Article Classification Between Real & Fake News

Ola Sanusi, Andrew Holland, Joseph Tanner, LaTonya Carter,
Danielle Scaramella & Adi Johnson
Georgetown University SCS Data Science Cohort 22*

January 18, 2021

Abstract

In this day and age, it is important that individuals are able to quickly identify real or fake news in order to determine whether or not to believe what they are reading. This research hypothesizes that there is a real, distinguishable difference between the word patterns of fake news when compared with real news. This hypothesis was analyzed using a 1GB data set of new articles that was ingested, cleaned, and manipulated to contain the article title, domain, content, word counts in title, word count in content, average word length in title, average word length in content, affinity score of content and sentiment of the title. The target variable was the label containing either real or fake. The models leveraged were Logistic Regression, Decision Tree Classifier, Linear SVC, Random Forest Classifier, SGDClassifier, Multinomial Naive Bayes, and Bi-Directional LSTM. Ultimately, the SGDClassifier yielded the best results as determined by its 1.0 scores for precision, recall, and F1 in culmination with the total amount of time it took to run the model.

*Code: https://github.com/georgetown-analytics/From-Russia-With-Love-fake-news-

1 Motivation

Over the past several years, there has been a media explosion over real versus fake news and how much trust the public should put in media releases. While robustly combating the spread of fake news will take a whole-of-society approach, machine learning solutions offer a quick way to distinguish between fake news and real news. In a perfect world, the United States would roll out a society wide media literacy campaign similar to the campaigns in Finland, Sweden, Denmark, and the Netherlands. Machine learning models would ideally be able to help the designers of the campaign determine the characteristics that mark a news narrative as fake and convey these characteristics to the populace. This research hypothesizes that there is a real, distinguishable difference between the word patterns of fake news when compared with real news. To test such a hypothesis, machine learning algorithms will be used as tools to assist in detecting, filtering, and classifying the word patterns of both real and fake news articles.

2 Methodology

2.1 Understanding the Problem

In order to begin work on classifying real versus fake news, it was of the greatest importance that the data selected contained article titles, domain/source, and content. With the understanding that an article can be partially fake or true and with consideration of processing time, this research decided to utilize data that was labeled fake or reliable, with exclusion to partial truthfulness. Such a set led to the use of a supervised learning approach.

2.2 Identifying Data Requirements

Several data requirements were considered when identifying the most suitable data resource for the project. At a minimum, the dataset had to have labels designating an article as fake or real to fit a supervised learning machine algorithm. The chosen data set included additional labels beyond real and fake. These included the following: satire, bias, conspiracy, state, junksci, hate, clickbait, unreliable, and political.

Tag	Description
fake	Sources that entirely fabricate information, disseminate deceptive content, or grossly distort actual news reports
satire	Sources that use humor, irony, exaggeration, ridicule, and false information to comment on current events.
bias	Sources that come from a particular point of view and may rely on propaganda, decontextualized information, and opinions distorted as facts.
conspiracy	Sources that are well-known promoters of kooky conspiracy theories.
state	Sources in repressive states operating under government sanction.
junksci	Sources that promote pseudoscience, metaphysics, naturalistic fallacies, and other scientifically dubious claims.
hate	Sources that actively promote racism, misogyny, homophobia, and other forms of discrimination.
clickbait	Sources that provide generally credible content, but use exaggerated, misleading, or questionable headlines, social media descriptions, and/or images.
unreliable	Sources that may be reliable but whose contents require further verification.
political	Sources that provide generally verifiable information in support of certain points of view or political orientations.

Figure 1: Label column descriptions.

However, these labels were not utilized as target variables. In consideration of using article content, the decision was made to exclude tweets and blog posts. This ensured that the requirements to apply NLP methods were met as each article had a title and sufficient content.

2.3 Data Collection

Several data collection methods were discussed throughout the beginning phases of work. One consideration was ingesting articles directly from news source APIs. This method would have enabled the creation of a uniquely compiled dataset, but only could have been successfully employed with an unsupervised machine learning approach. In order to ensure the data was supervised, the final source chosen was a Fake News corpus that included a label to be used as a target variable. This source [1] contains a 27 GB file with over 9.4 million articles. The fake articles were obtained prior to our use by scraping a list of 1001 domains from open source websites while the real articles were pulled from the New York Times and WebHose English News Articles. It contained the features id, domain, type, url, content, scraped at, inserted at, updated at, title, authors, keywords, meta keywords, meta description, tags, summary, and source.

