Fake News Analysis

Jerome Vincent Tagaro

2024-12-04

Introduction

The dataset was obtained at: https://www.kaggle.com/datasets/aadyasingh55/fake-news-classification

The dataset provides a list of 40,000+ news articles, with the corresponding title and text content. The dataset was designed for studying Fake news detection.

This project uses the aforementioned dataset to build a machine learning model that can detect Fake news based on the title and text content.

The modeling approach uses Sentiment Analysis in Natural Language Processing (NLP) to interpret the data. From Dela Cruz (2023), NLP is explained as:

> Natural Language Processing (NLP) models are a branch of artificial intelligence that enables computers to understand, interpret, and generate human language.

And Sentiment Analysis as:

> Sentiment analysis, also known as opinion mining, is a technique used in natural language processing (NLP) to identify and extract sentiments or opinions expressed in text data

By using Sentiment Analysis, a set of *sentiments* can be obtained, which can then be used to train a model for detecting Fake News. The model for detecting Fake News will be a binary classification model, which means that the model assigns either a *positive* or a *negative* class to an observation based on its set of predictors. The model used for training will be chosen from a list of models depending on the resulting model *Accuracy*, *Specificity* and *Sensitivity*. After the chosen model is optimized, the model will be retrained then evaluated.

Data set

The retrieved data is already split into 3 different files :
- *train (2).csv*
- *test (1).csv*
- *evaluation.csv*

As seen from the names of the files, the data set is already split into a *train set*, *test set*, and *evaluation set*. The *train set* is meant to be used for analyzing, building, and training the model. The *test set* is meant to be used for testing the built model, and for further training of the model. The *evaluation set* is used for evaluating the final model's performance.

The raw data looks like this:

Table 1: Train Set

...1	title	text	label
0	Palestinians switch off Christ ...	RAMALLAH, West Bank (Reuters) - Palestinians switched off Christmas lights at Je ...	1
1	China says Trump call with Tai ...	BEIJING (Reuters) - U.S. President-elect Donald Trump's call with Taiwan Preside ...	1
2	FAIL! The Trump Organization's ...	While the controversy over Trump s personal tax returns continues, business cred ...	0

Table 2: Test Set

...1	title	text	label
0	Live from New York, it's a Tru ...	NEW YORK (Reuters) - Veteran actor and frequent host Alec Baldwin returned to "S ...	1
1	Catalan separatists to lose ma ...	BARCELONA (Reuters) - Catalonia s independence movement could suffer a serious s ...	1
2	North Carolina governor conced ...	WINSTON-SALEM, N.C. (Reuters) - North Carolina Governor Pat McCrory on Monday co ...	1

Table 3: Evaluation Set

...1	title	text	label
0	Sanders back in U.S. Senate, b ...	WASHINGTON (Reuters) - Democratic U.S. presidential hopeful Bernie Sanders broug ...	1
1	Kremlin: Syria peoples' congre ...	MOSCOW (Reuters) - A proposal to convene a congress of all Syria s ethnic groups ...	1
2	Oregon Cop Convicted Of Shatte ...	In a baffling fit of rage, an Oregon State Police officer has been convicted of ...	0

With both title and text showing only the first few characters.

The three tables has these properties:
Train Set :
- Columns *: . . . 1, title, text, label.*
- Rows *: 24353.*
Test Set :
- Columns *: . . . 1, title, text, label.*
- Rows : 8117
Evaluation Set :
- Columns *: . . . 1, title, text, label.*
- Rows *: 8117.*

All the tables has the same columns:
- *. . . 1* : appears to be an index/row variable
- *title* : the title of the news article
- *text* : the full text content of the news article
- *label* : either *1* if it is deemed to be *Fake News*, *0* otherwise.

Methodology

Fix Tables

The tables appear to be immediately usable except for the first column *. . . 1.* The column name may make introduce unnecessary errors the code. Thus, the column name will be changed into *index* as it appears to serve the same function.

The *train set* table now looks like this:

Table 4: Fixed Train Set

index	title	text	label
0	Palestinians switch off Christ . . .	RAMALLAH, West Bank (Reuters) - Palestinians switched off Christmas lights at Je . . .	1
1	China says Trump call with Tai . . .	BEIJING (Reuters) - U.S. President-elect Donald Trump's call with Taiwan Preside . . .	1
2	FAIL! The Trump Organization's . . .	While the controversy over Trump s personal tax returns continues, business cred . . .	0

Find Predictors

Word Tokens

The raw *title* and *text* data cannot be used for any useful machine learning model. It is necessary to extract *predictors* that can be used to train a model. These *predictors* can be obtained using NLP.

