GAURAV JHANWAR

Paper Submission

Table of Contents

Abstract

The web and therefore Social media is full of fake news, bogus accounts, and sham posts. Stanford University defines fake news as news articles which are intentionally and verifiably false and could mislead readers. The intention is commonly to deceive readers and manipulate them into buying or believing something that isn't true. By the help of this project, a reader can know the likelihood of what they are reading is real or fake news.

There is a wide range of classes falsehood can fall into. Some articles are explicitly false, some items that give an honest occasion yet then make some false translations, some articles that are unscientific, materials that are exceptionally just an opinion camouflaged as news, pieces that are sarcastic. That's why appropriate response isn't distinguishing counterfeit news, but identifying real news. The real news is significantly less troublesome to classify. It is verifiable and to the point. Also, there are a lot of respectable sources to get it from.

Introduction

A large part of our lives are spent on social media and this has introduced a new type of problem. People usually believe that the information they get from these platforms are real and prefer news from these sources rather than news channels. This incorporate a craving to uncover profitable and engaging substance to others; to characterize themselves; to develop and sustain connections and to get the word out about brands and causes they like or support. The explanations behind this adjustment in utilization practices are intrinsic in the nature of these online networking stages. Firstly, it is more convenient and affordable than the usual means of news gathering, i.e. newspapers and TV. Secondly, it is more convenient to post views for the consumer as well as reply on others' views and comments and further share these posts. These elements have changed these social websites from being a helpful means for staying in contact with loved ones to being utilized as a part of ways that really affect society. Online networking is being utilized as a part of society that shapes governmental issues, business, world culture, industry, professions, development, and that's just the beginning.

Fake News Characterization

Fake news has always been a weapon to help political propaganda and powerful individuals. Octavian broadly utilized a crusade of disinformation to help his triumph over Marc Anthony in the last war of the Roman Republic. In its result, he changed his name to Augustus, and dispatched a complimenting and energetic picture of himself all through the Empire, keeping up its utilization in his seniority. This proves that fake news is not a newly developed idea but has been used for thousands of years.

Establishment of Fake News in the Era of Social Media

Prior to the web, it was substantially costlier to disperse data, developing trust took years, and there were considerably more straightforward meanings of what constituted news and media, making control of self-direction less demanding. Yet, the ascent of web-based life has separated a large number of the limits that kept the phony news from spreading in. Specifically, it has enabled anybody to make and disperse data, particularly those that have demonstrated most proficient at bluffing how informal communities work. Facebook and Twitter enabled individuals to trade data on a significantly more noteworthy scale than at any other media, while distributing stages like WordPress enabled anybody to make a dynamic site effortlessly. To put it plainly, the boundaries to making counterfeit news have been broken. Be that as it may, scams and misrepresentations have been related with the web since its initial days, however it is just over the most recent two years that sorted out, deliberate deception battles, regularly connected to governments, have risen, and their impact on vote based system and society examined. Critics say the main idea behind this is spreading falsehood, with the compass of a story reliant on its capacity to circulate around the web, be sensational and scandalous- something that frequently relies upon drama and passionate responses more than truth itself.

Datasets for the Project

The breakdown of the data used in this study is given hereunder:

- 13,000 recent fake news articles were extracted from Kaggle.com. These articles were explored and cleaned based on our requirement.
- 2000 Real news from The Guardian API
- 2000 Real news from The New York Times (NYT) API

To account for oversampling of fake news data, all the data sets were combined and a total of 6,316 articles were finally used for building the training, validation and testing sets. The dataset was finally cleaned to only have 3 columns, namely:

- Title: Title of the article
- Body: Text inside the article
- Label: Real or Fake

Scrapping the Datasets

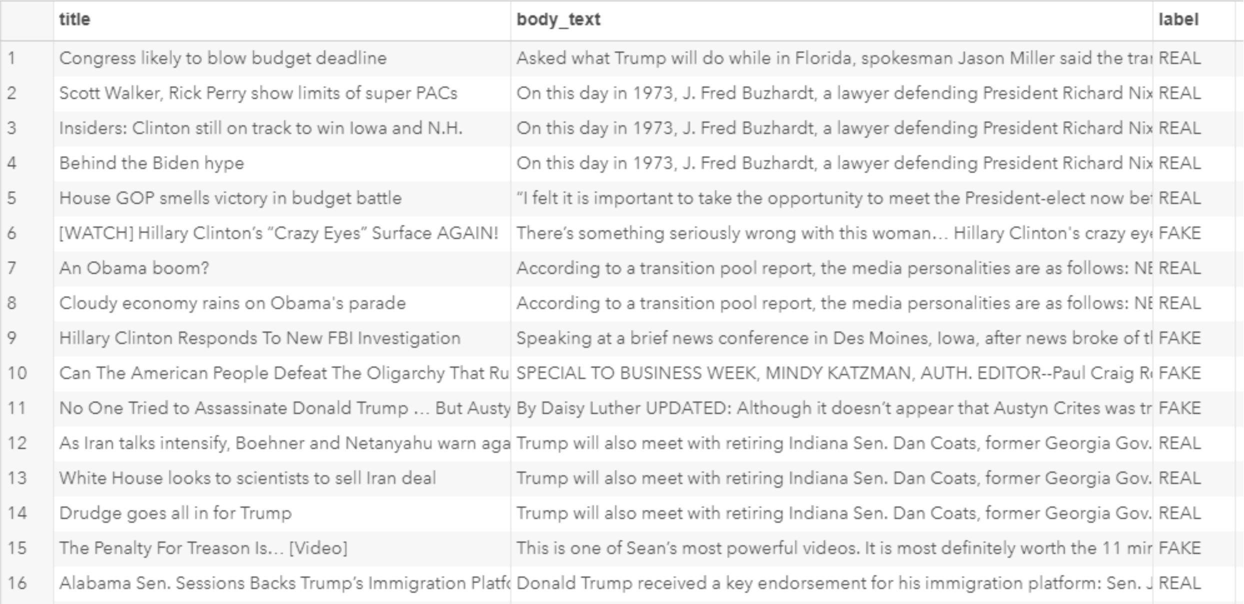
We scrapped the data from both Guardian API and NYT API. The steps mentioned below were performed for scrapping data from both the sources:

1. Initiated the Mongo client.
2. Retrieved the database and collection created for articles.
3. Added the 'page' parameter to the complete payload.
4. Got the requested URL. Error handling for bad requests was done in the calling function.
5. Returned to the metadata and docs including URLs.
6. Passed header to NYT when scraping article text.
7. Scraped the doc's URL, return a soup object with the URL's text.
8. Joined the resulting body paragraphs' text (returned in a list).
9. Requested all of the newest articles matching the search term.
10. Called the API with Base URL including parameters and page.
11. Incremented the page before we encounter any potential errors.
12. Got the metadata and documents from the request.
13. Initialized database and collection.
14. Set the initial end date (scraper starts at this date and moves back in time sequentially).
15. Passed the database collection and initial end date into main function for further processing.

Data Gathering

Below are the steps performed for cleaning and exploring the dataset:

1. Imported the data sets and combined them all together.
2. Got body text from the data frame and for building a column.
3. Created a column with body text.
4. Got headlines from the data frame for building another column in for final data.
5. Created a column with title of the article.
6. Got rid of the empty bodies. These would have just taken space without any influence on the model.
7. Overviewed the data frame for final outlook.
8. Saved the cleaned data into csv file for being used in the model.



	title	body_text	label
1	Congress likely to blow budget deadline	Asked what Trump will do while in Florida, spokesman Jason Miller said the trai	REAL
2	Scott Walker, Rick Perry show limits of super PACs	On this day in 1973, J. Fred Buzhardt, a lawyer defending President Richard Nix	REAL
3	Insiders: Clinton still on track to win Iowa and N.H.	On this day in 1973, J. Fred Buzhardt, a lawyer defending President Richard Nix	REAL
4	Behind the Biden hype	On this day in 1973, J. Fred Buzhardt, a lawyer defending President Richard Nix	REAL
5	House GOP smells victory in budget battle	"I felt it is important to take the opportunity to meet the President-elect now be	REAL
6	[WATCH] Hillary Clinton's "Crazy Eyes" Surface AGAIN!	There's something seriously wrong with this woman… Hillary Clinton's crazy eye	FAKE
7	An Obama boom?	According to a transition pool report, the media personalities are as follows: NE	REAL
8	Cloudy economy rains on Obama's parade	According to a transition pool report, the media personalities are as follows: NE	REAL
9	Hillary Clinton Responds To New FBI Investigation	Speaking at a brief news conference in Des Moines, Iowa, after news broke of tl	FAKE
10	Can The American People Defeat The Oligarchy That Ru	SPECIAL TO BUSINESS WEEK, MINDY KATZMAN, AUTH. EDITOR--Paul Craig Ro	FAKE
11	No One Tried to Assassinate Donald Trump … But Austy	By Daisy Luther UPDATED: Although it doesn't appear that Austyn Crites was tr	FAKE
12	As Iran talks intensify, Boehner and Netanyahu warn aga	Trump will also meet with retiring Indiana Sen. Dan Coats, former Georgia Gov.	REAL
13	White House looks to scientists to sell Iran deal	Trump will also meet with retiring Indiana Sen. Dan Coats, former Georgia Gov.	REAL
14	Drudge goes all in for Trump	Trump will also meet with retiring Indiana Sen. Dan Coats, former Georgia Gov.	REAL
15	The Penalty For Treason Is… [Video]	This is one of Sean's most powerful videos. It is most definitely worth the 11 mir	FAKE
16	Alabama Sen. Sessions Backs Trump's Immigration Platfc	Donald Trump received a key endorsement for his immigration platform: Sen. J	REAL

Table:1

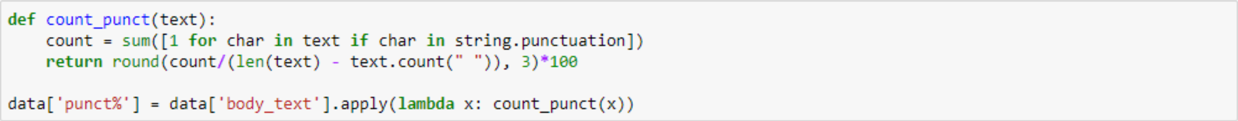
Data Manipulation

Using SAS Studio on demand, features such as body length and punctuation% per article were extracted. Body length was calculated using the LENGTHN() function for each article. This function returns the location of the last non-blank character and excludes the trailing blanks. This function was preferred over other length functions as it returns 0 for completely blank articles.

The punctuation% was calculated using the formula given below:

Punctuation% = No. of punctuations (Calculated by adding a python call in SAS Code Node)/Body length

Python Code:



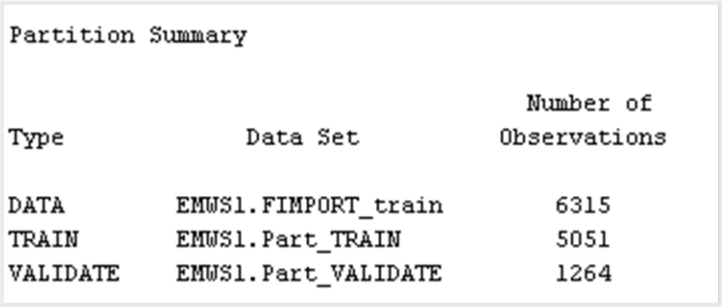
```
def count_punct(text):
    count = sum([1 for char in text if char in string.punctuation])
    return round(count/(len(text) - text.count(" ")), 3)*100

data['punct%'] = data['body_text'].apply(lambda x: count_punct(x))
```

Code:1

For the next steps of data manipulation, Enterprise Miner was used to partition the data. A new project is created in SAS EM 14.3 and a data source is created with the fake news data set. In the edit variable section, the role for column "label" is selected as the Target variable and its level is selected as Binary.

