Liar, liar, pants on fire?

Fake news detection with supervised machine learning

Jørgen Baun Høst
University of Copenhagen
pjz633@alumni.ku.dk

Marius Heltberg Lassen
University of Copenhagen
pgb206@alumni.ku.dk

Abstract

In this paper, we analyze the linguistic characteristics of fake news and propose a method to detect it. This involves scraping annotated fact checks from PolitiFact.com to train a supervised machine learning model by logistic regression. Through careful data cleaning and evaluation of model performance, we select a model with an accuracy of 71 pct. For illustration, we apply this model on data scraped from Peacedata.net, a short-lived website banned from Facebook and Twitter in the lead-up to the 2020 US election. Of 154 articles, our model labels 82 of them as 'fake'.

Introduction to Social Data Science: Project
Department of Economics
University of Copenhagen
Submission date: August 23, 2022
Character count: 27,663 (max. 28800)

Contributions:

Jørgen Baun Høst:	Sections 1, 2.1.0, 2.1.2, 2.3, 3.1, 3.3, 4.1, 4.2, 5.
Marius Heltberg Lassen:	Sections 1, 2.0, 2.1.1, 2.2, 3.0, 3.2, 4.0, 4.2, 5.

Contents

1 Introduction

In light of the 2016 US election, 'fake news' as a phenomenon rose to global attention. Were it not for this intentional interference, it has been suggested that Hillary Clinton would have pipped Donald Trump to become the 45th President of The United States, see Parkinson (2016) and Dewey (2016). This begs the question: What can we do to detect this spread of false information? One possibility is to employ machine learning techniques. Specifically, our goal with this paper is: *How precisely can we automatically detect 'fake news' using supervised machine learning?* We do this by leveraging scraped data of fact checked news stories and statements from PolitiFact.com to train our supervised machine learning model.

One may ask how do we define *fake news*? Kshetri and Voas (2017) break up false information in two sub branches: *disinformation* and *misinformation*. Disinformation is false information spread to intentionally deceive, manipulate and mislead. Misinformation, on the other hand, is simply wrong information provided *unintentionally* or information perceived erroneously, for example that all classes at University of Copenhagen start at 8.00 sharp and not actually 15 minutes later. Like Kshetri and Voas (2017), we define 'fake news' as *disinformation* typically produced manually by individuals and/or state entities with the intention to deceive readers.

Our model classifies statements with an accuracy of 70-80 pct. on our test data set from PolitiFact, depending on the model specification. We apply our classification model on text data from Peacedata.net, a short-lived website that was banned from Twitter and Facebook. This site was owned by the Internet Research Agency, an entity that has proven connections to the Russian state (DiResta et al. 2019). Our model classifies 82 out of 154 articles as 'fake' news.

This paper adds to the growing literature on 'fake news' and its implications for the political climate and outcomes. This includes Allcott and Gentzkow (2017), who take a theoretical approach by discussing the economics of fake news and sketches a model that illustrates why fake news appears in the media landscape. They find that pro-Trump fake stories were shared on Facebook a total of 30 million times in the lead-up to the 2016 US election, significantly more than fake pro-Clinton stories. Reis et al. (2019) employ more advanced machine learning models to detect fake news. They achieve around 85 pct. accuracy in a random forest model with supervised learning using annotated BuzzFeed news articles relating to the 2016 US election. Reis et al. (2019) make use of 141 different textual features, which include bag-of-words, like we do, and psycholinguistic features that capture biased and persuasive language.

This paper is structured as follows: Section 2 goes into detail how we have acquired the data, the ethics involved doing this and some descriptive statistics. In Section 3, we describe the methodical approach to the supervised machine learning model. In Section 4, we present our results, including evaluation of model performance, and in section 5 we summarize and conclude.

2 Data

To investigate our research question, we collected data from PolitiFact, a website constructed by the Poynter Institute specifically to fact check statements, articles and social media posts from politicians and the general public. The fact checking process is done by 'real' journalists and thus is ideal for supervised machine learning techniques, provided that we adhere to the terms of use.

We supplement our analysis by collecting data from Peacedata.net, a now defunct news website owned by the Internet Research Agency (IRA). IRA is an entity directly linked to the Russian state with the intention to spread disinformation and interfere with US elections (DiResta et al. 2019). The motivation behind us including Peacedata.net in our analysis is how they specifically tried (and in some cases succeeded) to recruit Western journalists to write columns for them, as long as they were focused on *"anti-war, anti-corruption, abuse of power, or human rights violations"* (Delaney 2020; Stubbs 2020). And so, it would be interesting to see which stories, if any, our model predicts to be 'fake'.

