Detection of Fake News Using Machine Learning

Shravan Kruthick Sridhar
Global Indian International School, Tokyo, Japan
Email: shravankruthicksridhar@gmail.com

Abstract – Machine learning and artificial intelligence are essential parts of technology today. They help reduce human effort and may also be able to perform certain tasks better than humans. An example of this is the classification of news articles as real or false. Whereas humans would have to do extensive research just to check the validity of a single article, a machine learning algorithm can do that in a much shorter timeframe by recognizing words and patterns that mostly occur in false news articles using techniques such as keyword analysis. Checking the validity of news articles is essential to prevent the spread of misinformation, especially due to the prevalence of fake news nowadays.

Key Words – Machine learning, news classification, keyword analysis, validity of articles

INTRODUCTION

The use of artificial intelligence is extremely common in a wide variety of tasks, from customer service chatbots to helping doctors interpret chest X-rays. In many cases, using machine learning algorithms helps reduce human error and the time required for the completion of an otherwise complex task.

The classification of news articles is a great way to use machine learning. Given a large enough dataset, a suitable algorithm can predict if the information presented in an article is real or not. Using more sophisticated algorithms and larger datasets, we may even be able to compute how much of the article is true.

For this study, I used the Information Security and Object Technology dataset from the University of Victoria, as prescribed by Ahmed et al. in publications from 2017 and 2018. The truthful articles in the dataset were obtained from Reuters.com and the false ones from various sources flagged as unreliable by PolitiFact, a fact-checking organization. This dataset contains over 40,000 entries in total, which is enough to predict with over 95% accuracy using the right algorithm.

I chose this topic to search for a more efficient solution to fact-check articles on the spot, as having people read and validate each article is extremely time-consuming.

Instead of following this long method, I used the fact that false news articles are written in a much different style compared to true articles. False news articles often feature informal, conversational language and words that directly mock or criticize someone. For instance, one of the false articles in the dataset contains the following lines:

"Donald Trump just couldn't wish all Americans a Happy New Year and leave it at that. Instead, he had to give a shout out to his enemies, haters and the very dishonest fake news media. The former reality show star had just one job to do and he couldn't do it. As our Country rapidly grows stronger and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and Healthy New Year, President Angry Pants tweeted."

This is a blatant example of a false news article. "President Angry Pants" seems to be a name given by the author and not by formal quotes. It can be assumed that the author is biased against the person in question (President Donald Trump), which resulted in the production of false news.

I. Background

I used four algorithms to classify the news articles: logistic regression, Support Vector Machines (SVM), K Nearest Neighbors (KNN), and decision trees. All of these algorithms operate on different principles. Thus, some may be more accurate than others.

I chose these algorithms because they are computationally simple. They are the ones taught to anyone beginning to learn machine learning. Whereas researchers use sophisticated techniques, such as Hidden Markov Models or Natural Language Processors, for text processing, I was intrigued by the possibility of these simple algorithms yielding accurate results.

Another reason I chose these algorithms is that it only takes a couple of minutes to execute them (except for SVM) and generate output. This can be handy in case an app is developed based on this paper.

Let us take a look at the algorithms I used.

a. Logistic Regression

Logistic regression uses the gradient descent algorithm on a linear model to classify data. This process generates a linear equation with multiple coefficients that determine how much each parameter affects the result, along with a bias that signifies the value of the output when all of the input parameters are 0. Given a list of parameters, this equation outputs a real number. For example, it is possible to predict the price of a house using this equation given the values of parameters, such as the neighborhood crime rate and the number of rooms in the house.

Gradient descent is the algorithm used to produce the line of best fit for data. This is done by calculating the mean deviation of the predicted value for one set of values, scaling it by a factor called the learning rate, and subtracting it from the existing coefficients and bias. The calculated numbers serve as the new coefficients and bias.

Performing this action for each set of values gives us a linear model that we can use to make the predictions. However, in logistic regression, we need a probability that lies between 0 and 1. For this, we need to apply a sigmoid function (Figure 1) to each predicted value.

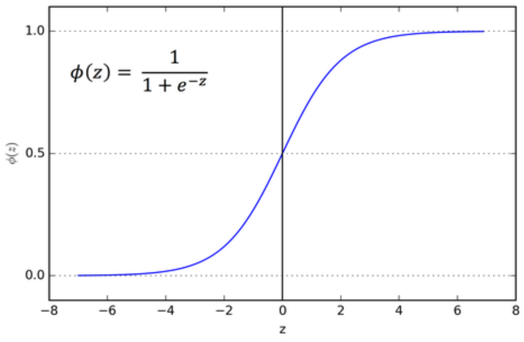

FIGURE 1 [3]: The sigmoid function. It takes in a real number and gives an output between 0 and 1.

b. SVM

In the SVM algorithm, we plot each data point as a point in an n-dimensional space, with n being the number of features. Then, we draw a hyperplane that separates the data point clusters and is as far as possible from the points. The margin is the distance between the hyperplane and the points on either side nearest to it. These points are called support vectors.