To demonstrate the extraction of useful predictors, one sample row will be used. In the following procedures, the row 7 of the *train set* will be used. The sample row has the following content:

Table 5: Index and label of 1 row

index	label
6	0

Table 6: Title of 1 row

title
WATCH: John Oliver Presents GOP Debates As 'Clowntown F*ck-the-World Sh*tshow 2016'

Table 7: Text of 1 row

text
John Oliver isn t known for mincing words when it comes to his description of Republicans. Last night was no exception as he left no hold barred when he discussed the insanity of the Republican candidates during the debates.His primary focus was on the man that has an uncanny tendency for outdoing himself for his insane behavior: Donald Trump. This time, however, he had a partner in Marco Rubio. After showing a clip of Rubio s size-of-hands-equals-penis joke and Trump s response, Oliver said: That s right, Donald Trump just talked about his dick during a presidential debate. A dick which I presume looks like a Cheeto with the cheese dust rubbed off! Building on the momentum, Oliver included clips of many Republicans disavowing their support for Republican front runner Donald Trump, including commentator Doug Heye who called a possible Trump nomination, an abortion on our own party and the end of the Republican Party as we know it. Oliver said: Holy shit! You know they re getting desperate when they re just throwing in their favorite buzzwords. He won t just be an abortion, he ll be a sharia law wrapped in a Benghazi-themed gay wedding. Oliver then went on to give a reading of from former wife Ivana Trump s creepy thinly-veiled autobiographical soft-core novel For Love Alone, as read by actress Morgan Fairchild: In a moment, they were on the floor, on top of the mink coat, Katrinka s legs gripping his waist, as he moved deeper and deeper into her. As Adam pulled away from her, his c*ck fell for a moment into the valley between her legs, leaving a smear of semen on the dark silk. He smiled with satisfaction. Now you have to keep the coat. Watch the video here: Featured image via video screenshot.

Since, the *index* column is only for indexing, and the *label* column is the output. Thus, only the *title* and *text* columns can be used for predictors. Both columns contain a number of words that can be attached categories or values for prediction.

For example, the previous *title* column can be separated and turned into a list of words or tokens:
- watch, john, oliver, presents, gop, debates, as, clowntown, f, ck, the, world, sh, tshow, 2016 .

The above list also contains articles or words that are not useful for analysis or *stop words*. Stop words are defined as:

> Stop words are a set of commonly used words in a language. Examples of stop words in English are “a,” “the,” “is,” “are,” etc. Stop words are commonly used in Text Mining and Natural Language Processing (NLP) to eliminate words that are so widely used that they carry very little useful information. (Ganesan, 2023)

Example stop words in the `tidytext` library:
- a, a’s, able, about, above, according, accordingly, across, actually, after .

Based on the list of stop words in `tidytext` library, the title of the sample row can be processed into:
Title words without stop words:
- watch, john, oliver, gop, debates, clowntown, ck, world, sh, tshow, 2016.

Looking at the stop words removed from Title:
- *presents, as, f, the .*
The removed stop words indeed provide little information for analysis.

Trying out stop word removal for *text* column, show first 20 words:
- *john, oliver, isn, mincing, words, description, republicans, night, exception, left, hold, barred, discussed, insanity, republican, candidates, debates.his, primary, focus, uncanny .*

Sentiment Lexicons

To be able to extract information from useful words, a look up table of words to category/value will be used. The *tidytext* package contains 4 lexicons for sentiment analysis: *afinn*, *bing*, *loughran*, and *nrc*.

Looking at each lexicon:

Bing:
The first 5 words contained in the *bing* lexicon as sample:

Table 8: Bing Lexicon

word	sentiment
2-faces	negative
abnormal	negative
abolish	negative
abominable	negative
abominably	negative

The Bing Lexicon attributes the categories - *negative, positive ,* to each word. While it is useful, it overlaps with the function of the next lexicon.

