Detection of Fake News With Machine Learning

Aditya Gupta

Adlai E. Stevenson High School | Grade 9

Experimental

Table of Contents

Acknowledgements

I would like to thank my family for their support in pursuing this research. A special thanks to my sponsor, Dr. Brontman, for investing her time to help me improve and implement this experimental project.

Purpose

The purpose of this experiment is to determine the most accurate and effective machine learning algorithm for the binary classification of news into fake and authentic journalism. With the rise of social media and rapid information dissemination, the need for accurately distinguishing between unreliable and factual reporting has become quintessential, especially due to growing political and racial tensions fueled by readily available information. Autonomously filtering content that may be hurtful, untrue, or inappropriate has become a priority for many websites. Notably, the number of credible media platforms leveraging methods to flag inaccurate journalism—thereby preventing users from being misinformed—has grown rapidly over the last several years. Thus, the need to find the most efficient and accurate technique to carry out this action is becoming increasingly necessary.

Hypothesis

If the dataset of text from news articles is fit into the decision tree, logistic regression, random forest, support vector machine, or passive-aggressive algorithms, then the most accurate algorithm will be the passive-aggressive classifier. This algorithm is expected to perform better than other algorithms because it has the strongest integration with the TF-IDF vectorizer, which is used to convert text to vectors that an algorithm can understand. The passive-aggressive classifier also incrementally updates and adapts the model. This differs from the other algorithms, which construct their model with all or most of the data at once. The vast amount of information and broadness of the dataset may make it difficult to consistently make an effective model using that method.

Experimental Components

Independent Variable: The machine learning algorithm used to classify the news articles; specifically a logistic regression classifier, decision tree classifier, random forest classifier, and passive-aggressive classifier.

Dependent Variable: The percent accuracy of the model's predictions on the authenticity of news articles as fake or authentic.

Constants: The constants include the input parameters for the algorithm, computer used, programming language used (Python 3.9.5), software used to run the program (Jupyter Notebook), libraries used (Pandas, Numpy, and Scikit-Learn), testing and training splits, dataset (Bisaillon's True and Fake News Dataset), vectorization method used (TF-IDF bag of words model), and the ratio of data used to train and test model (80:20).

Comparison Group: A linear support vector machine classification algorithm that makes a prediction on whether the news is fake or real. This algorithm is the industry standard for text classification due to being able to adjust to multidimensional data.

Review of Literature

The rapid development of data-driven software and computational advancements over the last decade have enabled machine learning to become prevalent in nearly every field. Today, machine learning (ML) is vital to services such as fraud prevention. It has become critical to removing dangerous and inaccurate information, known as fake news, which has caused problems such as political riots and the murders of innocent people. Optimizing algorithmic interactions is necessary to improve the functionality of machine learning in many of these essential services. It can enable systems to both imitate and elevate human observations, allowing an efficient resolution to several problems currently associated with the digital world, such as the spread of misinformation (*Machine Learning: What it is and Why it Matters*, n.d.).

Machine Learning

Machine learning is an area of artificial intelligence that discovers trends and unique structures in data to make decisions without human interference. One of the largest subsets of machine learning is supervised learning (Dickson, 2019). Supervised learning algorithms must be trained on labeled data; the algorithm is given a set of inputs and the desired outputs. The algorithm compares the input to the output to find structures in the data and create a model accordingly. The goal is to appropriately predict the output from an input (Wilson, 2019).

Unsupervised learning is another subset mostly used in exploratory analytics. It uses unlabeled data to organize the data and draw conclusions directly. Semi-supervised learning involves a combination of labeled and unlabeled data, using the labeled data as an example to learn from the unlabeled data. (*Machine Learning: What it is and Why it Matters*, n.d.).

Logistic Regression

Supervised classification algorithms deal with categorizing and labeling data. A common classification algorithm is logistic regression. Binomial logistic regression is used for binary classification, which involves sorting data into one of only two distinct categories, treated as class 0 or class 1. The first step of binomial logistic regression is to plot each data point on a Cartesian graph using the features or vectors of the data as the X-axis and the corresponding classification as the Y-axis—since the task is binomial, the Y-value of any data point must be 0 or 1. The line of best fit is found using linear regression (Ponraj, 2020).

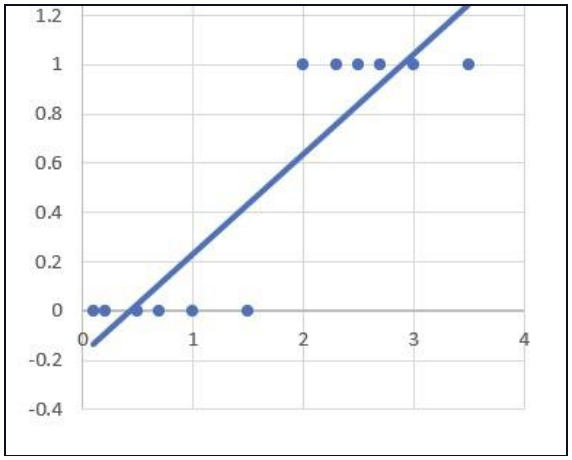


Figure 1. The linear function (Source: Ponraj, 2020).

The linear regression function is useful when it comes to producing continuous results. However, when data is binary, such as in Figure 1, the results become impractical and skewed; the algorithm fails to produce results that can be interpreted as a binary result, especially for values below 0 or exceeding 1. To remedy this, the sigmoid function is applied, which maps any real number to a decimal between 0 and 1, exclusive. The x-axis is then maintained while the Y-axis is the probability of each data point belonging to one of the two classes (Ponraj, 2020).