2.4 Preprocessing and Modeling

The methodology in preprocessing and modeling was based heavily on the exploratory data analysis that was performed. Preprocessing considerations included having a sound understanding of what features would have any effect on whether an article was deemed fake or real and what features would lend themselves to feature engineering such as average word count and length for the title and content features. The models that were leveraged included logistic regression, decision tree classification, linear support vector classification, random forest classification, multinomial naive bayes classification, and bi-directional LSTM. In order to evaluate model performance, confusion matrices and classification reports were generated for each model and the length of time for each step was throughout the running process.

3 Data Ingestion and Wrangling

3.1 Ingestion

Referring to the previously discussed source, this research ultimately decided to use this source because it had significant volume and veracity. However, the volume turned out to cause significant challenges in data ingestion. In order to effectively manage the data, the size of the corpus was reduced by selecting two of the eleven label types to include only reliable [real] and fake. Although there were other labels that could allude to being either real or fake, those were removed so as not to bias results with subjectivity. Prior to reducing the size, the data was stored using Amazon Web Services's SagerMaker S3 bucket and explored in chunks using cloud-hosted Jupyter Notebooks through SageMaker. However, once it was reduced, the data was able to be stored and used locally.

3.2 Munging and Wrangling

The subset of the Fake News Corpus was cleaned to remove noises such as special/accented characters, extra whitespaces, newlines, punctuations, digits, stopwords, and words with less than three characters. In addition, contractions were expanded and additional features necessary for training the models were created. The following additional features were extracted from the given features: character counts, word counts, average word length, sentiment, affinity score and part of speech (POS) tags. These features were created to help identify patterns in the dataset that can assist in filtering fake news contents from real news contents. After noise removal, the texts in the title and

content columns were tokenized to separate them into individual words/tokens and finally normalized to their root form. Upon completion of the cleaning and wrangling, that dataset had a total of nine features: title, domain, content, word counts in title, word count in content, average word length in title, average word length in content, affinity score of content and sentiment of the title.

During the data munging and wrangling stages, a few visualization techniques were leveraged to better understand the data. The first use of a visualization technique was plotting a bar chart of the counts for labels fake and real. Visualizing these counts helped shed light on the significant class imbalance in the dataset, which resulted in downsampling the majority class. Another standard visualization technique used was a histogram to show the distribution of characters in the title and content features. Histograms are effective to see the shape of the distributions to quickly determine if the distribution is normal or skewed, as heavily skewed data can have a negative impact on model performance.

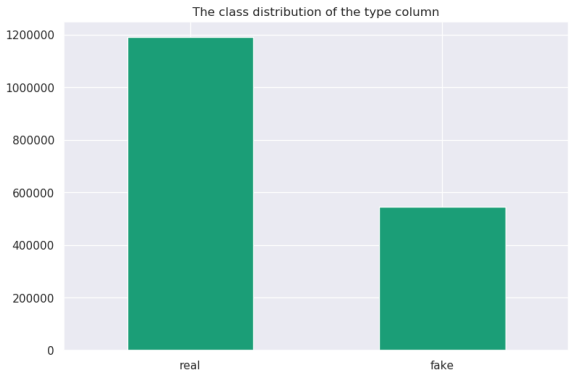

Figure 2: Class distribution of the type column.

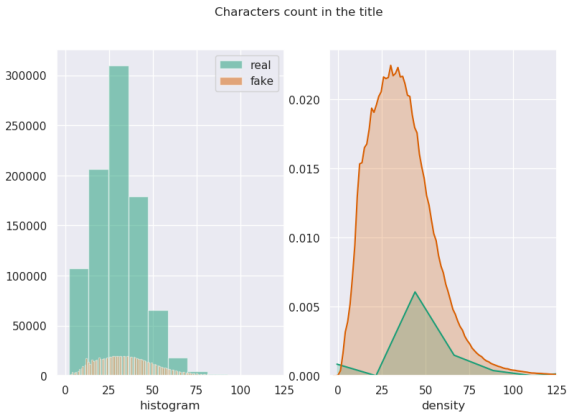

Figure 3: Character counts in the title.

