Fake News Detection

Ritika Nair, Shubham Rastogi, Tridiv Nandi
Northeastern University

Abstract

In our modern era where internet is ubiquitous, everyone relies on various online resources for news. Along with the increase in use of social media platforms like Facebook, Twitter etc. news spread rapidly among millions of users within a very short span of time. The spread of fake news has far reaching consequences like creation of biased opinions to swaying election outcomes for the benefit of certain candidates. Moreover, spammers use appealing news headlines to generate revenue using advertisements via click-baits. In this project, we aim to perform a binary classification of various news articles available online with the help of concepts pertaining to Artificial Intelligence, Natural Language Processing and Machine Learning.

1 Introduction

With the growing popularity of mobile technology and social media, information is accessible at one's fingertips. Mobile applications and social media platforms have overthrown traditional print media in the dissemination of news and information. It is only natural that with the convenience and speed that digital media offers, people express preference towards using it for their daily information needs. Not only has it empowered consumers with faster access to diverse data, it has also provided profit seeking parties with a strong platform to capture a wider audience.

With the outburst of information, it is seemingly tedious for a layman to distinguish whether the news he consumes is real or fake. Fake news is typically published with an intent to mislead or create bias to acquire political or financial gains. Hence it may tend to have luring headlines or interesting content to increase viewership.

In the recent elections of United States, there has been much debate regarding the authenticity of various news reports favoring certain candidates and the political motives behind them. Amidst such growing concerns, the detection of fake news gains utmost importance to prevent its negative impacts on individuals and society.

The most common algorithms used by fake news detection systems include machine learning algorithms such as Support Vector Machines, Random Forests, Decision trees, Stochastic Gradient Descent, Logistic Regression and so on. In this project we have attempted to implement two out of these algorithms to train and test our results. We have used a combination of both off the shelf datasets as well as expanded it by crawling content on the web. The main challenge throughout the project has been to build a set of uniform clean data and to tune parameters of our algorithms to attain the maximum accuracy.

We observed that the Random Forests algorithm with a simple term frequency-inverse document frequency vector performed the best out of the four algorithms we tested. In section 2 we describe the data collection, building of the dataset and the text preprocessing techniques used. In section 3, the three different models and their algorithms are discussed. Section 4 discusses evaluation of results and the scope for future enhancements.

2 Data Collection

The dataset for this project was built with a mix of both real and fake news. Most of the data was manually crawled and extracted, whereas some were used off the shelf. The entire dataset amounted to 125,600 news articles out of which 15,600 were fake news and 1,10,000 were real news.

To efficiently collect such huge data, we created a multi-threaded web crawler. We ran the crawler using up to 100 threads at a time and download the raw HTML body content of the crawled pages.

The sources of real news include Yahoo News, AOL, Reuters, Bloomberg and The Guardian among many. Sources for fake news include TheOnion. UsaNewsFlash, Truth-Out, The Controversial Files and so on.

To extract important content from the crawled pages we used two strategies. First was to reduce noise by removing

insignificant and irrelevant information like images, tables, headers, footers, special symbols, navigation bars etc. The second strategy was to extract HTML div tags from the remaining content having the id property as 'content' or some variations of 'content'. With this we noticed we were able to extract most of the important information across many webpages. Since each website has its own style of layout and parameters, a one size fit all strategy would have failed, and hence we leveraged a generic approach.

The collected data was processed using various text preprocessing measures, as explained later and stored in CSV files. The real and fake data were then merged and shuffled to get a CSV file containing a consolidated randomized dataset.

From the consolidated randomized dataset we picked 20845 records at random which contained approximate 50% real news and 50% fake news articles. From these records, 80% was used for training the detection model and 20% was reserved for testing the model.

2.1 Real News

The News Aggregator Dataset from the UCI Machine Learning Repository was used to extract real news. This dataset consists of links to the originally published news articles in their websites. We extracted these URLS and crawled them to download the news content using BeautifulSoup.

We extracted the body content of the articles by removing unnecessary information such as headers, footers, images, advertisements, tables etc. Further, we extracted the text from div tags having content and performed preprocessing steps on them before saving them into CSV files.

2.2 Fake News

For fake news we used Kaggle's 'Getting Real about Fake News' dataset. The CSV file with data was available off the shelf for use, and we had to perform minimal text processing on this data.