2.1 Collection

In general, we have focused on creating 'custom' functions that concisely illustrate our approach with ample documentation to follow. Crucially, whenever we made a request to scrape a webpage, we had to define a 'header' in a dictionary format. This dictionary had our names and non-commercial/academic intentions for acquiring data via scraping. Further, we integrated the use of log functions and auxiliary 'error' lists to inspect pages, where the scraping process did not go to plan.[1]

2.1.1 PolitiFact.com

To scrape PolitiFact, we made use of the `requests` package in `Python`. This package allows us to send HTTP (HyperText Transfer Protocol) requests to obtain the

1. We acknowledge and thank the lecturers/TA's from the course *Introduction to Social Data Science* at University of Copenhagen for making codes publicly available.

underlying HTML (HyperText Markup Language) code of the webpage in question. The `Beautiful Soup 4` package then allowed us to navigate the HTML-code and pull out the desired data. The HTML-code of the PolitiFact website has a fairly logical composition and so the scraping process was relatively smooth. First, we accessed the latest fact checks by PolitiFact which contains all fact checks in chronological, split up in 728 pages. We made a list of these 728 unique URL's and then looped through each page, extracting all information from the headline of every article: Date, statement, classification and link to the article. Throughout the webscraping process, we made sure to backup the collected data, as to avoid unnecessary 're-scraping'. We stored backups and data in general in the fast and efficient file format of `Apache Parquet`, which also stores data types (Vohra 2016). Finally, PolitiFact also checks whether politicians did go back on their promises and/or statements made in the past. We drop these articles, as they would provide considerable noise to our dataset.

Next, we looped through every article extracting the article paragraphs, sources, the journalist's conclusion and tags associated with each article. Here, we made use of the URL we scraped earlier to avoid using dynamic tools, e.g., the `Selenium` package in Python. We estimated that this would have substantially slowed the process. In the end, we ended up with 21,000 articles, with each statement categorized on a scale from "true news" to "fake news", namely the categories 'true', 'mostly true', 'half-true', 'barely true', 'false', and 'pants-on-fire'.

2.1.2 Peacedata.net

Scraping Peacedata.net was a bit of a challenge. As the website no longer exists in its 'news form', we scraped The Web Archive for available snapshots. We find snapshots available from April 2020 to the end of September 2020, before it was effectively shut down. This can be done by using The Web Archive's API with a specified date range, fetching a JSON file with a list of 'available' URLs (Pezoa et al. 2016). With these, we followed the same approach as with PolitiFact and used `requests` and `BeautifulSoup` to obtain the HTML-code for the Peacedata.net webpages. As (i) these URLs are snapshots of Peacedata.net's front page at different times, and (ii) the HTML-code is slightly finicky to assess, we concluded that this had limited use to us. Instead, we extracted links to individual articles where we could (and avoid any dynamic JavaScript content). From these links, we scraped the title, article text, and dates where possible. This leaves us with a dataset of 154 articles after removing duplicated articles.

2.2 Ethics

Scraping data on the internet poses crucial questions to the terms of use. As a rule-of-thumb, you are not allowed to webscrape with for commercial purposes. Further, given the recent adoption of the General Data Protection Regulation (GDPR) in the EU, the concern of enforcing privacy rights has become a much debated topic in the public domain, particularly in light of the Facebook-Cambridge Analytica scandal (Illing 2018).

In general we think we adhere to two main rule-of-thumbs; (i) we clearly state our intentions in a header when requesting information from the web server, and (ii) we do not store any private information nor make any attempt in trying to find this information that may or may not be publicly available.

The two sites has slightly different terms of use. PolitiFact states that the site's content is for your personal, non-commercial use, which we completely and transparently adhere to. Additionally, we are not allowed to use the material for 'unlawful purposes', like to track down and post defamatory material guided towards, say, the journalists at PolitiFact. It becomes slightly complicated for Peacedata.net as the site only exists in 'snapshot' form from The Web Archive. Interestingly, Peacedata.net did very openly share they had a Creative Commons license in each article. In essence, this implied royalty-free open-use of article content, provided that you attributed properly (Commons 2022). In this regard, we are convinced that we adhere to these terms, even though they no longer are in effect. Looking at the terms of use from The Web Archive, it is stated that access to the archive is provided free of charge. Use of any content from the archive is for scholarship and research purposes only.

Finally, we tried to comply with potential `robots.txt`-files, which specifies what webcrawlers are allowed to access on a given website. Neither PolitiFact or The Web Archive have specified any `robots.txt`-files, and interestingly, The Web Archive made a blog post stating that *"Robots.txt meant for search engines don't work well for web archives."* Hence, with webscraping intentions of solely academic character, we feel certain that we comply with their regulations.

2.3 Descriptive statistics

To provide a visualize representation of our datasets, we make a few descriptive plots. From Figure 1, we see that Facebook is by far the most widely used tag in PolitiFact's fact-checking articles in the last five years.[2] This includes fact checking videos, posts or statements made on the social media platform. Interestingly, the 'Coronavirus' tag is third most used tag in the last five years, despite only being an actual subject in the last two years with over 1,300 tags in total and over 600 tags in 2020 alone. One can see that Donald Trump, fake news and elections has also been ever present in PolitiFact's fact checking in the last five years.

