A Project Report

on

FAKE NEWS DETECTION MODEL

Submitted in partial fulfillment of the

requirement for the award of the degree

of

Bachelor of Technology in Computer Science and Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

Dr. Vipin Rai

Associate Professor

Department of Computer Science and Engineering

Submitted By

18SCSE1010754 – ISHITA SINGH

18SCSE1010747 – ANIKET SHARMA

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"Fake News Detection Model"** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**

submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **DR. VIPIN RAI, Associate Professor**, **Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010754 – ISHITA SINGH

18SCSE1010747 – ANIKET SHARMA

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Dr.Vipin Rai, Associate Professor)

CERTIFICATE

The Final Project Viva-Voce examination of **18SCSE1010754 – ISHITA SINGH, 18SCSE1010747 – ANIKET SHARMA** has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021.

Place: Galgotias University, Greater Noida, Uttar Pradesh.

ACKNOWLEDGEMENT

The feeling if gratitude when we expressed a holy acknowledgement and it's with deep sense of gratitude that we acknowledge the able guidance.

We express our grateful thanks to Dr. Vipin Rai, Associate professor, Department of Computer Science and Engineering, Galgotias University for providing us an opportunity for the research report on "**Fake News Detection Model**" and for his keen interest and the encouragement, which was required for the fulfilment of our capstone project report. We would also like to thank him for giving us valuable guidance at all level, help and suggestion, which prove to be valuable for preparation of the report.

Finally, I would also like to thank all our friends for their cooperation and interest, which was necessary for completing our project report.

Date: 18/12/2021

Ishita Singh, Aniket Sharma.

School of Computing Science & Engineering,

Galgotias University,

Greater Noida, Uttar Pradesh

ABSTRACT

Fake news and hoaxes are there since before the arrival of the web. The broadly accepted definition of web fake or we can say wrong news is: spurious articles purposefully constructed to deceive readers. On the one hand, it's very cheap, rapid approach, and quick circulation of data lead people to hunt out and consume news from social media. On the opposite hand, it enables the wide spread of "fake news", i.e., inferiority news with intentionally false information. The extensive spread of faux news has the potential for very negative impacts on individuals and society. Firstly, fake news is purposely written to deceive readers to believe wrong information, which makes it tough to detect brace news content; therefore, we'd wish to comprise supplementary information, such as user social engagements on social media, to assist make a determination. Secondly, utilizing this supplementary information is challenging in and of it as users' social interaction with fake news produce data that's very large in number, not in structural manner, noisy, and are never complete. The purpose of the work is to return up with an answer which will be utilized by users to detect and filter sites containing false and misleading information. We use simple and punctiliously selected features of the title and post to accurately identify fake posts. Throughout the paper we have shown the research related zones, open complications, and future directions for the research on fake news detection on social media. We wind up the report by elevate perception about examine and chance for businesses that are presently on the hunt to assist spontaneously detecting fake news by providing web services.

Keywords: Machine Learning, Python, TF-IDF Vectorizer, Passive Aggressive Classifier

Table of Contents

List of Figures

List of Charts

1. INTRODUCTION

1.1 The Impact of Fake News

The internet is mainly driven by advertising. Websites with sensational headlines are very fashionable, which results in advertising companies capitalizing on the high traffic to the location. The question remains how misinformation would then influence the general public. The spreading of misinformation can cause confusion and unnecessary stress among the general public. We can term this as disinformation which is digital in nature. So, fake news is generally created intentionally to harm and deceive the public. Disinformation has the potential to cause issues, within minutes, for many people. Disinformation has been known to disrupt election processes, create unease, disputes and hostility among the general public.

1.2 Fake News and Social Media

These days, the internet has become a vital part of our daily lives. It was reported in 2017 that Facebook was the most important social media platform, hosting more 1.9 million user’s world-wide. From all the social media platforms, the biggest impact in spreading the fake news is from Facebook alone, even it have the highest impact in spreading wrong information. It was reported that 44% of worldwide users get their news from Facebook. 23% of Facebook users have specified that they have shared wrong info or we can call it fake news, either intentionally or not. The roll out of false information is charged by platforms which are more social and it's occurring at distress speed.

1.3 Research Problem:

The project cares with identifying an answer that would be wont to detect and filter sites containing fake news for purposes of helping users to avoid being lured by click baits. It is imperative that such solutions are identified as they're going to convince be useful to both readers and tech companies involved within the issue.

1.4 Proposed Solution:

The proposed solution to the issue concerned with fake news includes the use of a model that can identify and remove fake sites from the results provided to a user by a search engine or a social media news feed.

This advanced python project of detecting fake news deals with fake and real news. Using sklearn, we build a TfidfVectorizer on our dataset. Then, we initialize a PassiveAggressive Classifier and fit the model. In the end, the accuracy score and the confusion matrix tell us how well our model fares.

Once operational, the tool will use various techniques including those related to the syntactic features of a link to determine whether the same should be included as part of the search results. We conclude the paper by raising awareness about concerns and opportunities for businesses that are currently on the quest to help automatically detecting fake news by providing web services, but who will most certainly, on the long term, profit from their massive usage. We also discuss related research areas, open problems, and future research directions for fake news detection on social media.

2. REQURIMENTS

2.1 Language: Python 3.6.9

When it comes to data science, machine learning is one of the significant elements used to maximize value from data. With Python as the data science tool, exploring the basics of machine learning becomes easy and effective. In a nutshell, machine learning is more about statistics, mathematical optimization, and probability. It has become the most preferred machine learning tool in the way it allows aspirants to ‘do math’ easily.

2.2 Tool Used: Jupyter notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

2.3 Python Package: pip 20.2.3

(Note: if you have Python version 3.4 or layer, pip is included by default).

The package contains all the files you need for a module.

3. ANALYSIS

Our project can be implemented by using python language. The tools we need are available online. And the project is also managed that there will less expenditure and that can handle. This is android app is very easy to use i.e. user- friendly.

The objective of feasibility study is to determine whether or not the proposed system is feasible.

3.1 Feasible Solution:

1. Data Gathering
2. Data Extraction: web scraping
3. Data formatting: cleaning
4. Identifying Patterns: trends
5. Showing Correlations: Data Visualization
6. Training Data: Data modeling, using Machine Learning Algorithms
7. Predicting Outcomes: using Machine Learning Algorithms

3.2 Two Phases:

Phase 1: Research:

1. Choose a topic
2. Define the task and prepare a working theory
3. Brainstorm all possible sources for appropriateness for the project
4. Prepare a project write-up

Phase 2: Implementation:

1. Data Gathering
2. Data Extraction: web scraping
3. Data formatting: cleaning
4. Identifying Patterns: trends
5. Showing Correlations: Data Visualization
6. Training Data: Data modeling, using Machine Learning Algorithms
7. Predicting Outcomes: using Machine Learning Algorithms.

4. OBJECTIVE

Figure 1 : Scope and Objective

Detection of fake news online is important in today's society as fresh news content is rapidly being produced as a result of the abundance of technology that is present. In the world of false news, there are seven main categories and within each category, the piece of fake news content can be visual- and/or linguistic-based. In order to detect fake news, both linguistic and non-linguistic cues can be analyzed using several methods. The detection of fake news can also be achieved through predictive modelling based methods. One type would be the logisticregression model. In this model, positive coefficients increase the probability of truth while negative ones increase the probability of deception.