An additional visualization technique that was used during this stage is Word Cloud. Word Cloud was used to visualize the top words in both the title and content. Though aesthetically pleasing, Word Cloud is not very effective to visualize statistical information. It is easy to discern the top word, but it is difficult to truly understand the placement of the words after the top value.

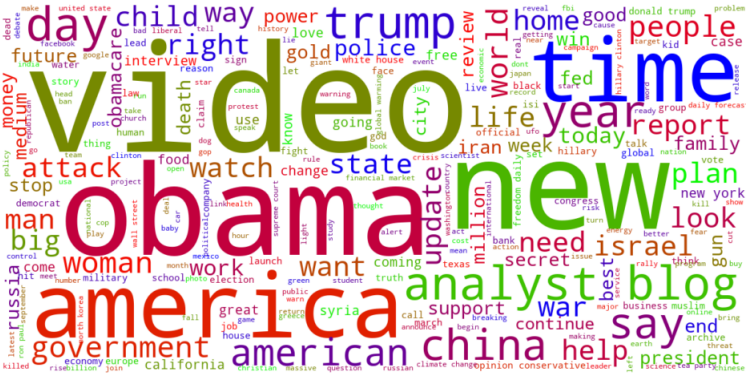

Figure 4: Word Cloud to discern top works for fake news.

4 Computation

Since the data contains heterogeneous data types, there are numerical, categorical, and text data. In order to use each type to its fullest extent, different vectorization methods were utilized to transform the data in those various columns. Specifically, One-hot encoding was used for categorical, robustscaler for numeric, and tfidfvectorizer for the text features. TfidfTransformer was used to reduce any impacts of frequently occurring tokens in a corpus that are less informative than features that occur in a small fraction of the corpus but may be more relevant [2]. Columntransformer was utilized to combine all of those transformations. In each deployed model, the target variable was label (real versus fake). The models leveraged were Logistic Regression, Decision Tree Classifier, Linear SVC, Random Forest Classifier, SGDClassifier, and Multinomial Naive Bayes. The split used for training and testing was 80:20.

4.1 Logistic Regression

The first model utilized was the scikit-learn Logistic Regression model. Seeing that this is a classic classification problem, it made sense to start here. There were two clear discrete classes, ultimately making this a binary logistic regression. To visualize model performance, a confusion matrix and classification report were used. The confusion matrix below for the Logistic Regression model shows the number of True Positive, False Positive, False Negative and True Negative from the predicted and true class. For this model, one can see there were 178,011 TP, 1 FP, 0 FN, and 178,321 TN. The classification report shows that the model had a precision, recall, and F1 scores of 1.

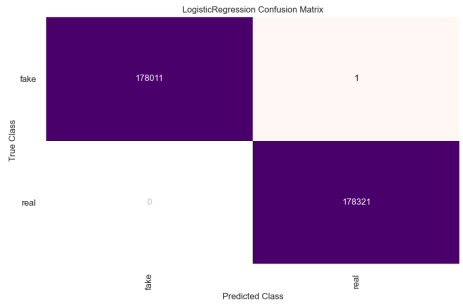

Figure 5: Logistic regression confusion matrix.

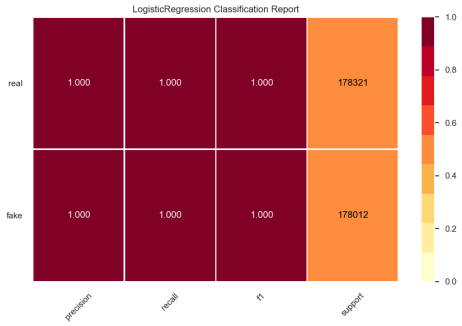

Figure 6: Logistic regression classification report.

