Introduction

Fake news can spread extremely fast. A 2018 article from *Science* found that fake news can spread 10 times faster than true news stories.[1] Morning Consult Brand Intelligence found that trust dropped on the two social media platforms (Facebook and Twitter) that were used most heavily for news consumption. In a survey conducted in September of 2022 they found that social media users trust social media outlets that limit fake news.[2]

There are currently many fake news detection algorithms used by these social media platforms and other entities. By and large they use models based on word usage (e.g. TF-IDF, N-gram, and Bag of Words Model. However, these models are extremely susceptible to changes in word use. For instance, a slang term used first and primarily in fake news articles might make its way to legitimacy. Obviously a model must constantly be fed new data to stay current, but there will be a delay. Especially, since to get labeled data a human expert has to perform the classification.

Legitimate news organizations have style guides, for instance the Associated Press style guide is on its 56th edition. And the New York Times discusses their style guide here (https://www.nytimes.com/2018/03/22/insider/new-york-times-stylebook.html). Assessing how readable an article could be a way to leverage this uniformity in writing style to help detect real news from fake news. Luckily there are already formulas and python libraries to determine readability and to easily obtain other metrics, such as word count, letter count, and syllable count (which are often components of the scores themselves).

Let's look at one popular formula before delving into the machine learning task at hand. Below is the formula for the Flesch Reading Ease test.

$$206.835 - (1.015 * \text{total words}/\text{total sentences}) - 84.6 * (\text{total syllables}/\text{total words}))$$

A high score indicates a greater readability, thus this readability formula defines a text as easy to read if it has short sentences and short words. And other readability scores work on similar principles. These scores and other metrics such as word count, letter count, and etc... are easily calculated by the python library textstat, will be useful features to add to machine learning models that detect fake news.

Data Wrangling/Cleaning

The data used for this project come from two Kaggle datasets that have separate datasets for real and fake news.[3] The datasets were labeled and then joined together. The primary challenges for wrangling and cleaning the data was to identify and remove duplicates and non-English language documents.

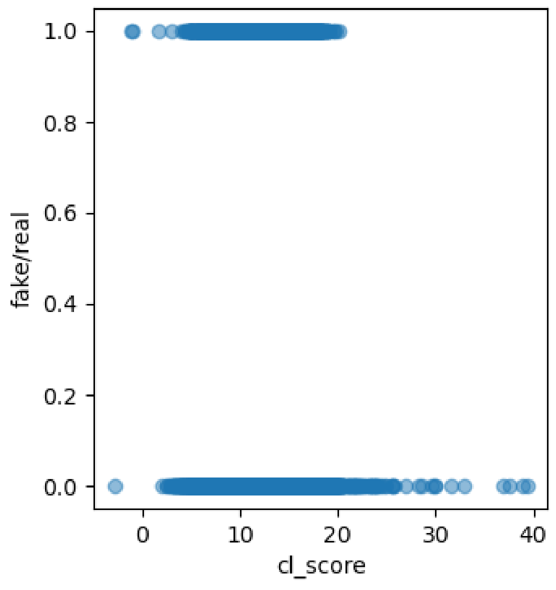
[1] https://www.science.org/doi/10.1126/science.aap9559
[2] https://pro.morningconsult.com/analysis/distributing-news-has-hurt-social-platforms-trust-among-consumers
[3] https://www.kaggle.com/datasets/algord/fake-news
https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets

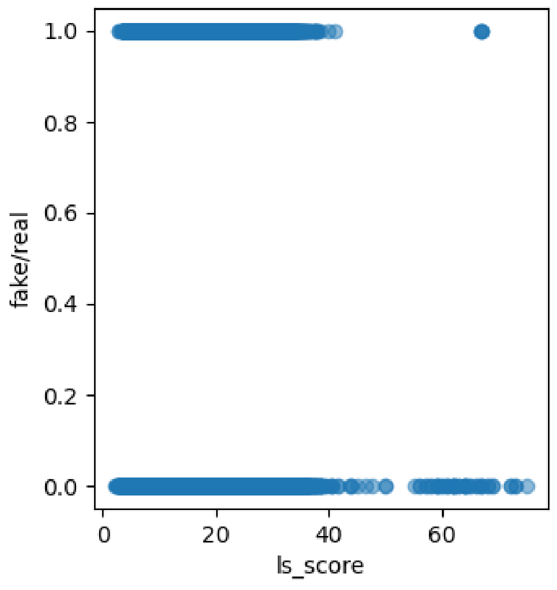
Simply removing exact duplicates was not sufficient, as some documents due to data gathering or news agencies copying nearly verbatim from another source. Traditional fuzzy matching was computationally expensive on such a large data set with many words, so a vectorized fuzzy matching was implemented. However, after doing readability scoring and sorting by the readability scores I found that there were still some documents that were nearly identical. Through trial and error I came up with a combination of readability scores and other metrics that identified more fuzzy duplicates. This process is prone to errors, but it might be helpful for deduplication in large textual datasets if refined further. The languages other than English that were in my dataset were Russian and a language with Arabic script. I used regEx for those languages and if a document had characters in those languages I removed them from the dataset.