The biggest merit of the work is to come up with a solution that can be utilized by users to detect and filter out sites containing false and misleading information. The aim of this report is to walk through the process of creating a machine learning model using python

and NLP in order to successfully detect fake news.This advanced python project of detecting fake news deals with fake and real news. Using sklearn, we build a TfidfVectorizer on our dataset. In the end, the accuracy score and the confusion matrix tell us how well our model fares.

5. PROBLEM FORMULATION

The work cares with identifying an answer that would be able to spot and sieve articles comprising fake news for purposes of serving users to evade being decoyed by click baits. It is authoritative that such solutions are acknowledged as they're going to convince be useful to both readers and tech corporations intricate within the subject. Throughout the paper we have shown the research related zones, open complications, and future directions for the research on fake news detection on social media.

One of the most potent and exciting technologies which come in the real-world application is machine learning. Machine Learning techniques drastically change the computer application, and it simulates human-decision making using neural networks.

Machine Learning is a domain of Artificial intelligence, where we bring AI into the equation by learning the input data. The process of making machines learn through the provided data is nothing but machine learning. Devices that are trained with a massive volume of data perform the task more accurately, and it can predict the result more precisely.

In general, an algorithm uses some mathematics and logic and takes some input to produce the output, but an AI algorithm takes a combination of both inputs and outputs in order to learn and train the data and produce beneficial outputs. Algorithms in each group perform the same task of forecasting outputs on the given unknown inputs. However, here the data is the backbone when it comes to selecting the right and correct algorithm.

6. DESIGN

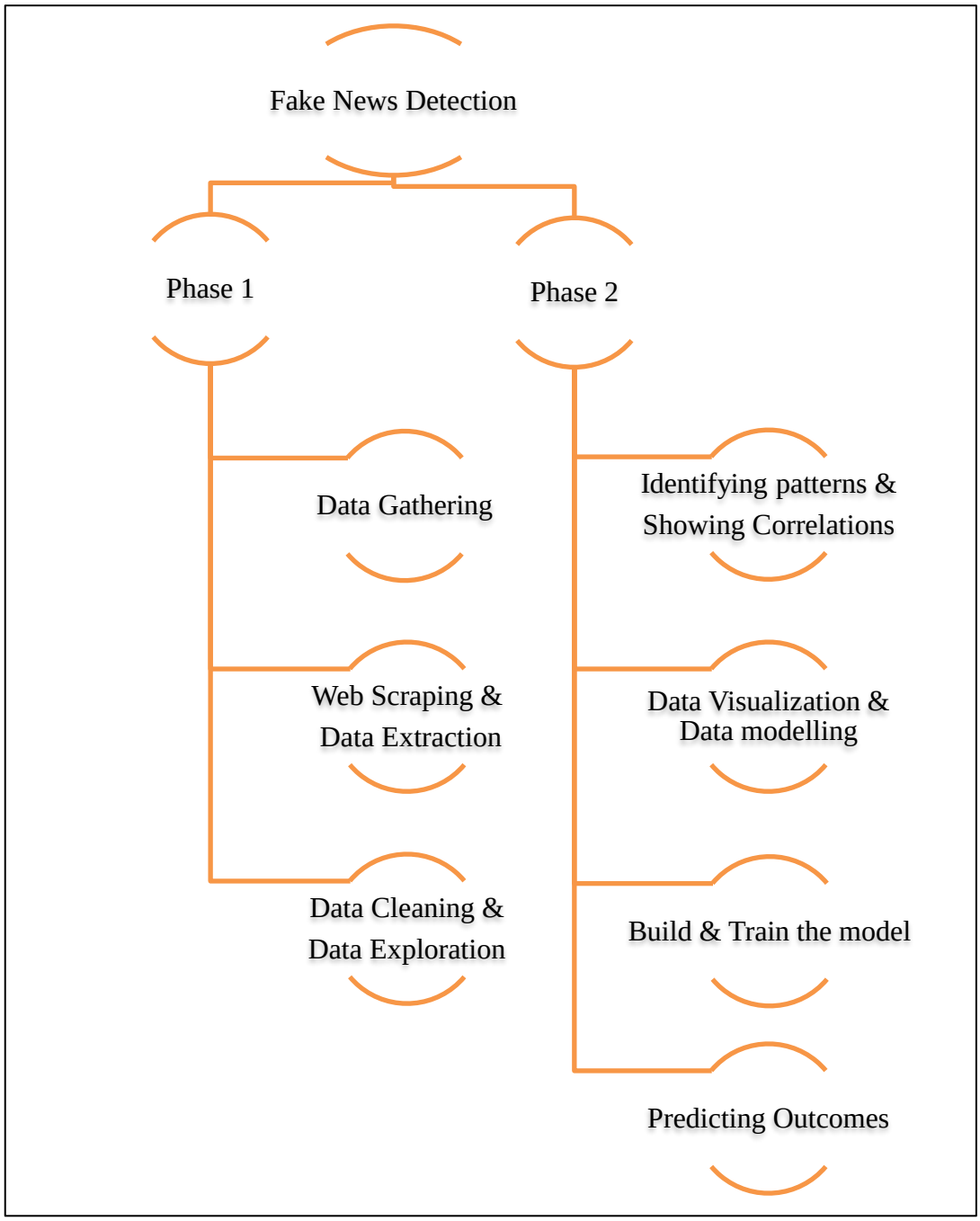


Figure 2: Design

7. IMPLEMENTATION

We have used various modules to get our model work better. We will be showing the implementation of all of some modules (as the explanation of two modules has being asked).

We have used numpy, pandas, matlpotlib, seaborn, sklearn, nltk, wordcloud etc.

We will be explaining about: Pandas and sklearn

7.1 About the data:

We have collected the data from Kaggle. It is very feasible to download the data in zip file from Kaggle. On downloading the dataset, we will get two csv files namely, true and fake. The "True.csv" dataset contains 21,418 rows while "Fake.csv" dataset contains 23,503 rows. The true dataset contains all the articles which are considered as "Real news". The fake dataset contains all those articles which are considered as "Fake news". We will merge both the dataset in the later part of our analysis.

7.2. Pandas:

Pandas are a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

1. **Loading Dataset using pandas:** we use .read_csv() method to load data in our notebook.

```
#Loading data, Dataframe1=fake, Dataframe2=true
fake = pd.read_csv("Fake.csv")
true = pd.read_csv("True.csv")
```

Figure 3: Load Data

2. **Concatenating and Shuffling both the Dataset:** using shuffle() and .concat() method.

```
#concatenating the data frames
data = pd.concat([fake, true]).reset_index(drop = True)
```

Figure 4: Merging DataFrames

```
#shuffle the data to prevent bias
from sklearn.utils import shuffle
data = shuffle(data)
data = data.reset_index(drop=True)
```

Figure 5: Shuffling DataFrame

3. **Viewing the Dataset:** df.head() method gives the first five rows of the dataset.

```
#cchecking the first five rows of the data set....
data.head()
```

	title	text	subject	date	target
0	Yemen's Saleh says ready for 'new page' with S...	ADEN (Reuters) - Former Yemeni President Ali A...	worldnews	December 2, 2017	true
1	Former soccer star Kaladze runs for mayor in G...	TBILISI (Reuters) - Kakha Kaladze climbed to t...	worldnews	October 19, 2017	true
2	Bosnian pensioners stage street protests for p...	SARAJEVO (Reuters) - Thousands of pensioners f...	worldnews	October 25, 2017	true
3	India, China need to do more to avoid border d...	NEW DELHI (Reuters) - Indian Prime Minister Na...	worldnews	September 5, 2017	true
4	Prospects for House vote on gun control measur...	WASHINGTON (Reuters) - Prospects dimmed on Mon...	politicsNews	July 12, 2016	true

Figure 6: Viewing dataset

4. **Checking the shape, missing and duplicate values of the dataset:**

```
In [6]: data.shape
Out[6]: (44898, 5)
```

Figure 7: shape of Dataframe

```
#checking the missing values in dataset
data.isnull().sum()
```

```
title      0
text       0
subject    0
date       0
target     0
dtype: int64
```

Figure 8: Checking for any Null Value

```
#checking for the duplicate data
sum(data.duplicated())
```

```
209
```

Figure 9: Checking Duplicate Entries

5. **Dropping and Rechecking the duplicated values:** by using .drop_duplicates() method.

```
#droping the duplicate values by using drop
# and then re-checking the duplicated values and shape of dataset
data.drop_duplicates(inplace=True)
print(sum(data.duplicated()))
print(data.shape)
```

```
0
(44689, 5)
```

Figure 10: Drop Duplicates

6. **Dropping attributes which we will not use in the analysis:**

```
#dropping 'date' and 'title' attributes
data.drop(["date"],axis=1,inplace=True)
data.drop(["title"],axis=1,inplace=True)
```

Figure 11: Drop Irrelevant Columns

```
#Converting the text to lowercase:
data['text'] = data['text'].apply(lambda x: x.lower())
```

Figure 12: Converting Text to LowerCase

7. **Other cleaning of the data:**

We will remove the punctuations and stopwords in the dataset. It makes the data noisy and later creates problems in the analysis. So better to get rid of them!

```
#Remove punctuation:
import string
def punctuation_removal(text):
    all_list = [char for char in text if char not in string.punctuation]
    clean_str = ''.join(all_list)
    return clean_str
data['text'] = data['text'].apply(punctuation_removal)
```

```
#Remove stopwords:
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split()
                                                      if word not in (stop)]))
```

Figure 13: Removal of Punctuation and StopWords

8. **Checking the final dataset:** we can view the first five rows of the dataset by using .head()

```
#check
data.head()
```

	text	subject	target
0	aden reuters former yemeni president ali abdul...	worldnews	true
1	tbilisi reuters kakha kaladze climbed top worl...	worldnews	true
2	sarajevo reuters thousands pensioners across b...	worldnews	true
3	new delhi reuters indian prime minister narend...	worldnews	true
4	washington reuters prospects dimmed monday us ...	politicsNews	true

Figure 14: Final Dataset

9. Data visualization:

Dividing data on basis of subject

```
In [21]: #articles per subject: here we are grouping data by subject and text
         print(data.groupby(['subject'])['text'].count())
         data.groupby(['subject'])['text'].count().plot(kind="bar")
         plt.show()
```

```
subject
Government News     1570
Middle-east          778
News                9050
US_News              783
left-news           4459
politics            6838
politicsNews       11220
worldnews           9991
Name: text, dtype: int64
```

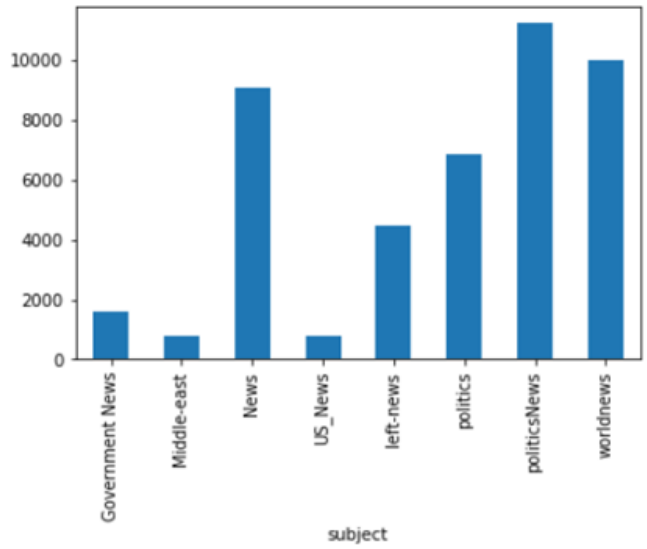


Figure 15: Visualization based on Subject

Fake News vs True News

```
In [22]: #fake or real article: here we are grouping data by target and text
         print(data.groupby(['target'])['text'].count())
         data.groupby(['target'])['text'].count().plot(kind="bar")
         plt.show()
```

```
target
fake    23478
true    21211
Name: text, dtype: int64
```

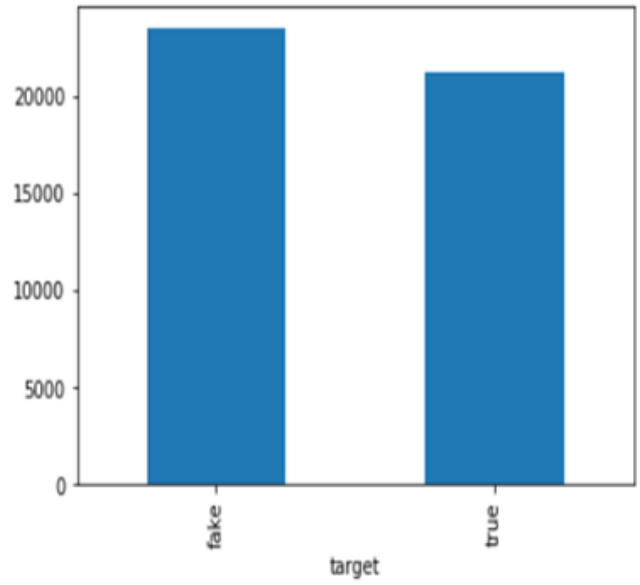


Figure 16: True vs. Fake Data

10. Word Cloud:

One of the best ways to visualize *Textual Data*. Word Cloud is a data visualization technique which is used to visualize text on the basis of frequency or occurrence of words in it.

Word Cloud for fake news:

```
In [23]: #Word cloud for fake news
         from wordcloud import WordCloud

         fake_data = data[data["target"] == "fake"] #selecting the "fake" target
         all_words = ' '.join([text for text in fake_data.text]) #combining all the words
         # Create and generate a word cloud image:
         wordcloud = WordCloud(width= 800, height= 500,max_font_size = 110, background_co
                               collocations = False).generate(all_words)

         plt.figure(figsize=(10,7))
         plt.imshow(wordcloud, interpolation='bilinear')
         plt.axis("off")
         plt.show()
```