4.2 Decision Tree Classification

The second model leveraged was a Decision Tree Classifier. This model was selected because the decision trees are known for "their robustness to noise, tolerance against missing information, handling of irrelevant, redundant predictive attribute values," [6] and more. When selecting the hyperparameters for the model it was decided to utilize "Gini Impurity" over "entropy". Both measure the quality of the data split; however, entropy is thought of as how unpredictable a dataset is while Gini impurity is the probability of incorrectly classifying a randomly chosen element [4]. Since the hypothesis centers around accurately classifying articles, and the goal was to call out wrong classifications over unpredictability, the decision was made to prioritize that measure in parameter selection. In regards to the max depth hyperparameter, it was initially set at 20. Upon, further investigation it was determined that it would be best to utilize the default setting of "None." This decision was made with consideration that, "the deeper you allow your tree to grow, the more complex your model will become because you will have more splits and it captures more information about the data and this is one of the root causes of overfitting in decision trees because your model will fit perfectly for the training data and will not be able to generalize well on test set" [6]. The confusion matrix below for the Decision Tree model shows 178,008 True Positive, 4 False Positive, 1 False Negative and 178,320 True Negative. The classification report shows that the model had a precision, recall, and F1 scores of 1.

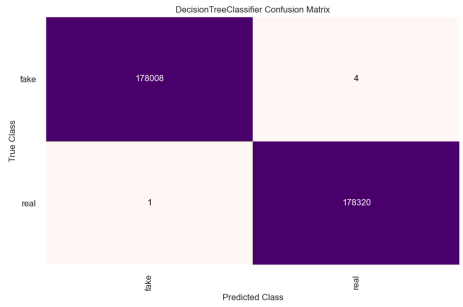

Figure 7: Decision tree classifier confusion matrix.

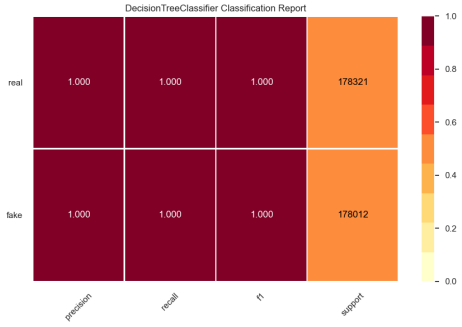

Figure 8: Decision tree classifier classification report.

4.3 Linear SVC

Linear SVC models are aimed at finding a clear hyperplane between data of two classes, making it a clear contender to investigate this classification problem [7]. While the default setting for dual in the LinearSVC model is True, the False parameter is often used. Reason being, False allows for a setting where the number of samples is greater than the number of features, which is applicable with this dataset [**?**]. In terms of selecting the regularization parameter C, it was decided to select a low value in order to avoid overfitting the training data, which can be a result of increasing C, the penalty parameter of the error term [9]. The confusion matrix below for the Linear SVC model shows 178,012 True Positive, 0 False Positive, 0 False Negative and 178,321 True Negative. The classification report shows that the model had a precision, recall, and F1 scores of 1.

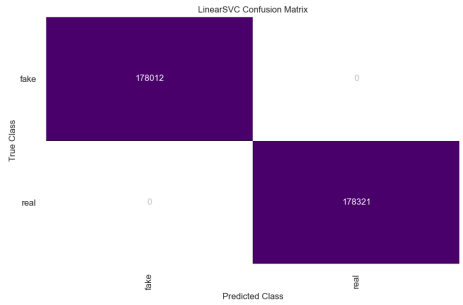

Figure 9: Linear SVC classifier confusion matrix.

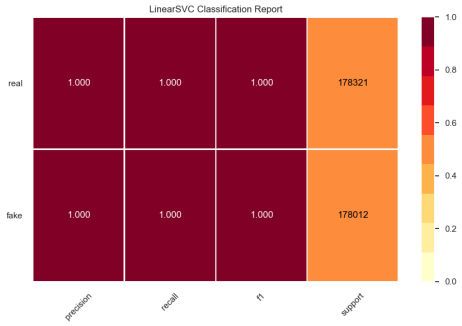

Figure 10: Linear SVC classification report.