Data Partition:

A data partition node is added to divide the data into training (80%) and validation sets (20%).

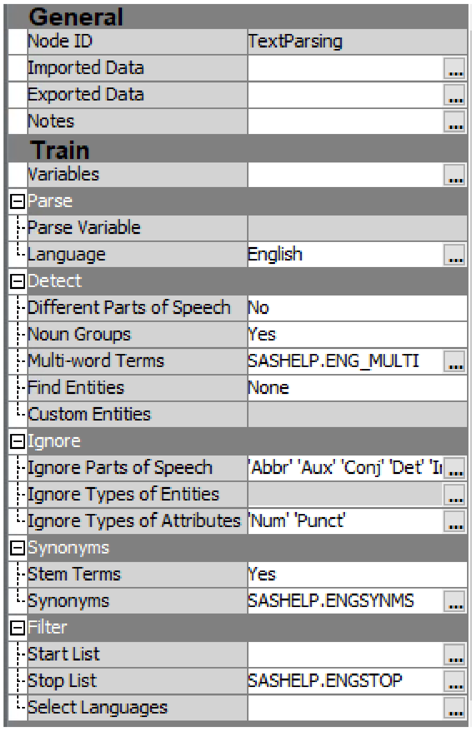


```
Partition Summary

                                        Number of
Type           Data Set                Observations

DATA        EMWS1.FIMPORT_train          6315
TRAIN       EMWS1.Part_TRAIN             5051
VALIDATE    EMWS1.Part_VALIDATE          1264
```

Figure:1

Text Parsing:

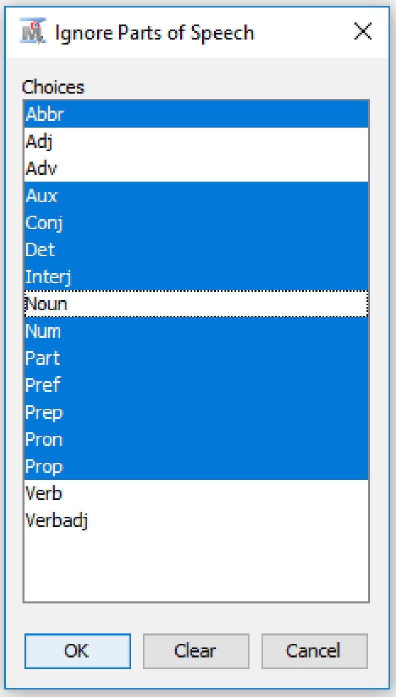
The text parsing node was used to ignore the parts of speech and attributes such as numbers and punctuations. Stem terms was set to yes for stemming the words thereby removing –ing, -es, -ed, etc. from words and making them in the same tense. Parsing language is set to English and different parts of speech detection is set to “No” for treating all similar words as same. Stop words, Synonyms and Multi-word terms were taken from a pre-defined table in the SASHELP library.The changes in the properties panel can be made as given under:



General	
Node ID	TextParsing
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
⊟Parse	
Parse Variable	
Language	English ...
⊟Detect	
Different Parts of Speech	No
Noun Groups	Yes
Multi-word Terms	SASHELP.ENG_MULTI ...
Find Entities	None
Custom Entities	
⊟Ignore	
Ignore Parts of Speech	'Abbr' 'Aux' 'Conj' 'Det' 'I ...
Ignore Types of Entities	...
Ignore Types of Attributes	'Num' 'Punct' ...
⊟Synonyms	
Stem Terms	Yes
Synonyms	SASHELP.ENGSYNMS ...
⊟Filter	
Start List	...
Stop List	SASHELP.ENGSTOP ...
Select Languages	...

Figure:2

The parts of speech ignored can be seen from the figure:



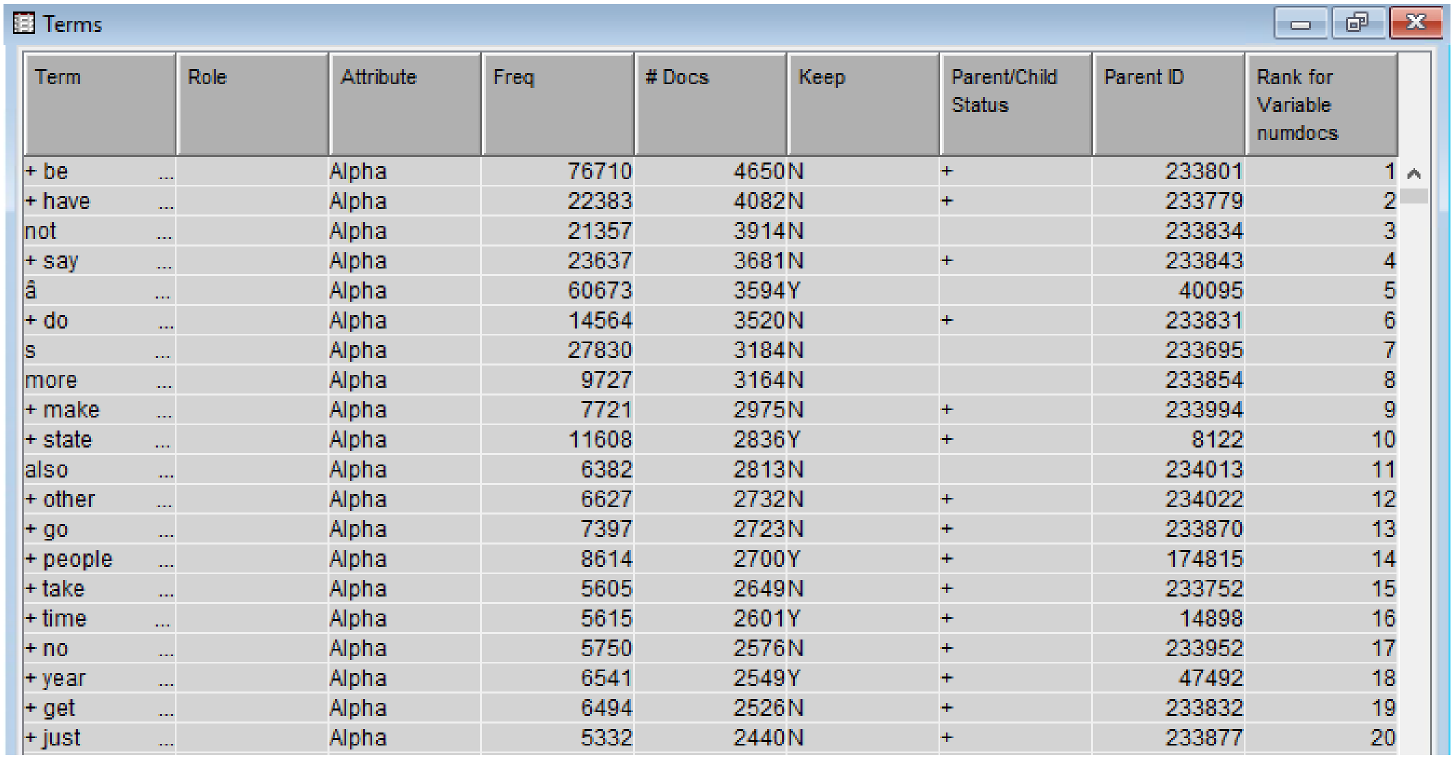
Ignore Parts of Speech

Choices

[✓ Abbr]
Adj
Adv
[✓ Aux]
[✓ Conj]
[✓ Det]
[✓ Interj]
Noun
[✓ Num]
[✓ Part]
[✓ Pref]
[✓ Prep]
[✓ Pron]
[✓ Prop]
Verb
Verbadj

Figure:3

Output:



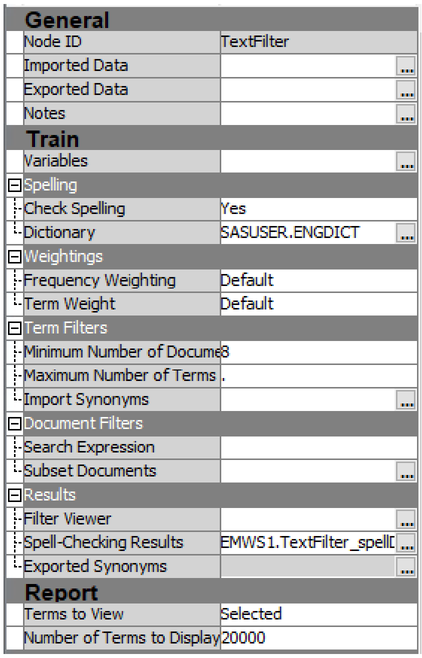
Terms

Term	Role	Attribute	Freq	# Docs	Keep	Parent/Child Status	Parent ID	Rank for Variable numdocs
+ be ...		Alpha	76710	4650	N	+	233801	1
+ have ...		Alpha	22383	4082	N	+	233779	2
not ...		Alpha	21357	3914	N		233834	3
+ say ...		Alpha	23637	3681	N	+	233843	4
â ...		Alpha	60673	3594	Y		40095	5
+ do ...		Alpha	14564	3520	N	+	233831	6
s ...		Alpha	27830	3184	N		233695	7
more ...		Alpha	9727	3164	N		233854	8
+ make ...		Alpha	7721	2975	N	+	233994	9
+ state ...		Alpha	11608	2836	Y	+	8122	10
also ...		Alpha	6382	2813	N		234013	11
+ other ...		Alpha	6627	2732	N	+	234022	12
+ go ...		Alpha	7397	2723	N	+	233870	13
+ people ...		Alpha	8614	2700	Y	+	174815	14
+ take ...		Alpha	5605	2649	N	+	233752	15
+ time ...		Alpha	5615	2601	Y	+	14898	16
+ no ...		Alpha	5750	2576	N	+	233952	17
+ year ...		Alpha	6541	2549	Y	+	47492	18
+ get ...		Alpha	6494	2526	N	+	233832	19
+ just ...		Alpha	5332	2440	N	+	233877	20

Figure:4

Text Filter:

The parsed terms were then reduced by reducing the number of documents analyzed. Thus, noisy and irrelevant data is removed. The properties panel for the text filter node is given hereunder:



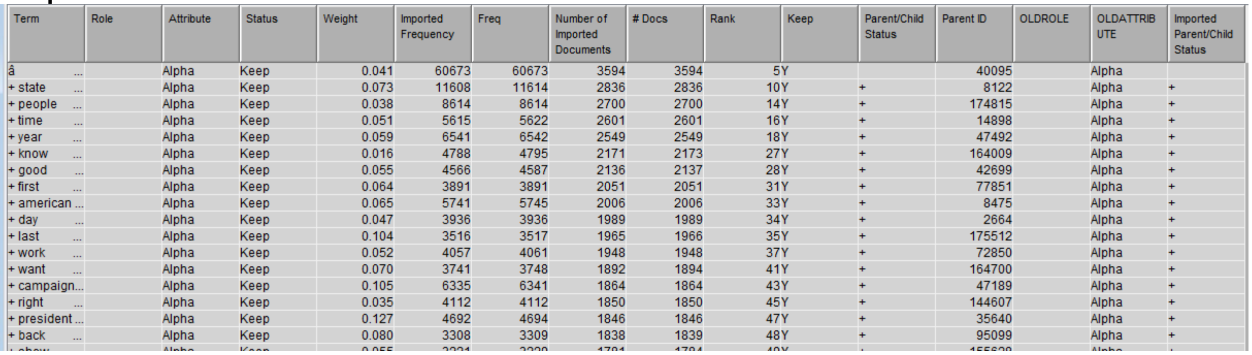
General	
Node ID	TextFilter
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
⊟Spelling	
Check Spelling	Yes
Dictionary	SASUSER.ENGDICT ...
⊟Weightings	
Frequency Weighting	Default
Term Weight	Default
⊟Term Filters	
Minimum Number of Docume	8
Maximum Number of Terms	.
Import Synonyms	...
⊟Document Filters	
Search Expression	
Subset Documents	...
⊟Results	
Filter Viewer	...
Spell-Checking Results	EMWS1.TextFilter_spellD ...
Exported Synonyms	...
Report	
Terms to View	Selected
Number of Terms to Display	20000

Figure:5

Changes made were:

- Spell check was set to "Yes" for rectifying spellings and creating synonyms for the misspelt words.
- A custom English dictionary was added.
- Minimum documents are set to 8.
- Terms to view was changed to "Selected" in the reports section.

Output:



Term	Role	Attribute	Status	Weight	Imported Frequency	Freq	Number of Imported Documents	# Docs	Rank	Keep	Parent/Child Status	Parent ID	OLDROLE	OLDATTRIBUTE	Imported Parent/Child Status
â ...		Alpha	Keep	0.041	60673	60673	3594	3594	5	Y		40095		Alpha	
+ state ...		Alpha	Keep	0.073	11608	11614	2836	2836	10	Y	+	8122		Alpha	+
+ people ...		Alpha	Keep	0.038	8614	8614	2700	2700	14	Y	+	174815		Alpha	+
+ time ...		Alpha	Keep	0.051	5615	5622	2601	2601	16	Y	+	14898		Alpha	+
+ year ...		Alpha	Keep	0.059	6541	6542	2549	2549	18	Y	+	47492		Alpha	+
+ know ...		Alpha	Keep	0.016	4788	4795	2171	2173	27	Y	+	164009		Alpha	+
+ good ...		Alpha	Keep	0.055	4566	4587	2136	2137	28	Y	+	42699		Alpha	+
+ first ...		Alpha	Keep	0.064	3891	3891	2051	2051	31	Y	+	77851		Alpha	+
+ american ...		Alpha	Keep	0.065	5741	5745	2006	2006	33	Y	+	8475		Alpha	+
+ day ...		Alpha	Keep	0.047	3936	3936	1989	1989	34	Y	+	2664		Alpha	+
+ last ...		Alpha	Keep	0.104	3516	3517	1965	1966	35	Y	+	175512		Alpha	+
+ work ...		Alpha	Keep	0.052	4057	4061	1948	1948	37	Y	+	72850		Alpha	+
+ want ...		Alpha	Keep	0.070	3741	3748	1892	1894	41	Y	+	164700		Alpha	+
+ campaign...		Alpha	Keep	0.105	6335	6341	1864	1864	43	Y	+	47189		Alpha	+
+ right ...		Alpha	Keep	0.035	4112	4112	1850	1850	45	Y	+	144607		Alpha	+
+ president ...		Alpha	Keep	0.127	4692	4694	1846	1846	47	Y	+	35640		Alpha	+
+ back ...		Alpha	Keep	0.080	3308	3309	1838	1839	48	Y	+	95099		Alpha	+

Figure:6

Concept Linking diagram are given under numbered by most weighted terms:

1. Indigenous:
   Frequency-109, Documents-39, Weight-0.305
   High Association with water protector, pipeline, sioux.

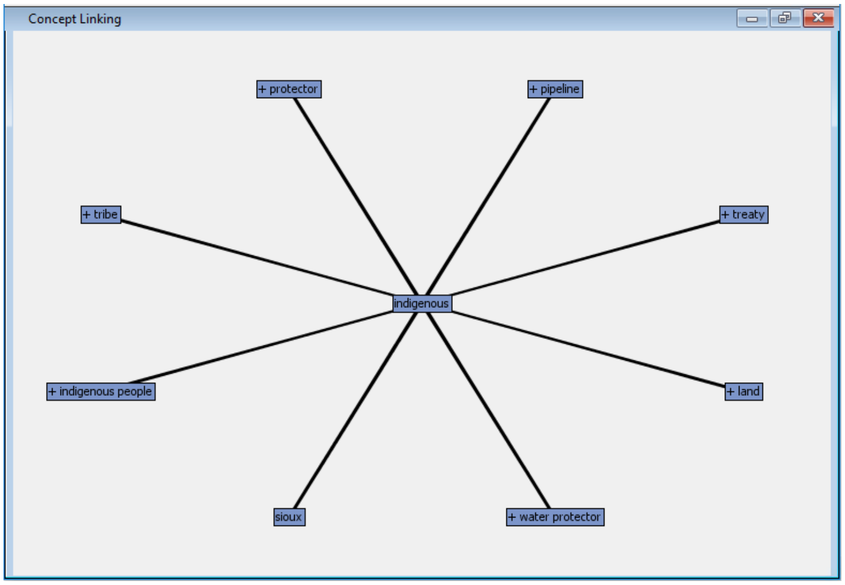


Figure:7

2. Wikileaks emails:
   Frequency-22, Documents-28, Weight-0.305
   High association with mortgage, Clinton campaign, email, reveal, dump