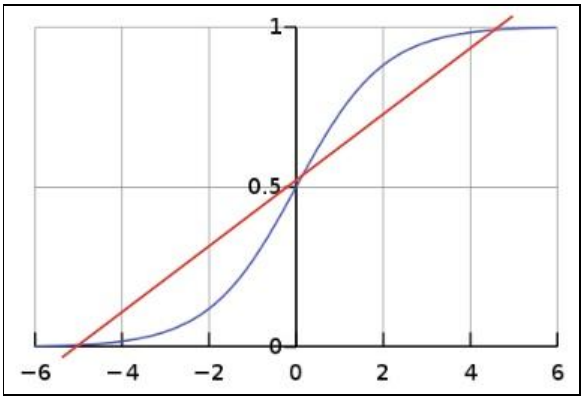


Figure 2. The sigmoid function (Source: Ponraj, 2020).

In Figure 2, the linear function (red) is transformed to form the sigmoid function (blue). The sigmoid function is asymptotically bounded between 0 and 1. Since the Y-axis represents probability, the algorithm can now easily classify data; input an X-value and the corresponding Y-value on the function is the probability (P) that the data point belongs to class 1. If $P \geq 0.5$, the data point will be classified as belonging to class 1, while if $P < 0.5$ the data point will be classified as belonging to class 0. This allows for the discrete binary classification of data. The linear function and sigmoid transformation are mathematically defined as:

$$z(x) = \theta_1 x + \theta_0$$
$$p = 1 / (1 + e^{-z})$$

Figure 3. Linear and sigmoid equations (Source: Jordan, 2017).

In Figure 3, it can be noted that *z(x)* is in slope-intercept form. Logistic regression algorithms are vulnerable to outliers and overfitting due to dependency on the line of best fit (Jordan, 2017). Overfitting occurs when the model is trained on an excess amount of data and results in the model over-accounting for the inherent randomness in data. The model may perform better on training data, but is less generalized, so during testing and real-world usage, the model will be hypersensitive to any noise in the data and less accurate (Yildirim, 2020).

Decision Trees and Random Forest Classifier

The decision tree is another common supervised learning algorithm commonly used for classification. It divides the data into multiple distinct categories based on the features of the data. Then, it continues to divide each category further so that there are sections within the broader categories. By repeating this method, it classifies data down to a precise group; for binary classification, all the paths would end at class 0 or class 1 (Wolff, 2020).

The algorithm makes divisions that try to maximize the most examples get properly classified. Each feature is taken into consideration and identified as a "branch" of the tree. The algorithm also prunes the tree, simplifying it by removing parts related to irrelevant features or those making unnecessary divisions. Decision trees also consider the inherent randomness or imbalance in the data, which is known as entropy (E), which is calculated as:

$$E = \sum -p_i \log_2(p_i)$$

Figure 4. Entropy equation (Source: Roy, 2020).

In Figure 4, the starting bound of the summation is $i=1$ and the limit is c, the number of classes; for binary classification, this would be 2. Each p_i is the likelihood that a class is selected if an example was chosen at random from the input data. Therefore, for a dataset where one class appears significantly more than another, the entropy will be higher, whereas a dataset where both datasets have a more equal distribution will have a lower entropy (Roy, 2020).

The random forest classifier is an expansion of the decision tree algorithm that creates several different, but accurate decision trees based on the training examples. For each data point, it feeds the data to all the decision trees, and then outputs the prediction that the majority of the trees return (Wolff, 2020).

Passive-Aggressive Classifiers and TF-IDF

The passive-aggressive classifiers are a group of online-learning algorithms that incrementally take in data; unlike other classification algorithms, they do not need to store all the data they are trained with in the random-access memory (RAM). Instead, they rely on taking in examples individually, updating the model, and then discarding each data point. This method allows the passive-aggressive model to be far more efficient and less resource-consuming than most batch-learning algorithms. It is also ideal when there are very large amounts of data to analyze, such as entire media platforms. However, since the algorithm can never understand the general trend of the data, it may have accuracy limitations (Bora, 2020).

The passive-aggressive classifier adjusts the model based on a few specific rules, in addition to the parameters set. During training, it makes a binary prediction to classify a data point. If the decision is correct, it remains passive and does not make any adjustments to the model. However, if the decision is incorrect, it becomes aggressive and updates the model to account for the example. Naturally, it does so while limiting the incorrect classification of previous data (Bora, 2020).

Passive-aggressive classifiers have a very tight integration with the term frequency—inverse document frequency (TF-IDF) vectorizer and the features it produces (Bora, 2020). Vectorization is a process that transforms text into numerical values known as vectors; this is the beginning of a process known as natural language processing (NLP), which is necessary for a machine learning algorithm to interpret the data. By using vectorization, any classification model can find patterns in text, such as news. There are several methods of vectorization, but a common one is the bag of words technique (Stecanella, 2017).

In this approach, each word is given values corresponding to certain measures. The most common of these is the TF-IDF feature extraction. Term frequency correlates to the frequency of a word in a document, and finds the significance of the word relative to the article. Inverse document frequency compares how frequently the word appears in a larger corpus or selection of texts. These values are taken as a ratio to yield a large amount of information, including the significance of words. For example, even though words such as “like” may be used commonly in a document, they are also present in numerous other documents and can be deemed insignificant (Stecanella, 2017).

TF-IDF metrics can be improved with various tools. Stopwords are a selection of the most common words in a language that can be completely ignored by the vectorizer, such as “the” (Stecanella, 2017). Since Zipf’s Law stipulates that these words are so common, it can significantly speed up the process of the conversion; the 100 most common words in English make up approximately one-half of all printed text (Manning et al., 2008). Lemmatizing words reduces them to the root word, so tenses and extensions such as “-ing” are removed (Stecanella, 2017).

In text classification, the vectors are inputted into a machine learning algorithm to organize the documents in the corpus by various content tags. Since the algorithms rely on rule-based systems and observations, machine learning has far less bias than manual text classification does. Most commonly, text classification is used in sentiment analysis, which is a process that statistically determines whether a writing is presenting positive, neutral, or negative feedback. The industry standard for text classification is support vector machines for various reasons (*Text Classification with Machine Learning & NLP*, n.d.).