Afinn:
The first 5 word of the *afinn* lexicon as sample:

Table 9: Afinn Lexicon

word	value
abandon	-2
abandoned	-2
abandons	-2
abducted	-2
abduction	-2

The Afinn Lexicon attributes each word the values:
- -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5.
This value ranges from positive to negative value with a neutral *0* value. The afinn lexicon attributes a positive, negative, or neutral numeric value to words. This range of value is of better use for the model as, aside from the sign (positive / negative / neutral), it also provides magnitude.

Loughran:
The first 5 word of the *loughran* lexicon as sample:

Table 10: Loughran Lexicon

word	sentiment
abandon	negative
abandoned	negative
abandoning	negative
abandonment	negative
abandonments	negative

Part of the loughran lexicon showing words with *superfluous* category:

Table 11: Loughran Lexicon for superfluous category

word	sentiment
aegis	superfluous
amorphous	superfluous
anticipatory	superfluous
appertaining	superfluous
assimilate	superfluous

Part of the loughran lexicon showing words with *uncertainty* category:

Table 12: Loughran Lexicon for uncertainty category

word	sentiment
abeyance	uncertainty
abeyances	uncertainty
almost	uncertainty
alteration	uncertainty
alterations	uncertainty

Loughran lexicon attributes each word the categories:
- negative, positive, uncertainty, litigious, constraining, superfluous .

NRC:
The first 5 word of the *nrc* lexicon as sample:

Table 13: NRC Lexicon

word	sentiment
abacus	trust
abandon	fear
abandon	negative
abandon	sadness
abandoned	anger
abandoned	fear
abandoned	negative
abandoned	sadness
abandonment	anger
abandonment	fear

NRC lexicon attributes each word the category:
- trust, fear, negative, sadness, anger, surprise, positive, disgust, joy, anticipation.
This lexicon uses emotional categories of words for analysis.

Lexicon Selection

Only part of the 4 lexicons will be used so that the model will not be too complex with too many columns. First, the *Bing* and *Afinn* Lexicons overlap in usage, assigning positive and negative categories or values to each word. However, *Afinn* derives more information than *Bing*, since the assigned values vary in magnitude. This means that the *Afinn* lexicon will be used instead of *Bing*. Second, the *Loughran* and *NRC* lexicon both assign categories to each of the words.

Next step is evaluating the last three lexicons (without *Bing*). From base number of words without stop words of 142,
- *Afinn* Lexicon attributed a total of 14 words.
- *Loughran* Lexicon attributed a total of 6 words.
- *NRC* Lexicon attributed a total of 34 unique words with some words attributed to multiple categories, bringing the total categorized words to 66

Since both *Loughran* and *NRC* assigns categories to each of the words, one of the two can be selected. However, from the results above, the *Loughran* Lexicon could remove too many words from the text. Thus the *NRC* lexicon will be used instead of the *Loughran* lexicon.

The two lexicons to be used are *Afinn* and *NRC*.

Sentiment Extraction

Using the lexicons *NRC* and *Afinn*, it is now possible to extract a useful predictor from the sentiments/value.

For *Afinn*, the cumulative value can be used to identify the net sentiment of the content. Using *Afinn* to get the cumulative value for the first 5 rows:

Table 14: Afinn Lexicon net value

index	value
0	-5
1	0
2	-12
3	6
4	15

For *NRC*, the total count of word per category can be used to identify the common sentiment category in the text. For words not in any *NRC* category, they are put into the *neutral* category. Using *NRC* to get word counts per category:

Table 15: NRC lexicon sentiment total

index	anger	anticipation	disgust	fear	joy	negative	positive	sadness	surprise	trust	neutral
0	7	3	2	5	3	6	13	3	8	9	87
1	0	1	0	1	1	2	6	0	1	6	27
2	8	12	6	9	13	19	24	14	20	16	82
3	5	7	0	14	11	14	35	5	5	25	163
4	4	6	10	2	5	11	21	3	6	10	120

Previously, the text column was used. However, it may be possible to also use title sentiments for predictors.