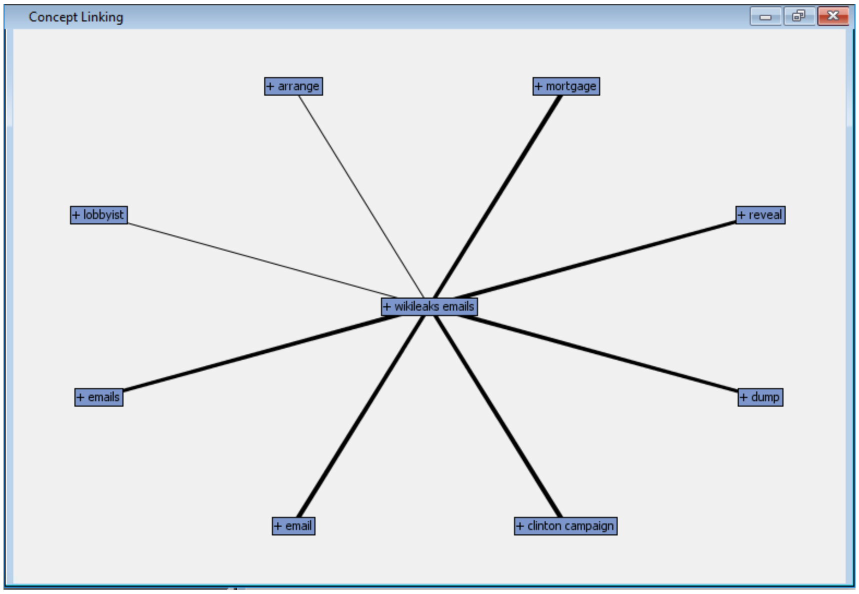


Figure:8

3. Ruling Class:
   Frequency-22, Documents-28, Weight-0.305
   High association with words such as capitalists, supremacy, rule, imperialist.

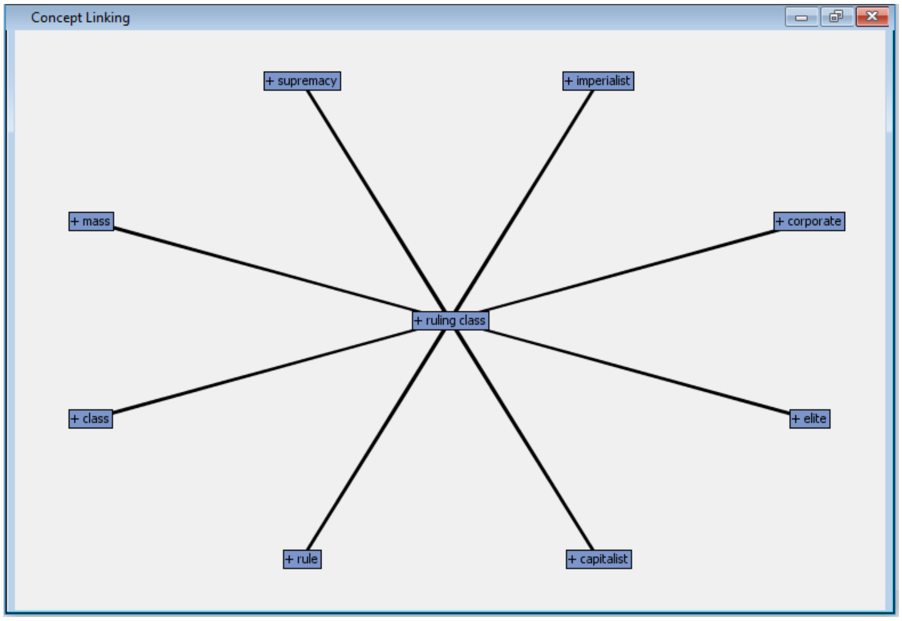
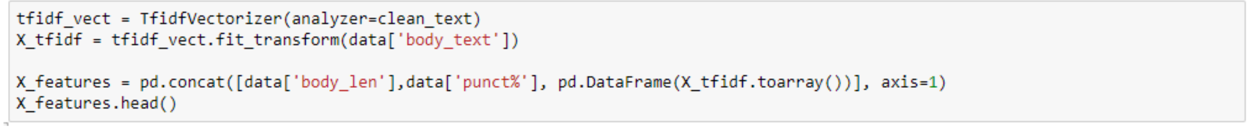


Figure:9

Vectorization

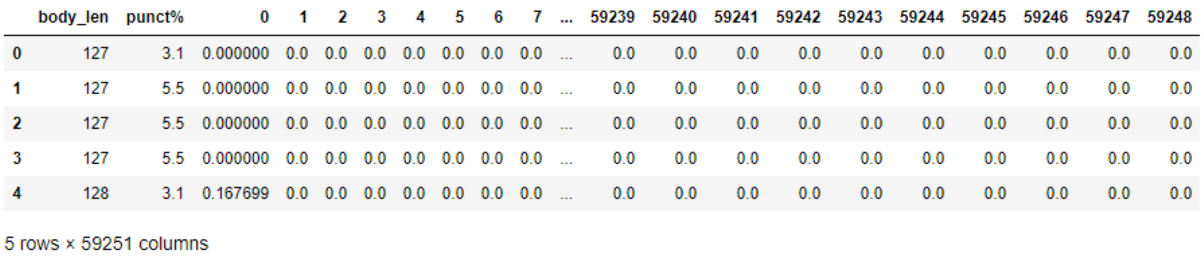
The data is now exported to Python for further analysis. TF-IDF (Term Frequency – Inverse Document Frequency) statistic is used to identify how important is a word in a document. The parsed and tokenized words are all given a number as representation and these are now included in the data set as variables and their probability in each document is taken out and set for the corresponding articles. A total of 59,251 columns were made. The python code given below is for the same:



```
tfidf_vect = TfidfVectorizer(analyzer=clean_text)
X_tfidf = tfidf_vect.fit_transform(data['body_text'])

X_features = pd.concat([data['body_len'],data['punct%'], pd.DataFrame(X_tfidf.toarray())], axis=1)
X_features.head()
```

