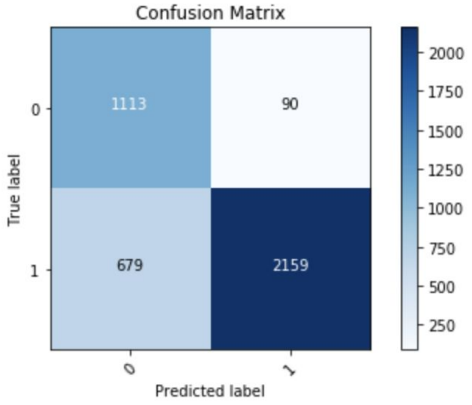
Table 1 : Performance metrics across different models for dataset-1				
Model	Precision	Recall	F-1 Score	Accuracy
Naïve Bayes	0.84	0.74	0.78	0.72
SVM	0.95	0.76	0.84	0.80
Logistic Regression	0.94	0.73	0.82	0.78
Random Forest	0.94	0.73	0.82	0.78

Table 2 : Performance metrics across different models for dataset 2				
Model	Precision	Recall	F-1 Score	Accuracy
Naïve Bayes	0.41	0.51	0.45	0.39
SVM	0.37	0.56	0.45	0.31
Logistic Regression	0.40	0.67	0.50	0.35
Random Forest	0.40	0.67	0.50	0.35

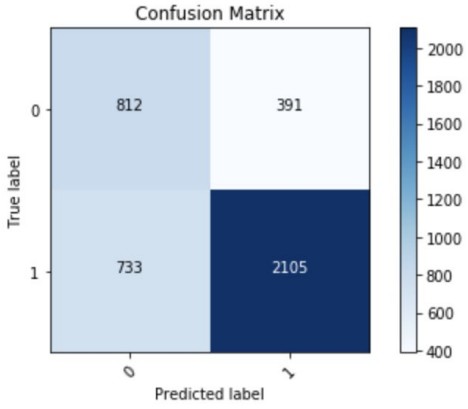
The confusion matrices obtained from the different models are given below:

Dataset 1:

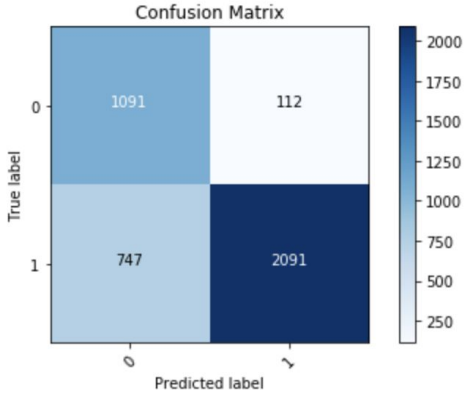
SVM



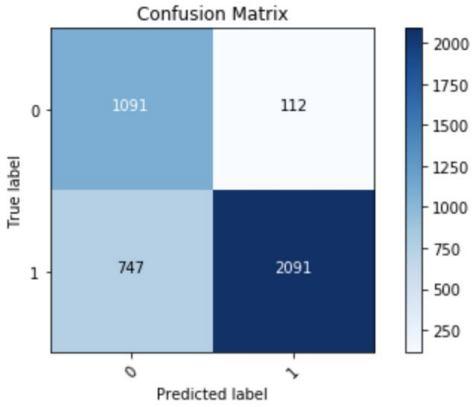
Naive Bayes



Logistic Regression

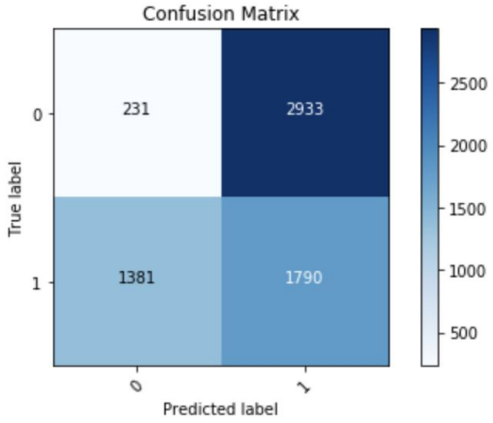


Random Forest

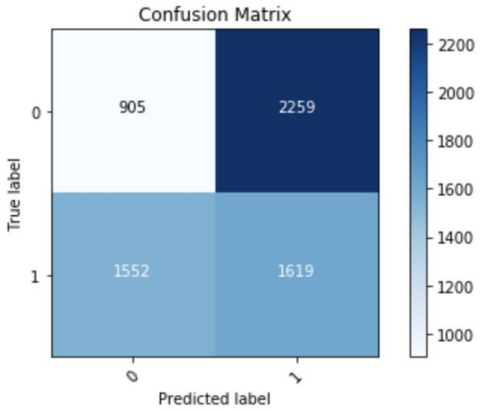


Dataset 2:

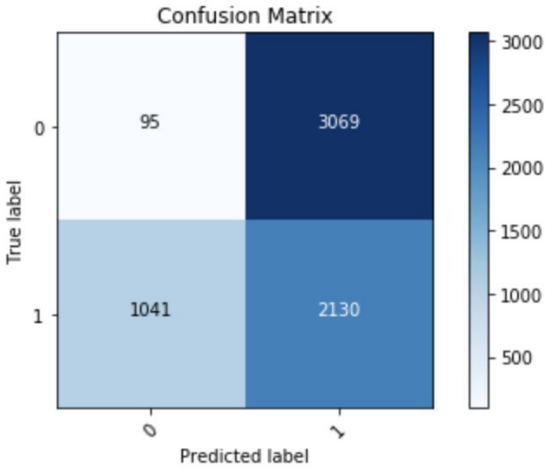
SVM



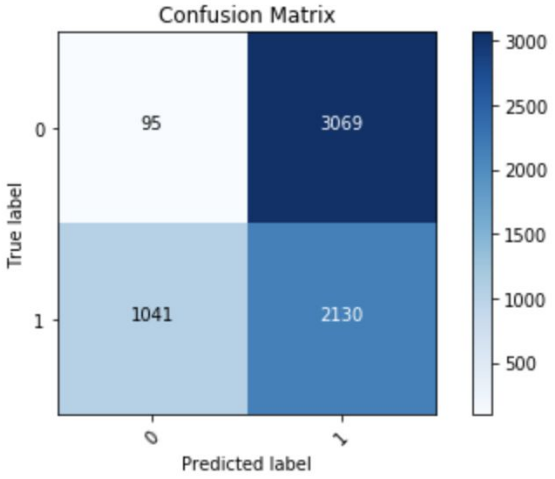
Naive Bayes



Logistic Regression



Random Forest



4. Summary

In this project, we have compared the results obtained from implementing classification models on news articles with the intention of combating fake news. A crucial part of the project revolved around pre-processing the data. The process of creating wore embeddings is a complex albeit necessary step before fitting the data to the respective models. A lot of interesting work has been done on the topic of fake news detection. We came across a myriad of effective techniques used by researchers in this regard. We encountered quite a few challenges while working on the project. One huge challenge was the presence of unclean data combined with a significant amount of incomplete data. It is worthwhile to note that the source of a news article plays an important part in determining if it's fake or not. We make extensive use of the scikit-learn module and its implementations of the different models used

in the project. The figures and metrics provide detailed insight into the efficacy of the different classification techniques. From the performance metrics it is obvious that out of all the models, SVM performed better with the data set that it was trained on and Naive Bayes was better for a dataset that was never seen before.

5. Conclusion and Future Scope

This paper demonstrates the effectiveness of classification techniques in the arduous task of identifying fake news. A better result can be obtained by extensive tuning of the classifier along with careful feature selection. Another way to increase performance is by using an aggregation algorithm like bootstrap aggregation on the well performing classifiers to achieve better accuracy. We would also like to explore the use of recurrent architectures as a means to classify fake news. Some news articles contain a large number of named entities which can be interpreted as unknown words resulting in the news source being mislabeled. We also hope to address this issue in a future update. It can be difficult to identify individual features in the news due to the vectorized approach undertaken in this project. This can be a worthwhile topic for future work in this area.

6. References

[1] D.Mrowcs, E. Wang & A.Kossow, "Stance Detection for Fake News Identification", URL: "web.stanford.edu/class/cs224n/reports/2760496.pdf"

[2] S. Gilda, "Evaluating machine learning algorithms for fake news detection" URL: "www.ieeexplore.ieee.org/document/8305411"

[3] Y. Kim, "Convolutional Neural Networks for Sentence Classification" URL: "www.aclweb.org/anthology/D14-1181"

[4]Understanding LSTM Networks URL: "colah.github.io/posts/2015-08-Understanding-LSTMs/"

[5] Naïve Bayes Documentation URL: "scikit-learn.org/dev/modules/naïve-bayes.html"

[6] Quoc, L., Mikolov, T., Distributed Representations of Sentences and Documents, URL: "https://arxiv. org/abs/1405.4053", May, 2014

[7] S. Feng, R. Banerjee, and Y. Choi, "Syntactic stylometry for deception detection," in Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, Association for Computational Linguistics, 2012, pp. 171–175

[8] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," Proceedings of the Association for Information Science and Technology, vol. 52, no. 1, pp. 1–4, 2015.

[9] Christopher, M. Bishop, Pattern Recognition and Machine Learning, URL: "http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf", April, 2016.

[10] Goldberg, Y., A Primer on Neural Network Models for Natural Language Processing, URL: " https://arxiv. org/pdf/1510.00726.pdf," October, 2015.

[11] Hochreiter, S., Jrgen, S., Long short-term memory, URL: “http://www.bioinf.jku.at/publications/older/2604.pdf”, October, 1997.