In many cases, the data may not be separable using a simple hyperplane. To solve this problem, a method called the kernel trick can be used, which converts a lower-dimensional point into a higher-dimensional one so that it is linearly separable.

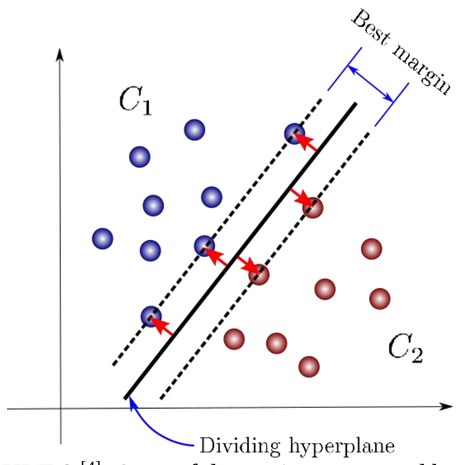

FIGURE 2 [4]: A set of data points separated by a hyperplane.

After this, we only need to see on which side of the hyperplane the given point falls.

c. KNN

The KNN algorithm uses the distance between points to classify data. For each data point's features, we compute the distance to every other data point.

Then, for each new point fed into the algorithm, the distance is computed between it and every other point. After that, the parameter k makes the algorithm consider the k nearest points (points with least distance) from the new point.

Thus, this process becomes a classification by majority. The new point is put into whichever data class occurs the most after the previous step of narrowing down.

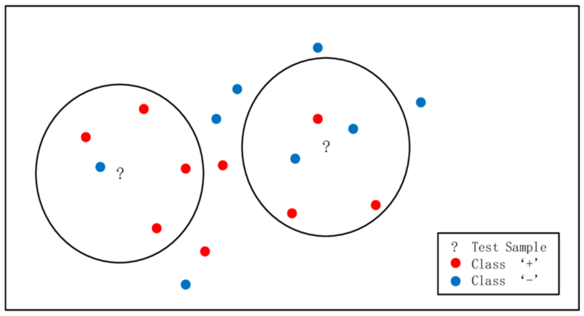

FIGURE 3: A simple example of a KNN algorithm.

If k is very small, the algorithm may overfit the training data, which means that it fits the training data too closely and cannot spot general data trends.

If k is very large, however, the algorithm underfits the data. Underfitting means that the algorithm does not sufficiently model the training data and cannot generalize to new data.

Since this algorithm only groups similarly worded articles, it may fail in case a true and false article contain similar words, despite having completely different meanings and contexts.

d. Decision Trees

The decision tree algorithm uses recursive binary splitting to categorize data [5]. For a decision tree classifier, the algorithm decides which condition to split the data on using a measure called the Gini impurity, which determines how pure a classification is. Specifically, it shows how many data points from multiple classes remain unseparated after one level of classification. The Gini impurity score of a node is the product of the probabilities of each class in the node. The lower this number is, the more accurate the classification.

II. Approach

I used Python to execute the algorithms since it has a machine learning library that helps speed up the process. To perform this task, I first cleaned up the data. After inspecting the data and removing characters such as the newline character \n, I used a TFIDF (Term Frequency Inverse

Document Frequency) [6] vectorizer to transform the strings into numerical data.

A TFIDF vectorizer works as follows: It first computes the number of occurrences of each word in each article; this is its TF value. Then, it computes the number of documents in the entire corpus in which the word occurs; this is its IDF value. The TFIDF is calculated by the following formula:

$$w_{\{i,j\}} = tf_{\{i,j\}} \times \log\left(\frac{N}{df_i}\right)$$

Next, I split the data, designating 20% for testing the model and the rest for training it. Then I was ready to feed the numerical data into the machine learning algorithm.

I used separate instances of TFIDF vectorizers to compute the TFIDF values for both the texts and titles, as I considered that even the article titles may be a good indicator of article validity. I also used two different instances of each algorithm for the same purpose.

One of this method's limitations is that it is not able to detect false statistics in a given article. For example, suppose an article says that 1,000 patients were infected with a disease, but the real number was closer to 400. The algorithms will not be able to detect this error because the method I used generates output based on the combinations of the words used in the articles.