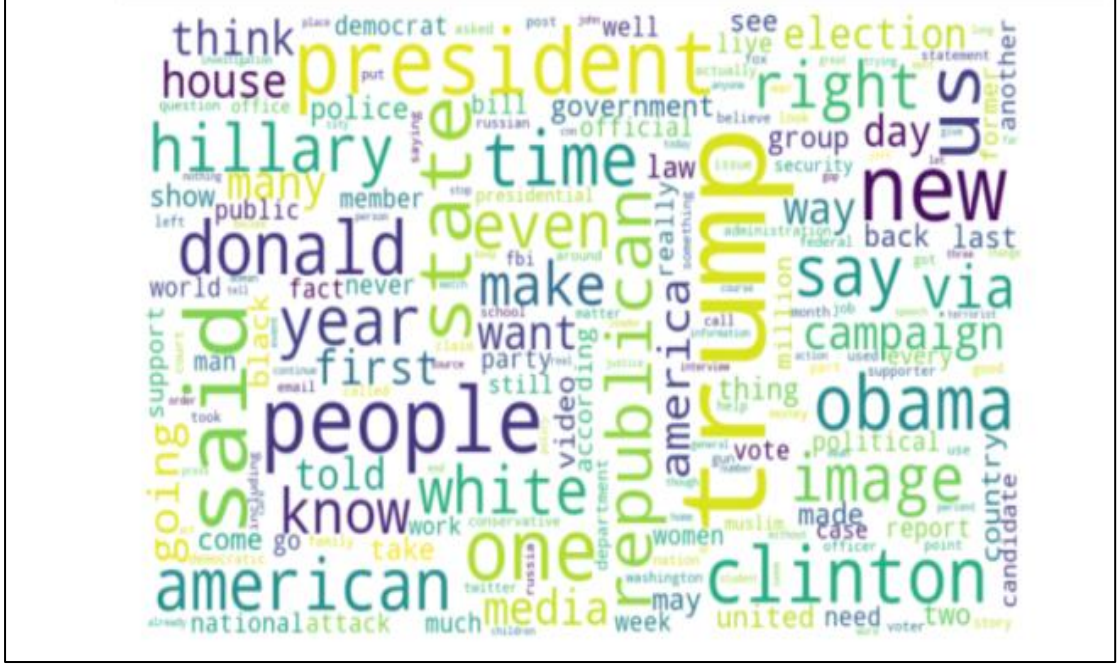


Figure 17 : Word Cloud of Fake Data

Word Cloud for real news:

```
In [24]: #Word cloud for real news
from wordcloud import WordCloud

real_data = data[data["target"] == "true"] #selecting the "fake" target
all_words = ' '.join([text for text in real_data.text]) #combining all the words
# Create and generate a word cloud image:
wordcloud = WordCloud(width= 800, height= 500,max_font_size = 110, background_co
                      collocations = False).generate(all_words)

plt.figure(figsize=(10,7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

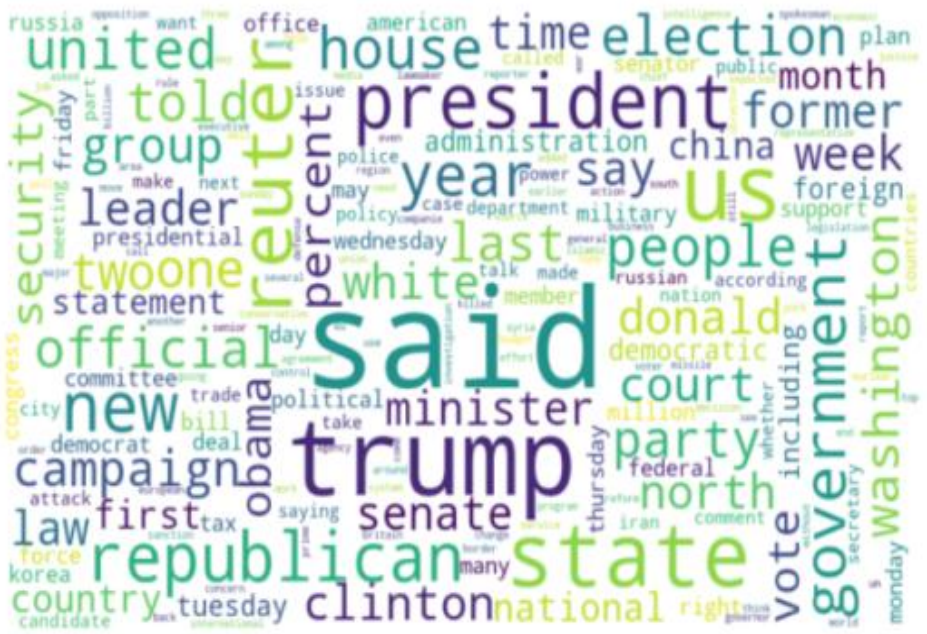


Figure 18: Word Cloud of True Data

11. Checking which words are the most frequent in the fake and real news respectively.

Most Frequent words in fake news:

```
In [26]: #Most frequent words in fake news
         counter(data[data["target"] == "fake"], "text", 20)
```

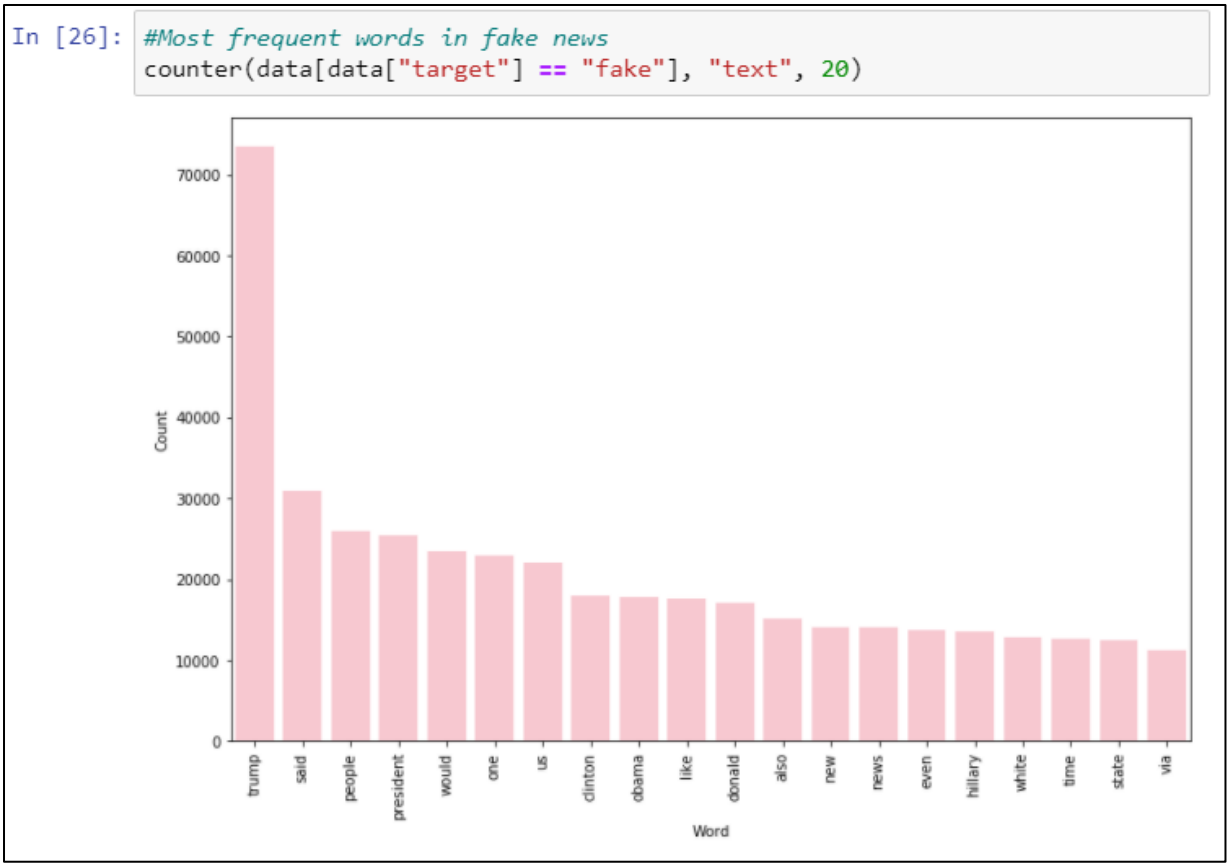


Figure19: Frequent Words in Fake Data

Most Frequent words in fake news:

```
In [27]: # Most frequent words in real news
         counter(data[data["target"] == "true"], "text", 20)
```

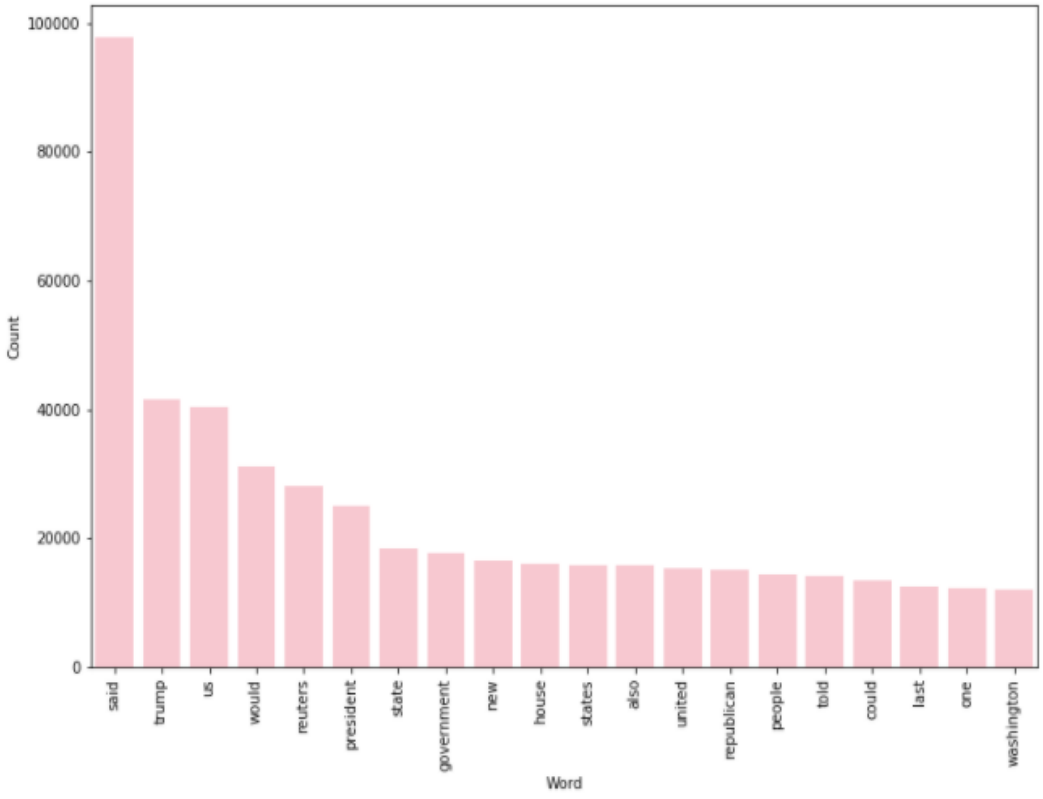


Figure 20: Frequent Words in True Data

7.9. Sklearn:

Scikit-learn is a free library in python used for machine learning. It comprise of some of the most important algorithms like support vector machine, random forests, and k-neighbours. And it also has a functionality to numerical python and other scientific libraries like NumPy and SciPy.

1. **Modelling:** using the function plot_confusion_matrix() to plot the confusion matrix of the models.

```
from sklearn import metrics
import itertools

def plot_confusion_matrix(cm, classes,normalize=False,
                          title='Confusion matrix',cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Figure 21 : Data Modeling

2. **Preparing the Data:** Split the data to train and test the model, using train_test_split() method, and giving the test_size, random_state in the parameters.

```
# Split the data
X_train,X_test,y_train,y_test = train_test_split(data['text'], data.target, test_size=0.2, random_state=42)
```

Figure 22 : Train - Test Split

3. **Logistic regression:**

Logistic Regression are used to classify data into categories like True or False, 1 or 0 etc. Extending Logistic Model to classify numerous classes of events such as whether the provided article is comprising of those words which are often used in Fake or Real News.

- It is used to define a relationship between one binary and one normal or more variables.
- Logistic Regression uses a complex cost function as compared to Linear Regression.
- It is known as 'Sigmoid Function.' The hypothesis of Logistic Regression makes sure that the limit of Sigmoid remains in between 0 to 1.

$$0 \leq h_\theta(x) \leq 1$$

Sigmoid Function:

- It is used to map the predicted values to the probability.
- Sigmoid Function maps any real value into the range of 0 to 1

$$S(x) = \frac{1}{1 + e^{-x}}$$

Hypothesis Representation:

Hypothesis equation of Logistic Regression is little bit different from that of Linear Regression

$$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$$

$Z = \beta_0 + \beta_1 X$

$h\Theta(x) = \text{sigmoid}(Z)$

i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Cost Function:

In Logistic Regression we use Binary Loss Entropy.

$$J(\theta) = -\frac{1}{m}\sum \left[y^{(i)} \log(h\theta(x(i))) + \left(1 - y^{(i)}\right) \log(1 - h\theta(x(i))) \right]$$

Binary Loss Entropy make sure that it would end up being a convex function and hence minimize the cost value.

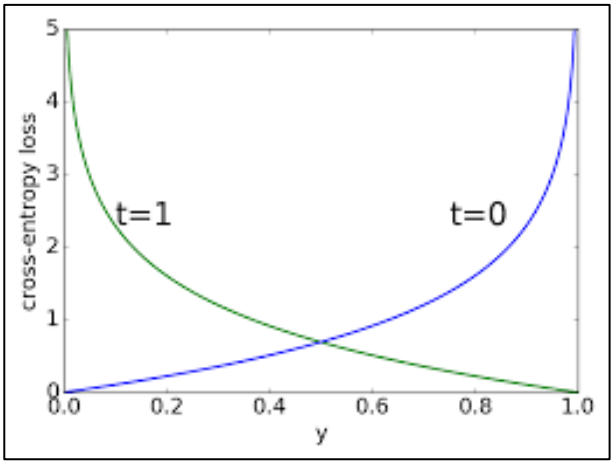


Figure 23: Algo of logistic reg

Gradient Descent:

Minimizing cost/loss with the help of cost function can be done using Gradient Descent.

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

That derivative term is called “Descent.” It is responsible for global minima.

And after putting values of Descent it looks like this :

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

We need to update theta up to a certain number of times so that it gets close to the ideal one.

Logistic regression

In [30]:
```
# Vectorizing and applying TF-IDF
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LogisticRegression())])

# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 98.81%
```

In [31]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```

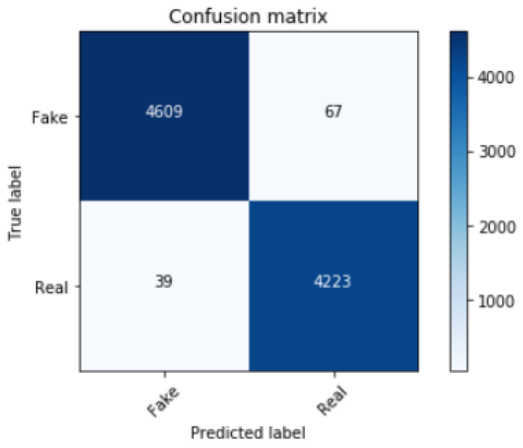


Figure 24: Logistic Regression

Decision Tree Classifier: It is an extrapolative modelling approach used in statistics, data mining and machine learning. It usages a decision tree to go from observations about an entry to inferences about the item's target value. Decision Trees are useful in Decision Analyis as it can be implemented to represent decision visuallly and explicitly and to infered decision. Decision Tree are mostly used in classification problems. Although it can also be used in regression problems as well. It is basically a tree like structure where nodes represent the features

of a dataset and branches shows the decision rule. The leaf node in decision tree is basically the outcome

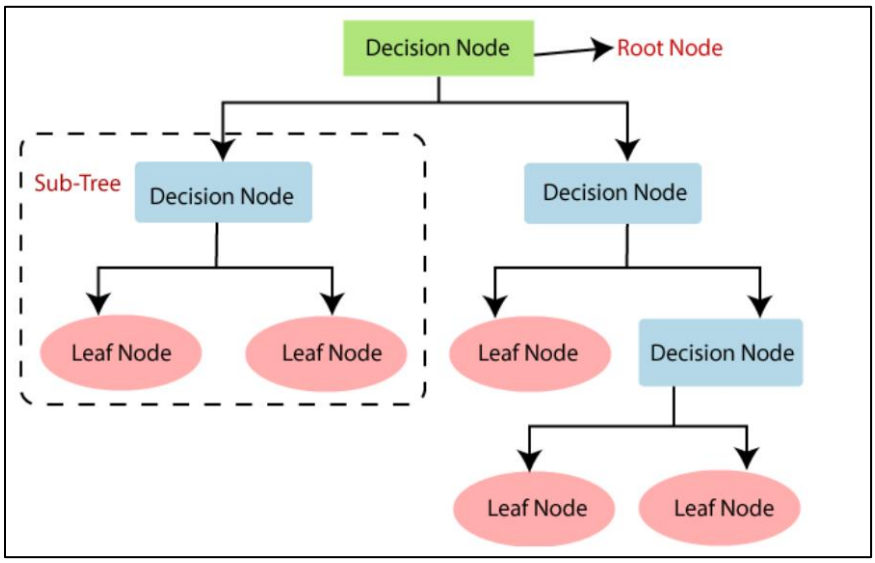


Figure 26: Algorithm of Decision Tree

1. Begin the decision tree with the root node that holds complete dataset.
2. Find out the best feature to split around using Attribute Selection Measure
3. Now, divide the root node into possible values of best attribute
4. Determine the best feature and create the decision tree node
5. Repeat the same process again and again, until it is impossible for you to classify the node further.

Decision Tree Classifier

In [32]:
```
from sklearn.tree import DecisionTreeClassifier

# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', DecisionTreeClassifier(criterion= 'entropy',
                                                  max_depth = 20,
                                                  splitter='best',
                                                  random_state=42))])
# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 99.5%
```

In [33]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```

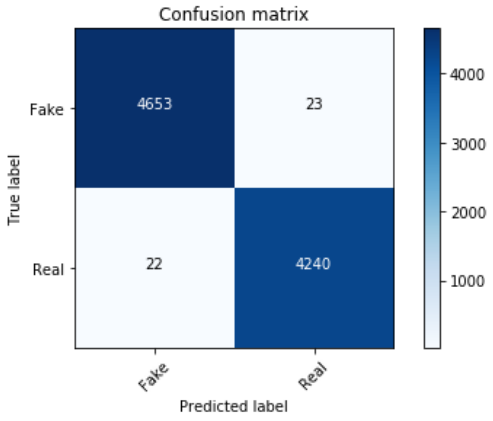


Figure 27: Decision Tree

Random Forest Classifier: It is based on ensemble tree-based learning algorithm. The Random Forest is made up of many different Decision Trees from a randomly selected subset of the training set. It collects the votes from different Decision Trees to conclude the final class of the test object. It can handle many of input variables without variable deletion. It produces a highly accurate classifier. Thus, it is one of the most accurate learning algorithms available.

As its name suggest, it is collection of different Decision Trees. The difference is that the process of finding the root node & splitting around the best feature node will run automatically here.

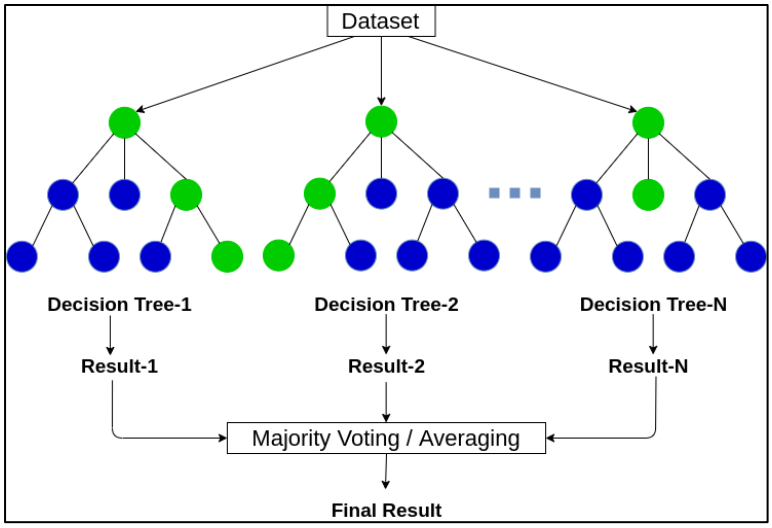


Figure 28: Algorithm of Random Forest

1. First step is to select 'K' features out of 'm' features where k << m
2. From K features, select best node d to split around
3. Split the node into children nodes using best split method
4. Now repeat first 3 steps upto a certain number of time.
5. Build different forest by repeating all four steps upto n number of times to create a random forest of n numbers of decision trees.

Random Forest Classifier

In [34]:
```
from sklearn.ensemble import RandomForestClassifier

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', RandomForestClassifier(n_estimators=50,
                                                  criterion="entropy"))])
model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
```

```
accuracy: 98.88%
```

In [35]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```

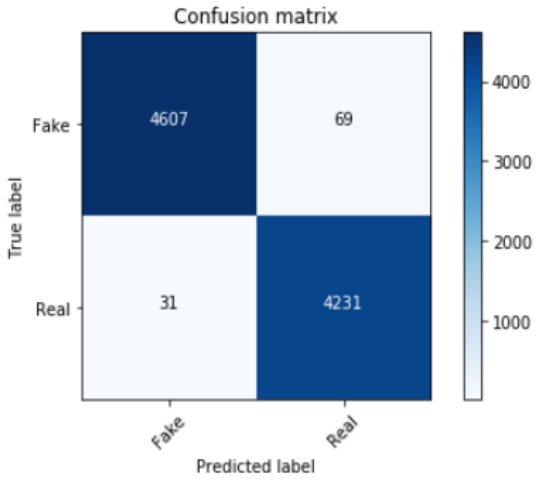


Figure 3 : Random Forest

Gradient Boosting Classifier: They are a gathering of AI calculations that consolidate numerous feeble learning models together to make a solid prescient model. Choice trees are generally utilized while doing slope supporting. The Python AI library, Scikit-Learn, upholds various executions of slope supporting classifiers. Gradient Boosting Classifiers are explicit sorts of calculations that are utilized for grouping assignments, as the name proposes.

Gradient Boosting Classifier

In [38]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(X_train)
xv_test = vectorization.transform(X_test)

GBC = GradientBoostingClassifier(random_state=0)
GBC.fit(xv_train, y_train)
pred_gbc = GBC.predict(xv_test)
print(classification_report(y_test, pred_gbc))
```

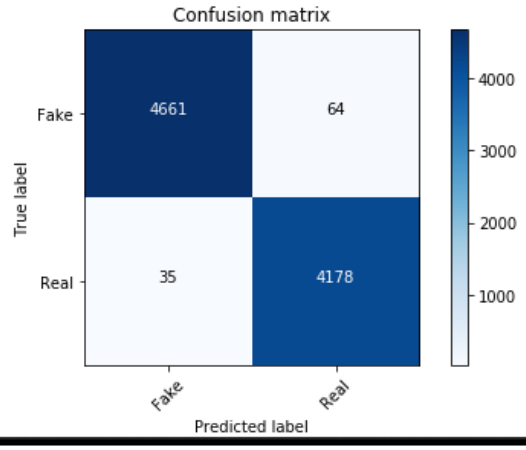
```
              precision    recall  f1-score   support

        fake       1.00      1.00      1.00      4725
        true       0.99      1.00      1.00      4213

    accuracy                           1.00      8938
   macro avg       1.00      1.00      1.00      8938
weighted avg       1.00      1.00      1.00      8938
```

In [40]:
```
GBC.score(xv_test, y_test)
```

Out[40]:
```
0.9961960170060417
```

In [41]:
```
cm = metrics.confusion_matrix(y_test, prediction)
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
```