Figure 1: PolitiFact: 10 Most Common Tags in the Last Five Years

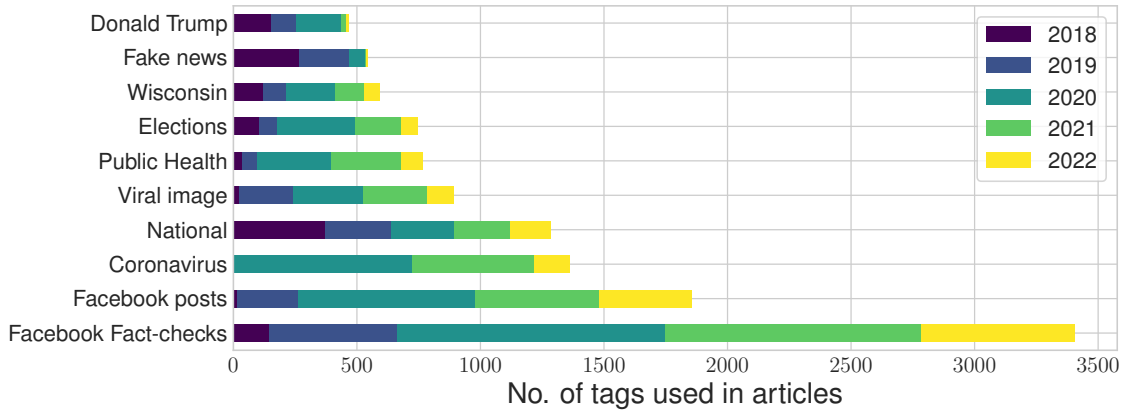


Figure 2 depicts the cumulative development in PolitiFact's statement classification, where we find a somewhat even distribution before 2020, except for the 'pants-on-fire' classification. Even in the lead-up to the 2016 US election, PolitiFact did not identify an abnormal amount of untrue statements. Of course, this may be due to selection. Interestingly, in 2020, the amount of statements classified by PolitiFact as 'false' surges to almost 6000 in total, with around 2000 coming in the last two years or so.

While we do not nearly have the same sample size for the Peacedata.net dataset, it is clear from Figure 3 that issues fact checked by PolitiFact is more-or-less the same issues most commonly found by the articles scraped from Peacedata.net. This ties in well with the most common words used by PolitiFact and Peacedata.net from Figure 4. The word 'people' and 'state'/'government' can even be found in both PolitiFact and PeaceData's most common words. At a glance the most used words from Peacedata seem to have more assertive connotations than what is found

2. This figure and all figures to come are made with the `pandas`, `seaborn`, and `matplotlib.pyplot` packages in `Python` (McKinney et al. 2010; Waskom 2021; Hunter 2007).

in the PolitiFact dataset. The fact that 'says' is the most common word in the PolitiFact.com dataset can be plausibly linked to how these statements are written. It could be that *person A says...(fact checked statement)* or simply that statements have an accusatory nature.

Figure 2: PolitiFact: Cumulative Statement Classification

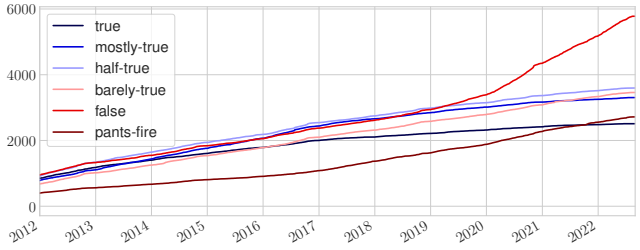


Figure 3: Peacedata.net: 5 Most Common Tags

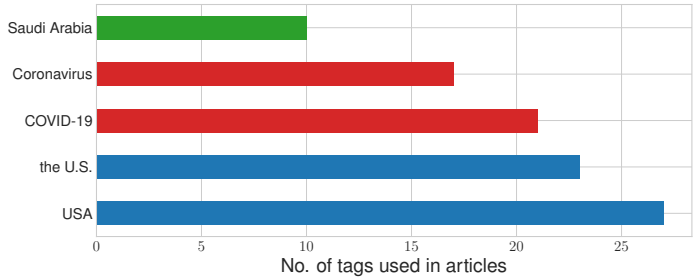
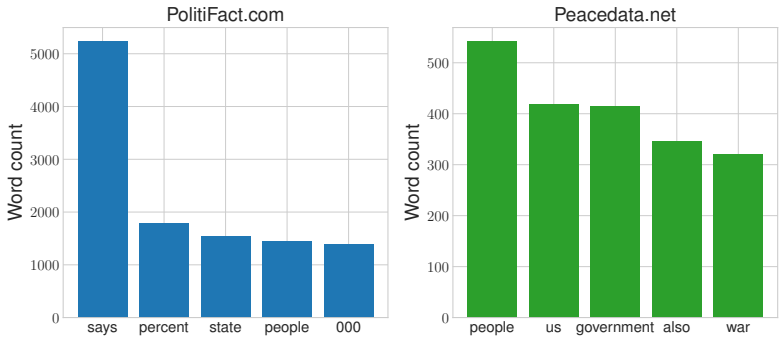