Exploratory Data Analysis

The first steps for EDA was to score the documents using readability scores from the python library textstat. I used most of the scores from this library. This library scores primarily English language documents, but has scorers that can be applied to other languages and some specific scorers for other non-English languages. Of note since this is not being used for its primary purpose and the English language scorers did return a score on non-English language documents. It may be possible to use these for real/fake classification in those languages as well.

After performing EDA on the readability scores, it seemed likely that using readability scores as features likely could boost prediction of Fake News, but likely would not be able to be a significant factor alone. A large portion of both fake and real news had a similar range of readability scores. Below are a few examples:





Fake news is coded as 0 and real news as 1. As you can see from these graphs there are a number of fake news articles on the extremes in comparison to real news. That being said, perhaps they can serve as a useful feature along with traditional TF-IDF vectorization.

Pre-Processing and Feature Extraction

The text of the news articles was cleaned (made lowercase, punctuation removed, and lemmatized). I also performed a TF-IDF vectorization fit on the training data and then transformed the test data to the fit of the training data to prevent data leakage. At this point I made the decision to limit the TF-IDF Vectorizer to the 500 best features based on the training data. Leaving me a total of 517 features total. In future projects different number of features should be assessed, but for a proof of concept project, this should be sufficient. All data was scaled using sklearn's standard scaler. Table of accuracy, precision, recall, and f1 score for each model is found in appendix 1.

Modeling

When deciding what models to try I considered which models work best on binary classification. I chose the classic Logistic Regression, Random Forest Classifier, and Linear SVC, a linear variation of a Support Vector Machine. Finally since I had three models I used a Voting Classifier to aggregate them into one model. All the models including the Voting Classifier performed approximately between 92-93.2% accuracy at a .5 threshold, with better performance on real news. A Random Forest Model was chosen as a result of a slightly better ROC curve. Textstat scores and metrics were fairly significant features in both the best performing models. All accuracy, precision, recall, and f1 scores are calculated at a .5 threshold unless otherwise specified.

Logistic Regression:

Logistic Regression was used as it is a classic model to use for binary classification tasks. I tuned the hyperparameters using GridSearchCV for penalty, C, and solver. Both the l1 and l2 penalties were searched. 7 values of C were assessed using the logspace between -3 and 3. And three solvers were assessed: newton-cg, lbfgs, liblinear. The best parameters for the search were: C = 1000.0, penalty = l2, solver = newton-cg. Its accuracy on the training data was 92.7%, or approximately a .4 improvement over the base model and on the test data the improvement was 92.5% or approximately a .2% improvement. It had a similar profile of performing better when classifying real news. Its values for precision, recall, and f1 score was between 94-95% for real news and 88-90% for fake news.

The top 5 features of this model were textstat metrics such as letter count and character count, most of the rest were readability scores. The only TFIDF vector in the top 10 feature list was 'reuters'.

Random Forest Classifier:

I performed a RandomizedSearchCV over most of the hyperparameters: bootstrap, criterion, max depth (10-100, in increments of 10), max features, minimum samples per leave (1,2,4), minimum samples split (2, 5, 10), number of estimators (10, 50, 200, 400, and increments of 200 until 2000). The final parameters of the RandomizeSearchCV were: n_estimators = 1600, min_samples_split = 2, min_samples_leaf = 1, max_features = 'auto', max_depth =50, criterion = 'entropy', bootstrap = False.

This model overfit the training data and the accuracy score on the training data was a perfect 100%. The score on the test data for this model was 92.8%. The concern with this model is not the accuracy, but it's generalizability. This model also performed better on real news. The precision, recall, and f1 scores were in the range of 87%-91%, this is the largest range of any of the models on the test data.

The top ten features of the model only had the difficult word score enter the top ten from the textstat metrics and scores. The rest were all TF-IDF vectors. Though the vast majority of the metrics and scores did appear in the top 10% of features.

Linear Support Vector Classifier:

I tuned the hyperparameters using GridSearchCV with the following potential settings: C consisted of 7 values between the logspace of 0-5, penalty was l1 or l2, loss was either hinge or squared hinge. The best parameters found were: C = 14677.992676220705, loss = 'squared_hinge', penalty = 'l2'. This model had a 93.2% accuracy on the training data and 92.4% accuracy on the test data. This was an improvement over the Logistic Regression model of about .1%, but overfit slightly more.