2.3 Text Preprocessing

Since most of the data was crawled and extracted manually, we had to first go through the data to understand organization and formatting of text. The data was made uniform and comparable by converting it into a uniform UTF-8 encoding. There were some cases where we encountered weird symbols and letters incompatible with the character set which had to be removed. We noticed that the data from news articles were often organized into paragraphs. So, we performed trimming to get rid of extra spaces and empty lines in text.

4. Feature Generation

For generation of features from the given data, we first performed tokenization on the raw text of articles. We then generated tf-idf feature vectors as described below.

4.1 Term Frequency - Inverse Document Frequency

The tf-idf is a statistical measure that reflects the importance of a particular word with respect to a document in a corpus. It is often used in information retrieval and text mining as one of the components for scoring documents and performing searches. It is a weighted measure of how often a word occurs in a document relative to how often it occurs across all documents in the corpus. Term frequency is the number of times a term occurs in a document. Inverse document frequency is the inverse function of the number of documents in which it occurs.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Figure 1. TF-IDF formula for weight of term i in document j. (Source: researchgate.net)

Hence a term like "the" that is common across a collection will have lesser tf-idf values, as its weight is diminished by the idf component. Hence the weight computed by tf-idf represents the importance of a term inside a document.

The tokenized data was used to generate a sparse matrix of tf-idf features for representation. This represented our feature vector and was used in subsequent prediction algorithms.

3. Prediction Algorithms

We implemented two different algorithms from scratch for the prediction model which were: Logistic Regression model and the Naïve Bayes classifier model. The algorithms and the details of implementation have been explained in the sections below. In addition to these we also trained and tested our dataset on two other models: Random Forests model and Support Vector Machine model. Given the short time frame of the project, the last two algorithms were prudently implemented with the help of scikit-learn libraries.

3.1 Logistic Regression

Logistic Regression is a Machine Learning technique used to estimate relationships among variables using statistical methods. This algorithm is great for binary classification problems as it deals with predicting probabilities of classes, and hence our decision to choose this algorithm as our baseline run. It relies on fitting the probability of true scenarios to the proportion of actual true scenarios observed. Also, this algorithm does not require large sample sizes to start giving fairly good results.

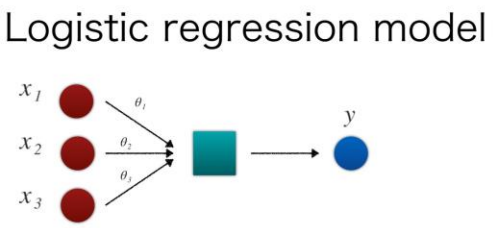


Figure 2. Logistic Regression Model (Source: towardsdata-science.com)

Algorithm 1 Logistic Regression with L1 regularization

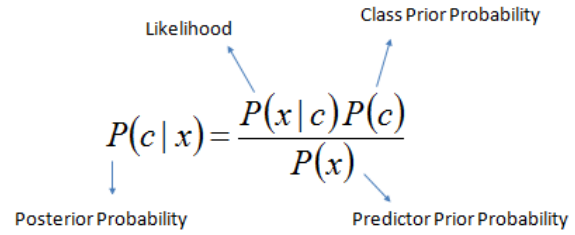
procedure StochasticGradientDescent(D, $Labels$, $Iter$)
Input: $Dataset\ D$, $Labels\ of\ Dataset$, $Iteration\ num$
Output: optimal weight of logistic regression
  $w \leftarrow [1, 1, ..., 1]$
  Initialize q_i with zero for all i
  for $k = 1 \rightarrow Iter$ **do**
    $chooseData = D$
    for $i = 1 \rightarrow m$ **do**
      $\gamma \leftarrow Learning\ Rate$
      $\lambda \leftarrow Regularization\ Lambda$
      $u = u + \gamma\lambda$
      Select a index of chooseData idx randomly
      $x \leftarrow chooseData[idx]$
      $del\ chooseData[idx]$
      for $i \in featuresinsamplex$ **do**
        $w_i = w_i - \gamma \frac{\partial loss(w,x_i)}{\partial w}$
        $wh \leftarrow w$
        if $w_i > 0$ **then**
          $w_i \leftarrow \max(0, w_i - (u + q_i))$
        else if $w_i < 0$ **then**
          $w_i \leftarrow \min(0, w_i + (u - q_i))$
        end if
        $q_i \leftarrow q_i + (w_i - wh)$
      end for
    end for
  end for

Figure 3. Logistic Regression Pseudo-code(Source:whatbeg.com)

The Logistic Regression algorithm works by assigning observations to a discrete set of classes and then transforms it using a sigmoid function to give the probability value which can be mapped to a discrete class.