4.4 Random Forest Classifier

Random Forest Classifier, due to its increased number of trees, reduces the probability of over-fitting in the model. It is used frequently in classification tasks and provides higher accuracy through cross validation. The confusion matrix below for the Random Forest model shows 177,513 True Positive, 499 False Positive, 361 False Negative and 177,940 True Negative. The classification report shows that the model had a precision of .997, recall of .998, and F1 of .998 for real and a precision of .998, recall of .997, and F1 of .998 for fake.

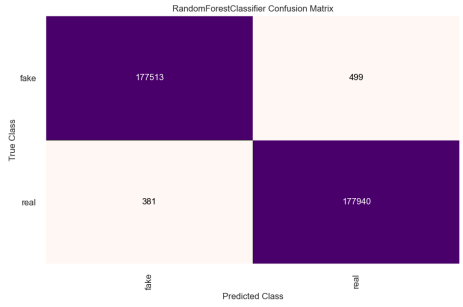

Figure 11: Random forest classifier confusion matrix.

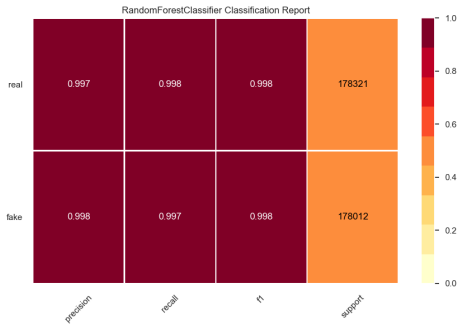

Figure 12: Random forest classification report.

4.5 SGDClassifier

Stochastic Gradient Descent Classifier has been successfully applied to machine learning problems often encountered in text classification and natural language processing. SGD Classifier is a simple yet very efficient approach to fitting linear classifiers and regressors under convex loss functions such as (linear) Support Vector Machines and Logistic Regression [10]. For hyperparameter tuning, the max iter was increased slightly from the default of 1000 to 1500, allowing it more passes over the training data.

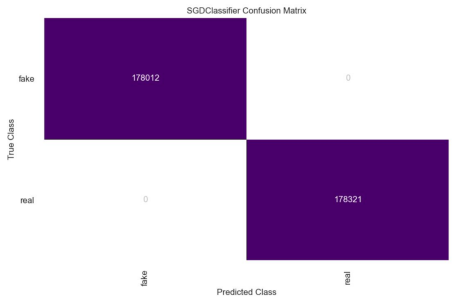

Figure 13: SGD confusion matrix.

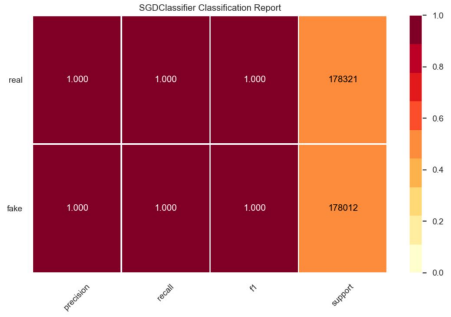

Figure 14: SGD classification report.

4.6 Multinomial Naive Bayes

Multinomial Naive Bayes implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification [11]. There are other alternatives when coping with NLP problems, such as Support Vector Machine (SVM) and neural networks. However, the simple design of Naive Bayes classifiers make them very attractive for such classifiers. Moreover, they have been demonstrated to be fast, reliable and accurate in a number of applications of NLP [12]. No hyperparameter tuning was done for this modal and the default parameters were used.

4.7 Bi-Directional LSTM

In addition to the previously listed models, a Recurrent Neural Network was also implemented since bidirectional LSTMs have shown to be quite effective for text classification. To develop this model, a guided project on Coursera was used as a template. While RNNs have been overtaken by BERT as state of the art for text classification, the model used still performed well as it achieved an F1 score of .95.

In this RNN, keras and tensorflow were used. Because of this, all of the work on the RNN was in Amazon's Sagemaker cloud environment. Originally, all 1.8 million news articles were used for training and testing but some issues arose keeping Sagemaker running for the entirety of the training time. So instead, the data was down sampled using pandas with approximately 800,000 articles. In a perfect world, one would be able to use all 1.8 million articles and train the model in a local environment, but the downsampling was necessary to at least attempt a based RNN. Additionally, only one epoch was run with a batch size of 64 due to the training time. Optimally, one would have been able to decrease the batch size slightly and run far more epochs to increase model performance in addition to trying different combinations of number of hidden layers and number of nodes per layer.