Code:2

Output:



	body_len	punct%	0	1	2	3	4	5	6	7	...	59239	59240	59241	59242	59243	59244	59245	59246	59247	59248
0	127	3.1	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	127	5.5	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	127	5.5	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	127	5.5	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	128	3.1	0.167699	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 59251 columns

Table:2

Modeling

After many trials, Random Forest method through Holdout set was taken as the final model. The steps followed are:

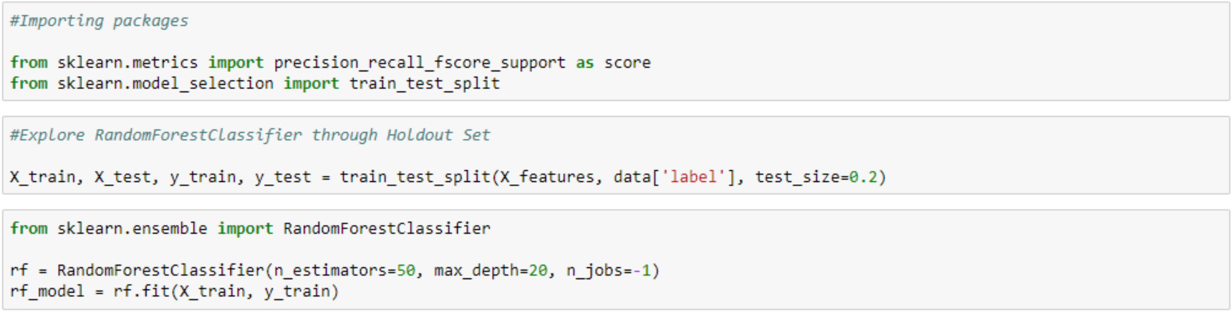
1. Importing packages

Packages such as metrics and model selection from sci-kit learn module were imported.

2. Training and testing sets were split and random forest model was run with the below mentioned parameters:

- Number of estimators- 50
- Maximum depth- 20
- Number of jobs - -1

The number of jobs=-1 ensures that all models are running simultaneously rather than one after the other.



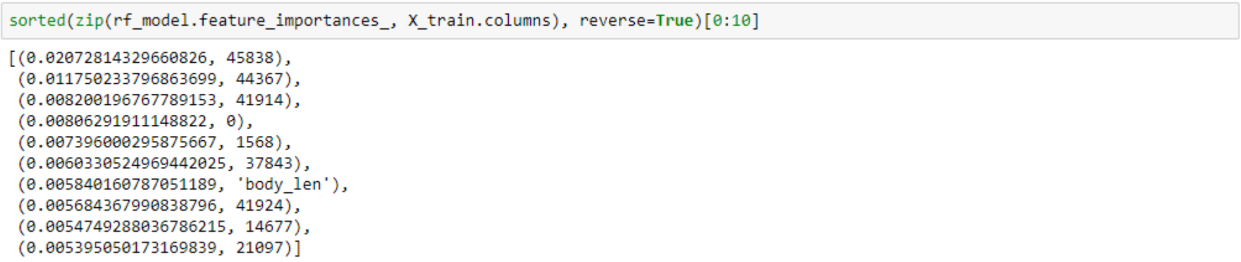
```
#Importing packages

from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.model_selection import train_test_split
```

```
#Explore RandomForestClassifier through Holdout Set

X_train, X_test, y_train, y_test = train_test_split(X_features, data['label'], test_size=0.2)
```

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=50, max_depth=20, n_jobs=-1)
rf_model = rf.fit(X_train, y_train)
```

Code:3

3. Variable importance output received.

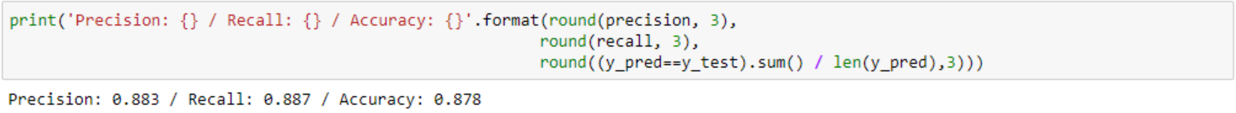


```
sorted(zip(rf_model.feature_importances_, X_train.columns), reverse=True)[0:10]
```