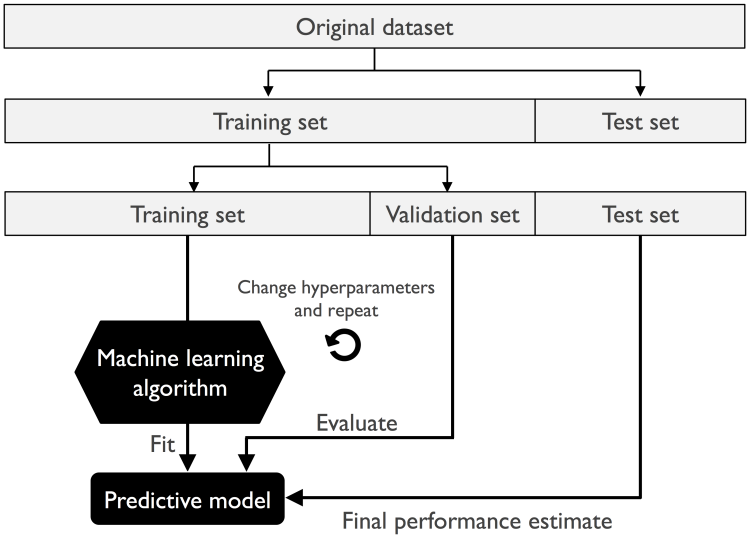
Figure 4: Word Count



3 Methodology

This section explains the route from data to results (McKinney et al. 2010). It will cover the basics behind working with text as data, the various packages used and calculations made in `Python` and the theory of the (penalized) logistic regression model. This route and its different steps closely follows Raschka and Mirjalili (2019) and are depicted in Figure 5. Throughout the analysis, we make use of the `numpy`, `pandas`, `sklearn`, and `tqdm` packages (Harris et al. 2020; McKinney et al. 2010; Pedregosa et al. 2011).

Figure 5: Model validation



Source: Bjerre-Nielsen et al. (2022, Lecture 12)

3.1 Text as Data

In order for us to infer meaningful predictions, we had to appropriately clean our text data. As is standard, we converted every letter to lower case to avoid identical words being interpreted differently. We then removed HTML-markup and non-alphanumeric characters with Regular Expression Operations, (`re`) – a standard package in Van Rossum (2020). Next, we took advantage of the extensive library of Natural Language Toolkit, `nltk` (Bird et al. 2009). Specifically, we use the their `tokenize` package to divide the text (or the strings) of the articles into lists of substrings, where, in our context, a substring is a word. Then, we use their dictionary of English "stopwords", to remove words that carry next to no meaning in the analysis, such as "is", "the" and "a". The full list of stopwords can be found

here. Finally, we applied their `lemmatizer` to determine the lemma of the words, i.e., align different conjugations and word endings, such that the words are grouped based on their meaning and not their grammatical form. This procedure follows Jurafsky and Martin (2018, ch. 2.4).

Then, after cleaning the strings, we take the *bag-of-words* approach, which *"allows us to represent text as numerical feature vectors"* (Raschka and Mirjalili 2019, p. 262). The first step is to count the occurrence of each word, and then, secondly, assess the word relevancy with the *term frequency-inverse document frequency* (tfidf) technique. The approach is carried out by applying the `TfidfVectorizer` from Pedregosa et al. (2011) (see section 6.2.3.4 in the documentation for details).

3.2 Machine Learning and the Logistic Regression Model

To classify statements from the PolitiFact as either 'true' or 'fake', we use logistic regression. Given that we map a binary classification problem, we are essentially inferring 'probabilities', e.g., $p(y = 1|\mathbf{x})$, where our target variable y classifies whether a statement or article is 'fake news' ($y = 1$) or 'true' ($y = 0$), and $\mathbf{x}$ is the features, namely the vectorized bag of words. Obviously, these probabilities translate directly into the predictions of the model – for $p(y = 1|\mathbf{x}) \geq 0.5 \Rightarrow \hat{y} = 1$. In the logistic regression model, the conditional probabilities are calculated with the sigmoid function (or, the inverse form of the logit function), which can be stated as

$$\phi(z) = \frac{1}{1 + e^{-z}} (= p(y = 1|\mathbf{x}), \tag{1}$$

where $z = \mathbf{w}^T\mathbf{x} = w_0 + w_1x_1 + ... + w_mx_m$ is the net input, i.e. m features with corresponding weights, and a bias term w_0. Later, when interpreting estimated weights, it is worth noting that $\hat{y} = 1$ if $z \geq 0$ and $\hat{y} = 0$ if $z < 0$.