Table 16: Afinn Net Value of Title

index	value
0	-3
1	0
2	-1
3	0

Table 17: NRC Sentiment Totals of Title

index	surprise	neutral	fear	positive	trust	anticipation	joy
0	1	7	0	0	0	0	0
1	1	4	1	2	1	0	0
2	3	2	0	3	1	1	2
3	1	5	1	1	0	0	0

Extract Predictors From Dataset

The extracted predictors can now be joined into one table / data frame that will then be used for training a model. The extracted predictors for *train set*:

Table 18: Text Predictors of Train Set

index	value	anger	anticipation	disgust	fear	joy	negative	positive	sadness	surprise	trust	neutral	words
0	-5	7	3	2	5	3	6	13	3	8	9	87	112
1	0	0	1	0	1	1	2	6	0	1	6	27	37
2	-12	8	12	6	9	13	19	24	14	20	16	82	136
3	6	5	7	0	14	11	14	35	5	5	25	163	223
4	15	4	6	10	2	5	11	21	3	6	10	120	156
5	-30	8	3	1	13	2	13	8	6	0	5	80	105

Table 19: Title Predictors of Train Set

index	value	surprise	neutral	fear	positive	trust	anticipation	joy	disgust	negative	sadness	anger	words
0	-3	1	7	0	0	0	0	0	0	0	0	0	8
1	0	1	4	1	2	1	0	0	0	0	0	0	8
2	-1	3	2	0	3	1	1	2	0	0	0	0	6
3	0	1	5	1	1	0	0	0	0	0	0	0	8
4	5	1	4	0	4	3	1	1	1	1	1	0	9
5	-6	0	7	2	0	0	0	0	0	2	2	1	9

Notice that, alongside the (Afinn) *value* and the (NRC) sentiments, the total word count without stop words was also included as another predictor.

Get A Working Model

Selected Models that are to be used: - *naive_bayes, glm, qda, lda, rf, knn* .
-naive_bayes: uses Naive Bayes model Classification for classification.
-glm: uses Generalized Linear Model for classification.
-qda: uses Quadratic Discriminant Analysis for classification.
-lda: uses Linear Discriminant Analysis for classification.
-rf: uses Random Forest model for classification.
-knn: uses k-Nearest Neighbors for modeling.

These models are to be used to predict output, then evaluated based on *Accuracy*, *Sensitivity* and *Specificity*. The better model will then be further optimized to be used for the final model.

Accuracy is the percentage of the prediction that is of the correct category. *Sensitivity* and *Specificity*, similar to *Accuracy*, is the percentage of the correct prediction for "*positive*" and "*negative*" class respectively.

Using the following terms:
True Positives (TP) are the *positive* class that are predicted correctly.
True Negatives (TN) are the *negative* class that are predicted correctly.
False Positives (FP) are the *negative* class that are predicted (incorrectly) as *positive*.
False Negative (FN) are the *positive* class that are predicted (incorrectly) as *negative*.

The three statistics are defined by the formulas:

$$Accuracy = \frac{TP + TN}{TotalPredictions}$$

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

Training each of the models, the results are shown in a table:

Table 20: Model Training Statistics

model_name	Accuracy	Sensitivity	Specificity	Prevalence
naive_bayes	0.670814	0.975023	0.317080	0.537637
glm	0.775040	0.855866	0.681055	0.537637
qda	0.736479	0.863657	0.588596	0.537637
lda	0.776518	0.884968	0.650413	0.537637
rf	0.818159	0.883822	0.741807	0.537637
knn	0.738573	0.860678	0.596589	0.537637

Visualizing Specificity, Sensitivity differences:

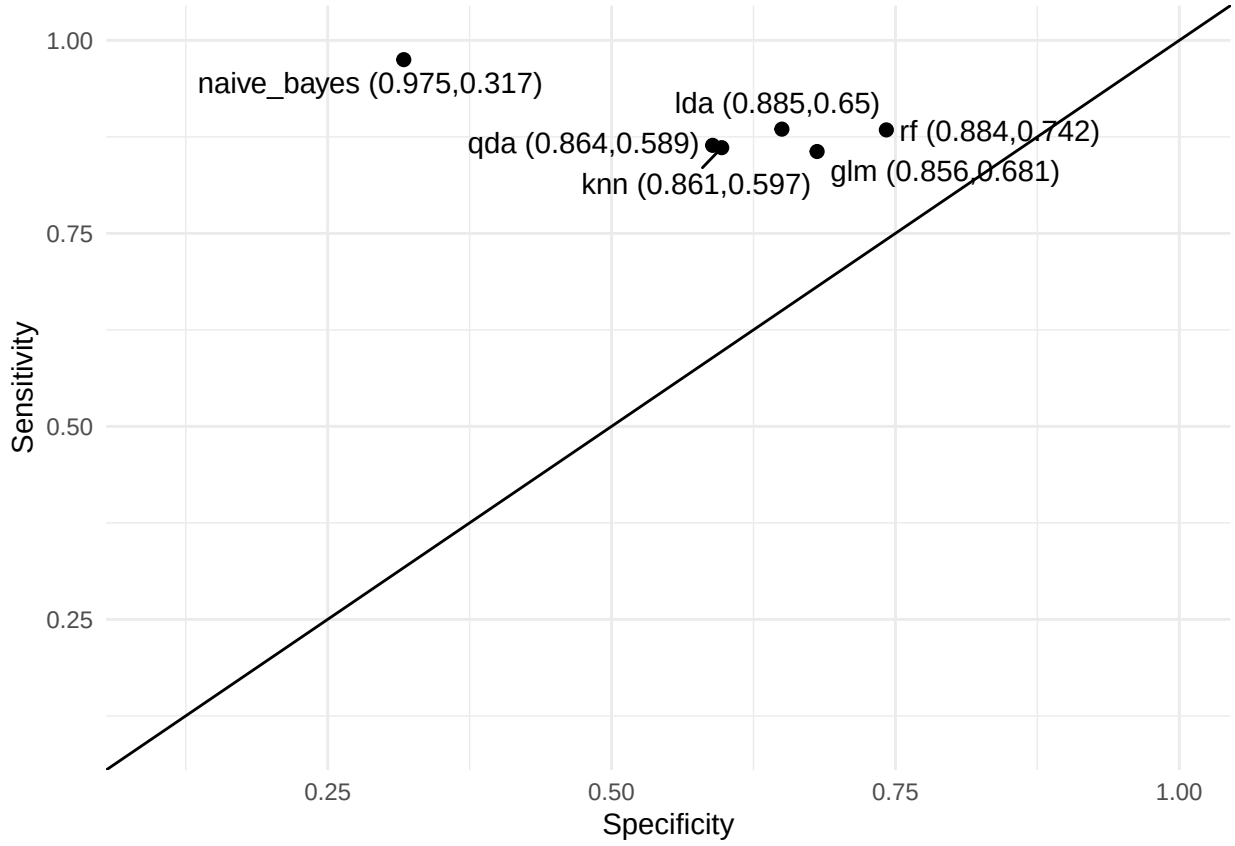

Figure 1: Training Models' Specificity vs. Sensitivity

Random Forest model has the highest Accuracy with reasonable Specificity and Sensitivity.

Optimize Selected Model

Looking at the obtained Random Forest model:

```
## Random Forest
##
## 24353 samples
##    26 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 24353, 24353, 24353, 24353, 24353, 24353, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.818287  0.630107
##   14    0.815756  0.625781
##   26    0.811539  0.617274
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The model used the tuning parameter *mtry*. The initial *Accuracy* within the sample is around 0.8.

From the previous model, the important variables / predictors can be obtained and shown with its overall importance.

Table 21: Predictors arranged based on overall importance

	Overall
title_words	100.000000
title_neutral	63.240827
text_words	58.962427
text_neutral	58.485995
text_trust	49.278773
text_surprise	47.337496
text_disgust	38.280300
text_positive	37.475907
text_value	35.626755
text_negative	27.732134
text_fear	27.130621
text_anticipation	25.690756
text_joy	25.175516
title_value	24.784173
text_anger	24.221265
text_sadness	21.888428
title_disgust	8.371059
title_negative	7.483234
title_positive	7.044451
title_surprise	6.920214
title_fear	6.591258
title_trust	5.224828
title_anger	2.622995
title_anticipation	2.436658
title_sadness	0.761833
title_joy	0.000000

One of the columns are not necessary for training. By changing the input to only use the important columns, the left over columns are:
- text_anger, text_anticipation, text_disgust, text_fear, text_joy, text_negative, text_neutral, text_positive, text_sadness, text_surprise, text_trust, text_value, text_words, title_anger, title_anticipation, title_disgust, title_fear, title_negative, title_neutral, title_positive, title_sadness, title_surprise, title_trust, title_value, title_words.

The non-important column is *:title_joy .*

By removing the unimportant column and retraining the model, the results are:

```
## Random Forest
##
## 24353 samples
##    25 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 24353, 24353, 24353, 24353, 24353, 24353, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.817796  0.629343
##   13    0.814963  0.624252
##   25    0.810498  0.615203
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Table 22: Model Result with removed unimportant predictors

model_name	Accuracy	Sensitivity	Specificity	Prevalence
rf	0.819268	0.886343	0.741274	0.537637

Next is optimizing the model parameter. From the model earlier, as well as the model details from `modelLookup()`:

Table 23: RF model lookup details / tuning parameters

model	parameter	label	forReg	forClass	probModel
rf	mtry	#Randomly Selected Predictors	TRUE	TRUE	TRUE

Optimization parameter is *mtry*, and the final model used to get the highest accuracy is at `mtry = 2`. To find an even better fit, try training the model using *mtry* values close to *2*.

Trying an mtry sequence from 2 to 5:

```
## Random Forest
##
## 24353 samples
##    25 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 24353, 24353, 24353, 24353, 24353, 24353, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##   2     0.818416  0.630410
##   3     0.819133  0.632153
##   4     0.819256  0.632498
##   5     0.819212  0.632497
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

Table 24: Model Result with optimized mtry.

model_name	Accuracy	Sensitivity	Specificity	Prevalence
Optimized RF	0.819761	0.885655	0.743139	0.537637

Results

Finally, add *test set* onto the *train set*, then retrain model in Random Forest with mtry parameter set to 4.

Table 25: Final Model Statistics

model_name	Accuracy	Sensitivity	Specificity	Prevalence
Optimized Final RF	0.813108	0.87019	0.748357	0.531477

The final model achieved an 81.31% *Accuracy*. With a 87.02% of labeling a Fake News article as Fake News (*Sensitivity*). With this level of *Accuracy* and *Sensitivity*, the model provides a reliable way of automatically detecting a Fake News article. However, the model still has a 25.16% chance of mislabeling a genuine article as fake news.

Conclusion

The Final Model was obtained through the use of Sentiment Analysis. Two lexicons, *Afinn* and *NRC*, were used to evaluate each word in the *title* and *text* columns. The values and categories obtained from the lexicons was then used to build a set of predictors that can be used to train a machine learning model. From the set of predictors, a list of models was trained and evaluated. The *Random Forest* model was then selected for the Final Model based on *Accuracy*, *Sensitivity*, and, *Specificity*. The *Random Forest* model's optimization parameter *mtry* was further optimized to get the Highest Accuracy possible from the model. Then, both the *train set* and *test set* was used for training the Final *Random Forest* model with `mtry = 4`. Finally, The Final Model was evaluated using extracted predictors from *evaluation set*. The *Accuracy* of the Final model was 81.31%, with a *Sensitivity* of 0.87019 and a *Specificity* of 0.748357. The model has a higher *Sensitivity* than *Specificity*. High *Sensitivity* means that the model has a high rate of detecting Fake news articles. However the relatively lower *Specificity* means that it can label genuine articles as Fake news.

The Model achieved a reasonably high *Accuracy* and *Sensitivity*. However, the model can be further improved. First, the lexicons used was only able to evaluate parts of the text. More extensive lexicons may improve the model. Secondly, the model was unable to evaluate the links or videos inside the articles. The dialogue / transcript / script of these videos could be used to further improve the model. However, this requires accessing each link and obtaining the script, which would add a lot of complexity to the model. Lastly, the model could be modified and improved to, instead of directly labeling Real/Fake News, determine the chance that a given input is Fake news or not. This would probably be more useful than directly labeling the news as Real or Fake.

References

Dataset obtained at:

Singh, A. (2024, October 22). Kaggle : Fake news classification. https://www.kaggle.com/datasets/aadyasingh55/fake-news-classification

Other references for the report:

De La Cruz, R. (2023, December 21). Sentiment Analysis using Natural Language Processing (NLP). Medium. https://medium.com/@robdelacruz/sentiment-analysis-using-natural-language-processing-nlp-3c12b77a73ec

Ganesan, K. (2023, March 12). What are Stop Words? Opinosis Analytics. https://www.opinosis-analytics.com/knowledge-base/stop-words-explained/