3.2 Naïve Bayes Classifier

This is a simple yet powerful classification model that works remarkably well. It uses probabilities of the elements belonging to each class to form a prediction. The underlying assumption in the Naïve Bayes model is that the probabilities of an attribute belonging to a class is independent of the other attributes of that class. Hence the name 'Naive'.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 4. Naïve Bayes formula (Source: techleer.com)

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count\ (w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

Figure 5. Naïve Bayes Pseudocode (Source: web.stanford.edu)

In this model we multiply the conditional probabilities of each attribute given the class value, to get the probability of the test data belonging to that class. We arrive at the final prediction by selecting the class that has the highest of the probabilities for the instance belonging to that class.

The advantages of using Naïve Bayes is that it is simple to compute, and it works well in categorizing data as we are using ratios for computation. The formula used for this model is as follows:

3.3 Random Forest Classifier

Random Forests are a machine learning method of classification that work by building several decision trees while training the model. It is a kind of additive model that makes predictions from a combination of decisions from base models. Decision trees have huge depth and tend to overfit results. Random forest utilizes multiple decision trees to average out the results.

The Random forest classifier creates a set of decision trees from a subset of the training data. It aggregates the results from different decision trees and then decides the final classification of the test data. The subsets of data used in the decision trees may overlap.

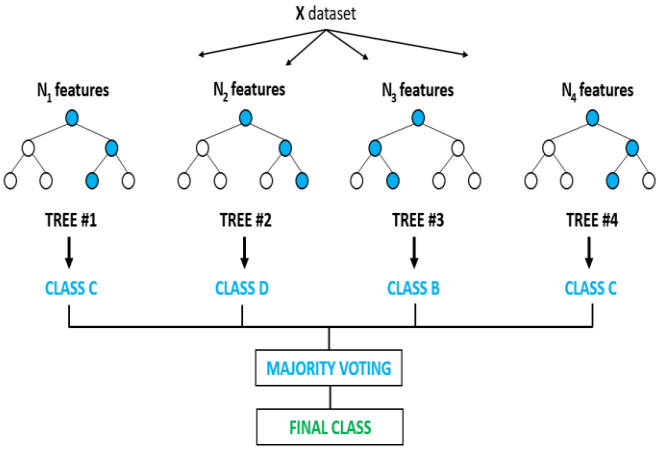


Figure 6. Random Forest Model (Source: globalsupportsoftware.com)

3.4 Support Vector Machine

Support Vector Machines are machine learning models that perform supervised learning on data for classification and regression. When given a labeled training dataset, it computes the optimal hyperplane that categorizes the test data.

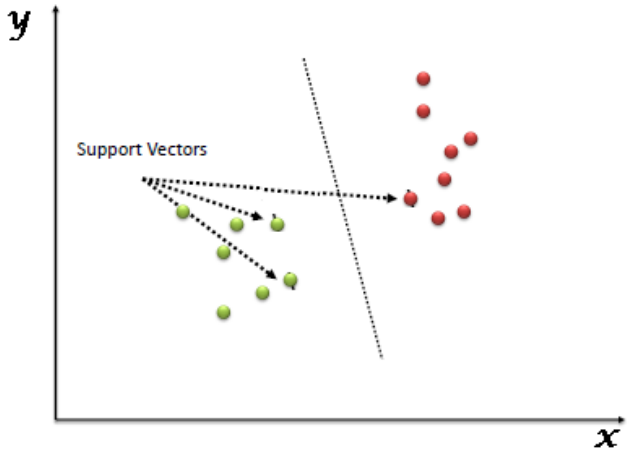


Figure 7. Support Vector Machine (Source: analyticsvidhya.com)

Data points are plotted in a multidimensional space, where the dimension is determined by the number of features at our disposal. The value of each feature is mapped to a point in the coordinate system. The algorithm then performs classification by finding the hyperplane that differentiates the two classes well. The hyperplane having the maximum margin between the two classes in chosen.