Support Vector Machines

The support vector machine (SVM) algorithm is a very useful and adaptable binary classification algorithm. Support-vector machines rely on choosing the hyperplane—a separating boundary between data belonging to different classes—that best separates the data linearly. Hyperplanes can be multiple dimensions, and defined with the following formula:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = 0$$

Figure 5. SVM hyperplane equation (Source: Sirohi, 2019).

In Figure 5, β_0 denotes the intercept, β_1 denotes the first axis, β_2 denotes the second axis, p represents the number of dimensions, and each x value defines a position of the datapoint, otherwise known as a vector. In the Cartesian plane, for instance, a two pair coordinate system is used: (x, y) or, in this case, (x_1, x_2). If the equation sums to zero, the point is on the hyperplane, whereas a greater value is above the hyperplane and a lesser value is below the hyperplane. Often, the original data is not linearly separable, so a transformation is applied to map the data into a higher dimension feature space. The goal is to map the data so it can be linearly separated by a hyperplane in one less dimension (p−1 dimensions) (Sirohi, 2019).

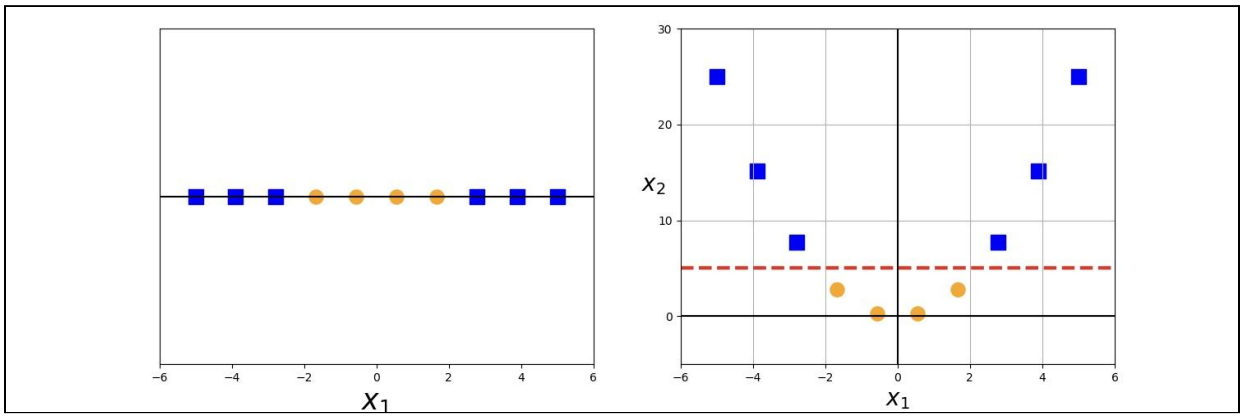


Figure 6. Graphs of SVM transformation (Source: Willimitis, 2018).

In Figure 6, a transformation changes a non-separable one-dimensional line of data into a two-dimensional parabola, after which a one-dimensional hyperplane is used to separate the data ($p-1$ dimensions). Similarly, more complex data could have been mapped into three or more dimensions (Sirohi, 2019). However, this approach becomes impractical when complex transformations are required to map the data into many dimensions, leading to unrealistic computational demands. That can be shortcutted by leveraging kernels (Willimitis, 2018).

The kernel function uses the dot products of vectors to map the data onto the higher dimensional feature space, without applying complex transformations to each vector. By significantly reducing the total calculations, the kernel function provides a shorter method allowing support-vector machines to map data into a near-infinite number of dimensions (Willimitis, 2018). Various kernel functions can be utilized in support vector machines, including the linear, polynomial, sigmoid, and radial basis function (RBF) (Pedregosa et al., 2011).

However, even when the data is linearly separable, there are still an infinite number of hyperplanes that can be chosen, as there is always an area or range in which the hyperplane can be placed. Therefore, support-vector machines choose the hyperplane by maximizing the margin width (w) or the Euclidean distance from the closest data points around the separating area, following the general shape of the data. These data points are known as support vectors and are given precedence over other data points as they influence the position of the hyperplane. The algorithm then defines the margin boundaries, a set of lines offset in either direction from the main hyperplane (Sirohi, 2019).

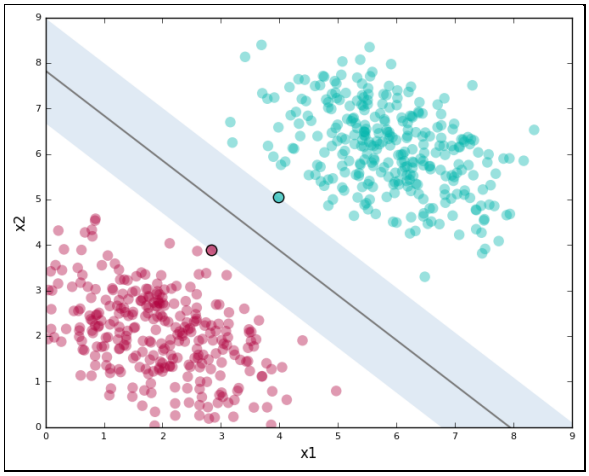


Figure 7. The SVM hyperplane (Source: Ghose, n.d.).

The size of the margin may vary, as the algorithm will attempt to minimize the data in the margins (Ghose, n.d.). For a single outlier in the margin, the hyperplane and margins may drastically shift position, making support-vector machines subject to overfitting. Support-vector machines can diminish consequences with the slack variable. The slack variable allows support-vector machines to increase the margin at the expense of misclassifying some data (Sirohi, 2019). Similarly, the cost or C parameter allows the algorithm to set a penalty value for misclassifying a data point. A lower C allows the algorithm to misclassify a few data points to increase the margin, as there is a lower penalty, while with a higher C the algorithm will choose a hyperplane that classifies the data better overall but has a narrower margin (Ghose, n.d.).

