VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

Comparison of Fake News Detection using Machine Learning Classification Algorithms

By

Ankana Chatterjee (20BCE1212), Mallapalli Kusuma Sai (20BCE1553), J Jayanthi (20BCE1819), Kovi Yasaswini (20BCE1470)

Under the kind guidance of

Prof. Brindha S

In partial fulfillment of

CSE3505
Fundamentals of Data Analytics

Abstract

In a time when we can spread and receive information quickly, media literacy is very important. But it is the duty of media organizations to crack down on fake news and focus on the “real news” for the sake of a peaceful society. Internet is one of the most significant inventions, and many people utilise it. These people employ it for various functions. These users have access to a variety of social networking channels. Any person can publish a message or share news using these internet platforms. The individuals or their posts are not verified on these platforms. Consequently, some users attempt to spread through these sites, bogus news These false reports may serve as propaganda against a person, society, a group, or an administration. A person cannot discern all of these false reports. Therefore, there is a need for machine learning classifiers that can automatically identify these phoney news. This comprehensive overview of the literature describes the use of machine learning classifiers for identifying fake news.

However, the biggest challenge is to differentiate between real news and fake news. Consumers are creating and disseminating more information than ever because to the widespread use of social media platforms, some of which is false and irrelevant to reality. The automated identification of misinformation or disinformation in a written article is a difficult undertaking. Before passing judgement on an article's veracity, even a subject matter expert must consider a number of factors. In this work, we propose to use the classification algorithms of machine learning, namely, Logistic Regression, Naive-Bayes and Decision Tree, to distinguish fake news from real. Experimental evaluation confirms the superior performance of Logistic Regression algorithm as compared to the other two classification algorithms for the detection of fake news.

Keywords - stemming, logistic regression, data analysis, naïve-bayes, decision tree, classification, machine learning

Scope/Objectives

Our aim is to compare and analyze the results produced by the classification algorithms of machine learning, namely, Logistic Regression, Naive-Bayes and Decision Tree, to distinguish fake news from real, in order to be able to identify the algorithm with the highest accuracy.

Problem statement

Deceptive content, such as fake news, reviews and opinion spam has become a more serious threat to online users in recent years. Fake reviews have an impact on businesses, consumers and shops alike. Additionally, the issue of fake news has come to light during elections. As a result of the vast amount of user-generated content, it has rapidly become a growing research subject.

Spammers utilize appealing news headlines to mislead readers into clicking on their ads. They intentionally publish half-truths, propaganda and disinformation asserting to be real news for people often using social media to drive web traffic and magnify their effect. The majority of fake news websites' objectives are to change public perceptions of certain issues. Currently, the hardest task is determining the difference between a true review and a false one.

Contributions

As we can see, despite our best efforts to identify bogus news spreading on social media, we still require a suitable algorithm to identify it more accurately. As a result, we have chosen to compare several machine learning models in this work. We have outlined a methodical process in our work for selecting the optimal machine learning model from among Logistic Regression, Naive Bayes, and Decision Tree that would enable us to identify false news with the greatest degree of accuracy.

Introduction

In the modern world, anyone can publish content online. Unfortunately, fake news attracts a lot of attention online, especially through web-based networking platforms. People are misled and don't stop to think before sending such inaccurate information to the arrangement's farthest point. Such behaviors are bad for society because they cause rumors or hazy news to spread, which in turn causes people or a certain group of people to think negatively. To deal with such actions, preventative measures must advance at the same rate as technology. The general population is greatly impacted by broad communications, and as is customary, some persons try to take advantage of this. There are several websites that provide misleading information.

Under the guise of being factual news, they consciously try to spread purposeful publicity, deceptions, and lies. Their primary responsibility is to manage the information that could undermine public trust. There are many examples of these sites all across the world. Therefore, false information affects how people's brains work. According to the study, scientists agree that many artificially created computations can aid in exposing false information.

The main goal is to identify bogus news, which is a straightforward solution to a traditional text classification problem. Building a model that can distinguish between "Real" and "Fake" news is necessary. This has negative effects on social networking sites like Facebook, Instagram, microblogging sites like Twitter, and instant messaging apps like WhatsApp and Hike, where false information gains significant traction and spreads rapidly around the nation and the world. The suggested system aids in determining the veracity of the news. If the news is false, a relevant news story is recommended to the user.

The spread of fake news cannot be compromised since it has the potential to have long lasting detrimental consequences on the population. Problematic issues can result from fake news, including slanders, misunderstandings, and inflammatory lies, up to and including sentimental concerns caused by irresponsible parties or persons that enjoy sowing discord and division among one another. Fake news is defined as information that has been altered or produced using unrelated, entirely or partially untrue information. It can be very challenging to tell whether news is phoney or not.

However, this is made feasible to identify bogus news by the capabilities of artificial intelligence in machine learning. A number of nations have shown their dedication to combating false news because of the potential harm it could cause to society. In other words, false news actually misleads and persuades people to believe in untrue and likely manipulative information. Fake news may offer a variety of subjective explanations that are appropriate for comprehension. Separating the word false news could be helpful in this situation. While news is described as a report of recent or upcoming current events, the word fake is defined as something that is not authentic, counterfeit, or a forgery of something.

With the development of communication technologies and social media, the fake news phenomena is expanding quickly. A new field of research that is receiving a lot of attention is fake news detection. Due to the restricted resources, including datasets, processing, and analysis methods, it does, nevertheless, confront some difficulties.

False information, statements, or propaganda that has been disseminated under the guise of news is referred to as "fake news." The Fake News epidemic has grown significantly over the past ten years, helped along by social media. Various motives can be used to broadcast this false information. Some are created solely to enhance the amount of clicks and site visitors. Others seek to sway public opinion regarding political or financial market decisions. For instance, through affecting the online reputation of businesses and institutions. Social media fake news about health poses a threat to overall health. The COVID-19 outbreak had been followed by a significant infodemic, or an overflow of information, the WHO warned in February 2020. This made it challenging for individuals to obtain dependable sources and trustworthy information when they needed it. Uncertainty, fear, anxiety, and bigotry are being propagated on a scale unseen in earlier pandemics as a result of disinformation overload.

In our society, fake news and a lack of faith in the media are serious issues that have far-reaching effects. This continues to overstate the price of particular services or maintain lies about certain statistics in a country, which could cause instability in some nations.However, their reach is so constrained because they rely on manual identification by humans, which is neither reliable nor practical in a world where millions of items are removed or published every minute.

Fake news is created to mislead the recipient or to draw attention and gain publicity which creates a false impact on that particular issue or information. This continues to overstate the price of particular services or maintain lies about certain statistics in a country, which could cause instability in some nations.