5 Discussion of Results

5.1 Results

The overarching goal of this research was to determine if using machine learning techniques could accurately predict if an article was real or fake. To accomplish this, extensive wrangling and comprehensive exploratory data analysis was used to obtain a deep understanding of the dataset. All models employed performed exceptionally well, even to the point where a few of models did not find any false positives or false negatives. This was attributed to the fact that the domain and content features could have led to overfitting. It was hypothesized that any article with New York Times in the content may have been labeled true automatically, falsely skewing the performance. In reviewing the time spent training and running each model in culmination of its overall results, the most successful model was determined to be Linear SVC. To recap, this model received precision, recall, and F1 scores of 1 and correctly classified every article. The total runtime for this model was 27.5 seconds compared with significantly longer run times for the execution of other models. All of this led to the conclusion that one could in fact find clear patterns that emerge that could help differentiate real and fake news.

5.2 Lessons Learned and Next Steps

Throughout the process of classifying real versus fake articles, a number of additional considerations arose. First, was the computation cost. At each step, the sheer volume of the data created hours worth of runtime at step. Specifically, the content and title features took a significant amount of time to train models due to the length of the fields. Additionally, memory issues caused some local machines to have Jupyter Notebook kernel failures. While the volume of data was great for determining correct classification, the computation cost was incredibly high. A second consideration was the possibility of overfitting. Given the fact that the articles that were used as real came primarily from the New York Times, there is a possibility that the model began using sources alone as means to classify. Efforts regarding feature importance could have been explored to understand the baseline feature utilized for model inputs in order to increase understanding of how overall model performance was affected. Finally, the subjectivity of real and fake classification has become increasingly high. There will always be an inherent bias in using labeled data. Therefore, future work should consider unsupervised learning. Especially so given that inspection of unlabeled data in conjunction with the most mentioned topics for content and title features as engineered features prove to be advantageous for overall model performance.

Although the results are exceptional for this research, there are a number of steps that could further its impact. These include adding more media types into the analysis such as blogs, videos, press conferences, tweets, etc. Another addition would be to create a user interface or application that allows the input of an article and returns an output of the real and fake percentage of it. This would expand the usage to correctly classifying articles that are partially real or partially fake. While leveraging machine learning to aid in the analysis of the truthiness of news sources is a step in the right direction, it will still take time to uncover how best to build them in a fair and ethical way to fully remove bias. Exploring NLP even further, the research could be pushed into the realm of language agnostic models to look at fake news in languages other than English.

References

[1] Szpakowski, Maciej.*Fake News Corpus.* GitHub, 24 January 2020. https://github.com/several27/FakeNewsCorpus

[2] *Scikit-learn: Machine Learning in Python*, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

[3] *Scikit-learn: Machine Learning in Python*, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011, https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

[4] Zhou, Victor. *A Simple Explanation of Gini Impurity.* 29 March 2019. https://victorzhou.com/blog/gini-impurity/,https://victorzhou.com/blog/information-gain/

[5] Pant, Ayush. *Introduction to Logistic Regression.* 22 January 2019. https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

[6] Mithrakumar, Mukesh. *How to tune a Decision Tree.* 11 November 2019. https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680

[7] Pupale, Rushikesh. *Support Vector Machines (SVM) - An Overview.* 16 June 2018. https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989#:~:text=SVM%20or%20Support%20Vector%20Machine,separates%20the%20data%20into%20classes.

[8] Yang, Limin. *When to use dual=False in LinearSVC?.* 2020. GitHub Repository, https://github.com/scikit-learn/scikit-learn/issues/17339

[9] Fraj, Mohtadi Ben. *In Depth: Parameter tuning for SVC.* 5 January 2018. https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

[10] *Scikit-learn: Machine Learning in Python*, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/modules/sgd.html#sgd

[11] *Scikit-learn: Machine Learning in Python*, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes

[12] Huang, Olli. *Applying Multinomial Naive Bayes to NLP Problems: A Practical Explanation.* 17 July 2017. https://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf