



**KALINGA INSTITUTE OF
INDUSTRIAL TECHNOLOGY (KIIT)**

Deemed to be University U/S 3 of UGC Act, 1956

Neural Network and machine learning
Report On

Fake news Classification using NLP

Group No: 9

Members

Anshul Kumar 1907010

Saurabh Singh 1907042

Shivam Singh 1907046

Yash Gautam 1907060

Statement

People have the right to express their opinions and thoughts to each other, it's their right as human beings. But how does one tell the difference whether that opinion is fake or genuine. There are people who believe in discovering the truth for themselves, and then there are people who carry on with the fake information believing it and spreading the information with no regard to its authenticity.

Fake news is not something that gained leverage after the expansion of social media and the internet. It has its roots buried deep down before the idea of internet struck the minds of its fathers **Vinton Cerf** and **Bob Kahn**.

The biggest fake news or misinformation of the 2nd Century was the **Geocentric Model**, founded by **Ptolemy of Alexandria**. According to him and his model, all the planets, stars and even the sun, revolve around the earth. Concisely, it assumed the earth to be the center of the solar system.

That was the era when visual evidence was everything compared to legal written evidence. Many Astronomers came up with evidences but they all were silenced until **Copernicus came up with Heliocentric model**. We won't go in the detail on how he did so but the catholic astronomers were so furious that they assassinated him publicly.

However there was no assassination, we gave you a fake information. You would have believed it if we wouldn't have told you the truth. So this is how a fake news operates.

In the 21st Century, there are multiple streams for fake news to spread, Social media is the most preferred option if you're looking for fake news. Such news or informations have no good purpose but to cause riots and disbelief in our society. There are people who actually stand up for such news and keep on arguing even after the truth is revealed just because they think admitting to such thing would hurt their confidence and ego. If we compare the 21st century with the 2nd century, you would notice that only technology has changed, the people are still the same with most of them playing the role of a fake Copernicus.

Overview

In this project, we have decided to use machine learning algorithms and libraries such as NLP, NLTK, Sci-kit Learn, Pandas, etc to process a database of approximately **20800** news articles. The database can be accessed on Kaggle.com by the link provided below:

<https://www.kaggle.com/c/fake-news>

The main purpose of this project is to reduce the time an individual or a large group of people would take to go through those articles in the database classifying whether which article out of them is fake and which is real.

However the database already has the articles labeled as 0's and 1's which means the articles labeled 0 are Real and those labeled 1 are fake. But still, how would anyone remember if article 10,000 is fake or real. That where the use of machine learning takes place.

On the upcoming page, we'll explain everything in detail on how we managed to create a system that tells just by entering the article no. whether that article is fake or legit.

The only limitation of this project is that this project is limited to the database provided to it and it cannot justify the authenticity of an information outside the dataset fed to it. Most of the articles inside the dataset are political in nature because political news are the hardest to differentiate. The dataset contains political news from around the world.

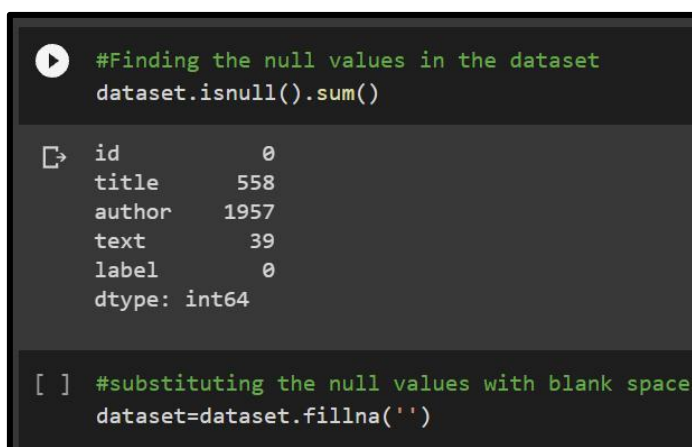
The Python Libraries used in this project are listed below:

1. Numpy
2. Pandas
3. RE (Regular Expression)
4. NLTK (Natural Language Tool Kit)
5. Sci-Kit Learn
6. Logistic Regression

Theory

The first process in this project was to import all the required libraries for classification of words inside the dataset since it contains only text formats. Also, inside the dataset, many **NaN** values or **empty cells** were present that may have caused errors when predicting the accuracy score of the model. So the first step was to clean the data of the empty values. To check the dataset for the Null values, we used '**IsNull**' Command and to clear the NaN values, we used '**dropna**' command.

As you can see, the title feature contained **558 null values**, the author



```
#Finding the null values in the dataset
dataset.isnull().sum()

id          0
title      558
author     1957
text        39
label       0
dtype: int64

[ ] #substituting the null values with blank spaces
dataset=dataset.fillna('')
```

feature contained **1957 empty values** and similarly the NaN values of all features was shown. The next step was to allocate blank spaces to the NaN values so that they are skipped during the text processing and the data gives a fast result. So we used '**fillna**'

command to assign blank spaces to all the rows that contains NaN values.

The next essential step was to determine which factor to consider for classifying the articles as fake or true. The below image contains all the features available inside the database.

Inside that database, the most number of words were available inside the 'text' column. To consider our classification based on full details of the article given inside the 'text' feature, we would have suffered a huge time loss as well as the need of high processing powers, powers that our systems are not able to handle or provide properly. The image below will show the amount of words inside the text feature.

D

text

House Dem Aide: We Didn't Even See Comey's Letter Until Jason Chaffetz Tweeted It By Darrell Lucas on October 30, 2016 Subscribe Jason Chaffetz on the stump in American Fork, Utah (image courtesy Michael Jolley, available under a Creative Commons-BY license)

With apologies to Keith Olbermann, there is no doubt who the Worst Person in The World is this week—FBI Director James Comey. But according to a House Democratic aide, it looks like we also know who the second-worst person is as well. It turns out that when Comey sent his now-infamous letter announcing that the FBI was looking into emails that may be related to Hillary Clinton's email server, the ranking Democrats on the relevant committees didn't hear about it from Comey. They found out via a tweet from one of the Republican committee chairmen.

As we now know, Comey notified the Republican chairmen and Democratic ranking members of the House Intelligence, Judiciary, and Oversight committees that his agency was reviewing emails it had recently discovered in order to see if they contained classified information. Not long after this letter went out, Oversight Committee Chairman Jason Chaffetz set the political world ablaze with this tweet. FBI Dir just informed me, "The FBI has learned of the existence of emails that appear to be pertinent to the investigation." Case reopened

— Jason Chaffetz (@jasoninthehouse) October 28, 2016

Of course, we now know that this was not the case . Comey was actually saying that it was reviewing the emails in light of "an unrelated case"—which we now know to be Anthony Weiner's sexting with a teenager. But apparently such little things as facts didn't matter to Chaffetz. The Utah Republican had already vowed to initiate a raft of investigations if Hillary wins—at least two years' worth, and possibly an entire term's worth of them. Apparently Chaffetz thought the FBI was already doing his work for him—resulting in a tweet that briefly roiled the nation before cooler heads realized it was a dud.

But according to a senior House Democratic aide, misreading that letter may have been the least of Chaffetz' sins. That aide told Shareblue that his boss and other Democrats didn't even know about Comey's letter at the time—and only found out when they checked Twitter. "Democratic Ranking Members on the relevant committees didn't receive

It would have been very difficult even for a machine to recognize such huge amount of words and classify them into further more features. So, to handle this situation we came up with the idea of **concatenating** to features inside the database, the features being: **'Author'** and **'Title'**.

```
[ ] #merging author and title features
    dataset['Combined']=dataset['author']+' '+dataset['title']

▶ print(dataset['Combined'])

0      Darrell Lucas House Dem Aide: We Didn't Even S...
1      Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2      Consortiumnews.com Why the Truth Might Get You...
3      Jessica Purkiss 15 Civilians Killed In Single ...
4      Howard Portnoy Iranian woman jailed for fictio...
...
20795   Jerome Hudson Rapper T.I.: Trump a 'Poster Chi...
20796   Benjamin Hoffman N.F.L. Playoffs: Schedule, Ma...
20797   Michael J. de la Merced and Rachel Abrams Macy...
20798   Alex Ansary NATO, Russia To Hold Parallel Exer...
20799   David Swanson What Keeps the F-35 Alive
Name: Combined, Length: 20800, dtype: object
```

The reason for concatenating these features was to consider each unique identity for each unique article and also the title also had some

brief information about the incident, so the need of analyzing the whole text is not required.

The concatenated data is stored inside another feature created using python command and stored inside '**Combined**' column.

The next step is to separate the 'label' feature from the given dataset to make it independent and to use this feature with respect to the 'combined' feature inside the logistic regression test. We separate these two features by creating two new variables for storing those two features by the name of 'X' and 'Y' respectively.

```
[ ] X=dataset.drop(columns='label', axis=1)
    Y=dataset['label']

▶ print(X)
  print(Y)
```

	id	...	Combined
0	0	...	Darrell Lucas House Dem Aide: We Didn't Even S...
1	1	...	Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2	2	...	Consortiumnews.com Why the Truth Might Get You...
3	3	...	Jessica Purkiss 15 Civilians Killed In Single ...
4	4	...	Howard Portnoy Iranian woman jailed for fictio...
...
20795	20795	...	Jerome Hudson Rapper T.I.: Trump a 'Poster Chi...
20796	20796	...	Benjamin Hoffman N.F.L. Playoffs: Schedule, Ma...
20797	20797	...	Michael J. de la Merced and Rachel Abrams Macy...
20798	20798	...	Alex Ansary NATO, Russia To Hold Parallel Exer...
20799	20799	...	David Swanson What Keeps the F-35 Alive

```
[20800 rows x 5 columns]
0      1
1      0
2      1
3      1
4      1
```

As explained earlier, the main reason behind opting for author and title feature rather than text feature was to eliminate the large amount of processing power required by the system and also to reduce the amount of time taken to fetch the result. If you observe the image above carefully, you'll see that it contains words such "as, to, we, keeps, the,

and", and so on. Such words have no resemblance in text classification and hence are termed as '**Stopwords**' in Machine Learning.

Another feature is the **Stemming** function available through NLTK library. *This function converts a word to its root origin*, for example: Swimming and Swum becomes Swim, Ran and Running becomes Run and flew and flying becomes fly. This function also simplifies the dataset for easier classification by assigning similar words inside the entire dataset into different columns and hence we get more column features for a good accuracy score.

```
[ ] stem=PorterStemmer()

[ ] def stemming(Combined):
    stemmed_content = re.sub('[^a-zA-Z]', ' ', Combined)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [stem.stem(word) for word in stemmed_content if not word in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
```

Inside the def stemming(combined) function, we used the **Regular expression function for stemming similar words together for conversion** to their origins. Then we substituted everything inside the concatenated feature except those words. The next step was to convert everything to lower case and then split those words in a list.

Vectorization and Logistic Regression Testing and Training:

After accomplishing most of the data pre-processing task, the next hurdle was how do we train our model to calculate the accuracy score for conclusion. This is the part where *Tfidf Vectorizer played a very essential role*. Since computer cannot understand words, it can definitely understand numbers, the **main use of Tfidf Vectorizer is to assign different number to different collection of texts inside the Corpus**. The full form of Tfidf means Term frequency and Inverse document frequency. By executing this command, our model becomes suitable for training and testing using Logistic Regression.

Splitting the Dataset and applying Logistic Regression

To calculate the accuracy of our data, we need to split them in test and train model. For that we used sklearn.model_selection feature.

Splitting the dataset into training and test data

```
[ ] X_train,X_test,Y_train,Y_test= train_test_split(X,Y, test_size=0.2, stratify=Y, random_state=2)
```

Training the model using Logistic Regression

```
[ ] model=LogisticRegression()
```

```
[ ] model.fit(X_train,Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
warm_start=False)
```

Evaluation of Accuracy score

1. Training Model:

For calculating the accuracy score, we used model.predict function from sklearn.metrics library. Therefore, the accuracy score of the training model is:

0.98 or 98%, which depicts a very precise accuracy score.

2. Testing Model:

Similarly the accuracy score of the testing data is:

0.97 or 98%, which also represents a good accuracy score.

Evaluation of Accuracy score

```
[ ] # Accuracy score of training data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction, Y_train)
```

```
[ ] print("Accuracy score of training data: ",training_data_accuracy)
```

```
Accuracy score of training data:  0.9865985576923076
```

```
[ ] # Accuracy score of testing data
X_test_prediction=model.predict(X_test)
testing_data_accuracy=accuracy_score(X_test_prediction, Y_test)
```

```
[ ] print("Accuracy score of testing data: ",testing_data_accuracy)
```

```
Accuracy score of testing data:  0.9790865384615385
```


Building a predictive System:

The last step was to implement a system that will give the result whether a the article number input by the user is fake or true. Inside the system the user inputs the article number he wants to search for its authenticity and based on the model we made, the result is shown.

```
Designing a predictive system

▶ News=X_test[5]
  prediction=model.predict(News)
  print(prediction)

  if (prediction==0):
    print("The news is Real.")
  else:
    print("The news is Fake.")

[ ] [1]
    The news is Fake.

[ ] print(Y_test[5])

1
```

Conclusion

By performing this project we understood how to interpret a corpus file for classification using machine learning techniques and calculate its accuracy by using Logistic Regression Method. This project also helped us solidify the basic concepts of Neural Network and Machine Learning. However none of this would have been possible without the constant support of our group members who came up with bright ideas to execute this project.

Therefore , the group project on Fake news Classification using Natural Language processing has been successfully completed .
Thank You.