Together, the values of the slack variable and cost determine whether the algorithm uses a soft or hard margin hyperplane. In a soft margin hyperplane, some of the support vectors may be misclassified or in the margin to increase the margin and confidence that the optimization has (Yildirim, 2020). In a hard margin hyperplane, the optimization chooses the hyperplane that best classifies the data regardless of the margin (Sirohi, 2019).

Fake News

Fake news is a systematically created form of disinformation used to purposefully delude and manipulate an audience, often exploiting pre-existing biases and prejudices, such as political affiliation (Preston et al., 2021). Experts conclude that fake news includes authentic information that has been manipulated or used in misleading contexts, falsified information intended to mislead an audience, and sources attempting to impersonate credible websites. However, fake news doesn't include humorous or satirical content, such as memes (Wardle, 2016). By utilizing charged language, alluring headlines, and fabricated content, fake news seeks to incite a response from the reader, while professionally presenting itself in an attempt to look credible. The journalism is typically intended for a specific audience and augments their beliefs, making it difficult to differentiate from authentic sources (Tandoc Jr., 2021).

A study conducted by Hansrajh et al. (2021) found that humans have only a 54% chance at accurately identifying fake news, and that more effective methods rely on using expanding machine learning technologies. Still, they discovered that success was highly dependent on the techniques used, observing instances of spectacular and mixed success. These findings were supplemented with research done by Ahmad et al. (2020), who discovered that employing multiple machine learning algorithms—known as an ensemble method—were in general more effective than most singular methods. The usage of machine learning techniques to flag misleading information has even been expanded beyond fake news. A 2018 study by Brogly and Rubin found that support vector machines, the industry standard for natural language processing (NLP), worked effectively at identifying clickbait and misleading hyperlinks.

In recent years, fake news has spread and grown tremendously in magnitude by leveraging social media and crowdfunded websites, which typically don't use any factual verification or peer reviewing like most publishers (Ahmad et al., 2020). The problem has only been growing; a prolific number of fake news campaigns may have influenced results in the 2016 presidential election, reducing trust in the democratic system. More recently, fake news was abused in the COVID-19 pandemic to share false information that threatened the health of the general public (Preston et al., 2021). Tandoc Jr. et al. (2021) concluded that the growth of fake news has resulted in a growing distrust in the validity of both legitimate news sources and questionable ones. In response to fictional posts, riots and mobs have resulted in the murder of innocent individuals that were wrongfully tagged as criminals.

Consequently, organizations and companies are devoting substantial resources to combat the spread of this disinformation, especially to promote credible journalists. Facebook has begun using machine learning to flag suspicious content, while Twitter is known for its "Verified" tags. YouTube has been utilizing machine learning to improve their search engine and hide irrelevant videos and comments as well as investing in other alternatives, such as a verified "Breaking News" panel. Google created a team solely dedicated to scrubbing fake news related to the 2020 pandemic, as well as devoting additional resources into their engine to prioritize credible news sources (Jain, 2021).

Materials

- 32 bit/64 bit PC (running Windows 8/10, macOS, or Linux)
- Internet Access
- Bisaillon's Fake and Real News Dataset (Retrieved December 28, 2021).
- Anaconda 3.8 Individual Edition with Pandas, Numpy, Scikit-Learn, and Jupyter Notebook

Procedure

Install and Configure Anaconda 3.8 and Jupyter Notebook

1. Download the version of Anaconda 3.8 Individual Edition for your corresponding operating system, available from https://www.anaconda.com/products/individual.
2. Open the downloaded installer and follow the configuration prompts presented by the executable file. All necessary Python packages come with Anaconda Distribution.
3. Disable the PATH variable when installing Anaconda 3.8.

Run the program

4. Download the Fake and Real News dataset available from https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset.
5. Extract all the files from the downloaded zip folder.
6. Rename the downloaded files in the "true" and "fake" folders to "true.csv" and "fake.csv", respectively.
7. Download source code from https://github.com/aaggupta07/detecting-fake-news.git. The full code has been provided in Appendix A as well.

8. On Windows, move the downloaded folder to the C drive under your user profile.
9. Move the dataset files (.csv files) to the same folder housing the downloaded notebook. Make sure that the CSV files are not housed under a sub-folder.
10. Launch Jupyter Notebook using the shortcut created by downloading Anaconda 3.8.
11. Navigate to the correct folder using the Jupyter Notebook interface.
12. Open the downloaded code (.ipynb file) via Jupyter Notebook.
13. Click the “Run All Cells” button available through the menu. Each algorithm receives a copy of the original data, which is left unchanged.
14. Record the output of the last 5 cells, which shows the algorithm used as well as the percent accuracy and confusion matrix, which is a 2x2 grid showing the number of true negatives, false positives, false negatives, and true positives when read left to right, top to bottom.
15. Repeat steps 13–14 for each trial. Previous algorithms are automatically discarded when the code is rerun.