This limitation can be counteracted by adding another layer of fact-checking to the process. This layer may include the checking of statistics. This can be done either by a human or a computer program that utilizes web crawling and parsing information from websites.

EVALUATION

I used multiple metrics to evaluate each of my models [7]:

Accuracy

This is one of the simplest ways by which a model can be judged. A model's accuracy is given by the quotient of the number of correct predictions and the total number of predictions.

However, accuracy alone is not a good measure of a model because of possible class imbalances. For example, if two classification labels are positive and negative, we can make a model that always predicts negative. If the number of true negatives in the data is more than the number of true positives, the accuracy increases, even though we have not built a model that represents the data to the highest possible extent.

Confusion Matrix

As its name suggests, the confusion matrix is a matrix that contains the number of each type of prediction made by the model (true positives, false positives, true negatives, false negatives).

Using the confusion matrix, we can calculate other metrics, such as precision. When used along with accuracy, these metrics can be used to arrive at a good estimate of the classifier's effectiveness.

F1 score

The f1 score of a classification model is given by the following formula:

$$\text{f1} = \frac{(\text{Precision}^{-1} + \text{Recall}^{-1})^{-1}}{2}$$

In this equation, precision is defined as the ratio of the relevant instances and retrieved instances, and recall is the fraction of the total amount of relevant instances retrieved. In other words, precision is the quotient of true positives and the total number of predicted positives. Recall is the quotient of the true positives and total number of actual positives (true positives + false negatives).

Using the harmonic mean instead of the arithmetic mean ensures that the f1 score is low even if one of the two quantities is low.

RESULTS

The results for each model are arranged in tables as follows:

Logistic Regression

Feature	Accuracy	Precision	Recall	F1 score
Title	95.53%	94.52%	96.07%	95.29%
Text	98.70%	98.46%	98.79%	98.62%

SVM

Feature	Accuracy	Precision	Recall	F1 score
Title	96.18%	95.56%	96.52%	96.04%
Text	99.46%	99.32%	99.55%	99.44%

KNN

Feature	Accuracy	Precision	Recall	F1 score
Title	89.95%	86.37%	93.37%	89.73%
Text	69.00%	94.50%	36.22%	52.37%

Decision Trees

Feature	Accuracy	Precision	Recall	F1 score
Title	91.83%	90.72%	92.12%	91.41%
Text	99.52%	99.50%	99.48%	99.49%

CONCLUSION

We can see from the results that the best algorithm for the task of classifying news articles as real or false based on title is SVM, with an accuracy of 96.18% and f1 score of 96.04%. Based on classification by text, however, the decision trees method is marginally better than SVM, with an accuracy of 99.52% and f1 score of 99.49%.

These are not the exact scores that will be obtained if the entire experiment is replicated due to the random separation of training and testing data, but the results obtained will always be in the same ballpark.

The classifier algorithms could be made more accurate if the title and text data are fed into a single classifier. However, this method is beyond the scope of this paper and was not conducted. Future research could explore this possibility.

REFERENCES

[1] Ahmed, Hadeer, et al. “Detecting Opinion Spams and Fake News Using Text Classification.” *Wiley Online Library*, John Wiley & Sons, Ltd, 29 Dec. 2017, onlinelibrary.wiley.com/doi/full/10.1002/spy2.9.

[2] Ahmed, Hadeer, et al. “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques.” *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, edited by L. Traore, I. Woungang, and A. Awad, Springer, 2017, pp. 127-138.

[3] Sharma, Sagar. “Activation Functions in Neural Networks.” *Towards Data Science*, 2019, www.towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[4] Carrasco, Oscar Contreras. “Support Vector Machines for Classification.” *Towards Data Science*, 2019, www.towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3. Accessed 16 June 2020.

[5] Starmer, Josh. “StatQuest: Decision Trees.” *YouTube*, uploaded by StatQuest with Josh Starmer, 23 January 2018, www.youtube.com/watch?v=7VeUPuFGJHk.

[6] Mamun, Iftekher. “Creating a TF-IDF in Python.” *Medium*, 2020, www.medium.com/@imamun/creating-a-tf-idf-in-python-e43f05e4d424. Accessed 18 June 2020.

[7] Yıldırım, Soner. “How to Best Evaluate a Classification Model.” *Towards Data Science*, www.towardsdatascience.com/how-to-best-evaluate-a-classification-model-2edb12bcc587. Accessed 23 June 2020.