Fake News Detection Using Readability Scores and TDIF Vectorization

Daniel Ansted

Executive Summary:

Fake news is a major problem for news organizations and social media sites. In this paper we apply readability measures and introduce a machine learning model that is able to detect and flag 74% of fake news, which could potentially reduce churn and increase revenue by $16 million per month.

Introduction

Fake news can spread very quickly. 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]

Many social media platforms do use algorithms to detect fake news. These algorithms commonly use TF-IDF vectorization, N-grams, and other features based on words and phrases. The proposed model includes TF-IDF vectorization, but supplements this with commonly used readability scores.

Legitimate news organizations have style guides. 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 the readability of an article could leverage this writing style uniformity helping to detect real news from fake news. Luckily there are already formulas and python libraries to determine readability. Let's look at one popular formula, the Flesch Reading Ease test.

206.835 - (1.015 * total words/total sentences) - 84.6 * (total syllables/total words))

This readability formula defines a text as easy to read if it has short sentences and short words. 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.

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 articles.

Simply removing exact duplicates was not sufficient, as some articles were inexact duplicates. Traditional fuzzy matching was computationally expensive on such a large data set

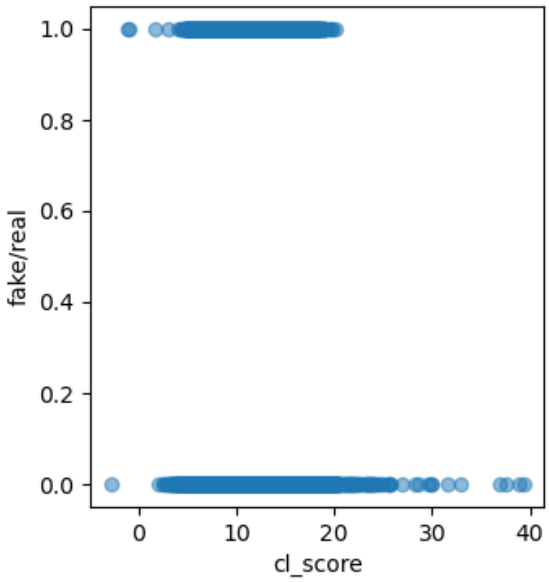
---

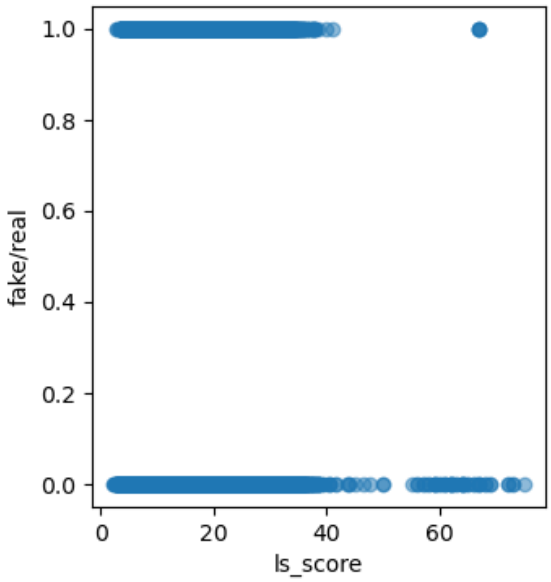
[1] https://www.science.org/doi/10.1126/science.aap9559
[2] https://pro.morningconsult.com/analysis/distributing-news-has-hurt-social-platforms-trust-among-consumers
[3] https://www.kaggle.com/datasets/algord/fake-news
https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets

with large numbers of words, so a vectorized fuzzy matching was implemented. However, there were still some articles that were nearly identical. These were discovered by having duplicate or near duplicate readability scores. Through trial and error, a combination of readability scores and other metrics 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 these datasets were Russian and a language with Arabic script. regEx was used to identify characters in those languages, which were then removed from the final dataset.

Exploratory Data Analysis

First the documents were scored for readability.[4] 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. However, as will be seen, they can serve as a useful feature along with traditional TF-IDF vectorization.

Pre-Processing and Feature Extraction

[4] Most of the scores from this library were used. These scores are primarily for English language documents. There are some scorers for other languages. However, the readability scores are not being used for their intended 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.

The text of the news articles was cleaned (made lowercase, punctuation removed, and lemmatized). A TF-IDF vectorization was fitted on the training data and then this was used to transform the test data preventing data leakage. For computational reasons, only the 500 best TF-IDF features from the training data were chosen. This made a total of 517 features. In future projects a different number of features should be assessed. All features were scaled using sklearn's standard scaler. Accuracy, precision, recall, and f1 score for each model is found in appendix 1.

Modeling

The models that were chosen were Logistic Regression, Random Forest Classifier, and Linear SVC. A Voting Classifier was then used 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 chosen as it is a classic model to use for binary classification tasks. The hyperparameters were tuned 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. 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 TF-IDF vector in the top 10 feature list was 'reuters'.

Random Forest Classifier:

Random Forest Classifier was chosen for performance and simplicity of use 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. 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 only textstat feature that entered the top ten was the difficult word score. 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:

This model's hyperparameters were 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%.

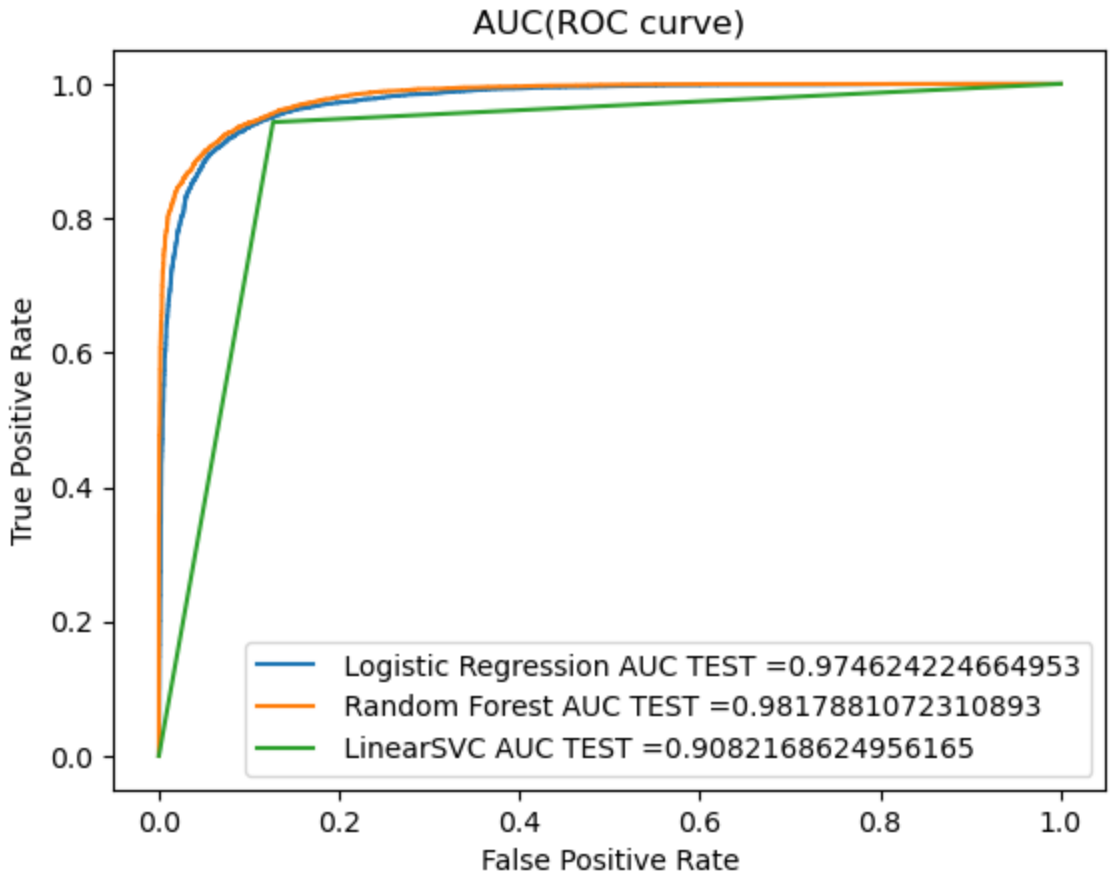
Unlike the other models LinearSVC is a little harder to peer into to get the best features. Given that one point of interest for this project was readability scores, Using the same parameters GridSearchCV was performed on the same dataset not including readability scores. Its accuracy decreased to 92.5% on the training data and 91.9%. About a .6% 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 chosen as the final model.

Voting Classifier:

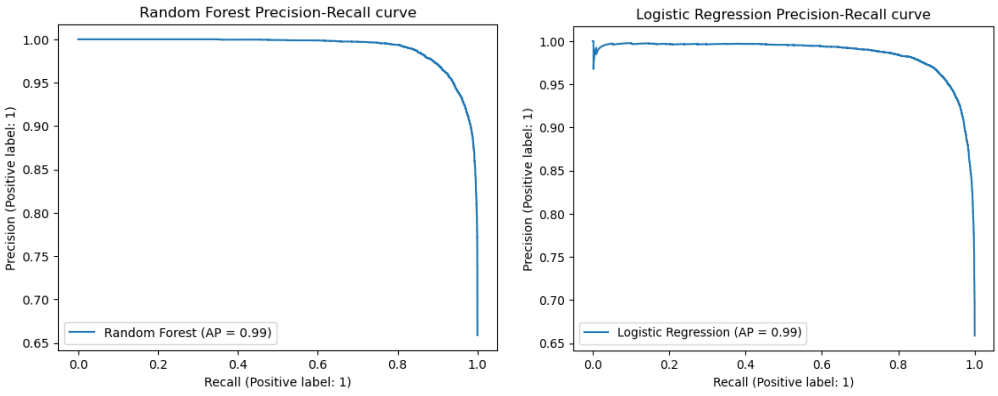
The voting classifier consisted of the tuned versions of the three above models and performed in about the same range, achieving a .01 percent increase over LinearSVC and performing equally well with Logistic Regression. It also performed less well on fake news with precision, recall, and f1 score in 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.



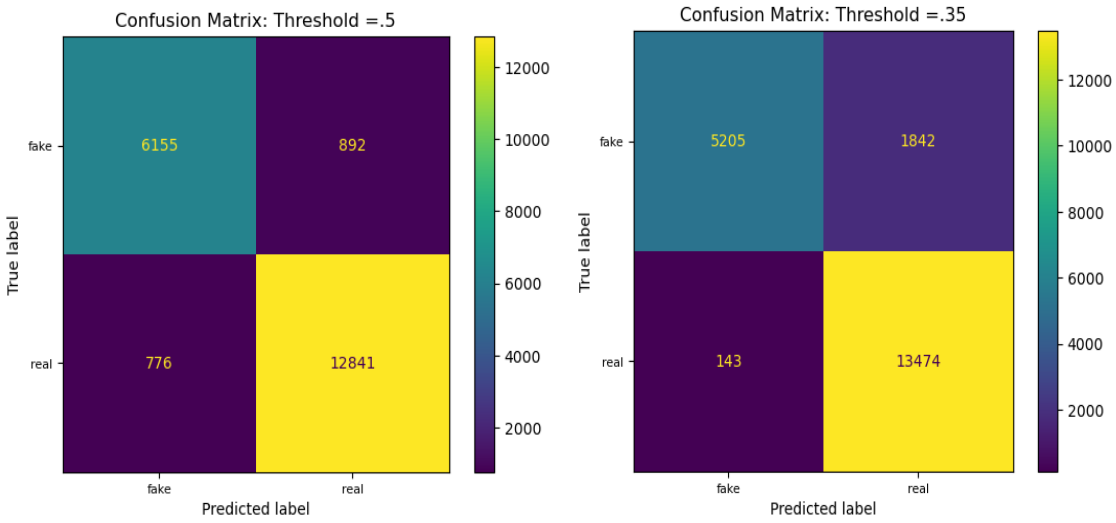
Further the precision recall curves were virtually identical:



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 for the purposes of this report.

Random Forest Threshold:

The threshold of .35 was chosen as a comparison to the default of .5 because 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 a threshold of .35 is recommended 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:

Monthly churn for MyFace is estimated to be around 1-2%. 44% of users who stopped using a social media platform said a major reason why was fake news. Average Revenue Per User is estimated to be about 3.33$ per month. MyFace has about 2.96 billion monthly users. Incorrectly labeling real news is costly, thus the threshold choice prioritized minimizing this. This came at a cost to the recall score which is .74.

The following calculation takes the lower value of the monthly churn of MyFace (1%). Assumes half of the 44% of people who said fake news was a major reason for leaving a platform would stay.

.01 x .22 x 3.33$/month per user x 2.96 users x.74 ≈ $16 million/month

This is a conservative estimate and does not factor in any considerations about marketing being able to promote these new additions to our service.

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%