Results

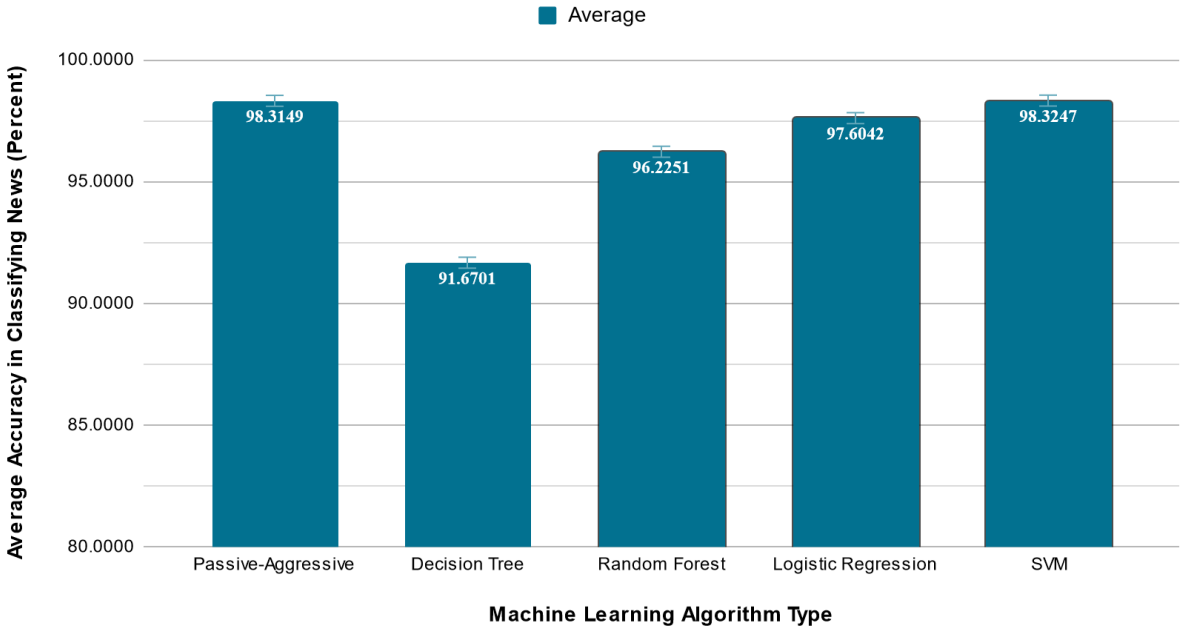
A k-fold cross-validation was performed for every algorithm, which is used to estimate the skill of a machine learning model. Each algorithm was trained and tested ten times with unique training and testing splits of the dataset, after which it was discarded. For any given trial, all algorithms received the exact same splits of the dataset.

Figure 8: Accuracy in Classifying News Versus Machine Learning Algorithm Type

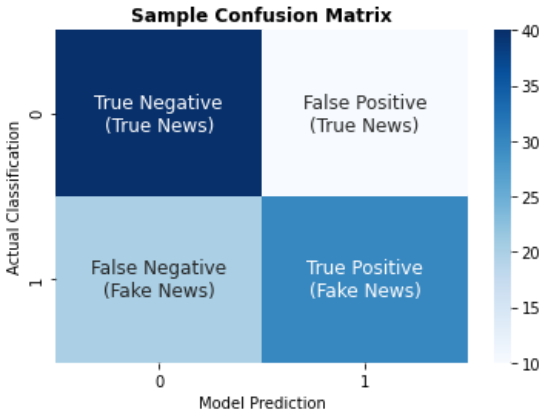
Accuracy in Classifying News (Percent)	Machine Learning Algorithm Type				
Trial	**Passive-Aggressive**	**Decision Tree**	**Random Forest**	**Logistic Regression**	**SVM**
1	98.4217	91.7594	96.2096	97.6197	98.3312
2	98.1113	91.6818	95.8473	97.5162	98.2536
3	98.2277	91.9534	96.4942	97.5162	98.0336
4	98.6417	92.1475	96.1449	97.6843	98.5640
5	98.3751	91.8887	96.0931	97.3868	98.2924
6	98.3312	91.5783	96.3648	97.8137	98.3959
7	98.4088	91.4360	96.3519	97.6197	98.4864
8	98.2018	91.2807	96.2484	97.7232	98.3700
9	98.2277	91.3454	96.1061	97.4386	98.1759
10	98.2018	91.6300	96.3907	97.7232	98.3441
Average	98.3149	91.6701	96.2251	97.6042	98.3247
Std. Dev.	0.1539	0.2761	0.1877	0.1371	0.1505

Models performed somewhat similarly with the exception of the decision tree classifier, which performed worse than the other models. Support vector machines had the highest overall accuracy, at approximately 98.32%, while the decision tree algorithm had the lowest accuracy at approximately 91.67%. Models performed consistently, with the average standard deviation being just 0.18%. Accuracy was directly measured as the number of correct classified data points over the total number of data points during testing.

Figure 9: Average Accuracy in Classifying News Versus Machine Learning Algorithm Type



Overall, the confidence margins were small, with the average standard error being just 0.057% wide. The algorithm that had the largest performance difference was the decision tree classifier, being both least effective and least consistent. Logistic regression was the most consistent out of all algorithms, with a 95% confidence interval between 97.506%–97.702%.



A sample confusion matrix is provided to the left. The y-axis shows the correct classification of data as 0 (true) or 1 (fake). The x-axis shows the model prediction. True quadrants indicate accurate predictions while false quadrants indicate incorrect predictions. These identifications should be applied when evaluating Figures 10–14.