In accordance with the theory of maximum likelihood estimation (MLE), the weights are determined by maximizing the log-likelihood function:

$$\ell(\mathbf{w}) = \sum_{i=1}^{n} \left[y_i \log\left(\phi(z)\right) + (1 - y_i) \log\left(1 - \phi(z)\right) \right] \tag{2}$$

Actually, as is common practice in optimization problems, we will be minimizing a cost function. Moreover, we will include a penalty term to alleviate issues with "over-fitting" the data. The resulting cost function can be stated as

$$J(\mathbf{w}) = \sum_{i=1}^{n} \left[y_i \log\left(\phi(z)\right) + (1 - y_i) \log\left(1 - \phi(z)\right) \right] + \frac{\lambda}{2} ||\mathbf{w}||_2^2, \tag{3}$$

where λ is the regularization parameter, and $\frac{1}{2}||\mathbf{w}||_2^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is an L2-penalty, which works by shrinking the absolute size of the weights.[3]

3. Note, that the logistic regression from `sklearn` uses the inverse regularization parameter

3.3 Model Selection

The binary outcome classification leaves us with a choice of how we want to train our model. Given PolitiFact's six-way classification from 'true' to 'pants-on-fire', one may ask which classification of the PolitiFact statements to use. Do you only train the model on *true* vs *pants-on-fire* statements? Or do you include all 6 statement classifications and lump them together in a binary outcome ? There is no clear answer to these questions, which is why we make three different partitions of the full dataset and evaluate the performance of each of them. The partitions are specified in Table 1. Worth noting is that these partitions not only influence the total number of observations, but also the ratio between $y = 1$ and $y = 0$, cf. Figure 2. For each of these partitions, we follow Figure 5 and split the sets into training sets and test sets. We will refer to the training sets in this step as the *development* sets, as to avoid confusion between two different training sets later on.

Table 1: Dataset Partitions

$y = 0$	$y = 1$	n
true	*pants-on-fire*	5,227
true and *almost true*	*false* and *pants-on-fire*	14,311
true, *almost true*, and *half-true*	*barely true*, *false* and *pants-on-fire*	21,367

To illustrate the model performance for each development set in the dataset partitions, Figure 6 visualizes validation and learning curves for the three different model specifications. In the calculations of the learning and validation curves, we have imposed a 10-fold cross validation step, in each step splitting the development set into a training set and a validation set, again, in correspondence with Figure 5. The learning curve takes each training set and construct subsets of different sizes, which are then fitted to and transformed by the Tfidf-Vectorizer to make a *bag-of-words*. The bag is subsequently fitted to the Logistic Regression model. This compound model (or `pipeline`) then takes both (the subset of) the training set and the validation set as inputs to calculate predictions and evaluate accuracy

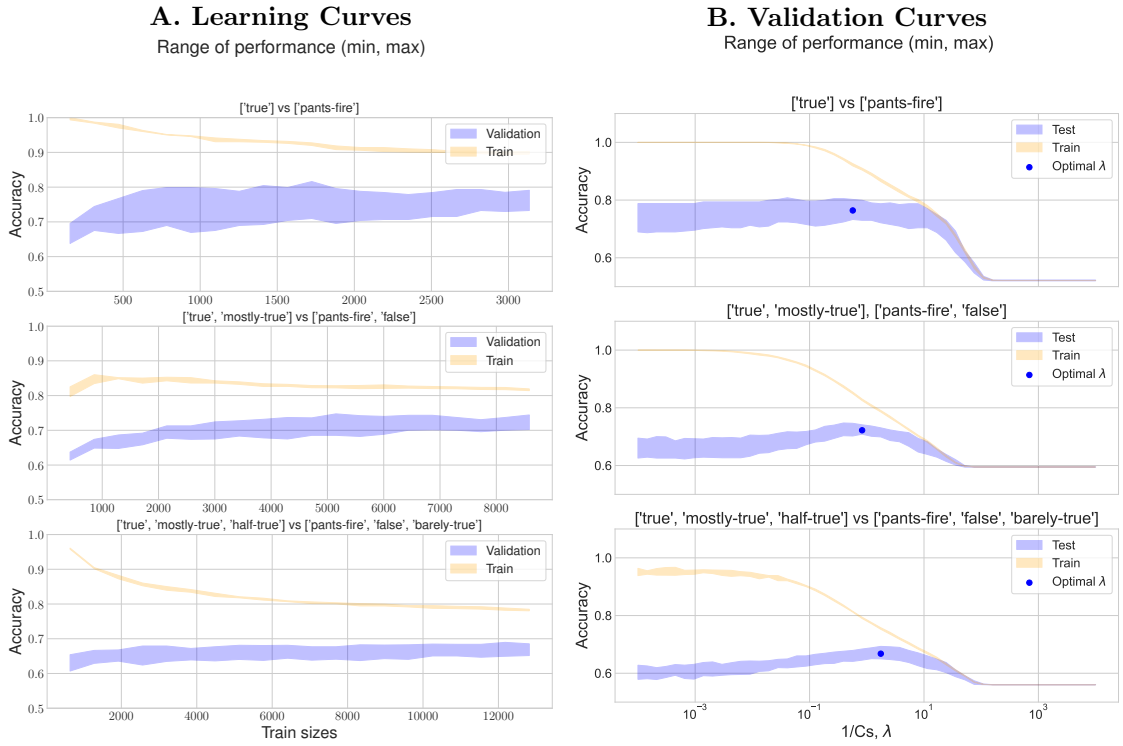
$C = 1/\lambda$. This implies the following cost function (Pedregosa et al. 2011):