```
Confusion matrix, without normalization
```



8. RESULTS

8.1 Experimental Outputs:

We have applied 4 algorithms i.e., Logistic Regression, Decision Tree, Random Forest Classifier and Gradient Boosting Classifier. We got best result from the **Decision Tree**.

ALGORITHMS	ACCURACY	CORRECT CLASSIFICATION
Logistic Regression	98.81%	8832
Decision Tree Classifier	99.70%	8889
Random Forest Classifier	98.89%	8838
Gradient Boosting Classifier	99.61%	8938

Chart 1: Result Table

1. **Logistic Regression**

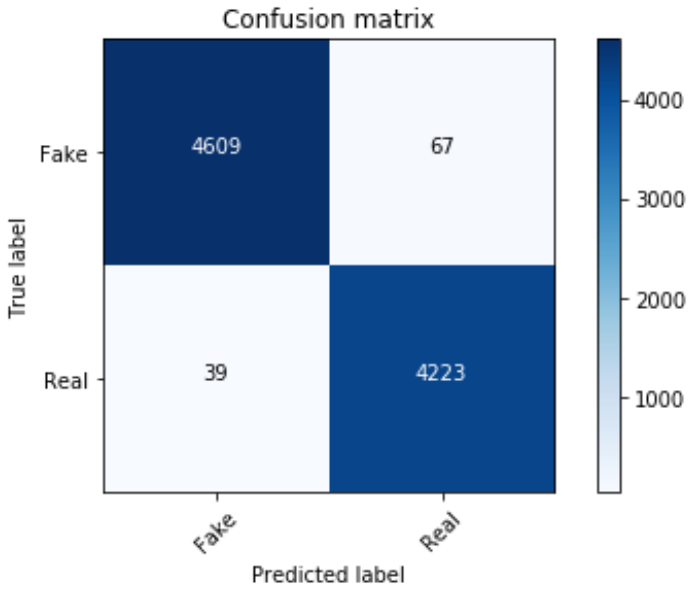


Figure 29: Confusion Matrix of Logistic Regression

- ✓ First cell denotes that we have correctly classified **4609** fake news examples.
- ✓ Second cell denotes that we have mistakenly classified **67** fake news examples as real news.
- ✓ Third cell denotes that we have mistakenly classified **38** real news examples as fake news.
- ✓ Fourth cell denotes that we have correctly classified **4223** real news examples.

2. **Random Forest Classifier**

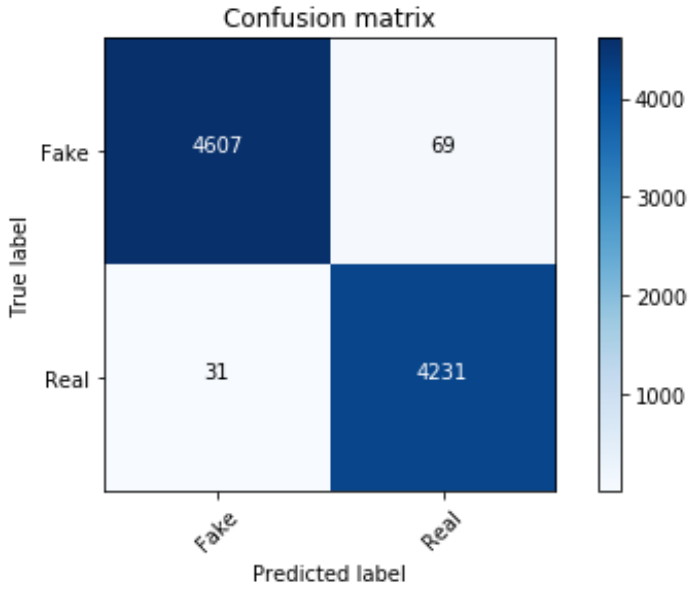


Figure 30: Confusion Matrix of Decision Tree

- ✓ First cell denotes that we have correctly classified **4607** fake news examples.
- ✓ Second cell denotes that we have mistakenly classified **69** fake news examples as real news.
- ✓ Third cell denotes that we have mistakenly classified **31** real news examples as fake news.
- ✓ Fourth cell denotes that we have correctly classified **4231** real news examples.

3. **Decision Tree**

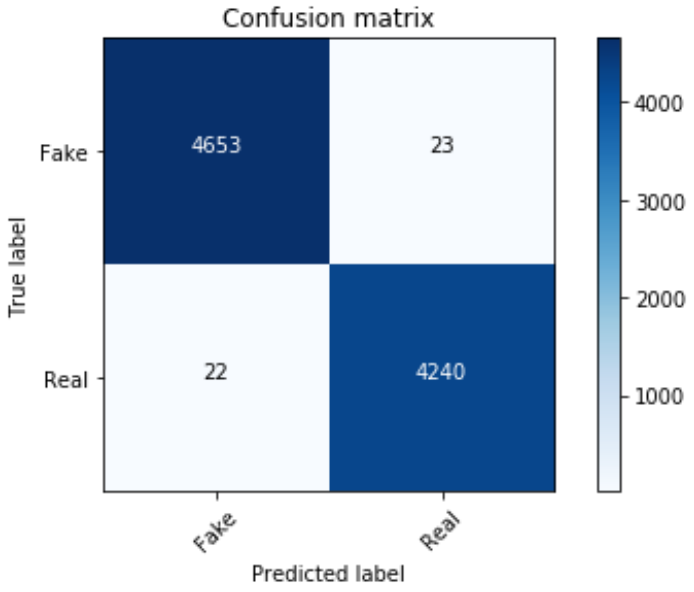


Figure 31: Confusion Matrix of Random Forest

- ✓ First cell denotes that we have correctly classified **4653** fake news examples.
- ✓ Second cell denotes that we have mistakenly classified **23** fake news examples as real news.
- ✓ Third cell denotes that we have mistakenly classified **22** real news examples as fake news.
- ✓ Fourth cell denotes that we have correctly classified **4240** real news examples.

4. Gradient Booster Classifier

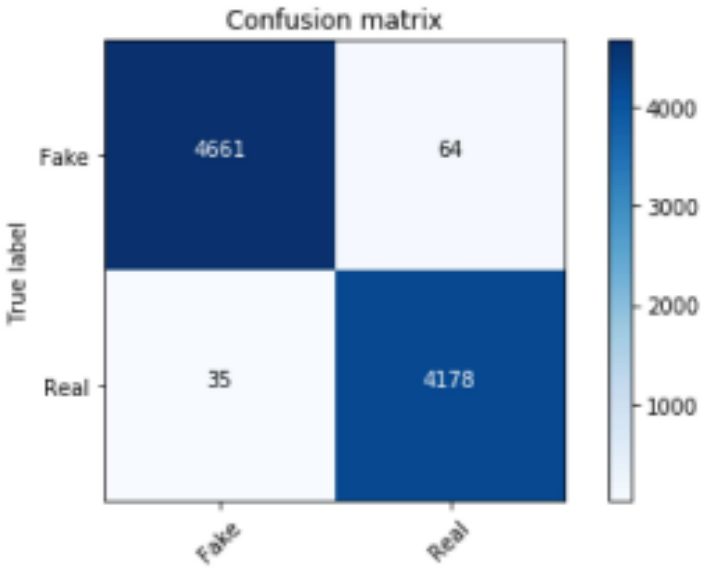


Figure 31: Confusion Matrix of Gradient Booster Classifier

- ✓ First cell denotes that we have correctly classified **4661** fake news examples.
- ✓ Second cell denotes that we have mistakenly classified **64** fake news examples as real news.
- ✓ Third cell denotes that we have mistakenly classified **35** real news examples as fake news.
- ✓ Fourth cell denotes that we have correctly classified **4178** real news examples.

8.2 Experimental Results:

Decision Tree has correctly classified maximum numbers of examples, followed by Random Forest, Gradient Boosting Classifier and Logistic Regression. So, **Decision Tree is suitable** for fake news classification.

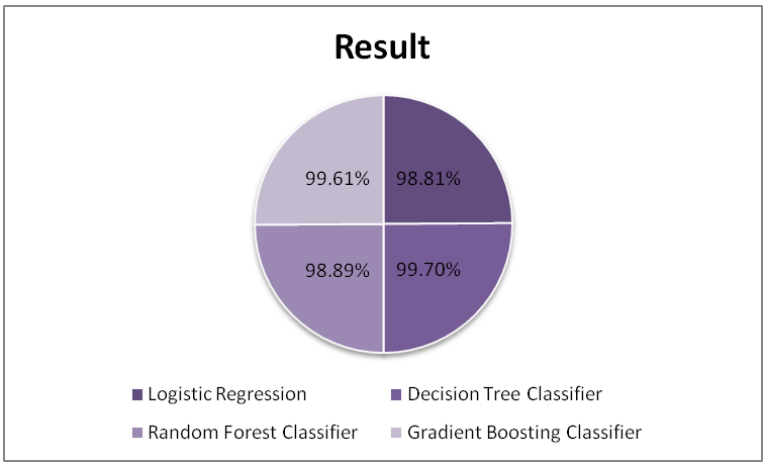


Chart 2: Classification Chart

As it is evident from previous figure that Decision Tree is performing well over given dataset. This can also be seen from the graph below:

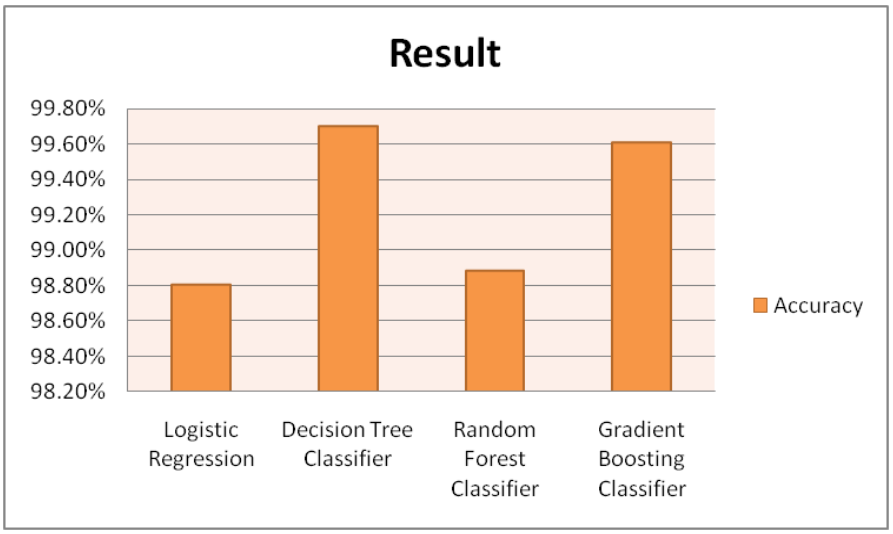


Chart 3: Accuracy Chart

9. AFTER EXPLORATION

9.1 Limitations:

The fact is, these models have achieved tremendous accuracy on existing examples of manipulated news, the analysis is naturally relatively shallow models check whether news articles conform to standard norms and styles used by professional journalists. This leads to two drawbacks.

First, these models can only detect fake news when they are under-written, for example when the information is dissimilar to the headline (termed as “clickbait”), or it could be when the articles are comprised of words that are deemed to be biased or provocative. While this criterion suits to detect many present examples of fake news, for instance, taking a well-written real news article and tampering the report in a targeted way. By conserving the original subject matter and linking the content strongly to the headline without introducing biased phrases, an adversarial article can easily dodge detection.

Adversarial Machine Learning is an emerging field of applied Machine Learning that focuses on how any malicious user can attack machine learning-based classifiers. Here are three kinds of adversarial example which focus on tempering different aspects of an article.

- Fact Distortion: Amplifying or adjusting some words. Person, time, location, relation etc. elements can be distorted
- Subject-Object Exchange: With such kind of attack, a user can be confused, whom to identify as performer and receiver of the action.
- Cause Cofounding: IT includes building a non-existent relationship between two independent events, or for manipulation, removing some part of a story and leaving the details which adversarial wants.

9.2 Solutions:

Due to some of these vulnerabilities, there's still a chance that our model could misclassify the given article or information. To counter such problems, there are some Machine Learning Algorithms which could provide a better accuracy over unseen examples such as: Recurrent Neural Network

- Deep Learning

There is another field which could be helpful to prevent the integrity of the data, i.e. to make sure that the provided data is not manipulated or tempered, we can use:

- Blockchain

9.3 Scope:

Later on, when we are sure about the accuracy and when the model is much more than just a black box to us, i.e. when the reason behind the classification of the model could be understandable, we can deploy it on the web. So that people do not have to go here and there to check the truthfulness of the given information. Web deployment could be followed by forming API of the model so that it could be integrated with any of applications, websites etc.

10. CONCLUSION

In this paper, we have studied about the need for "Fake News Detection" in the current era! How just a WhatsApp forward could lead to failure of the entire system. Along with some theoretical aspects, we have successfully built a machine learning model to classify whether the news is fake or genuine based on keywords present in it. It includes natural language processing, some of the machine learning algorithms like Logistic Regression, Decision Tree and RandomForest. We have checked which algorithm better suits to classify news. Furthermore, we are looking forward to implementing the Recurrent Neural Network and see how it performs on the categorical data.

11. REFERENCES

1. Fake News Detection via NLP is Vulnerable to Adversarial Attacks
2. Introduction to Natural Language Processing
3. Fake News Detection
4. Fake News Detection using Machine Learning guided project by Ryan Ahmed
5. Introduction to Natural Language Processing, Geeks for Geeks
6. Detecting Fake News using Machine Learning, DataFlair
7. Introduction to Logistic Regression by Ayush Pant
8. Decision Tree from Analytics Vidya
9. Understanding Random Forest by Tony Yiu
10. Ramasubramanian, K., et. al.," Machine Learning Using R", Springer, 2019.
11. Rodrigo Fernandes de Mello, et. al., "Machine Learning A Practical Approach on the Statistical Learning Theory", Springer, 2018.
12. QuanZou, et. al., "Advanced Machine Learning Techniques for Bioinformatics", IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), July 2019.