```
[(0.02072814329660826, 45838),
 (0.011750233796863699, 44367),
 (0.008200196767789153, 41914),
 (0.00806291911148822, 0),
 (0.007396000295875667, 1568),
 (0.006033052496944025, 37843),
 (0.005840160787051189, 'body_len'),
 (0.005684367990838796, 41924),
 (0.0054749288036786215, 14677),
 (0.005395050173169839, 21097)]
```

Code:4

4. Model accuracy output came out to be 87.8%. Precision (88.3%) as well as recall (88.7%) were also quite high. Overall, the model looks pretty promising and this model is finalized as the winning model. Precision is the most important statistic in this case because it has higher cost on false negatives which means recall is prioritized. So when it says the news is real, it better be real.

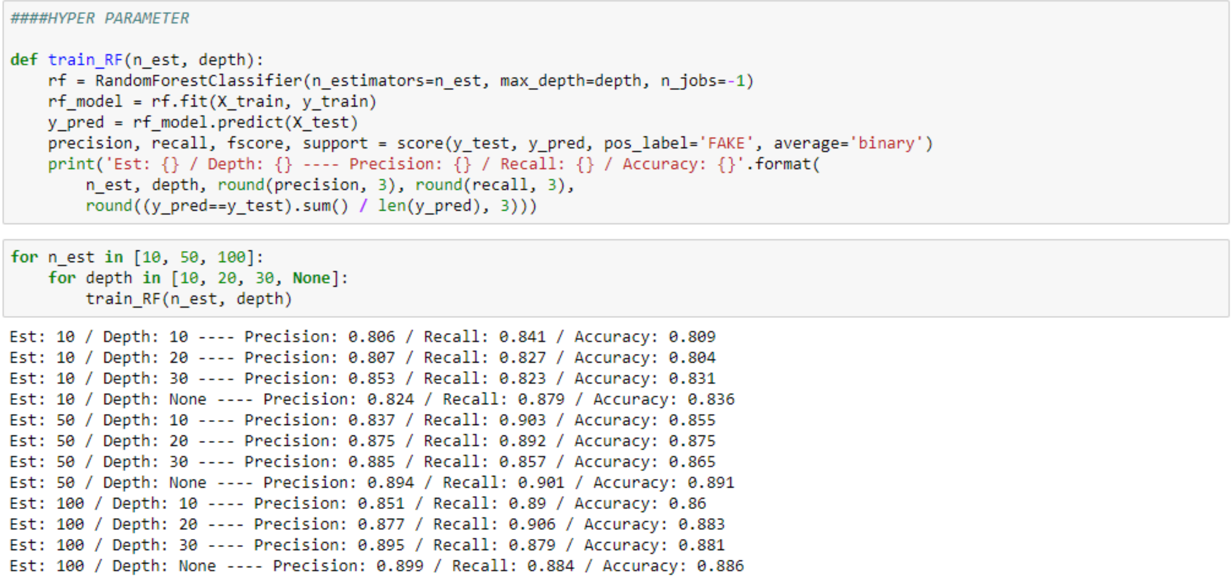


```
print('Precision: {} / Recall: {} / Accuracy: {}'.format(round(precision, 3),
                                                         round(recall, 3),
                                                         round((y_pred==y_test).sum() / len(y_pred),3)))
```

```
Precision: 0.883 / Recall: 0.887 / Accuracy: 0.878
```

Code:5

5. Moreover, hyper parameters were added and a grid for different number of estimators and various depths is created. This gives us an idea to change the parameters to get a good accuracy, precision and recall. We can then select the most appropriate model based on the various hyper-parameters. But in this case the earlier model with recall of 88.7% and accuracy of 87.8% is the best model.



```
####HYPER PARAMETER

def train_RF(n_est, depth):
    rf = RandomForestClassifier(n_estimators=n_est, max_depth=depth, n_jobs=-1)
    rf_model = rf.fit(X_train, y_train)
    y_pred = rf_model.predict(X_test)
    precision, recall, fscore, support = score(y_test, y_pred, pos_label='FAKE', average='binary')
    print('Est: {} / Depth: {} ---- Precision: {} / Recall: {} / Accuracy: {}'.format(
        n_est, depth, round(precision, 3), round(recall, 3),
        round((y_pred==y_test).sum() / len(y_pred), 3)))
```

```
for n_est in [10, 50, 100]:
    for depth in [10, 20, 30, None]:
        train_RF(n_est, depth)
```

```
Est: 10 / Depth: 10 ---- Precision: 0.806 / Recall: 0.841 / Accuracy: 0.809
Est: 10 / Depth: 20 ---- Precision: 0.807 / Recall: 0.827 / Accuracy: 0.804
Est: 10 / Depth: 30 ---- Precision: 0.853 / Recall: 0.823 / Accuracy: 0.831
Est: 10 / Depth: None ---- Precision: 0.824 / Recall: 0.879 / Accuracy: 0.836
Est: 50 / Depth: 10 ---- Precision: 0.837 / Recall: 0.903 / Accuracy: 0.855
Est: 50 / Depth: 20 ---- Precision: 0.875 / Recall: 0.892 / Accuracy: 0.875
Est: 50 / Depth: 30 ---- Precision: 0.885 / Recall: 0.857 / Accuracy: 0.865
Est: 50 / Depth: None ---- Precision: 0.894 / Recall: 0.901 / Accuracy: 0.891
Est: 100 / Depth: 10 ---- Precision: 0.851 / Recall: 0.89 / Accuracy: 0.86
Est: 100 / Depth: 20 ---- Precision: 0.877 / Recall: 0.906 / Accuracy: 0.883
Est: 100 / Depth: 30 ---- Precision: 0.895 / Recall: 0.879 / Accuracy: 0.881
Est: 100 / Depth: None ---- Precision: 0.899 / Recall: 0.884 / Accuracy: 0.886
```

Code:6

Conclusion

Social media is increasing in popularity and is a major source of information to most people. But when the information is fake, it messes up with reality. The whole point of this project was to minimize the spread of misinformation by classifying real and fake news. To summarize, a machine learning model was created which analyzes news based on the way it was written and tells whether it is real or fake.

References

1. Kai Shu, Suhang Wang, Amy Silva- Fake News Detection: A Data Mining Perspective, http://www.kdd.org/exploration_files/19-1-Article2.pdf
2. Mounika Kondamudi, Oklahoma State University- Classifying and Predicting Spam Messages Using Text Mining in SAS® Enterprise Miner https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2650-2018.pdf
3. Goutam Chakraborty, Murali Pagolu, Satish Garla, Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS® by SAS Institute Inc 2014, Accessed 20 Mar 2017.
4. Getting Started with SAS® Text Miner 13.2. Cary, NC: SAS Institute Inc., Accessed 20 Mar 2017.