$$\tilde{J}(\mathbf{w}) = C \sum_{i=1}^{n} \left[y_i \log\left(\phi(z)\right) + (1 - y_i) \log\left(1 - \phi(z)\right) \right] + \frac{1}{2} ||\mathbf{w}||_2^2$$

This cost function and equation (3) are equivalent from an optimization standpoint, as the latter is simply divided by λ (a constant), which is a linear transformation. We choose the formulation with λ following Raschka and Mirjalili (2019).

(accuracy$(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{\hat{y}_i = y_i\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function).[4] Similarly, in the calculations of the validation curves, the training set is fitted and predictions are made, but instead of varying the size of the training set, we now vary the size of the hyperparameter, λ. For both types of curves, Figure 6 plots the minimum and maximum accuracies obtained under the different calibrations. These bands provide a *quick view* into the variance of the models, while actual confidence intervals should be computed with Monte Carlo simulations, or similar.

Figure 6: Model Performance



What is evidently clear from the learning curves in Figure 6 is that the model does suffer from significant variance. While the 'true vs pants-on-fire' model tend to have the highest maximum accuracy, the 10-fold cross validation process highlights how sensitive the model is to a particular partition of the dataset. For a training sample size of $1,500$, the accuracy of the model jumps between 70-80 pct. depending on the dataset partition. This is not surprising, given we have quite a small training sample size of no more than $3,500$ statements for this model specification.

The validation curves in Figure 6 in conjunction with Table 2 show the regularization strength needed to achieve the highest accuracy for the different specifications. The optimal hyperparameter, λ, is calculated as the λ that gives the highest mean accuracy of the validation set. As expected, the more items we include from the

4. Note, learning curves are calculated with `sklearn`'s baseline calibration of $C = 1/\lambda = 1$.

PolitiFact scale, the higher the regularization parameter, λ, is needed to achieve the highest accuracy. This makes intuitive sense. If we lump the whole scale in a binary outcome, effectively fitting ambiguous statements that are classified as barely true or half-true in our model, the noise in our dataset substantially increases. To counteract this, we need to increase the regularization strength, or penalty term, to avoid over-fitting.

Table 2: Optimal Hyperparameter and Corresponding Accuracy

	Model 1	Model 2	Model 3
λ_i	0.569	0.829	1.758
Accuracy	0.764	0.722	0.668

Note: λ_i for $i = 1, 2, 3$ denotes the model that includes increasingly more items on the PolitiFact fact check classification scale.

4 Results

In the previous section, we have visualized our datasets and trained a machine learning model using logistic regression using different specific specification. The diagnostics of each model specification illustrate the trade-off between variance and accuracy. The '*(True, Mostly true) vs (False, Pants-on-fire)*' model seem to strike the best balance. This model hovers around 70 pct. accuracy compared to consistently under 70 pct. for the model that utilizes the full Politifact statement scale.

Table 3 show the coefficient sizes of this particular model with the optimal hyperparameter, $\lambda = 0.829$. The larger the absolute coefficient size, the higher the impact each word has on the predicted classification outcome. Interestingly, the common denominator seem to be American politics and the COVID-19 pandemic. Words such as 'biden', 'pelosi', 'vaccine' and 'covid/coronavirus' triggers the algorithm to predict a given statement to be fake. On the other hand, words such as 'percent', 'million', 'growth' - words that may have a 'quantitative' tone - tends to the swing the pendulum the other way, i.e., that statements typically are classified as 'true'. The bias term is found to be 0.79. This implies that for a 'neutral' statement, i.e., statements lacking impactful words, and only having features with coefficients close to zero, the model tends to classify these as fake news.