Fake reviews have affected consumers, companies and stores alike. Furthermore, the problem of fake news has gained attention in elections as well. It has quickly become a growing research area due to the abundance of user-generated content. Spammers use appealing news headlines to generate revenue using advertisements via click baits. The biggest challenge now is to tell the difference between a real review and a fake one.

In this project, we propose to create the model which uses the best features by analyzing classical ML algorithms like decision tree, Random forest, Logistic Regression and Naive Bayes.

Algorithm

```
Input: content – list of news titles or texts
Input: selected attribute – content to learn (title, author merged)
Input: selected algorithm – selected machine learning algorithm
Output: classifier
if selected attribute == ‟CONTENT‟ then
    content← select text(content, ‟CONTENT‟);
content←tokenize text(content);
data set←vectorize text(content);
train set, test set←split data(data set);
classifier←model learning(train set, selected algorithm);
model testing(test set);
result classifier;
```

Literature review

Mykhailo Granik et. al.[1] in their paper, shows a simple approach for fake news detection using naive Bayes classifier. This approach was implemented as a software system and tested against a data set of Facebook news posts. They were collected from three large Facebook pages each from the right and from the left, as well as three large mainstream political news pages (Politico, CNN, ABC News).

The procedure they followed is: articles information loading, articles filtering based on the presence of the content and relevant label, separating data in the training, validation and test datasets. training the naïve Bayes classifier, testing and accuracy evaluation. They achieved classification accuracy of approximately 74%. Classification accuracy for fake news is slightly worse. This may be caused by the skewness of the dataset: only 4.9% of it is fake news. The research showed, that even quite simple artificial intelligence algorithm (such as naive Bayes classifier) may show a good result on such an important problem as fake news classification.

Barbara Probierz et.al. [2] in their paper, proposed an approach to classifying news based on the title. The obtained results will be compared with classification based on the whole news text. The goal of this work is to propose a method that balances between data analysis time and quality of classification in fake news prediction. They used natural language processing (NLP) to describe the title and text of the news and made analysis of a real data set and results of news classification using the proposed model – including an ensemble of classifiers and single classifiers. Observations showed that the analysis of the text of articles allows for a very good classification of true or fake news. The problem is the running time of the SVM 5 classifier but the use of the random forest algorithm at the initial stage of experiments is a good trade-off between quality of classification and running time.

By contrasting two different feature extraction strategies and six different classification techniques, the authors of [3] present a false news detection model that employs n-gram analysis and machine learning methods. The results of the tests indicate that the so-called features extraction method yields the best results (TF-IDF). They employed the 92% accurate Linear Support Vector Machine (LSVM) classifier. The LSVM used in this model is restricted to handling only the situation where two classes are linearly separated.

Florian Sauvageau et. al. [4] explain how social network users can guarantee the veracity of information. They also explain how they are validated, the function of journalists, and what to anticipate from academics and government agencies. People who don't trust anything can use this work to see a tiny amount of the reality behind the news they read on social media.

The authors of [5] offer a variety of methods and indices that relate to various modalities (text, image, social information). In order to evaluate and validate shared knowledge, they also consider the benefit of integrating and merging these methodologies.

Junaed Younus Khan et. al. [6] provide a comprehensive performance analysis of various methodologies on three separate datasets. This work ignored details like the source, author, or publication date that could have a significant impact on the outcome and instead concentrated on the language of the information and the feeling it conveyed. Additionally, we will demonstrate in our work that including feelings in the detection procedure does not yield any useful information.

The author of the paper [7], has discussed the current solutions for detecting fake news by investigating how news spreads online. By involving the dissemination of information (i.e., social context) in fake news detection, propagation-based methods are more robust against writing-style manipulation by malicious entities. However, propagation-based fake news detection is inefficient for fake news early detection as it is difficult for propagation-based models to detect fake news before it has been disseminated, or to perform well when limited news dissemination information is available. The paper uses a mix of English (Twitter) and Chinese (Weibo) news articles from vague domains. The algorithms used in this paper are Deep Learning (DL) Models. Within a DL framework, learning the representation of news cascades often relies on neural networks, where a softmax function often acts as a classifier. Fake News Detection using Self-defined Propagation Graphs When detecting fake news using self-defined propagation graphs (networks), one constructs flexible networks to indirectly capture fake news propagation. (1) More-Spreader Pattern, i.e., more users spread fake news than true news; (2) Farther-Distance Pattern, i.e., fake news spreads farther than true news; (3) Stronger-Engagement Pattern, i.e., spreaders engage more strongly with fake news than with true news; and (4) Denser-Network Pattern, i.e., fake news spreaders form denser networks compared to true news spreader.

Proposed Method

The system we propose uses a news dataset to build a decision model based on logistic regression, naïve bayes and decision tree algorithms. The model is then used to classify novel news to fake or real.

The comparison of the accuracy of the three chosen ML models (Logistic Regression, Naive Bayes and Decision Tree) in predicting a fake news correctly is given as follows:

- train.csv dataset is collected on which machine learning(ML) models are to be applied, in order to detect whether the data is fake or real.
- Data preprocessing is done, which includes removal of NA values and duplicate rows in train.csv dataset using R programming. Now our dataset is named as clean_train.csv
- The three ML models are analyzed based on accuracy scores. The ML model having the highest accuracy (Logistic Regression) was found to be the most useful in predicting a fake news.

Dataset

The news dataset that we have used for our research is available openly in Kaggle (https://www.kaggle.com/competitions/fake-news/data?select=train.csv). The dataset consists of 4 columns namely, Title, Author, Text and Label. Title represents the Title of the news article. Author represents the author of the news article. Text represents the text present in the news article. Label represents whether the data is fake or real. If the news is fake, the value is 1, otherwise it is 0. The dataset consists of 25116 entries.

Pre-processing:

First of all, all the empty and NaN values present in the excel sheet have been replaced by NA, after which the rows containing NA values have been removed from the dataset. In order to avoid overfitting, the data is pre-processed before fetching it to the Natural Languange Processing (NLP) system. The pre-processing involves various steps like sentence segmentation, tokenization, stop-words removal and word stemming, as discussed below in detail:

Sentence Segmentation: Sentence segmentation is establish text borders and break the text into sentences. Exclamation (!),interrogation (?), and utter stop (.) signs are widely used as markers to segment the paragraph into sentences.

Tokenization: At this stage, the phrases and sentences are divided into separate words by dividing them into whitespaces such as tabs, blanks, and signs of punctuation, i.e. dot (.), comma (,), semicolon (;), colon (:), etc.These are the key indications for dividing the sentences into tokens.

Stop-words removal: Words which have occurred repeatedly are called stop-words in a sentence. These consist of prepositions(in, on, at, etc.), conjunctions (and thus, too, etc.), articles (a, an, a), etc. These words have little meaning in text documents and are more weighted, and removing them will help improve the system's performance.

Word Stemming: Stemming is used to bring the word to its basic form. Word stemming plays a significant role in pre-processing. In order to normalize the word token to a standard form, this step changes the derived words to its base or stem word. The famous stemming algorithm, Porter’s stemming (Porter, 1980), is adopted to remove suffixes like –ing, -es, -ers from the text words. For example, the words ‘looking’ and ‘looks’ will be modified to its base type ‘look’ after stemming.

Experimental Setting: After the pre-processing stage, the data is then split into two datasets. One for training and the other for testing the accuracy of our algorithm. In this research, we have used logistic regression, naïve bayes and decision tree machine learning algorithms for comparing the accuracy of our prediction.

Logistic Regression:

To estimate the likelihood of a categorical dependent variable, this machine learning classification approach is used. The dependent variable in logistic regression is a binary variable with data coded as 1 (yes, success, etc.) or 0 (no) (no, failure, etc.). In other words, P(Y=1) is predicted by the logistic regression model as a function of X.

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

Naïve Bayes:

A supervised machine learning algorithm that makes use of Bayes' theorem is called a Naive Bayes classifier. The variables that make up the model are unrelated to one another. It has been demonstrated that this classifier produces rather good results on its own.

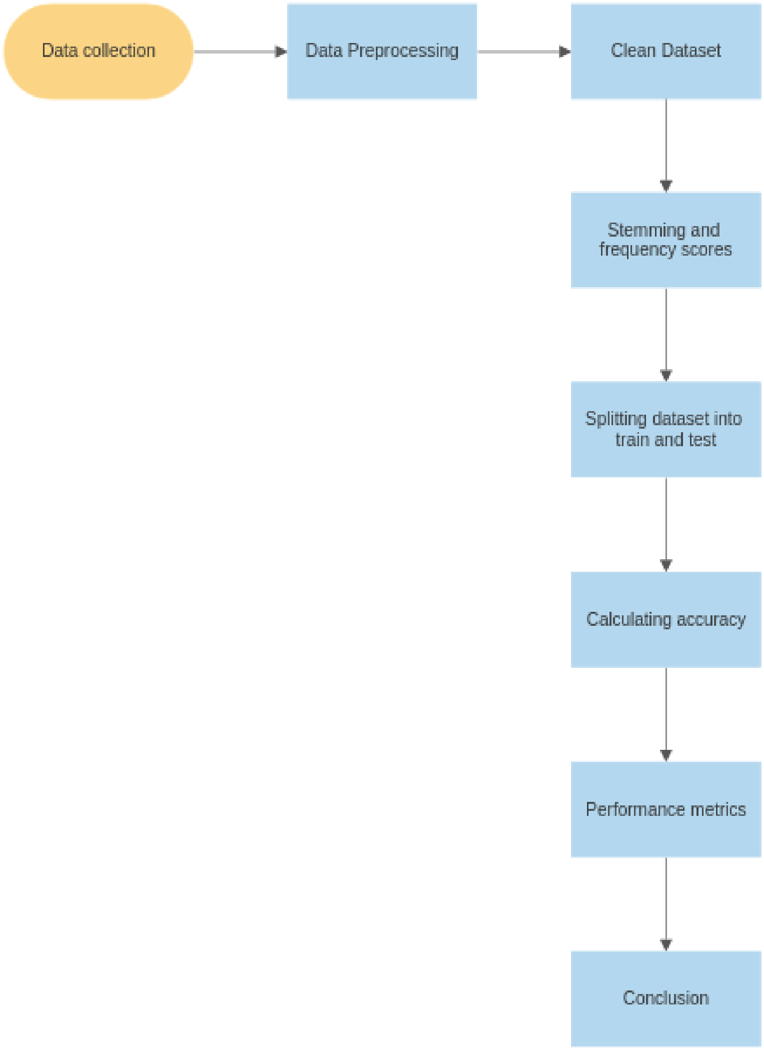
$P((X|Ci) = \prod P(xk|Ci) = P(x1|Ci) \times P(x2|Ci)nk=1\times \ldots \times P(xn|Ci)$

The Bayes theorem is used to carry out the classification by determining the maximum posterior, which is the maximal P(Ci|X) under the aforementioned assumption. By merely accounting for the class distribution, this assumption significantly lowers the computational cost. Multinomial Naive Bayes is a well-known method that is used to determine whether news is accurate and hence true or false. It is not the sole algorithm for training such classifiers because there are many that concentrate on common principles.

Decision Tree:

The J48 algorithm is one of the most widely used classification algorithms. It is based on the C4.5 method, and all of the data that needs to be studied must be of the categorical and numerical variety. As a result, continuous data types won't be looked at. J48 employs two distinct pruning techniques. The first technique is known as "subtree replacement," which refers to the potential for replacing nodes in a decision tree with its leaves in order to reduce the quantity of tests along the convincing path. Typically, the subtree raising has a negligible effect on the decision tree models. Although it may be wise to disable an option when the induction process takes longer due to the subtree's raising being relatively computationally challenging, there is typically no precise way to estimate an option's utility.

Work Flow Diagram



Modules/algorithm phases

Pre-processing:

Data is pre-processed in R-Cloud. Here, ‘train.csv’ contains the news dataset. The dataset is stored in the data frame ‘df’. Now, all the empty values as well as NaN values in the data frame ‘df’ are replaced by ‘NA’ so that it becomes easier to get removed using the built-in function na.omit() in R. By applying the built-in function sapply() on our dataset, it has been found that there are 558 ‘NA’ values in the column ‘title’, 1957 ‘NA’ values in the column ‘author’ and 39 ‘NA’ values in the column ‘text’. After applying the na.omit() function on our dataset to remove all the rows containing ‘NA’ values, we again check the presence of ‘NA’ values in our dataset using the sapply() function. This time, it is found that there are no ‘NA’ values in any of the columns, which implies that our dataset has been cleaned and all the ‘NA’ values have been removed successfully. This dataset is stored in a csv file named ‘clean_train.csv’.

After removal of ‘NA’ values, the columns ‘author’ and ‘title’ are merged into a single column named as ‘content’ using the in-built R method paste(). This dataset is the final dataset that has been used for our work and is stored in the csv file named as ‘clean_train2.csv’.

Splitting the data:

The dataset obtained after pre-processing is split into two, training and testing dataset. The training data is used for training our machine learning model whereas the testing data is used for finding the accuracy of our machine learning model.

Applying Machine Learning models to our data:

After splitting the data, NumPy and sklearn modules are imported, which make it easier for us to apply the various machine learning models on our dataset.

Logistic Regression:

From the sklearn module we will use the LogisticRegression() method to create a logistic regression object. This object has a method called fit() that takes the independent and dependent values as parameters and fills the regression object with data that describes the relationship.

The accuracy of our model is measured using the accuracy() method which gives us the accuracy score of the model on the training data.

After training the dataset, the same model is applied on the testing dataset to check the accuracy of the model on the testing dataset.

Naïve Bayes:

From sklearn module, we will import and make use of metrics in order to perform Naive Bayes algorithm analysis on our dataset. Here, classifier.fit() method takes the independent and dependent values as parameters and fills the naïve bayes object with data that describes the relationship.

The accuracy of our model is measured using the accuracy() method which gives us the accuracy score of the model on the training data.

After training the dataset, the same model is applied on the testing dataset to check the accuracy of the model on the testing dataset.

Decision Tree:

Decision Tree consists of two types, namely, Decision Tree Classification and Regression. In our case, we will be needing Decision Tree Classifier since we want to classify whether a news is fake or not. Therefore, Decision Tree Classification algorithm needs to be followed in our case.

From the sklearn module we will use the DecisionTreeClassifier() method to create a decision tree classifier object. This object has a method called fit() that takes the independent and dependent values as parameters and fills the decision tree object with data that describes the relationship.

The accuracy of our model is measured using the accuracy() method which gives us the accuracy score of the model on the training data.

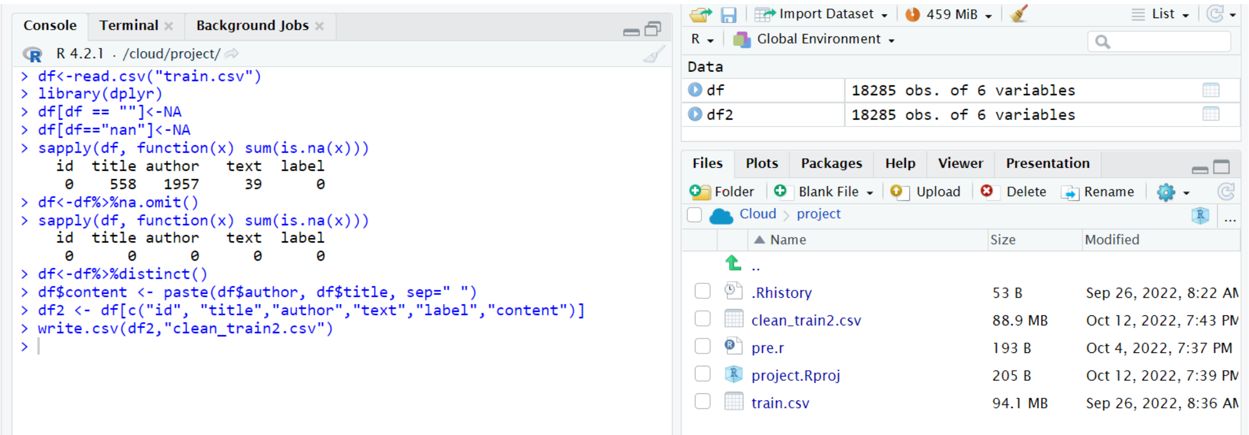
After training the dataset, the same model is applied on the testing dataset to check the accuracy of the model on the testing dataset.

Step –1:
Data cleaning using R programming
Step –2:
Stemming the clean_train2.csv dataset using Python programming
Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma.
For example: “Flying” is a word and its suffix is “ing”, if we remove “ing” from “Flying” then we will get base word or root word which is “Fly”.
Step-3:
Splitting the dataset into test and train dataset.
Step-4:
Training the dataset with Logistic Regression, Naïve Bayes and Decision Tree Classifier.
Step-5:
Testing the dataset with Logistic Regression, Naïve Bayes and Decision Tree Classifier.

Results

Data Cleaning:

df<-read.csv("train.csv")
install.packages("dplyr")
library(dplyr)
df[df == ""]Data<-NA
df[df=="nan"]<-NA
sapply(df, function(x) sum(is.na(x)))
df<-df%>%na.omit()
sapply(df, function(x) sum(is.na(x)))
df<-df%>%distinct()
df$content <- paste(df$author, df$title, sep=" ")
df2 <- df[c("id", "title","author","text","label","content")]
write.csv(df2,"clean_train2.csv")



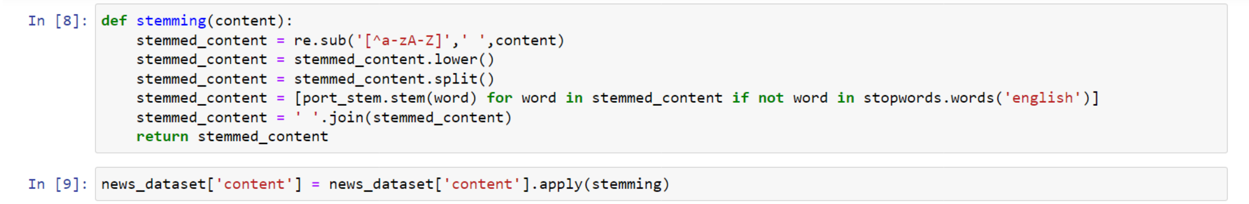
```
> df<-read.csv("train.csv")
> library(dplyr)
> df[df == ""]<-NA
> df[df=="nan"]<-NA
> sapply(df, function(x) sum(is.na(x)))
    id  title author   text  label
     0    558   1957     39      0
> df<-df%>%na.omit()
> sapply(df, function(x) sum(is.na(x)))
    id  title author   text  label
     0      0      0      0      0
> df<-df%>%distinct()
> df$content <- paste(df$author, df$title, sep=" ")
> df2 <- df[c("id", "title","author","text","label","content")]
> write.csv(df2,"clean_train2.csv")
>
```

Data

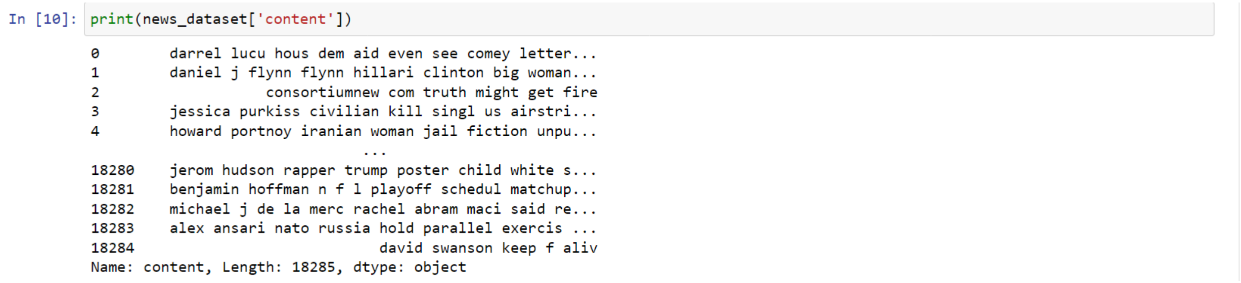
df	18285 obs. of 6 variables
df2	18285 obs. of 6 variables