Like the other models the LinearSVC performed better on real news. For real news the precision, recall, and f1 score were all between 94-95% and for fake news these scores were between 88-90%.

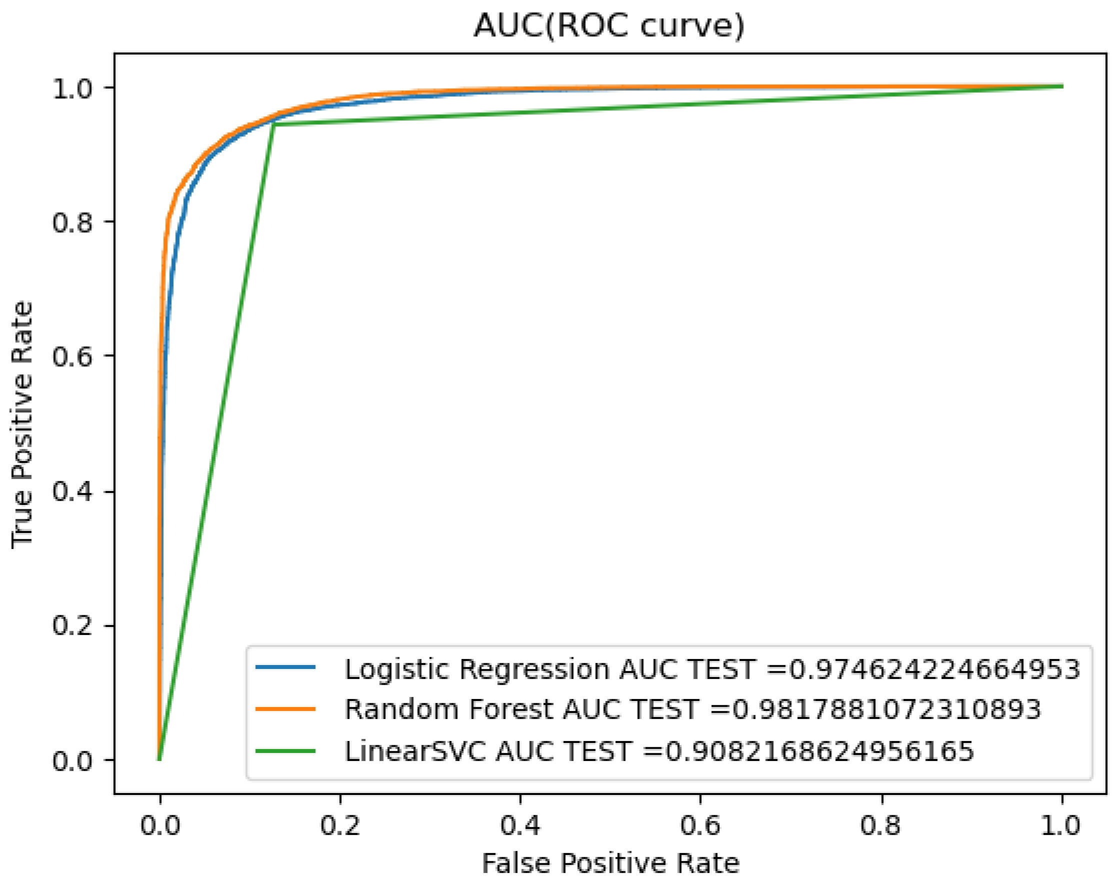
Unlike the other models LinearSVC is a little harder to peer into to get the best features. Given that one point of interest for this project was readability scores, Using the same parameters I performed a GridSearchCV on the same dataset not including readability scores. Its accuracy decreased to 92.5% on the training data and 91.9%. About a .5% decrease in performance. Whether this is a significant decrease is a business decision, but readability scores are relatively computationally cheap compared with tuning and running the models themselves. However, this was not the final model anyways

Voting Classifier:

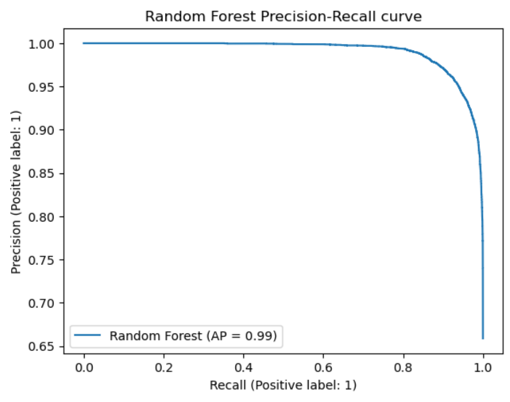
The voting classifier consisted of the tuned versions of the three above models and performed in about the same range, achieving a .01 percent increase over LinearSVC and performing equally well with Logistic Regression. It also performed less well on fake news with precision, recall, and f1 score in the 88-90% range. However, given a slightly inferior performance to Random Forest and a difficulty of assessing the model with further metrics, such as an ROC curve, this model was abandoned as an option.

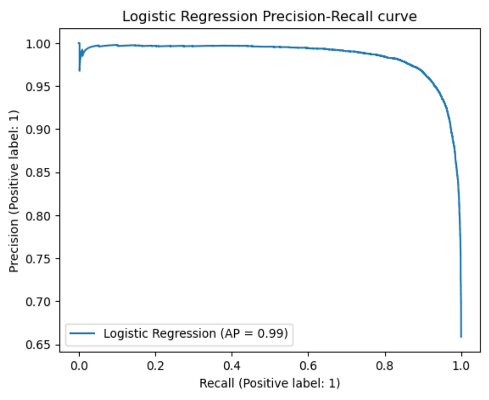
Comparing the Models:

The accuracy, precision, recall, and f1 scores of the above models at a threshold of .5 are approximately equal. However, when computing the ROC curve the LinearSVC model does not perform as well as the other models. And similarly to accuracy, the Random Forest model performs slightly better than logistic regression. See the ROC curves below with the AUC scores.



Further the precision recall curves were virtually identical:

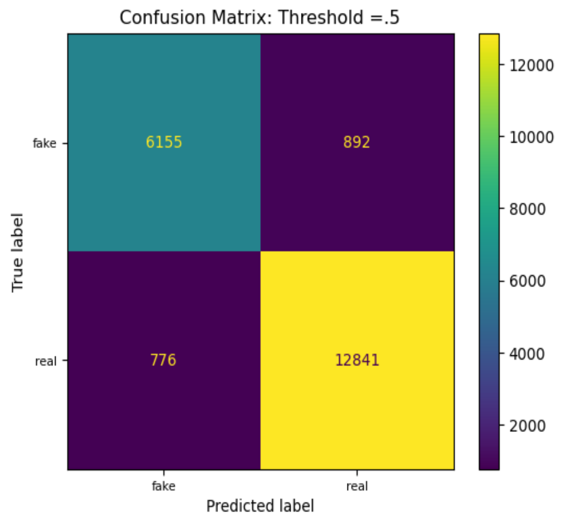


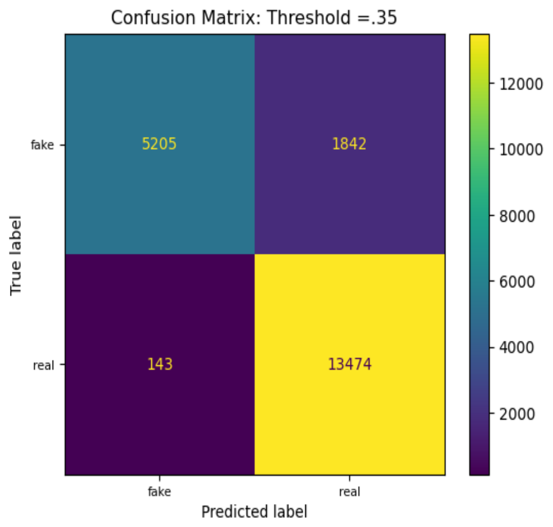


It is a virtual tie between these two models and it might be helpful to test both on real data. However, Random Forest has the edge by a very small margin and will be chosen as the final model

Random Forest Threshold:

However, the Random Forest threshold should be adjusted to limit the number of news articles incorrectly classified as real when they are fake. I chose the threshold of .35 to compare as that is the approximate ratio of fake to real news in the cleaned data. See the confusion matrices below for threshold at .5 and .35.





For our fake news warning system I would recommend a threshold of .35 as this would minimize the number of times we incorrectly call real news fake. It would increase the number of fake news that we call real by a factor of two and decrease the accuracy from ~92% to ~90%. However, that is a necessary trade off.

Conclusion:

Metrics such as those found in textstat, and readability scores should be considered in Fake News algorithms. They were the top 9 out of 10 of the best features in Logistic Regression and were common in the top 10% of features in the final Random Forest Model and were more prominent in the runner-up Logistic Regression Model.

Appendix 1: Results from tuned Models on Test Data at .5 threshold
*FN = Fake news, RN = Real News

	Logistic Regression	Random Forest	LinearSVC	Voting Classifier
Accuracy	92.5%	92.8%	92.4%	92.5%
Precision	FN: 90% RN: 94%	FN: 91% RN: 93%	FN: 90% RN: 94%	FN: 90% RN: 94%
Recall	FN: 88% RN: 95%	FN:87% RN:96%	FN: 88% RN: 95%	FN: 88% RN: 95%
F1	FN: 89% RN: 94%	FN: 89% RN: 95%	FN: 89% RN: 94%	FN: 89% RN: 94%