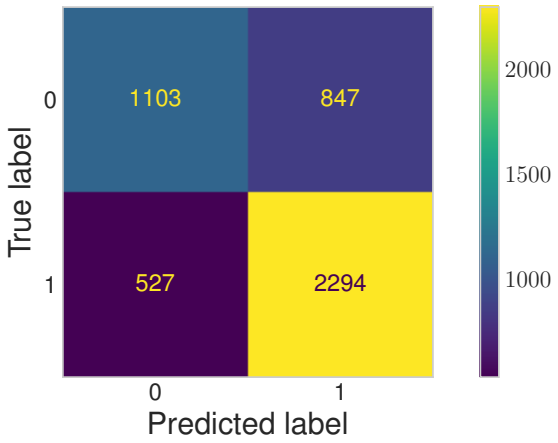
Table 3: Most Impactful Words

Largest		**Smallest**	
biden	4.027278	percent	-3.267239
show	3.273758	since	-2.847953
vaccine	3.120486	half	-2.485818
covid	2.787858	state	-2.305970
photo	2.298474	growth	-2.259099
pelosi	2.258160	le	-2.120455
coronavirus	1.991822	bipartisan	-2.081750
arrested	1.984314	three	-2.064798
face	1.879778	six	-2.015160
democrat	1.784491	twice	-1.968387

Note: We find the bias (intercept) term to be 0.79.

4.1 Evaluating Performance

Figure 7: Confusion Matrix



The confusion matrix in Figure 7 shows the model performance graphically on our test dataset. True positive (TP) and true negatives (TN) are instances where our model correctly predicts statements as either fake (TP) or true (TN). Statements that are incorrectly labelled as fake news are fake positives (FP) and statements incorrectly labelled as true are called fake negatives (FN). In Figure 7 the areas are TN, FP, FN, and TP in the direction of reading.

Furthermore, we calculate three model evaluation metrics:

$$Precision = \frac{TP}{TP + FP} = 0.73 \tag{4}$$

$$Recall = \frac{TP}{TP + FN} = 0.81 \tag{5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = 0.71 \tag{6}$$

Precision essentially measures the fraction of which all news classified as 'fake' are indeed fake. We obtain a precision of around 73 pct. However, it may simply be that we do not have enough true statements to challenge the model for correct predictions, i.e., that an artificially low false positive rate may boost precision.

The recall metric measures the fraction of all statements, which we know to be 'fake', is correctly captured by our model. For 81 pct. of the fake news statements, our model correctly labels them fake.

Accuracy is the overall measure of how well our model correctly labels fake news as fake and true news as true in our test data set. Recall, that we have trained our model on the *(True, mostly true) vs (False, pants-on-fire)*. This model correctly identifies 71 pct. of fake and true statements.

4.2 Model Application

Finally, we apply this trained model on the dataset from Peacedata.net obtained through The Web Archive:

Table 4: PeaceData.net Classification Count

'Fake' news	82
'True' news	72

The model predicts that of the 154 articles from PeaceData.net, 82 of them are 'fake news'. For illustration, we extract the most 'fake' and 'true' article from Peacedata.net according to our predictions. These are found in Table 5 below:

Interestingly, we find that the article with the highest probability of being 'fake' is about a Democratic presidential debate in Iowa aired on CNN in the lead-up to the 2020 American election. When reading the story, it is difficult to conclude whether this article is indeed fake, and we make no claim that it is despite our results. The fact that the story is labelled as fake ties in well with the article's

Table 5: Probabilities of Fake Articles from PeaceData.net

	Most 'fake' article	Most 'true' article
Title	#CNNIsTrash Trends As Push-back Grows Against Oligarchic Election Meddling	US Workers, Minorities Continuously Face Housing Insecurity during the Pandemic
$p(y = 1\|\mathbf{x})$	0.957	0.028

tags ('CNN', 'Democratic Party', 'the U.S.') and the most impactful words from our model in Table 3. At the other end of the scale, we find that an article about housing insecurity during the pandemic has the lowest probability of being fake. Since the article has quantitative keywords such as 'percent' and 'growth', the classification seems plausible with the results from Table 3 in mind. Likewise, we make no claim that this story is indeed 'true', but only that our supervised machine learning flags this as the least suspicious. A more sophisticated model that capture additional linguistic features, such as the one found in Reis et al. (2019), is beyond the scope of this paper and may yield different results. We leave this for future research.

5 Conclusion

In this paper, we scrape the website PolitiFact.com for their annotated fact-checked statements. We are careful to adhere to the terms of use, which in short stipulate that articles are used for non-commercial purposes only and with respect for the right to privacy. These statements are carefully cleaned so that they can be used in a supervised machine learning model that classifies statements via logistic regression.

As PolitiFact has a six-way scale from pants-on-fire to true when classifying statements, we need to make a choice for which specification to use. From our model selection, we inspect the learning and validation curves which shows (i) all models exhibits significant variance, and (ii) the more statements we include, the higher the regularization is needed to achieve the highest accuracy and avoid over-fitting. We argue that the model specification that disregard ambiguous statements that are classified as either 'half-true' or 'barely-true' seems to strike the best balance between bias, variance and accuracy. We then use this preferred model on our test set (with no data leakage), and evaluate its performance.

Finally, we apply this model (with the optimal hyperparameter) on a subset of articles from Peacedata.net, a website recently excluded from Facebook and Twitter for intentional spread of disinformation. We scraped these from The Web Archive. Of 156 PeaceData.net articles, 82 are labelled as 'fake'. We can trace back the two articles which have the highest probability of being 'fake' and 'true' to the most impactful words found in our model of choice.

While we believe that our model produces sensible results, it does not come without caveats and room for future research. Of particular interest is training the model on *much* larger datasets of articles, expand the model to detect patterns of words, e.g., with *n-grams*, and improve on the classification scheme, e.g, with a *multinomial logistic regression* model.

We would like to emphasize that this relatively simple model merely serves to show what text-as-data in conjunction with machine learning can do, rather to have claimed that we have found the panacea to fake news detection. For instance, our model predicts that there is a 69 pct. chance that this very paper is fake news! So take our results with a grain of salt.

References

Allcott, Hunt, and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of economic perspectives* 31 (2): 211–36.

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural language processing with python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Bjerre-Nielsen, Andreas, Hjalte Fejerskov Boas, and Tobias Gabel Christiansen. 2022. *Lecture slides: Introduction to Social Data Science.* Department of Economics, University of Copenhagen.

Commons, Creative. 2022. *Creative Commons licenses.* https://creativecommons.org/licenses/. (Accessed on 08/21/2022).

Delaney, Jack. 2020. I'm a freelance writer. A Russian media operation targeted and used me — Jack Delaney — The Guardian. (Accessed on 08/21/2022).

Dewey, Caitlin. 2016. Facebook fake-news writer: 'I think Donald Trump is in the White House because of me' - The Washington Post. (Accessed on 08/18/2022).

DiResta, Renee, Kris Shaffer, Becky Ruppel, David Sullivan, Robert Matney, Ryan Fox, Jonathan Albright, and Ben Johnson. 2019. The tactics & tropes of the Internet Research Agency - the United States Senate Selection Committee on Intelligence.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. Array programming with NumPy. *Nature* 585, no. 7825 (September): 357–362. https://doi.org/10.1038/s41586-020-2649-2. https://doi.org/10.1038/s41586-020-2649-2.

Hunter, J. D. 2007. Matplotlib: a 2d graphics environment. *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Illing, Sean. 2018. "It's pretty much the Wild West": why we can't trust Facebook to police itself - Vox. (Accessed on 08/20/2022).

Jurafsky, Daniel, and James H Martin. 2018. Speech and language processing (draft). *preparation [cited 2022 August 22] Available from: https://web. stanford. edu/~ jurafsky/slp3.*

Kshetri, Nir, and Jeffrey Voas. 2017. The economics of "fake news". *IT Professional* 19 (6): 8–12.

McKinney, Wes, et al. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th python in science conference,* 445:51–56. Austin, TX.

Parkinson, Hannah Jane. 2016. Click and elect: how fake news helped Donald Trump win a real election — The Guardian. (Accessed on 08/18/2022).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Pezoa, Felipe, Juan L Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. 2016. Foundations of json schema. In *Proceedings of the 25th international conference on world wide web,* 263–273. International World Wide Web Conferences Steering Committee.

Raschka, Sebastian, and Vahid Mirjalili. 2019. *Python machine learning: machine learning and deep learning with python, scikit-learn, and tensorflow 2.* Packt Publishing Ltd.

Reis, Julio CS, André Correia, Fabrício Murai, Adriano Veloso, and Fabrício Benevenuto. 2019. Supervised learning for fake news detection. *IEEE Intelligent Systems* 34 (2): 76–81.

Stubbs, Jack. 2020. Duped by Russia, freelancers ensnared in disinformation campaign by promise of easy money — Reuters. (Accessed on 08/21/2022).

Van Rossum, Guido. 2020. *The python library reference, release 3.8.2.* Python Software Foundation.

Vohra, Deepak. 2016. Apache parquet. In *Practical hadoop ecosystem: a definitive guide to hadoop-related frameworks and tools,* 325–335. Berkeley, CA: Apress. ISBN: 978-1-4842-2199-0. https://doi.org/10.1007/978-1-4842-2199-0_8. https://doi.org/10.1007/978-1-4842-2199-0_8.

Waskom, Michael L. 2021. Seaborn: statistical data visualization. *Journal of Open Source Software* 6 (60): 3021. https://doi.org/10.21105/joss.03021. https://doi.org/10.21105/joss.03021.