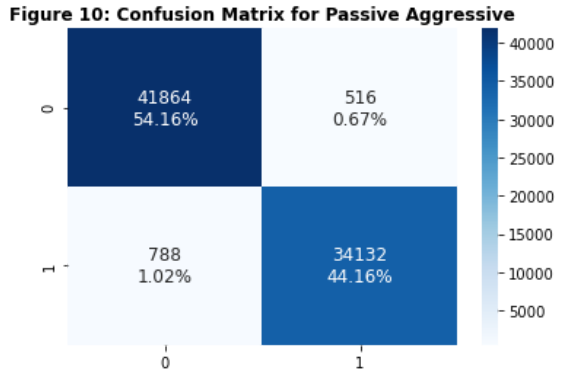


Figure 10: Confusion Matrix for Passive Aggressive

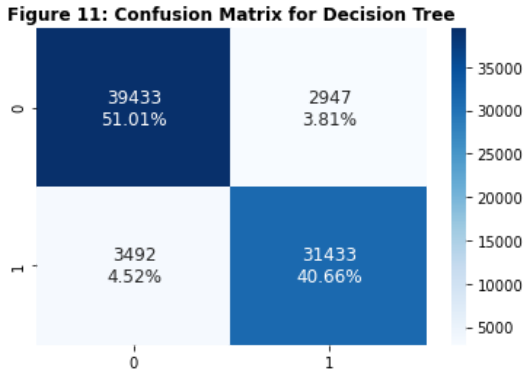


Figure 11: Confusion Matrix for Decision Tree

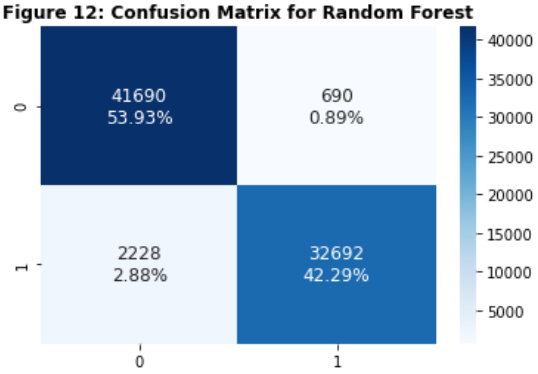


Figure 12: Confusion Matrix for Random Forest

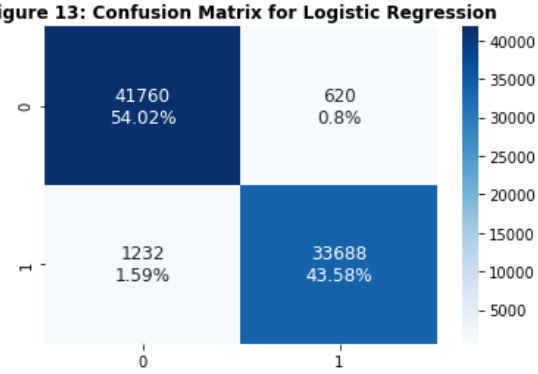


Figure 13: Confusion Matrix for Logistic Regression

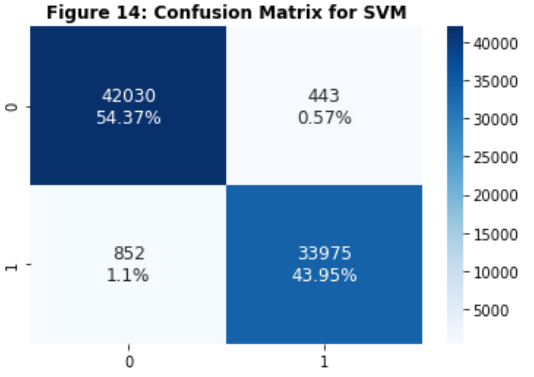


Figure 14: Confusion Matrix for SVM

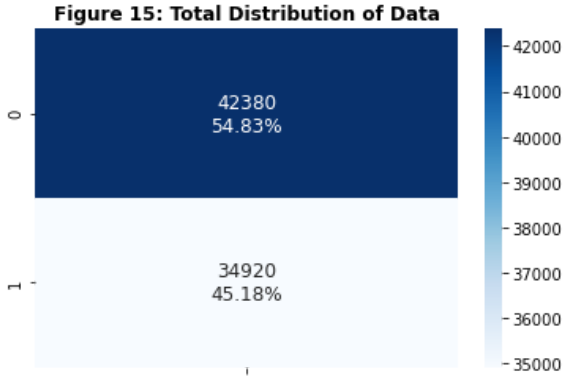


Figure 15: Total Distribution of Data

The confusion matrices above feature the aggregate total of all the data classified across all ten trials and the percentage of data in each quadrant compared to the total. All models struggled more to identify fake news, misclassifying more information as false negative than false positive. Figure 15 may explain this, as while the data had moderate distribution as fake or true news, there was still less data overall to train and validate the model on for fake news.

Discussion

A one-way anova t-test was conducted, resulting in a p-value < 0.0001. Since the threshold for significance was set at 0.05, at least one group was statistically significant from the others. Each pair of algorithms was then tested with a two-tailed t-test to determine if the two algorithms were statistically significant. Results are summarized below.

Figure 16: Two-Tailed Unpaired T-Tests for Significance

Algorithm 2	**Algorithm 1**				
	Passive Aggressive	Decision Tree	Random Forest	Logistic Regression	SVM
Passive Aggressive		t = 66.4812 p < 0.0001 SIGNIFICANT	t = 27.2300 p < 0.0001 SIGNIFICANT	t = 10.9016 p < 0.0001 SIGNIFICANT	t = 0.1439 p = 0.8868 INSIGNIFICANT
Decision Tree			t = 43.1565 p < 0.0001 SIGNIFICANT	t = 60.8826 p < 0.0001 SIGNIFICANT	t = 66.9151 p < 0.0001 SIGNIFICANT
Random Forest				t =18.7672 p < 0.0001 SIGNIFICANT	t = 27.5929 p < 0.0001 SIGNIFICANT
Logistic Regression					t = 11.1843 p < 0.0001 SIGNIFICANT
SVM					

Each t-test paired two algorithms and tested if using performances between the two algorithms were statistically significant. The results of the t-tests reveal that most relationships were indeed significant, with the majority having a p-value less than 0.0001, well below the 0.05 significance threshold.

However, there was one insignificant relationship that occurred between the support vector machine classifier and the passive-aggressive classifier. This shows that there was no tangible difference in accuracy between the two algorithms. These two algorithms performed the best, revealing that they are ideal algorithms for this task. Other algorithms would not be ideal choices, since they showed statistically significant performance gaps.

Most models were accurate in predictions. All models had more false negatives than false positives, inaccurately identifying more fake news as true than true news as fake. This resulted in lower recall scores than precision scores. Recall is calculated as the number of true positives (TP) divided by the number of true and false negatives (TP + FN). A perfect recall score is 1. This shows the percentage of positive (fake) examples that were classified correctly. Precision is calculated as TP divided by the number of true and false positives (TP + FP). This shows the percentage of examples the model classified as positive were correct.