The advantages of the SVM model are that it performs very well for high dimensional spaces and also creates a clear margin of separation between data points. The disadvantages of using SVM were that it takes greater time to train the model compared to other models, especially when the dataset is large.

4 Evaluation Metrics

We used the following three metrics for the evaluation of our results. The use of more than one matrix helped us evaluate the performance of the models from different perspectives.

4.1 Classification Accuracy

This depicts the number of accurate predictions made out of the total number of predictions made.

Classification accuracy is calculated by dividing the total number of correct result by the total number of test data records and multiplying by 100 to get the percentage.

4.2 Confusion Matrix

This is a great visual way to depict the predictions as four categories:

1. False Positive: Predicted as fake news but are actually true news.
2. False Negative: Predicted as true news but are actually fake news.
3. True Positive: Predicted as fake news and are actually fake news.
4. True Negative: Predicted as true news and are actually true news.

4.1 Precision and Recall

Precision which is also known as the positive predictive value is the ratio of relevant instances to the retrieved instances.

Precision = No. of True Positives / (No. of True Positives + No. of False Positives)

Recall which is also known as sensitivity is the proportion of relevant instances retrieved among the total number of relevant instances.

Recall = No. of True Positives / (No. of True Positives + No. of False Negatives)

5 Optimization

5.1 Feature Selection and Extraction

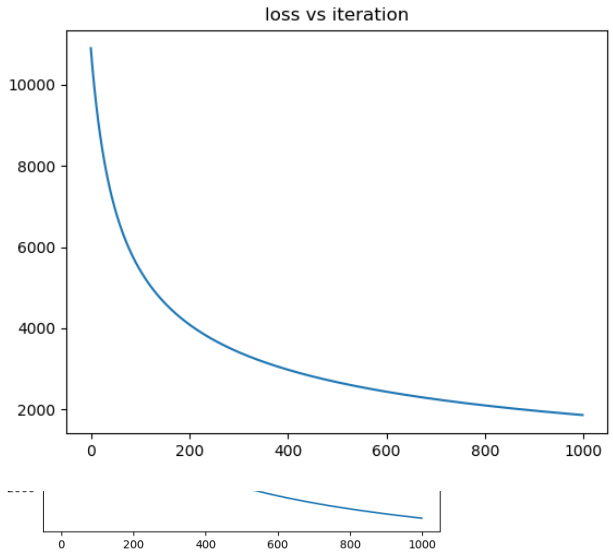
Feature selection was the major part of our text-based classification problem, we used tf-idf vectorization of the news data we collected. But there was a challenge as the dimensions of the vector was quite high which caused models like SVM and Logistic Regression to run for a very long time on large datasets. To resolve the issue we used some text based transformation techniques such as stopping. We passed a list of stop words generated using NLTK library, as a parameter to the sklearn TFIDF vectorization class. We also defined the max_feature parameter to be assigned to 50000, i.e. the TFIDF class generates a vocabulary and considers only the top max_features ordered by term frequency across the corpus. This was only possible after stop word removal as this could have caused an issue as stop words are the most frequent words in the document.

5.2 Naïve Bayes

For Naïve Bayes model, smoothing parameter was added as well as log probabilities were taken for accurate calculations.

5.3 Logistic Regression

In the implementation of Logistic Regression, we collected the loss in each iteration in an array. Then we plotted the curve of the loss in each iteration vs the number of iterations. The graph is given below:



5.4 Support Vector Machine

In SVM, we have used GridSearch from sklearn to find the optimized parameters that gives us the most accurate predictions. GridSearch produces better results but this also slows down the training process.
The parameters used are as follows :
parameters={'C' :[0.00001, 0.0001, 0.001, 0.01, 0.1, 1,10],'kernel':['linear'], 'random_state': [1]}

5.5 Random Forest

For Random Forest Model no tuning was required.