Name	Size	Modified
..		
.Rhistory	53 B	Sep 26, 2022, 8:22 AM
clean_train2.csv	88.9 MB	Oct 12, 2022, 7:43 PM
pre.r	193 B	Oct 4, 2022, 7:37 PM
project.Rproj	205 B	Oct 12, 2022, 7:39 PM
train.csv	94.1 MB	Sep 26, 2022, 8:36 AM

Stemming:

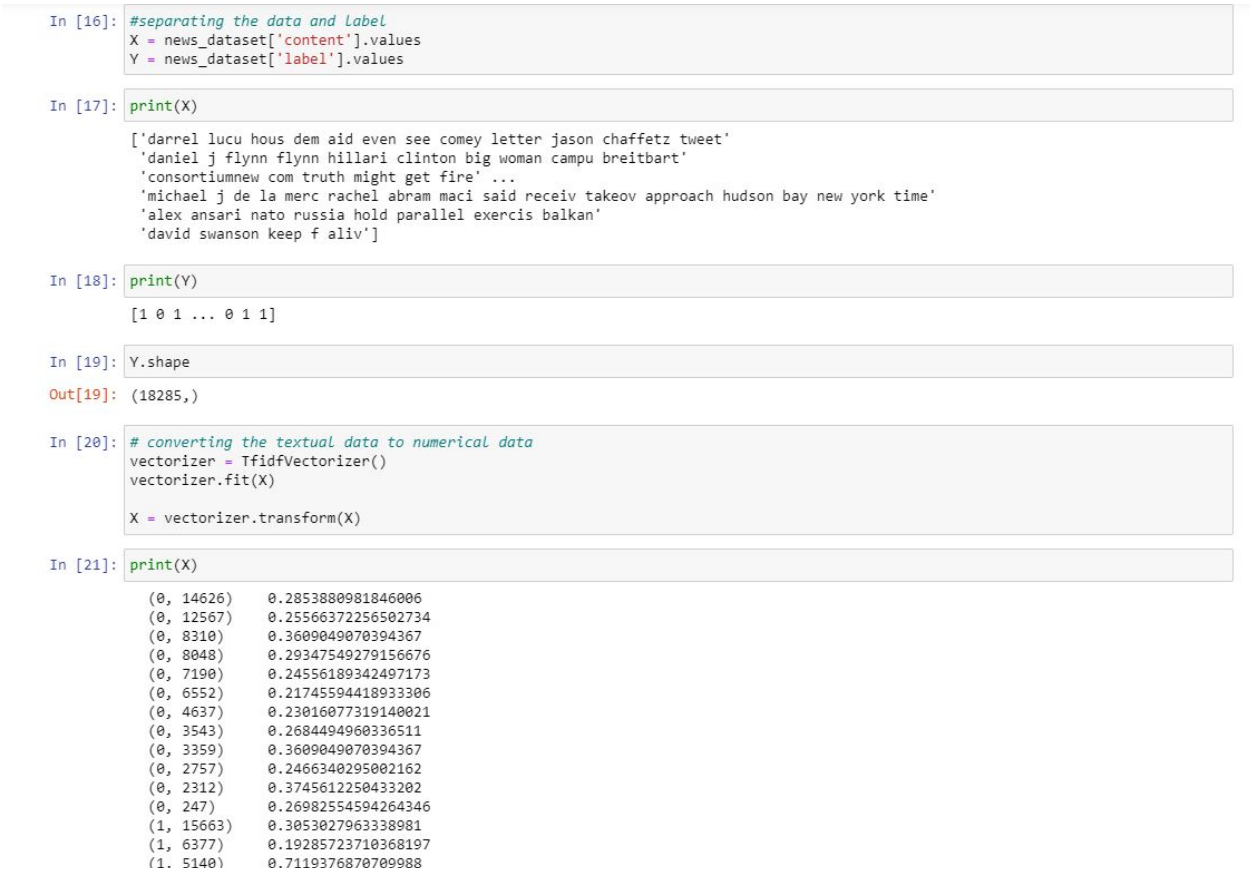


```
In [8]: def stemming(content):
            stemmed_content = re.sub('[^a-zA-Z]',' ',content)
            stemmed_content = stemmed_content.lower()
            stemmed_content = stemmed_content.split()
            stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words('english')]
            stemmed_content = ' '.join(stemmed_content)
            return stemmed_content

In [9]: news_dataset['content'] = news_dataset['content'].apply(stemming)
```

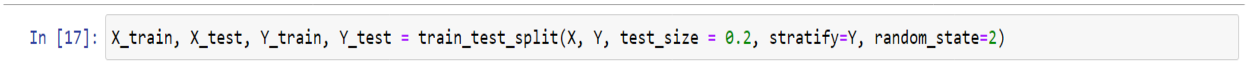


```
In [10]: print(news_dataset['content'])

         0        darrel lucu hous dem aid even see comey letter...
         1        daniel j flynn flynn hillari clinton big woman...
         2                    consortiumnew com truth might get fire
         3        jessica purkiss civilian kill singl us airstri...
         4        howard portnoy iranian woman jail fiction unpu...
                                        ...
         18280    jerom hudson rapper trump poster child white s...
         18281    benjamin hoffman n f l playoff schedul matchup...
         18282    michael j de la merc rachel abram maci said re...
         18283    alex ansari nato russia hold parallel exercis ...
         18284                             david swanson keep f aliv
         Name: content, Length: 18285, dtype: object
```

Splitting the dataset into test and train dataset:

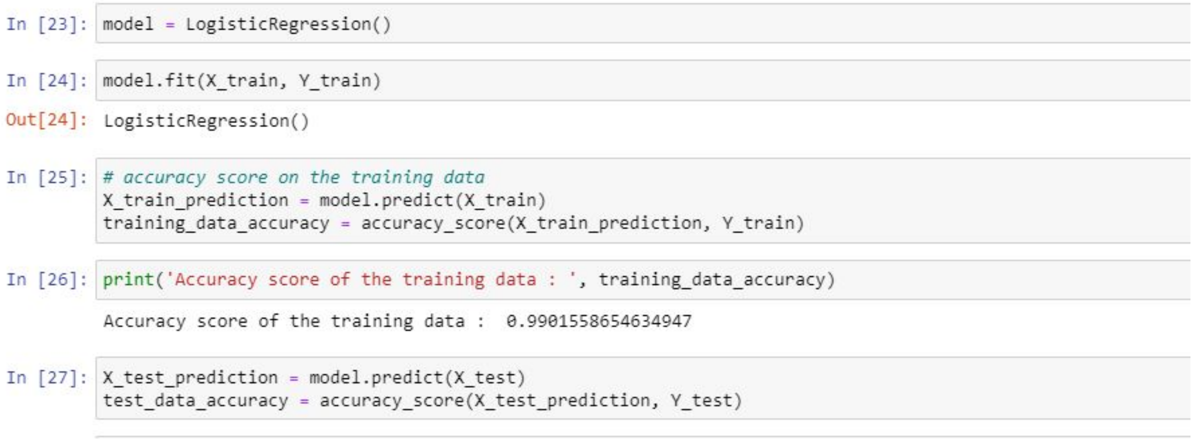


```
In [16]: #separating the data and label
         X = news_dataset['content'].values
         Y = news_dataset['label'].values

In [17]: print(X)

         ['darrel lucu hous dem aid even see comey letter jason chaffetz tweet'
          'daniel j flynn flynn hillari clinton big woman campu breitbart'
          'consortiumnew com truth might get fire' ...
          'michael j de la merc rachel abram maci said receiv takeov approach hudson bay new york time'
          'alex ansari nato russia hold parallel exercis balkan'
          'david swanson keep f aliv']

In [18]: print(Y)

         [1 0 1 ... 0 1 1]

In [19]: Y.shape

Out[19]: (18285,)

In [20]: # converting the textual data to numerical data
         vectorizer = TfidfVectorizer()
         vectorizer.fit(X)

         X = vectorizer.transform(X)

In [21]: print(X)

           (0, 14626)    0.2853880981846006
           (0, 12567)    0.25566372256502734
           (0, 8310)     0.3609049070394367
           (0, 8048)     0.29347549279156676
           (0, 7190)     0.24556189342497173
           (0, 6552)     0.21745594418933306
           (0, 4637)     0.23016077319140021
           (0, 3543)     0.2684494960336511
           (0, 3359)     0.3609049070394367
           (0, 2757)     0.2466340295002162
           (0, 2312)     0.3745612250433202
           (0, 247)      0.26982554594264346
           (1, 15663)    0.3053027963338981
           (1, 6377)     0.19285723710368197
           (1, 5140)     0.7119376870709988
```

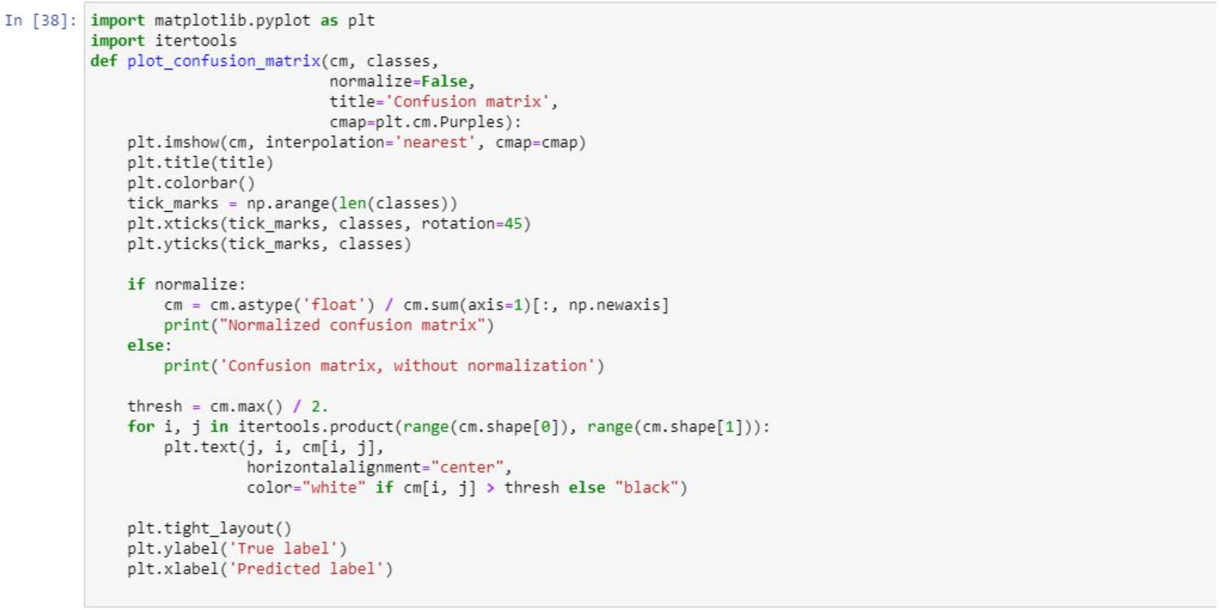


```
In [17]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify=Y, random_state=2)
```

Training and testing the dataset with Logistic Regression,Naive Bayes and Decision Tree Classifier:



```
In [23]: model = LogisticRegression()

In [24]: model.fit(X_train, Y_train)

Out[24]: LogisticRegression()

In [25]: # accuracy score on the training data
         X_train_prediction = model.predict(X_train)
         training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

In [26]: print('Accuracy score of the training data : ', training_data_accuracy)

         Accuracy score of the training data :  0.9901558654634947

In [27]: X_test_prediction = model.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

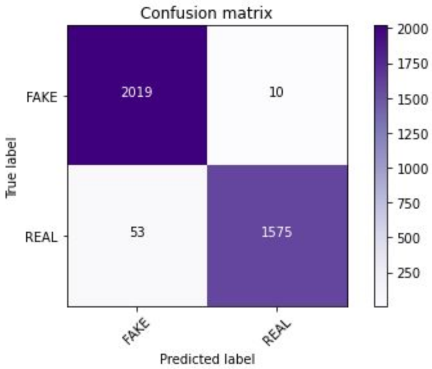


```
In [38]: import matplotlib.pyplot as plt
         import itertools
         def plot_confusion_matrix(cm, classes,
                                   normalize=False,
                                   title='Confusion matrix',
                                   cmap=plt.cm.Purples):
             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')
```

In [39]:

```
cm = confusion_matrix(X_test_prediction, Y_test)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
print ("Confusion Matrix : \n", cm)
```

```
Confusion matrix, without normalization
Confusion Matrix :
 [[2019   10]
 [  53 1575]]
```



In [44]:

```
from sklearn.metrics import classification_report
print(classification_report(Y_test,X_test_prediction))
```

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      2072
           1       0.97      0.99      0.98      1585

    accuracy                           0.98      3657
   macro avg       0.98      0.98      0.98      3657
weighted avg       0.98      0.98      0.98      3657
```

In [45]:

```
#classification accuracy,error and precision
TP=cm[0,0]
TN=cm[1,1]
FP=cm[0,1]
FN=cm[1,0]
```

In [47]:

```
classification_accuracy=(TP+TN)/float(TP+TN+FN+FP)
print("Classification accuracy is : {0:0.4f}".format(classification_accuracy))
```

```
Classification accuracy is : 0.9828
```

In [48]:

```
classification_error=(FP+FN)/float(FP+FN+TP+TN)
print("Classification error is : {0:0.4f}".format(classification_error))
```

```
Classification error is : 0.0172
```

In [50]:

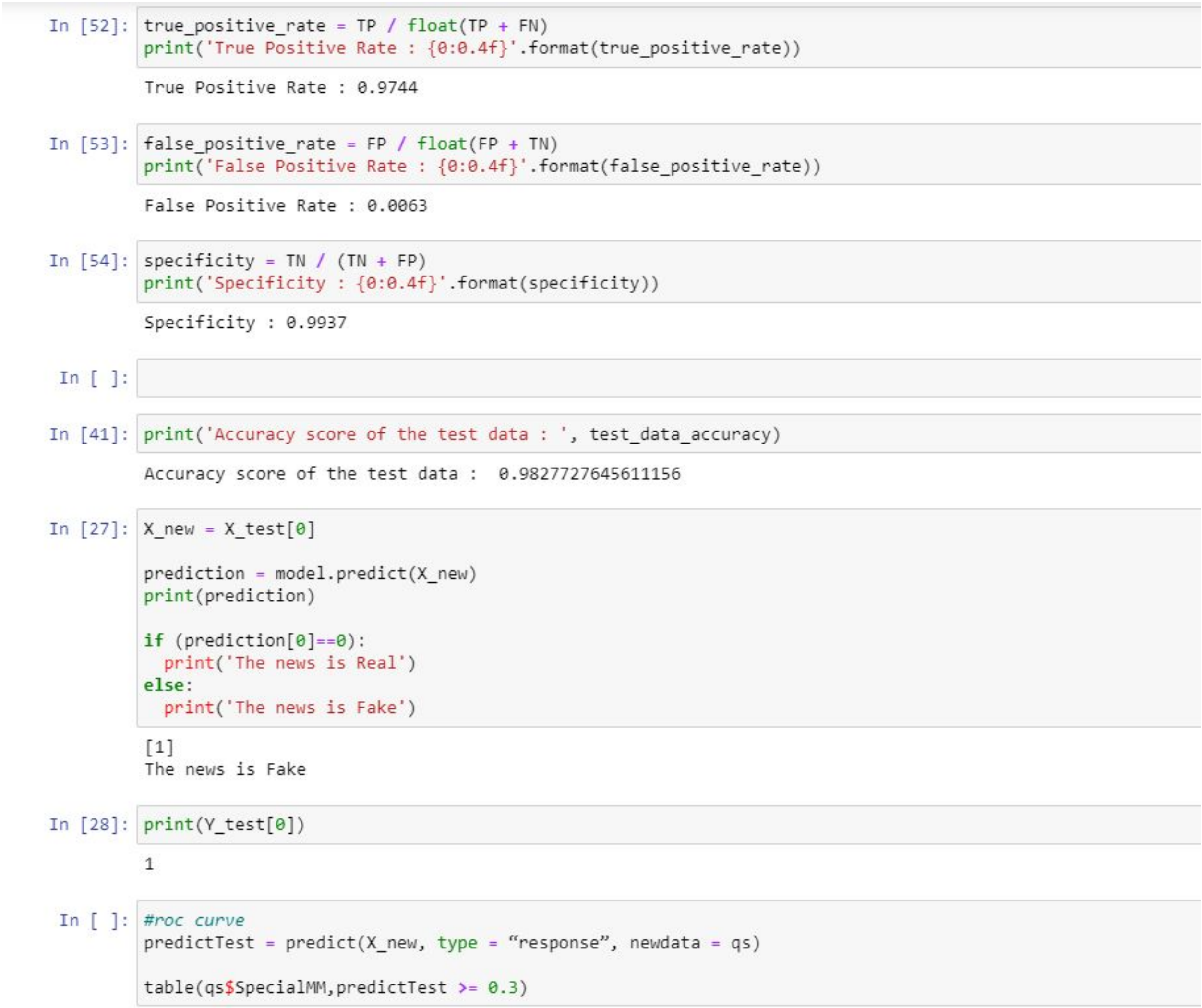
```
precision=TP/float(TP+FP)
print("Precision is : {0:0.4f}".format(precision))
```

```
Precision is : 0.9951
```

In [51]:

```
recall = TP / float(TP + FN)
print('Recall or Sensitivity : {0:0.4f}'.format(recall))
```

```
Recall or Sensitivity : 0.9744
```



```
In [52]: true_positive_rate = TP / float(TP + FN)
         print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))

         True Positive Rate : 0.9744

In [53]: false_positive_rate = FP / float(FP + TN)
         print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))

         False Positive Rate : 0.0063

In [54]: specificity = TN / (TN + FP)
         print('Specificity : {0:0.4f}'.format(specificity))

         Specificity : 0.9937

 In [ ]:

In [41]: print('Accuracy score of the test data : ', test_data_accuracy)

         Accuracy score of the test data :  0.9827727645611156

In [27]: X_new = X_test[0]

         prediction = model.predict(X_new)
         print(prediction)

         if (prediction[0]==0):
           print('The news is Real')
         else:
           print('The news is Fake')

         [1]
         The news is Fake

In [28]: print(Y_test[0])

         1

 In [ ]: #roc curve
         predictTest = predict(X_new, type = “response”, newdata = qs)

         table(qs$SpecialMM,predictTest >= 0.3)
```

In [57]:

```
#Multibayes algorithm
import matplotlib.pyplot as plt

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

In [58]:

```
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X, Y, test_size=0.33, random_state=42)
```

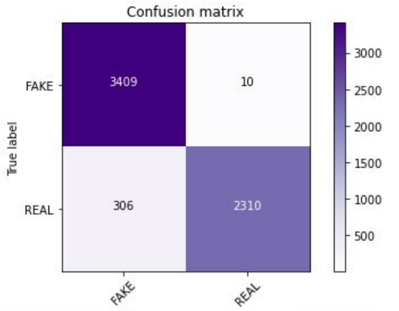
In [59]:

```
#Let's implement the model : Multinomial Naive Bayes
from sklearn.naive_bayes import MultinomialNB
classifier=MultinomialNB()
```

In [60]:

```
from sklearn import metrics
import numpy as np
import itertools

classifier.fit(X1_train, Y1_train)
prediction2 = classifier.predict(X1_train)
score2 = metrics.accuracy_score(Y1_train, prediction2)
print("Accuracy score of the training data : %f" % score2)

prediction1 = classifier.predict(X1_test)
score = metrics.accuracy_score(Y1_test, prediction1)
print("Accuracy score of the test data : %f" % score)
cm1 = metrics.confusion_matrix(Y1_test, prediction1)
plot_confusion_matrix(cm1, classes=['FAKE', 'REAL'])
print(classification_report(Y1_test,prediction1))
```

```
Accuracy score of the training data : 0.974367
Accuracy score of the test data : 0.947639
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      3419
           1       1.00      0.88      0.94      2616

    accuracy                           0.95      6035
   macro avg       0.96      0.94      0.95      6035
weighted avg       0.95      0.95      0.95      6035
```

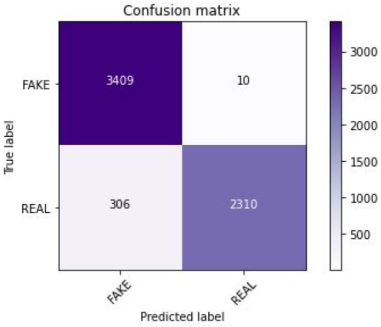


In [62]:
```
clf_gini = DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=3, min_samples_leaf=5)
clf_gini.fit(X_train, Y_train)
y_pred1 = clf_gini.predict(X_train)
y_pred2 = clf_gini.predict(X_test)
print("Predicted values:")
print(y_pred1)
print(y_pred2)
```

```
Predicted values:
[1 1 0 ... 1 0 0]
[1 0 0 ... 0 0 1]
```

In [63]:
```
cm2 = metrics.confusion_matrix(Y_test, y_pred2)
plot_confusion_matrix(cm1, classes=['FAKE', 'REAL'])
print(classification_report(Y_test,y_pred2))
print("Accuracy score of the training data : ",accuracy_score(Y_train,y_pred1)*100)
print("Accuracy score of the test data :",accuracy_score(Y_test,y_pred2)*100)
```

```
Confusion matrix, without normalization
              precision    recall  f1-score   support

           0       1.00      0.86      0.92      2072
           1       0.85      1.00      0.92      1585

    accuracy                           0.92      3657
   macro avg       0.92      0.93      0.92      3657
weighted avg       0.93      0.92      0.92      3657

Accuracy score of the training data :  91.78288214383375
Accuracy score of the test data : 92.04265791632486
```



In [35]:
```
X_new_decision = X_test[3]

prediction = clf_gini.predict(X_new_decision)
print(prediction)

if (prediction[0]==0):
  print('The news is Real')
else:
  print('The news is Fake')
```

```
[1]
The news is Fake
```

In [36]:
```
y_pred2[3]
```

Out[36]:
```
1
```

Tables

Performance metrics, comparisons

Sl.No.	Machine Learning Model	Accuracy	f1-score
1	Logistic Regression	0.98	0.98
2	Naive Bayes	0.94	0.95
3	Decision Tree	0.92	0.92

Inferences from results

The accuracy scores of logistic regression, naïve bayes and decision tree on the training data are 99.02%, 97.44% and 91.78% respectively whereas the accuracy scores of the same machine learning algorithms on the testing data are 98.28%, 94.76% and 92.04% respectively. Clearly, we can see that the accuracy of logistic regression model on our final dataset comes out to be much more than the other two models. Therefore, it can be said that logistic regression can be used as a reliable method for detection of fake news and real news.

Conclusion

The issue of false news garnered a lot of attention in 2016, during the most recent US presidential elections. As more data and study became available, it was shown that 62% of adults in the United States distributed false information on social media. Three different ML algorithms are being used by us. When employing the logistic regression model, the realized model has attained the highest level of accuracy. The accuracy rating at its highest was 98.28%. By producing results that are superior to the listed related work, this algorithm improves the classification accuracy. From the results above, we conclude that:

1. The logistic regression is better than naïve bayes and decision tree classifier in the classification accuracy of fake news dataset.

2. The pre-processing steps using our dataset give better results. These steps had a significant impact on increasing the accuracy of the classification.

3. The type of dataset collected also has a significant impact on the classification accuracy of this work.

Scope for future work

We aim to work on the following areas:

- **Detection of Non-traditional Fake News:**Based on how fake news was defined, fake news can also be news articles with (1) outdated knowledge, e.g., “Britain has control over fifty-six colonial countries” or (2) false claims in some parts of the news content (text and images), i.e., news content is partially (in)correct.
- **Fake News Early Detection:** Fake news early detection aims to detect fake news at an early stage before it becomes widespread so that one can take early actions for fake news mitigation and intervention. Early detection is especially crucial for fake news as the more fake news spreads, the more likely people are to trust it. To detect fake news at an early stage, one has to primarily and efficiently rely on news content and limited social context information, which leads to facing multiple challenges.
- **Identifying Check-worthy Content**:With new information created and circulated online at an unprecedented rate, identifying check-worthy content or topics can improve the efficiency of fake news detection and intervention by prioritizing content or topics that are check-worthy.Whether a given news content or topic is check-worthy can be measured by, e.g., (i) its newsworthiness or potential to influence society, or (ii) its historical likelihood of being fake news.

References

[1] Granik, Mykhailo, and Volodymyr Mesyura. "Fake news detection using naive Bayes classifier." 2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON). IEEE, 2017.

[2] Probierz, Barbara, Piotr Stefański, and Jan Kozak. "Rapid detection of fake news based on machine learning methods." Procedia Computer Science 192 (2021): 2893-2902.

[3] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. In International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, pages 127–138. Springer, 2017.

[4] Florian Sauvageau. Les fausses nouvelles, nouveaux visages, nouveaux défis. Comment déterminer la valeur de l’information dans les sociétés démocratiques? Presses de l’Université Laval, 2018.

[5] Cédric Maigrot, Ewa Kijak, and Vincent Claveau. Fusion par apprentissage pour la détection de fausses informations dans les réseaux sociaux. Document numerique, 21(3):55–80, 2018.

[6] Junaed Younus Khan, Md Khondaker, Tawkat Islam, Anindya Iqbal, and Sadia Afroz. A benchmark study on machine learning methods for fake news detection. arXiv preprint arXiv:1905.04749, 2019.

[7] Xinyi Zhou and Reza Zafarani. 2020. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. ACM Comput. Surv. 53, 5, Article 109 (September 2021), 40 pages.