However, the models were only tested on a single dataset due to availability. While this dataset was large, and included nearly 40,000 unique examples, there were still limitations. For example, the dataset used only a handful of news sources to retrieve fake and true news, and the data is mostly from around 2016–2020. Similarly, it mainly included world and political news, as well as social media statements; political news was mostly centered around the U.S. This limits the capabilities of the models to evaluate more diverse types of news, such as international political news or educational journalism intended for younger audiences. These errors may have produced slightly more optimistic results than if the dataset had been expanded. However, the use of a k-fold cross validation suggests that effects would have been mild at most.

Conclusion

The purpose of this experiment was to discover the most effective and accurate machine learning algorithm to classify news as fake or genuine. As fake news has intensified in magnitude and fueled a number of biases—even resulting in riots and innocent individuals being murdered—the need for accurately detecting false news has become a priority for many platforms, and doing so efficiently and effectively has become crucial.

The initial hypothesis predicted that of the five tested algorithms, the passive-aggressive classifier would have the most success in classifying the data, based on strong integration with the vectorizer and its adaptive tendencies. This hypothesis was only partially supported, as the data from Figures 8 and 9 show that there were two ideal algorithms for classifying news: the passive-aggressive classifier and the support vector machine classifier. Other algorithms performed statistically worse when accuracy was compared. Although the SVM classifier was slightly more accurate than the passive-aggressive classifier, results of the t-tests in Figure 16 show that the difference was not statistically significant. Thus, the passive-aggressive classifier was one of the best algorithms to classify news, but the SVM classifier performed equally well.

One possible experimental error is that the folds used when estimating the skill of the models were not completely pure. In a true k-fold cross validation, the dataset is pre-partitioned and then a single partition is used for testing, while all others are used for training. However, that restricts the number of trials or makes the testing size smaller per trial, so instead this experiment partitioned the dataset randomly at the start of each trial. Consequently, some data points may have been used for testing multiple times, and other data points not at all. However,

the impact of this error seems to be minimal, as standard deviations were low and each model was carefully discarded. This also suggests that data leakage did not occur.

It was also found that accuracy, and other interpretations, such as F-score, are only one factor when judging a model for effectiveness. Other factors, such as computational expense also matter, especially when an algorithm is scaled up for real world use. For example, while the passive-aggressive and SVM models were found to be equal in accuracy, the SVM model did take significantly more time and computing power to yield results.

There are various real world applications for this project. Applications such as Meta's Facebook and Google's YouTube have already begun using machine learning to detect fake news and flag it, as well as prioritizing credible and relevant information. Racial and political tensions have magnified in the last several years, and the world is currently in one of its most fragmented states yet—in large part because of information bias and fake news. In the last two years, media trust has dropped by 8% in the United States alone—resulting in over 71% of Americans claiming they distrust most news media. The problem has only been escalating as studies show there is only a 46% chance that a human accurately identifies fake news. Using classifiers with a greater-than 98% chance of correctly identifying fake news certainly helps.

Opportunities to expand this research certainly exist as well. In particular, using more data from a wider variety of sources and topics can be extremely beneficial. Fake news also exists in other forms besides text, such as disturbing images and deceptive clickbait. In addition, this experiment tested only singular algorithms. Developing a more sophisticated and blended model from this information by combining methods used by the SVM and passive-aggressive classifier might allow for even higher effectiveness.

Reference List

Ahmad, I., Yousaf, M., Yousaf, S., & Ahmad, M. O. (2020). Fake news detection using machine learning ensemble methods. *Complexity*, 1–11. https://doi.org/10.1155/2020/8885861.

Bisaillon, C. (2019). Retrieved October 10, 2021, from https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset.

Bora, A. (2020, July 17). Passive aggressive classifiers. GeeksforGeeks. Retrieved November 14, 2021, from https://www.geeksforgeeks.org/passive-aggressive-classifiers/.

Brogly, C., & Rubin, V. L. (2018). Detecting Clickbait: Here's How to Do It. *Canadian Journal of Information & Library Sciences*, *42*(3–4), 154–175. https://doi.org/10.3138/cjils.e14026.

Ghose, A. (n.d.). Support Vector Machine (SVM) tutorial: Learning SVMs from examples. KDnuggets. Retrieved November 18, 2021, from https://www.kdnuggets.com/2017/08/support-vector-machines-learning-svms-examples.html.

Hansrajh, A., Adeliyi, T. T., & Wing, J. (2021). Detection of online fake news using blending ensemble learning. *Scientific Programming*, 1–10. https://doi.org/10.1155/2021/3434458

Jain, R. (2021, April 1). What tech companies are doing to control the spread of fake news. iTMunch. Retrieved November 7, 2021, from https://itmunch.com/tech-companies-control-spread-fake-news/.

Jordan, J. (2017, June 8). Logistic regression. Jeremy Jordan. Retrieved November 27, 2021, from https://www.jeremyjordan.me/logistic-regression/.

Machine learning: What it is and why it matters. Statistical Analysis System. (n.d.). Retrieved October 23, 2021, from https://www.sas.com/en_us/insights/analytics/machine-learning.html.

Manning, C. D., Raghavan, P., & Schutze, H. (2008). Zipf's law: Modeling the distribution of terms. In Introduction to Information Retrieval (pp. 89–90). Essay, Cambridge University Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 2825–2830.

Ponraj, A. (2020, July 2). Logistic regression part I-transformation of linear to logistic. Analytics Vidhya. Retrieved November 26, 2021, from https://medium.com/analytics-vidhya/logistic-regression-part-i-transformation-of-linear-to-logistic-395cb539038b.

Preston, S., Anderson, A., Robertson, D. J., Shephard, M. P., & Huhe, N. (2021). Detecting fake news on Facebook: The role of emotional intelligence. *PLOS ONE*, *16*(3). https://doi.org/10.1371/journal.pone.0246757.

Roy, A. (2020, November 6). A dive into decision trees. Towards Data Science. Retrieved November 28, 2021, from https://towardsdatascience.com/a-dive-into-decision-trees-a128923c9298.

Sirohi, K. (2019, August 5). Support Vector Machine (detailed explanation). Towards Data Science. Retrieved November 16, 2021, from https://towardsdatascience.com/support-vector-machine-support-vector-classifier-maximal-margin-classifier-22648a38ad9c.

Stecanella, R. (2017, September 21). The beginner's guide to text vectorization. MonkeyLearn Blog. Retrieved November 8, 2021, from https://monkeylearn.com/blog/beginners-guide-text-vectorization/.

Tandoc Jr., E. C., Duffy, A., Jones-Jang, S. M., & Wen Pin, W. G. (2021). Poisoning the information well? *Discourses of Fake News*, *20*(5), 783–802. https://doi.org/10.1075/jlp.21029.tan

Text classification with machine learning & NLP. MonkeyLearn Blog. (n.d.). Retrieved November 20, 2021, from https://monkeylearn.com/text-classification/.

Wardle, C. (2016, November 18). *6 types of misinformation circulated this election season*. Columbia Journalism Review. Retrieved December 23, 2021, from https://www.cjr.org/tow_center/6_types_election_fake_news.php

Wilimitis, D. (2018, December 12). The kernel trick. Towards Data Science. Retrieved November 16, 2021, from https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f.

Wilson, A. (2019, October 1). A brief introduction to supervised learning. Towards Data Science. Retrieved October 26, 2021, from https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590.

Wolff, R. (2020, August 26). 5 types of classification algorithms in machine learning. MonkeyLearn Blog. Retrieved October 25, 2021, from https://monkeylearn.com/blog/classification-algorithms/.

Yildirim, S. (2020, June 1). Hyperparameter tuning for support vector machines—C and gamma parameters. Towards Data Science. Retrieved November 18, 2021, from https://towardsdatascience.com/Hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167.

Appendix A: Full Code

This appendix features all the code that was custom written for the training, testing, and implementation of the models used in this study.

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

# Read the Data
true_df = pd.read_csv('true.csv')
fake_df = pd.read_csv('fake.csv')
true_df = true_df.drop(columns=['title', 'subject', 'date'])
fake_df = fake_df.drop(columns=['title', 'subject', 'date'])

# Clean the Data and join into one dataset
true_df['text'] = true_df['text'].str.replace('Reuters', '')
true_df['label'] = 0
fake_df['label'] = 1
df = pd.concat([true_df, fake_df]).drop_duplicates(subset=['text']).dropna(subset=['text'])
df = df.sample(frac=1)
df.index = range(len(df))

X = df['text']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Convert the text into vectors or numerical values using TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
tfidf_train = vectorizer.fit_transform(X_train)
tfidf_test = vectorizer.transform(X_test)

# Train the model (Passive Aggressive Classifier)
model = PassiveAggressiveClassifier(max_iter=50)
model.fit(tfidf_train, y_train)
y_pred = model.predict(tfidf_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {round(accuracy * 100, 4)}%')

cm = confusion_matrix(y_test, y_pred)
counts = ['{0:0.0f}'.format(value) for value in cm.flatten()]
percents = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
labels = np.asarray([f'{i}\n{j}' for i, j in zip(counts, percents)]).reshape(2, 2)
sns.heatmap(cm, annot=labels, fmt='', annot_kws={'size': 12}, cmap='Blues')

# Train the model (Decision Tree Classifier)
model = DecisionTreeClassifier()
model.fit(tfidf_train, y_train)
y_pred = model.predict(tfidf_test)
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f'Accuracy: {round(accuracy * 100, 4)}%')

cm = confusion_matrix(y_test, y_pred)
counts = ['{0:0.0f}'.format(value) for value in cm.flatten()]
percents = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
labels = np.asarray([f'{i}\n{j}' for i, j in zip(counts, percents)]).reshape(2, 2)
sns.heatmap(cm, annot=labels, fmt='', annot_kws={'size': 12}, cmap='Blues')

# Train the model (Random Forest Classifier)
model = RandomForestClassifier()
model.fit(tfidf_train, y_train)
y_pred = model.predict(tfidf_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {round(accuracy * 100, 4)}%')

cm = confusion_matrix(y_test, y_pred)
counts = ['{0:0.0f}'.format(value) for value in cm.flatten()]
percents = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
labels = np.asarray([f'{i}\n{j}' for i, j in zip(counts, percents)]).reshape(2, 2)
sns.heatmap(cm, annot=labels, fmt='', annot_kws={'size': 12}, cmap='Blues')

# Train the model (Logistic Regression)
model = LogisticRegression()
model.fit(tfidf_train, y_train)
y_pred = model.predict(tfidf_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {round(accuracy * 100, 4)}%')

cm = confusion_matrix(y_test, y_pred)
counts = ['{0:0.0f}'.format(value) for value in cm.flatten()]
percents = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
labels = np.asarray([f'{i}\n{j}' for i, j in zip(counts, percents)]).reshape(2, 2)
sns.heatmap(cm, annot=labels, fmt='', annot_kws={'size': 12}, cmap='Blues')

# Train the model (Support Vector Machines)
model = SVC(kernel='linear')
model.fit(tfidf_train, y_train)
y_pred = model.predict(tfidf_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {round(accuracy * 100, 4)}%')

cm = confusion_matrix(y_test, y_pred)
counts = ['{0:0.0f}'.format(value) for value in cm.flatten()]
percents = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
labels = np.asarray([f'{i}\n{j}' for i, j in zip(counts, percents)]).reshape(2, 2)
sns.heatmap(cm, annot=labels, fmt='', annot_kws={'size': 12}, cmap='Blues')
```