6 Observed Results

The results observed in terms of evaluation metrics are as follows:

METRICS	MODELS			
	Naïve Bayes	Logistic Regression	SVM	Random Forest
Accuracy	57%	98 %	97%	97%
Precision	0.59	0.98	0.98	0.98
Recall	0.70	0.98	0.97	0.96

N.B: The above results are for evaluating the models on data set of size 20,000. (model trained on 80% and tested on 20% of the data)

The confusion matrix data for each model run on data set size 20,000 split into 80% Training and 20% Test data are as follows:

METRICS	MODELS			
	Naïve Bayes	Logistic Regression	SVM	Random Forest
True Positive	1538	2155	2125	2114
True Negative	756	1750	1754	1756
False Positive	1048	54	45	48
False Negative	658	41	76	82
Total Test Data	4000	4000	4000	4000

Scatter Plots for various Models on a small set of data i.e 1000 are shown below. In each figure the first subplot shows the prediction value plotted in red, whereas the second subplot shows the actual values plotted in green.

Naïve Bayes:

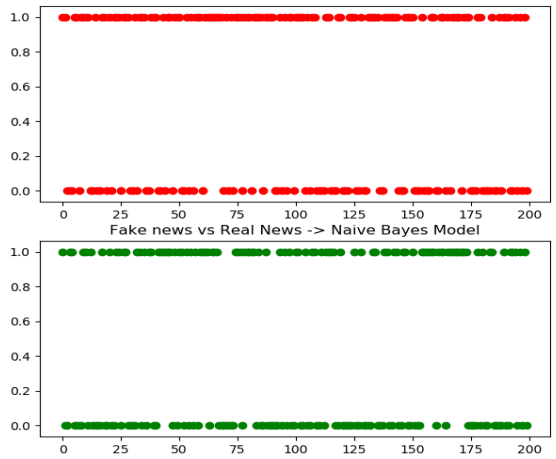


Figure 9. Scatter plot for Naïve Bayes Classifier

Logistic Regression :

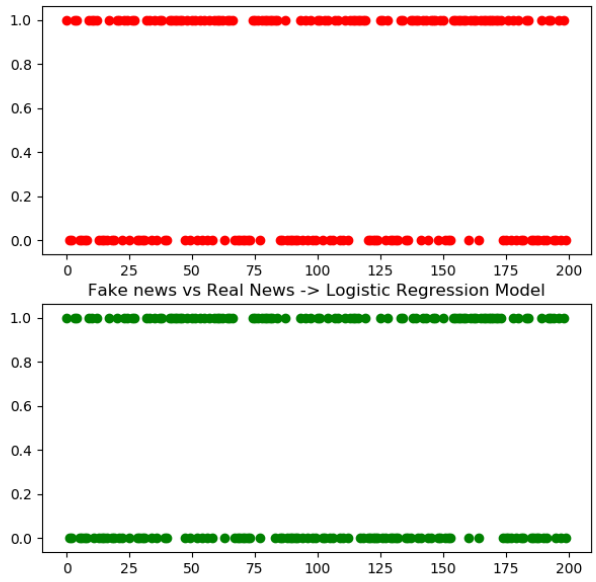


Figure 10. Scatter plot for Logistic Regression model

Support Vector Machine:

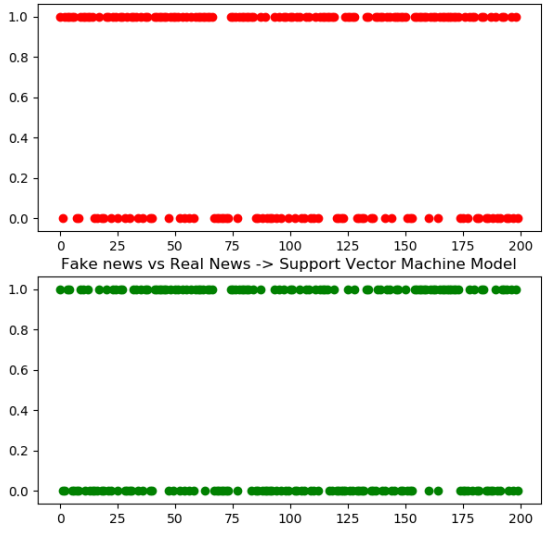


Figure 11. Scatter plot for Support Vector Machine

Random Forest:

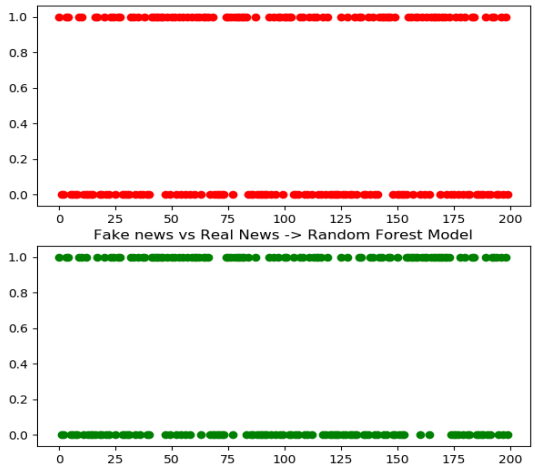


Figure 12. Scatter plot for Random Forest Classifier

7 Conclusion

Naïve Bayes performed very poorly in Bag of words model. Logistic Regression with tf-idf vector as a feature improved this result substantially.

Naïve Bayes performed very poorly. Logistic Regression surprisingly performed very well, as observed from the above results Logistic regression performed slightly better than Support vector machine and random forest models.

The result can further be improved if the n-grams are used to generate tf-idf vectors and then used as a feature.

8 Contributions

The contributions of individual group members were as follows:

Ritika was responsible for extraction of the fake news dataset, preprocessing the text and the collection and randomization of the cleaned data into CSV files. She also implemented the training of data using Random Forest classifier and the Support Vector Machine classifier using sklearn library and tested the results for various input sizes using the evaluation measures described.

Shubham was responsible was extraction of real news data by creating a multithreaded web crawler and analyzing and pre-processing the text. He was responsible for collection of this data into CSV files and merging with the real news data to form a consolidated dataset and cleaning up the data into usable formats. He was also responsible for the implementation of the Naïve Bayes classifier from scratch and the evaluation and visualization of results for the same.

Tridiv was responsible for analysis and cleaning of the consolidated dataset, randomizing it and segregating it into training and testing data. He was also responsible for the implementation of the Logistic Regression model from scratch. He trained and tested the data using this model, evaluated the results using the described evaluation measures and graphically plotted the results.

All three members were involved in the topic selection, analysis, design and algorithm selection and documentation phases.

References

https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

www.analyticsvidhya.com

[K Gunasekaran et al, 2016] Fake News Detection in Social Media. Vol 1, No. 1 Article 1, January 2016.

[Victoria L Rubin, Yimin Chen, and Niall J Conroy] Deception detection for news: three types of fakes. Proceedings of the Association for Information Science and Technology, 52(1):1–4, 2015.

[Niall J Conroy, Victoria L Rubin, and Yimin Chen] Automatic deception detection: methods for finding fake news. Proceedings of the Association for Information Science and Technology, 52(1):1–4, 2015.

En.wikipedia.org. Support vector machine. https://en.wikipedia.org/wiki/Support_vector_machine

Stat.berkeley.edu. Random forests - classification description. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_ home.htm.

[Nebel, 2000] Bernhard Nebel. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315, 2000.

[Daniel Jurafsky, Jones H. Martin] Naïve Bayes and Sentiment Classification, Speech and Language Processing, Chapter 6. https://web.stanford.edu/~jurafsky/slp3/6.pdf

http://scikit-learn.org/stable/

http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

http://scikit-learn.org/stable/modules/svm.html

Lecture Notes (Logistic Regression) of CS 6140: Machine Learning, CCIS, Northeastern University by Prof. Bilal Ahmed & Prof. Yu Wen

Lecture Notes (Logistic Regression) of CS 5100: Fundamentals of Artificial Intelligence, CCIS, Northeastern University by Prof. David Smith.

Lecture Notes (Naïve Bayes) of CS 5100: Fundamentals of Artificial Intelligence, CCIS, Northeastern University by Prof. David Smith.

Datasets:

https://www.kaggle.com/mrisdal/fake-news

[Dua, D. and Karra Taniskidou, E. (2017).] UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

NLTK :

[Bird, Steven, Edward Loper and Ewan Klein (2009).]

Natural Language Processing with Python. O'Reilly Media Inc.